Visual Studio

ユニバーサル Windows プラットフォーム ユーザー エクスペリエンス ガイドライン

更新 2015 年 12 月

この文書が翻訳版の場合、オリジナル版と内容に違いが認められた場合にはオリジナル版に従うものとします。記載された情報は発行日時点のものです。

本書に記載された情報について、Microsoft は明示または黙示を問わずいかなる保証をするものでもありません。

この文書に記載されている URL や参照しているウェブサイトを含む情報や見解は、予告なく変更される場合があます。断りがない限り、例として記載されている企業、組織、製品、ドメイン名、メールアドレス、ロゴ、人名、場所、事象は架空のもので、実在の企業、組織、製品、ドメイン名、メールアドレス、ロゴ、人名、場所、事象とは関連性がなく、また意図したり暗示するものではありません。利用者はすべての適用される著作権法にした従う責任を負うものとします。著作権により権利の制限がない場合は、Microsoft Corporation による明記された許諾がない限り、この文書のいずれの部分も複製、保管、復旧用システムへの取り入れ、いかなる方法 (電子的、機械的、複写式、記録、その他) もしくはいかなる目的で任意の形式への送信をすることができません。

Microsoft は、この文書にある事柄について特許、特許出願中、商標、著作権、もしくはその他の知的財産の権利を保有する場合があります。Microsoft からの文書による使用許諾契約に明示的に示された場合を除き、この文書の提供はそれらの特許、商標、著作権もしくはその他の知的財産のいかなる許諾を与えるものではありません。

Microsoft は、米国における Microsoft Corporation および、またはその他の国における登録商標もしくは商標です。

実在する会社の名称と製品はそれぞれの所有者に帰属する場合があります。

目次

はじめに

Windows 10 プラットフォームとは

開発者にとって、Windows 10 は、生産性や創造性の向上、そして娯楽のために Windows を毎日使う世界中のたくさんの人々との距離を縮めるすばらしいプラットフォームとなります。また、Windows 10 は、世界規模の顧客にアプローチするためにかつてない手段と機会を提供します。Windows 10 は開発者が使用できる最上位のプラットフォーム機会であり、すばらしい Windows ストアのアプリをたくさんの潜在的なユーザーに提供できるビジネス チャンスといえます。

アプリは Windows 10 エクスペリエンスの中心に位置付けられています。アプリはアクティビティやコンテンツによって生命を吹き込まれます。ユーザーはウィンドウ表示や全画面表示の Windows ストア アプリに夢中になって、オペレーティング システムではなくコンテンツに集中できます。

Windows 10 プラットフォームは、Windows 8 で導入されたアプリ モデルの進化形である Windows ランタイム (WinRT) をさらに進化させています。 Windows 10 の統合されたコアにまとめる、ユニバーサル Windows プラットフォーム (UWP) が導入されています。コアの一部として、UWP は Windows 10 を実行するすべてのデバイスで利用可能な共通アプリ プラットフォーム を提供し、生産性および流動性に適した「妥協のない」エクスペリエンスを実現するという約束を果たしています。UWP は、PC、Windows Phone、Xbox、IoT などのデバイスでも稼働し、さまざまなデバイスでのコンテンツの作成に関するガイドラインを定めると同時に、柔軟性やデバイス独自の機能も提供しています。

UWP は、Windows 8 と同様にアプリが主役であり、またシステムを操作する新しい方法が導入されています。Windows ストアでは、デバイスに依存しないアプリの配布場所であり、優れたマーチャンダイジングとアプリの検索のしやすさにより、優れた収益機会を引き続き提供します。もちろん、UWP では、プログラミング言語 (C#、C++、JavaScript、または VB)、プレゼンテーション テクノロジ (XAML、HTML、または DirectX) の独自の選択範囲を開発者に提供すると共に、Windows ストアによってビジネス モデルを選択できます。UWP を活用したアプリを作成するには、UWP が提供するエクスペリンスを活用するためにも デザイン ガイドラインを知ることが重要になります。本ホワイトペーパーでは、エクスペリエンスをデザインするために必要な、基本的な事項を説明します。

ユニバーサル Windows アプリとは

ユニバーサル Windows アプリは、ユニバーサル Windows プラットフォーム (UWP) 上に構築された Windows エクスペリエンスであり、Windows 8 で Windows ランタイムとして初めて導入されました。ユニバーサル Windows アプリの中核となるのは、ユーザーがすべてのデバイスでモバイル エクスペリエンスを手に入れて、目の前の作業に一番便利で効率的なデバイスを使いたいという考え方です。

Windows 10 を使うことで、UWP 用アプリの開発がこれまでよりも簡単になります。API セット、アプリ パッケージ、ストアをそれぞれ 1 つ使うだけで、すべての Windows 10 デバイス (PC、タブレット、電話など) で利用可能なアプリを作成できます。また、さまざまな画面サイズ、操作方式 (タッチ、マウスとキーボード、ゲームコントローラー、ペン) のサポートもより簡単になります。

では、ユニバーサル Windows アプリとは何でしょうか。

ユニバーサル Windows アプリの一番の特徴は何でしょうか。それらの特徴をいくつか挙げます。

- OS ではなくデバイス ファミリを対象にする。
  デバイス ファミリに基づいて、デバイス ファミリのデバイス全体で想定できる API、システム特性、動作を特定します。ストアからアプリをインストールできる一連のデバイスも決定します。
- アプリは .AppX パッケージ形式を使ってパッケージ化されて配布される。
  すべてのユニバーサル Windows アプリは、AppX パッケージとして配布されます。これにより、信頼できるインストール方法をユーザーに提供でき、アプリはシームレスに展開、更新できるようになります。
- 1 つのストアですべてのデバイスに対応する。

アプリ開発者として登録した後、アプリをストアに提出し、すべてまたは特定のデバイス ファミリ向けに販売できます。Windows デバイス向けのすべてのアプリを 1 か所で提出、管理できます。

- デバイス ファミリに共通の API セットが用意されている。
  ユニバーサル Windows プラットフォーム (UWP) のコア API はすべての Windows デバイス ファミリに共通です。アプリにコア API のみを使う場合、そのアプリはいずれの Windows 10 デバイスでも動作します。
- 拡張機能 SDK によりアプリを特定のデバイスで機能アップする。
  拡張機能 SDK により、各デバイス ファミリに特化した API が追加されます。アプリが特定のデバイス ファミリ向けであれば、これらの API を使ってそのデバイスで機能アップできます。この場合も、すべてのデバイスで動作する 1 つのアプリ パッケージを用意できますが、拡張 API の呼び出し前に、アプリが実行されるデバイス ファミリを確認するようにします。
- アダプティブ コントロールおよび入力。
  UI 要素では有効ピクセル (「レスポンシブ デザイン 101」を参照) が使われるため、UI はデバイスの画面のピクセル数に基づいて自動的に調整されます。また、キーボード、マウス、タッチ、ペン、Xbox One コントローラーなど、さまざまな種類の入力デバイスで問題なく機能します。 アプリが実行される特定のデバイスや画面サイズに合わせて UI をさらに調整する場合は、新しく追加されたレイアウト パネルとツールが便利です。

より詳細な情報が知りたい場合は、「ユニバーサル Windows プラットフォーム アプリのガイド」を参照してください。

使い慣れた言語の使用

ユニバーサル Windows アプリは、JavaScript、C#、Visual Basic、JavaScript と HTML、C++ と DirectX、C++ と XAML(Extensible Application Markup Language) など、使い慣れたプログラミング言語で作成できます。ある言語で作ったコンポーネントを、別の言語で作ったアプリで使うこともできます。

ユニバーサル Windows アプリは、オペレーティング システムに組み込まれているネイティブな API である Windows ランタイムを使うことができます。この API は C++ で実装され、JavaScript、C#、Visual Basic、C++ の各言語で自然な形でサポートされます。

Microsoft Visual Studio 2015 には、各言語のユニバーサル Windows アプリ用テンプレートが用意されており、すべてのデバイス向けのアプリを 1 つのプロジェクトから作成できます。作業が終わったら、Visual Studio 内からアプリ パッケージを生成し、Windows ストアに提出できます。これにより、すべての Windows 10 デバイスのユーザーがそのアプリを入手できるようになります。

Windows で ユニバーサル Windows アプリを魅力的なものにする

Windows では、アプリが、関連するリアルタイム情報をユーザーに表示し、ユーザーが何度も戻ってくるようにすることができます。最新のアプリ エコシステムでは、アプリをユーザーの生活の中で常に最初に思い出されるように魅力的なものにしておく必要があります。Windows では、ユーザーが繰り返しアプリを使うように、次のような多くのリソースを提供しています。

- ライブ タイルとロック画面は、コンテキストに関連したタイムリーな情報をひとめでわかるように表示します。
- プッシュ通知は、ユーザーが必要なときにリアルタイムの最新の通知に注目できるようにします。
- アクション センターでは、ユーザーが操作を実行する必要のある通知やコンテンツを整理して表示できます。
- バックグラウンドの実行とトリガーにより、ユーザーが必要とするときだけアプリが有効になります。

- アプリで音声と Bluetooth LE デバイスを使うと、 ユーザーはそれらのデバイスを取り巻く環境と対話できます。

最終的に、ローミング データと Windows 資格情報保管ボックスを使うと、ユーザーがアプリを実行するすべての Windows ベースのデバイスで一貫したローミング エクスペリエンスを実現できます。ローミング データを使うと、独自の同期インフラストラクチャを構築する必要なく、ユーザーの基本設定とその他の設定をクラウドに簡単に保存できます。資格情報保管ボックスには、ユーザーの資格情報を保存できます。このボックスにおける最優先事項はセキュリティと信頼性です。

ニーズに合わせてアプリの収益を得る

Windows では、ニーズに合わせて (電話、タブレット、PC などで) アプリの収益を得る方法を選ぶことができます。ここでは、アプリとアプリが提供するサービスで収益を得るさまざまな方法について説明します。必要なのは、ニーズに合った最適な方法を選ぶことだけです。

- 有料のダウンロードは最も簡単な方法です。必要な作業は価格の指定だけです。
- 試用版は、ユーザーがアプリを購入する前に試すことができる独自の方法を提供しているため、従来の"試供品"オプションよりも見つけやすく簡単にコンバージョンできます。
- アプリ内購入では、アプリで収益を得ることに関して柔軟性が最も高くなります。

作業の開始

より詳細な情報が知りたい場合は、「ユニバーサル Windows プラットフォーム アプリのガイド」を参照してください。次に、「準備」を確認し、アプリの作成を始めるために必要なツールをダウンロードしてください。

ユニバーサル Windows プラットフォーム (UWP) ガイド

このガイドでは、さまざまなデバイスで実行できる UWP アプリについて説明します。

はじめに

ここでは、次の項目について説明します。

- デバイス ファミリの定義およびターゲットにするデバイス ファイルを決定する方法。
- さまざまなデバイスのフォーム ファクターに合わせて UI を対応させることができる新しい UI コントロールおよびパネル。
- アプリで利用できる API サーフェスを理解および制御する方法。

Windows 8 では、Windows アプリ モデルの進化形である Windows ランタイム (WinRT) が導入されました。これは一般的なアプリケーション アーキテクチャを意図したものでした。

Windows Phone 8.1 が使用できるようになったとき、Windows ランタイムは Windows Phone 8.1 と Windows の両方に配置されました。これにより、開発者は共有コードベースを使って Windows と Windows Phone の両方をターゲットにするユニバーサル Windows 8 アプリの作成が可能となりました。

Windows 10 では、Windows ランタイム モデルをさらに進化させ、Windows 10 の統合されたコアにまとめる、ユニバーサル Windows プラットフォーム (UWP) が導入されています。コアの一部として、UWP は Windows 10 を実行するすべてのデバイスで利用可能な共通ア

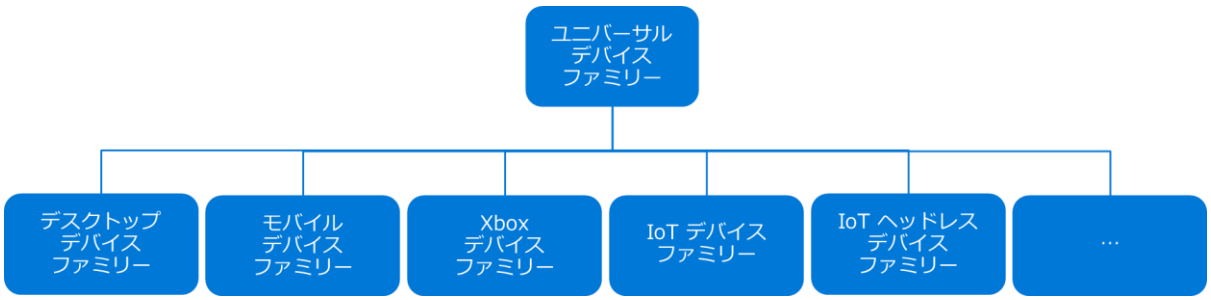
プリ プラットフォームを提供するようになりました。この進化により、UWP をターゲットとするアプリはすべてのデバイスに共通する WinRT API だけでなく、アプリが実行されているデバイス ファミリに固有の API (Win32 と .NET API を含む) も呼び出すことができます。UWP はさまざまなデバイスに保証されたコア API レイヤーを提供します。つまり、多様なデバイスにインストールできる単一のアプリ パッケージを作成することができます。そして、その単一のアプリ パッケージで、Windows ストアはアプリが実行できるすべてのデバイスの種類に利用可能な統合された配布チャネルを提供します。

UWP アプリはさまざまなフォーム ファクターと入力モダリティを備えた多様な種類のデバイスで実行されるため、各デバイスに合わせて調整して、各デバイスの独自の機能のロックを解除する必要があります。デバイスはデバイス固有の API を保証された API レイヤーに追加します。これらの固有な API に条件付きでアクセスするコードを作成して、アプリが他のデバイスで異なるエクスペリエンスを表示している間に 1 種類のデバイスに固有の機能の使い勝手をよくすることができます。アダプティブ UI コントロールと新しいレイアウト パネルを使うと、UI をさまざまな画面解像度で調整できます。

デバイス ファミリ

Windows 8.1 アプリと Windows Phone 8.1 アプリがターゲットとするオペレーティング システム (OS) は、Windows または Windows Phone です。Windows 10 では、オペレーティング システムをターゲットにする代わりに、1 つまたは複数のデバイス ファミリに設定されたアプリをターゲットにします。デバイス ファミリは、API、システム特性、デバイス フ

ァミリ内のすべてのデバイスで予想される動作を識別します。ストアからアプリをインストールできる一連のデバイスも決定します。次にデバイス ファミリの階層を示します。



デバイス ファミリは、まとめられて、名前とバージョン番号が指定された一連の API です。デバイス ファミリは OS の基盤です。PC ではデスクトップ デバイス ファミリに基づいているデスクトップ OS を実行します。電話やタブレットなどでは、モバイル デバイス ファミリに基づいているモバイル OS を実行します。その他のデバイス ファミリも同様の構造になります。

ユニバーサル デバイス ファミリは特殊です。直接には、いずれの OS の基盤でもありません。代わりに、ユニバーサル デバイス ファミリの一連の API は、子デバイス ファミリに継承されます。ユニバーサル デバイス ファミリ API はすべての OS に存在することが保証され、その結果、すべてのデバイスに存在することが保証されます。

各子デバイス ファミリでは、独自の API が継承する API に追加されます。結果として得られる子デバイス ファミリの API の和集合は、そのデバイス ファミリに基づく OS、つまり、その OS を実行するすべてのデバイス、に存在することが保証されます。

デバイス ファミリの利点の 1 つは、電話、タブレット、およびデスクトップ コンピューターから Surface Hub と Xbox コンソールまでのいずれか、またはすべてのデバイスでアプリを実行できることです。アプリはユニバーサル デバイス ファミリの外部のデバイスの機能を動的に検出して使うアダプティブ コードを使うこともできます。

アプリがターゲットにするデバイス ファミリに関する決定は自分で行うことができます。この決定はこれらの重要な点でアプリに影響します。次の内容を決定します。

- アプリが実行されるときに存在する (したがって、自由に呼び出すことができる) と期待できる一連の API。
- 条件ステートメント内のみに記述された安全な一連の API 呼び出し。

- ストア (および結果として、検討する必要のあるフォーム ファクター) からアプリをインストールできる一連のデバイス。

デバイス ファミリの選択には主に次の 2 つの結果があります。アプリによって無条件に呼び出せる API サーフェスとアプリをインストールできるデバイスの数です。これら 2 つの要素にはトレードオフがあり、逆相関します。たとえば、UWP アプリは特にユニバーサル デバイス ファミリをターゲットとするアプリで、その結果、すべてのデバイスに利用できます。ユニバーサル デバイス ファミリをターゲットとするアプリでは、ユニバーサル デバイス ファミリの API のみが存在することが期待できます (ターゲットとするものであるため)。その他の API は、条件付きで呼び出す必要があります。また、このようなアプリはさまざまなデバイスで実行できるため、高度なアダプティブ UI と包括的な入力機能を備えている必要があります。Windows モバイル アプリは、特にモバイル デバイス ファミリをターゲットとするアプリで、OS がモバイル デバイス ファミリ (電話、タブレット、類似したデバイスを含む) に基づくデバイスで利用できます。モバイル デバイス ファミリのアプリには、モバイル デバイス ファミリのすべての API が存在することが期待でき、その UI はある程度アダプティブである必要があります。IoT デバイス ファミリをターゲットとするアプリは IoT デバイスのみにインストールすることができ、IoT デバイス ファミリ内のすべての API が存在することが期待できます。このアプリは特定の種類のデバイスでのみ実行されることがわかっているため、UI と入力機能に特化することができます。

ターゲットにするデバイス ファミリを決定するために役立つ考慮事項をいくつか紹介します。

アプリの適用範囲の最大化

アプリを適用できるデバイスの範囲を最大限に広げ、できるだけ多くの種類のデバイス上で実行するために、アプリはユニバーサル デバイス ファミリをターゲットとします。これにより、アプリはユニバーサルに基づくすべてのデバイス ファミリを自動的にターゲットにします (図内、ユニバーサルのすべての子)。つまり、アプリはそれらのデバイス ファミリに基づくすべての OS で実行され、それらのオペレーティング システムを実行するすべてのデバイスで実行されます。それらのすべてのデバイスで利用できることが保証される API のみが、ターゲットとするユニバーサル デバイス ファミリの特定のバージョンで定義され

るセットです。(このリリースでは、バージョンは常に 10.0.x.0)アプリがターゲットのデバイス ファミリのバージョン以外の API を呼び出す方法については、このトピックの後方にある「コードの記述」をご覧ください。

アプリを 1 種類のデバイス に制限する

アプリをさまざまなデバイスで実行せずに、たとえば、デスクトップ PC や Xbox コンソールに特化する場合があります。その場合、子デバイス ファミリのいずれかでアプリをターゲットにすることができます。たとえば、デスクトップ デバイス ファミリをターゲットとする場合、アプリで利用可能であることが保証されている API には、ユニバーサル デバイス ファミリから継承された API とデスクトップ デバイス ファミリに固有の API が含まれます。

アプリを使用可能なすべてのデバイスのサブセットに制限する

ユニバーサル デバイス ファミリをターゲットとしたり、子デバイス ファミリのいずれかをターゲットとしたりする代わりに、2 つ以上の子デバイス ファミリをターゲットにすることができます。アプリのターゲットとして、デスクトップとモバイルが適している場合があります。また、デスクトップと Xbox が適している場合があります。また、デスクトップ、Xbox、Surface Hub が適している場合があります。

デバイス ファミリーの特定バージョンのサポートを除外する

まれに、特定のデバイス ファミリの特定のバージョンのデバイスを除くすべてでアプリを実行する場合があります。たとえば、アプリがユニバーサル デバイス ファミリのバージョン 10.0.x.0 をターゲットとするとします。将来的にオペレーティング システムのバージョンが、たとえば 10.0.x.2 に変更された場合、その時点でアプリのターゲットをユニバーサル 10.0.x.0 と Xbox 10.0.x.1 にして、アプリを Xbox のバージョン 10.0.x.1 以外のすべてで実行するように指定することができます。その後、アプリは Xbox 10.0.x.1 以前のデバイス ファミリのバージョンのセットでは使用できなくなります。

Microsoft Visual Studio は、既定で **Windows.Universal** が指定されます。つまり、アプリ パッケージ マニフェストのデバイス ファミリとして指定されているということになります。ストアに掲載するアプリの条件としてデバイス ファミリを指定するには、Package.appxmanifest ファイルの **TargetDeviceFamily** 要素を手作業で編集します。

UI とユニバーサルな入力

UWP アプリは、異なる形式の入力、画面の解像度、DPI 密度、その他の固有の特性を備えた、多くの異なる種類のデバイスで実行できます。Windows 10 には、新しいユニバーサル コントロール、レイアウト パネル、およびアプリが動作するデバイスに UI を対応させるためのツールが用意されています。 たとえば、アプリがデスクトップ コンピューターとモバイル デバイスで実行されている場合、画面の解像度の違いを利用して UI を調整することができます。

アプリの UI の一部はデバイスの違いに自動的に対応します。ボタン、スライダーなどのコントロールは、デバイス ファミリと入力モードに対して自動的に対応します。ただし、アプリのユーザー エクスペリエンスの設計は、アプリが実行されているデバイスへの対応が必要になる場合があります。たとえば、フォト アプリが小型のハンドヘルド デバイスで実行されている場合、UI を片手での使用に適するように調整する必要があります。フォト アプリがデスクトップ コンピューターで実行されている場合、UI は広い画面スペースを利用するように調整する必要があります。

Windows では、UI を次の機能を備えた複数のデバイスを対象としたものにすることができます。

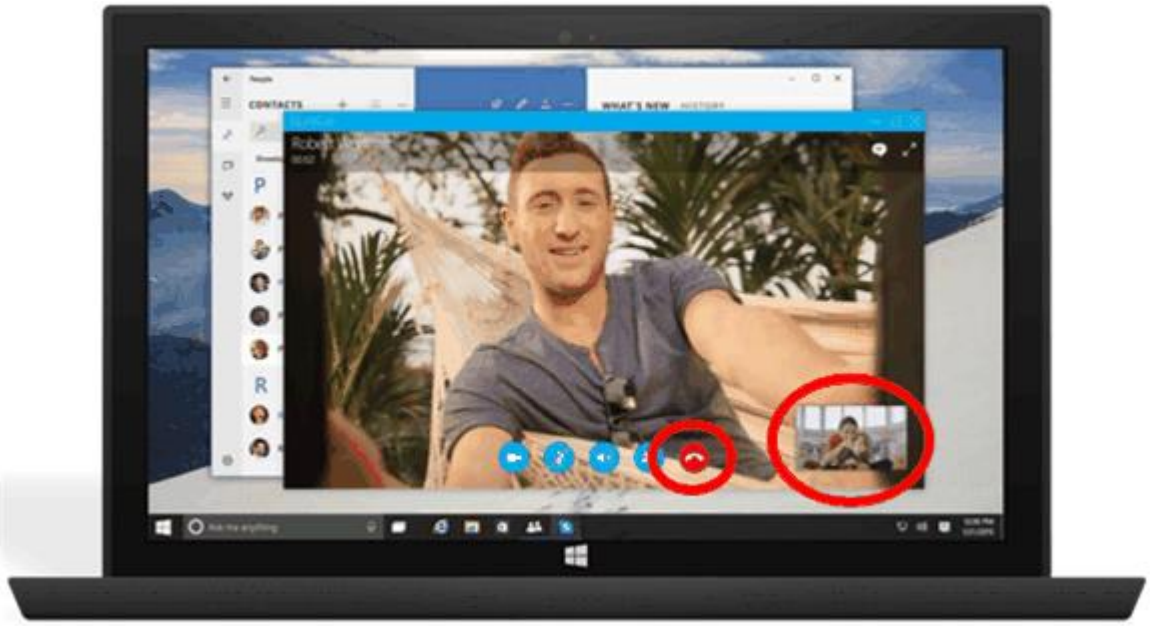
- UI をデバイスの画面の解像度に合わせて最適化するために役立つユニバーサル コントロールとレイアウト パネル
- 一般的な入力処理では、タッチ、ペン、マウス、キーボード、またはコントローラー (Microsoft Xbox コントローラーなど) による入力を受け取ることができます。
- さまざまな画面の解像度に合わせて変化する UI の設計に役立つツール
- デバイス間で異なる解像度と DPI の相違を調整するアダプティブ スケーリング

ユニバーサル コントロールとレイアウト パネル

Windows 10 には、Calendar や SplitView などの新しいコントロールが含まれています。以前は Windows Phone のみで使用できたピボット コントロールも、ユニバーサル デバイス ファミリで使用できるようになりました。

コントロールは、より大きな画面で適切に動作し、デバイスで使用できる画面ピクセル数に対応し、キーボード、マウス、タッチ、ペン、コントローラー (Xbox コントローラーなど) などの複数の種類の入力で適切に動作するように更新されています。

アプリが実行されるデバイスの画面の解像度に基づいて、全体的な UI レイアウトの調整が必要になる可能性があります。たとえば、デスクトップで実行されている通信アプリに呼び出し元のピクチャ イン ピクチャとマウス入力に適したコントロールが含まれている場合があります。



ただし、アプリを電話で実行する場合は、操作する画面の作業領域が小さいため、アプリではピクチャ イン ピクチャ ビューを削除して、片手で操作しやすいように通話ボタンを大きくする場合があります。

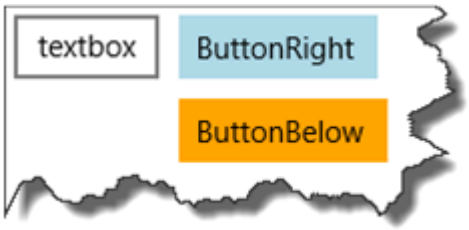
使用可能な画面領域の大きさに基づいて全体的な UI レイアウトを調整するために、Windows 10 ではアダプティブ パネルと VisualState が導入されています。

アダプティブ パネルを使った アダプティブ UI デザイン

レイアウト パネルでは、利用可能な領域に応じて、子にサイズと位置を指定します。 たとえば、**StackPanel** は子を連続的に順序付けます (横方向または縦方向)。 **Grid** は子をセルに配置する CSS グリッドのようなものです。

新しい **RelativePanel** は子要素間の関係で定義されるレイアウトのスタイルを実装します。画面の解像度の変更に対応できるアプリのレイアウトの作成で使用します。**RelativePanel** は要素間の関係を定義して要素を並べ替えやすくして、入れ子になったレイアウトを使わずにさらに動的な UI を作成できるようにします。

次に示す例では、**blueButton** が **textBox1** の右側に表示されています (この時点では、向きのやレイアウトの変更は気にしないでください)。**BrangeButton** は、**blueButton** の下に表示されています – テキストを入力することで **textBox1** の幅が変更されても配置は変わりません -。このようなレイアウトは、以前なら **Grid** の列や行を使うことで実現していましたが、より少ないマークアップを使うことで実現することができます。



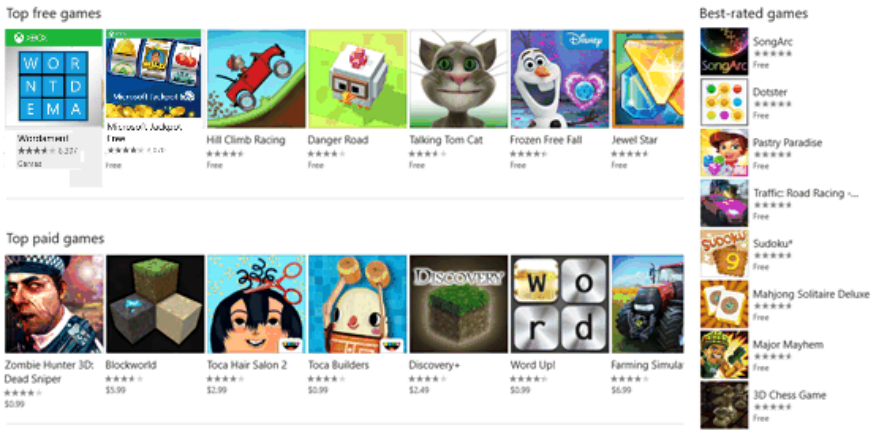

```
XAML
<RelativePanel>
    <TextBox x:Name="textBox1" Text="textbox" Margin="5"/>
    <Button x:Name="blueButton" Margin="5" Background="LightBlue"
                Content="ButtonRight" RelativePanel.RightOf="textBox1"/>
    <Button x:Name="orangeButton" Margin="5" Background="Orange"
                Content="ButtonBelow" RelativePanel.RightOf="textBox1"
                RelativePanel.Below="blueButton"/>
</RelativePanel>
```

有効なスクリーン サイズへ対応させる UI の構築には VisualState トリガーを使用する

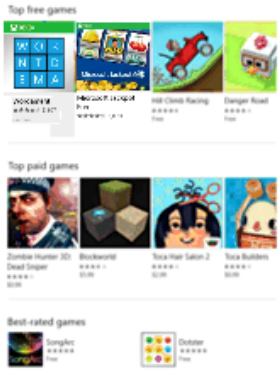
UI はウィンドウ サイズの変化への対応が必要になる場合があります。アダプティブな表示状態により、ウィンドウのサイズの変化に応じて表示状態を変更できます。

StateTriggers は表示状態がアクティブ化されるしきい値を定義し、状態の変更をトリガーしたウィンドウ サイズに適したレイアウト プロパティを設定します。

次の例では、ウィンドウ サイズの幅が 720 ピクセル以上の場合、**wideView** という名前の VisualState がトリガーされ、**[高評価ゲーム]** パネルが **[トップ無料ゲーム]** パネルの右側に、上部を揃えて表示されるように配置します。



ウィンドウが 720 ピクセル未満の場合、**wideView** のトリガーは適合せず、無効になっているため、**narrowView** という VisualState がトリガーされます。**narrowView** の VisualState は **[高評価ゲーム]** パネルを **[トップ無料ゲーム]** パネルの下に、左側を揃えて配置します。

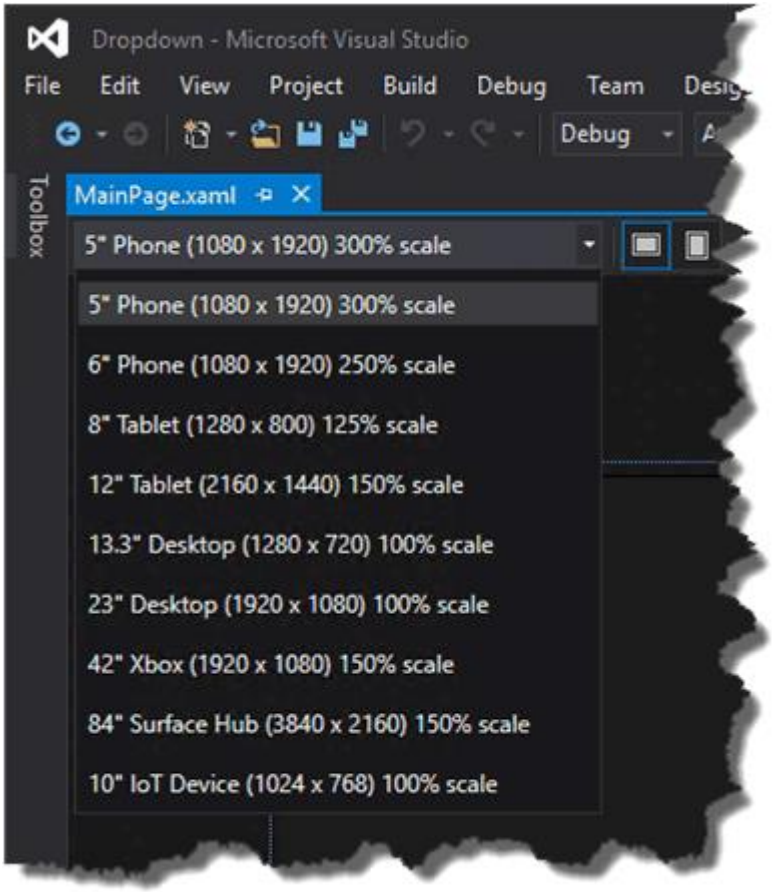


次に説明した VisualState トリガーの XAML を示します。以下の "..." でわかるように、わかりやすくするためにパネルの定義は削除しています。

```
XAML
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
   <VisualStateManager.VisualStateGroups>
      <VisualStateGroup>
         <VisualState x:Name="wideView">
            <VisualState.StateTriggers>
               <AdaptiveTrigger MinWindowWidth="720" />
            </VisualState.StateTriggers>
            <VisualState.Setters>
               <Setter Target="best.(RelativePanel.RightOf)" Value="free"/>
               <Setter Target="best.(RelativePanel.AlignTopWidth)" Value="free"/>
            </VisualState.Setters>
         </VisualState>
         <VisualState x:Name="narrowView">
            <VisualState.Setters>
               <Setter Target="best.(RelativePanel.Below)" Value="paid"/>
               <Setter Target="best.(RelativePanel.AlignLeftWithPanel)" Value="true"/>
            </VisualState.Setters>
            <VisualState.StateTriggers>
               <AdaptiveTrigger MinWindowWidth="0" />
            </VisualState.StateTriggers>
         </VisualState>
      </VisualStateGroup>
   </VisualStateManager.VisualStateGroups>
   ...
</Grid>
```

ツール

既定では、できるだけ幅広いデバイス ファミリをターゲットにします。特定のデバイスでのアプリの外観とレイアウトを表示する準備ができたら、Visual Studio のデバイス プレビュー ツール バーを使用して、小規模または中規模のモバイル デバイス、PC、または大画面テレビで、UI をプレビューします。このようにして、アダプティブな VisualState を調整して、テストすることができます。



サポートするすべてのデバイスの種類について事前に決定する必要はありません。プロジェクトには、追加のデバイス サイズを後で追加することができます。

アダプティブ スケーリング

Windows 10 には、既存のスケーリング モデルの進化形が導入されています。ベクター コンテンツの表示スケールに加えて、さまざまな画面サイズと画面の解像度で UI 要素に一貫したサイズを提供するスケール ファクターの統合されたセットがあります。スケール ファ

クターは、iOS や Android などの他のオペレーティング システムのスケール ファクターとも互換性があります。これにより、これらのプラットフォーム間でのアセットの共有が簡単になります。

ストアでは、デバイスの DPI に基づいて、ダウンロードするアセットが選ばれます。デバイスに最適なアセットのみがダウンロードされます。

共通の入力処理

マウス、キーボード、タッチ、ペン、コントローラー (たとえば、Xbox コントローラー) などのさまざまな入力を処理するユニバーサル コントロールを使ってユニバーサル Windows アプリをビルドすることができます。従来、手描き入力はペン入力のみに関連付けられていましたが、Windows 10 では、一部のデバイスでのタッチ、およびポインター入力によって手描き入力ができます。手描き入力は、多くのデバイス (モバイル デバイスなど) でサポートされており、数行のコードだけで簡単に組み込むことができます。

次の API は、入力へのアクセスを提供します。

- **CoreIndependentInputSource** は、メイン スレッド、あるいはバックグラウンド スレッドで生の入力を処理できるようにする新しい API です。
- **PointerPointer** は、生のタッチ、マウス、ペンのデータを、1 つの一貫したインターフェイスやイベントのセットに統合します。これらのインターフェイスやイベントは、**CoreInput** を使うことにより、メイン スレッドまたはバックグラウンド スレッドで利用できます。
- **PointerDevice** は、デバイスがサポートする機能を調べるデバイス API です。これを使うことで、デバイスで使用できる入力機能を決定できるようになります。
- 新しい **InkCanvas** コントロールと **InkPresenter** Windows ランタイム API は、インクのストローク データへのアクセスを可能にします。

コードの記述

Visual Stdudio の Windows 10 プロジェクトに使用するプログラミング言語オプションには、Visual C++、C#、Visual Basic、JavaScript が含まれます。Visual C++、C#、Visual Basic については、完全な再現性を持つ、ネイティブ UI エクスペリエンスのために XAML を使うことができます。Visual C++ については、XAML の代わりに、または XAML に加えて、DirectX での描画を選ぶことができます。JavaScript については、プレゼンテーション レイヤーは HTML になり、HTML はもちろん、クロスプラットフォーム Web 標準です。コードと UI の大部分はユニバーサルで、すべての場所で同様に実行されます。特定のデバイス ファミリに合わせて調整されたコードと特定のフォーム ファクターに合わせて調整された UI については、アダプティブ コードとアダプティブ UI を使用するオプションがあります。これらの異なる場合を詳しく見てみましょう。

ターゲット デバイス ファミリで実装されている API の呼び出し

API を呼び出す場合は、アプリがターゲットとしているデバイス ファミリでその API が実装されているかどうかを把握しておく必要があります。判断がつかない場合は、API リファレンス ドキュメントで検索することができます。関連するトピックを開いて、要件セクションを表示すると、実装するデバイス ファミリの内容が表示されます。たとえば、アプリがユニバーサル デバイス ファミリのバージョン 10.0.x.0 をターゲットにしていて、**Windows.UI.Core.SystemNavigationManager** クラスのメンバーを呼び出すとします。この例におけるデバイス ファミリは "ユニバーサル" です。呼び出そうとするクラス メンバーもターゲット内にあることを、この場合でもさらに確認することをお勧めします。したがって、この例では、API はアプリをインストールできるすべてのデバイスに存在することが保証されることがわかり、通常と同様に、コードに API を呼び出すことができます。

```
Windows.UI.Core.SystemNavigationManager.GetForCurrentView().BackRequested
                                                    += TestView_BackRequested;
```

もう 1 つの例として、アプリが Xbox デバイス ファミリのバージョン 10.0.x.0 をターゲットとしていて、呼び出す API のリファレンス トピックでは API は Xbox デバイス ファミリのバージョン 10.0.x.0 に導入されたと示されているとします。その場合も、API はアプリがイ

ンストールできるすべてのデバイス上に存在することが保証されます。そのため、コード内で通常の方法でその API を呼び出すことができます。

Visual Studio のインテリセンスはアプリのターゲット デバイス ファミリまたは参照している拡張 SDK によって実装されていない限り、API を認識しないことに注意してください。そのため、拡張 SDK を参照していない場合は、インテリセンスに表示される API は確実にターゲット デバイス ファミリにあり、自由に呼び出すことができます。

ターゲット デバイス ファミリに実装されていない API を呼び出す

API を呼び出そうとしてもドキュメントの一覧にターゲット デバイス ファミリがない場合があります。その場合、その API を呼び出すためにアダプティブ コードを記述することができます。

ApiInformation クラスを使ったアダプティブ コード

アダプティブ コードを記述するには 2 つの手順があります。最初の手順は、プロジェクトで使用可能な、アクセスする API を作成します。そのためには、条件付きで呼び出す API を所有しているデバイス ファミリを表す**拡張 SDK** への参照を追加します。拡張 SDK をご覧ください。

2 番目の手順は、呼び出す API の存在をテストするためにコードの条件で **Windows.Foundation.Metadata.ApiInformation** クラスを使います。このテストの条件は、アプリの実行時に必ず評価されますが、API が存在するデバイスに対してのみ true と評価され、呼び出しが可能になります。

いくつかの API のみを呼び出す場合は、次のように **ApiInformation.IsTypePresent** メソッドを使うことができます。

```
// 注: 繰り返し使用する場合は、結果をキャッシュします
    bool isHardwareButtonsAPIPresent =
        Windows.Foundation.Metadata.ApiInformation.IsTypePresent(
                                                "Windows.Phone.UI.Input.HardwareButtons");

    if (isHardwareButtonsAPIPresent)
    {
```

```
        Windows.Phone.UI.Input.HardwareButtons.CameraPressed +=
            HardwareButtons_CameraPressed;
    }
```

この場合、**HardwareButton** クラスが存在すると **CameraPressed** イベントが存在します。これはクラスとメンバーには同じ要件の情報があるためです。しかし、やがて、新しいメンバーは既に導入されたクラスに追加され、それらのメンバーは後でバージョン番号に "導入 " されます。このような場合、**IsTypePresent** を使う代わりに、**IsEventPresent**、**IsMethodPresent**、**IsPropertyPresent**、および同様のメソッドを使って個々のメンバーの存在をテストすることができます。次に例を示します。

```
bool isHardwareButtons_CameraPressedAPIPresent =
        Windows.Foundation.Metadata.ApiInformation.IsEventPresent
            ("Windows.Phone.UI.Input.HardwareButtons", "CameraPressed");
```

デバイス ファミリ内の API のセットは、API コントラクトと呼ばれる小項目にさらに分類されます。API コントラクトの存在をテストするには、**ApiInformation.IsApiContractPresent** メソッドを使うことができます。これは、すべてが API コントラクトの同じバージョンに存在する膨大な数の API の存在をテストする場合に便利です。

```
bool isWindows_Devices_Scanners_ScannerDeviceContract_1_0Present =
        Windows.Foundation.Metadata.ApiInformation.IsApiContractPresent
            ("Windows.Devices.Scanners.ScannerDeviceContract", 1, 0);
```

UWP における Win32 API について

C++/CX で記述された UWP アプリまたは Windows ランタイム コンポーネントは UWP の一部である Win32 API にアクセスします。これらの Win32 API は、すべての Windows 10 デバイス ファミリによって実装されます。アプリを WindowsApp.lib とリンクします。WindowsApp.lib は UWP API へのエクスポートを提供する "包括的な" lib です。WindowsApp.lib へリンクすると、すべての Windows 10 デバイス ファミリに存在する DLL にアプリの依存関係を追加することになります。

UWP アプリで使用する Win32 API の完全なリストは、**API Sets for UWP apps** と **Dlls for UWP apps** を参照してください。

ユーザー エクスペリエンス

ユニバーサル Windows アプリによって、実行されているデバイスの固有の機能を利用することができます。アプリでは、デスクトップ デバイスのすべての機能、タブレットの直接的な自然な操作 (タッチ/ペン入力を含む)、モバイル デバイスの移植性と便利さ、**Surface Hub** のコラボレーション機能を使うことができます。

適切な「**UWP アプリのデザイン**」は、ユーザーがアプリを操作する方法とアプリの外観と機能を決定するプロセスです。ユーザー エクスペリエンスは、ユーザーがアプリでどの程度満足するかを判断する場合に大きな役割を果たします。そのため、この手順は必ず守ってください。「デザインの基本」では、ユニバーサル Windows アプリの設計を紹介します。ユーザーを楽しませる UWP アプリの設計の情報については、「**デザイナー向けのユニバーサル Windows プラットフォーム (UWP) アプリの概要**」をご覧ください。コーディングを開始する前に、ターゲットにするすべての異なるフォーム ファクターについてのアプリの使用についての操作エクスペリエンスを検討するために役立つ「**デバイスの基本情報**」をご覧ください。

さまざまなデバイスでの操作に加えて、複数のデバイスで動作する利点を取り入れるように「**アプリの計画**」を行います。たとえば、次のようなものです。

- クラウド サービスを使用して、デバイス間で同期します。アプリのエクスペリエンスをサポートするにあたって、**Web サービスに接続する方法**について説明します。
- あるデバイスから別のデバイスに移動し、中断した場所から再開するユーザーをサポートする方法を検討します。**通知**と**アプリ内購入**を計画に盛り込みます。これらの機能はさまざまなデバイスで動作する必要があります。
- **ナビゲーション デザインの基本**を使用してワークフローをデザインして、モバイル、小型画面デバイス、大型画面デバイスに対応します。使用可能な解像度ではなく、特定のウィンドウのサイズに**ユーザー インターフェイスを配置**します。

- 小さな携帯電話の画面では役に立たないアプリの機能があるかどうかを検討します。また、固定されたデスクトップ コンピューターでは役に立たず、モバイル デバイスを使う必要がある領域がある場合もあります。たとえば、**位置情報**に関連するシナリオではモバイル デバイスを意味することが多いものです。
- 複数の入力モダリティの対処方法を検討してください。ユーザーが **Cortana**、**音声認識**、**タッチ操作**、**タッチ キーボード**などを使ってアプリを操作する方法については、「**インタラクションのガイドライン**」をご覧ください。

一般的なインタラクションについては、**テキスト入力のガイドライン**を参照してください。

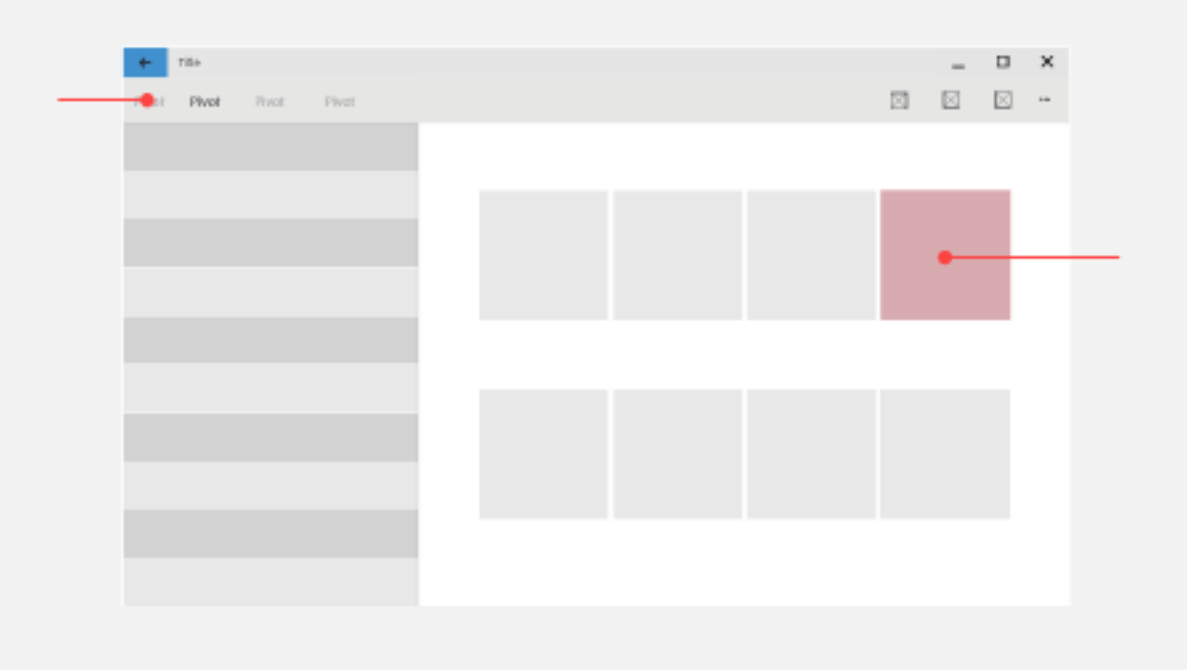
ストアへユニバーサル Windows アプリの提出

新しい統合 Windows デベロッパー センター ダッシュボードにより、Windows デバイス向けのすべてのアプリの管理と申請を 1 か所で行うことができます。新しい機能が追加されたことで、より高度な管理が可能になった一方、プロセスは簡単になりました。また、詳しい**分析レポート**に加えて、**支払いの詳細**も得られるようになりました。いずれも**アプリの宣伝と顧客エンゲージメントの獲得**に役立ちます。

Windows ストアにアプリ公開の申請をする方法については、「**統合 Windows デベロッパー センター ダッシュボード**」の使用をご覧ください。

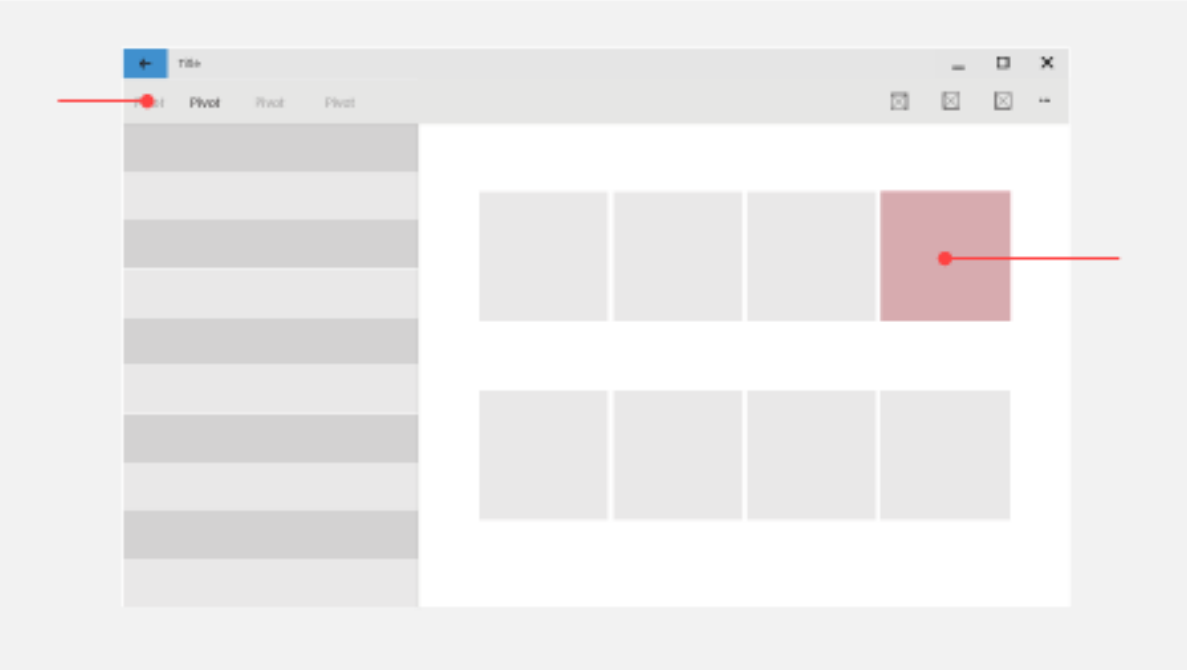
UWP アプリのデザイン

優れたアプリは、優れたユーザー インターフェイスで始まります。携帯電話から、タブレット、PC、Surface Hub に至るまで、Windows 10 ベースのすべてのデバイスで高性能の画面を持つユニバーサル Windows プラットフォーム (UWP) アプリをデザインする方法について説明します。



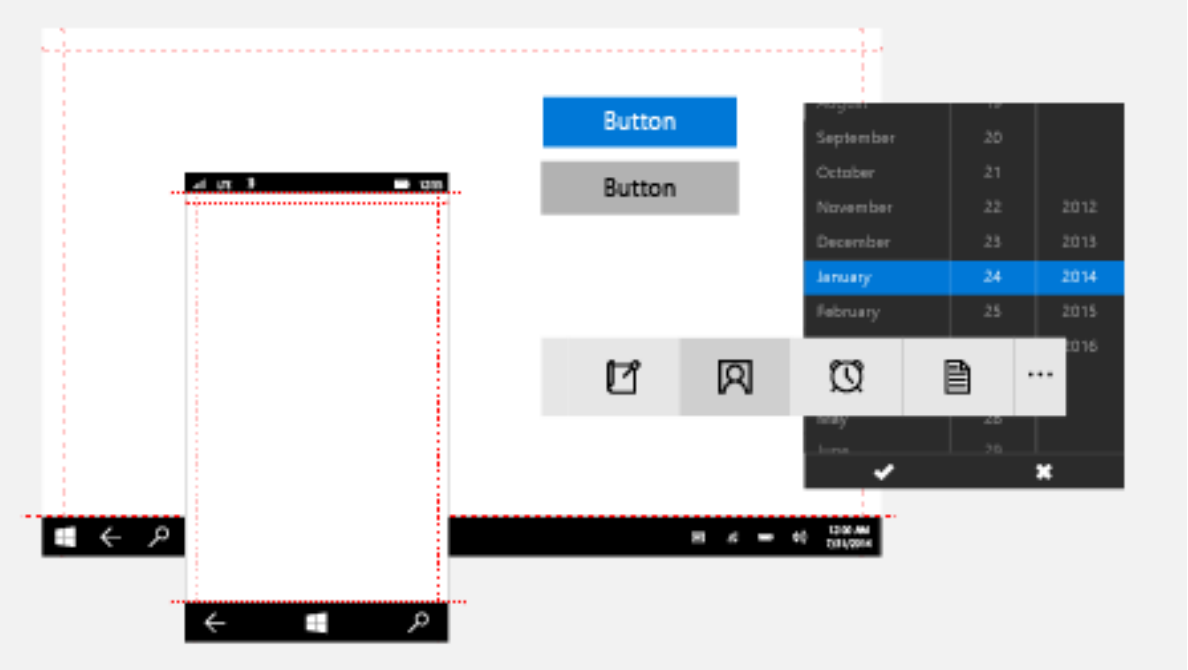
デザインの基本

UWP アプリのデザインを始めるには、プラットフォーム、UI デザインの基本、レスポンシブ デザイン テクニックについて理解します。



ガイドライン

レイアウト、コントロール、ユーザー操作、テキストなどについて、包括的なガイドラインの一覧を参照できます。

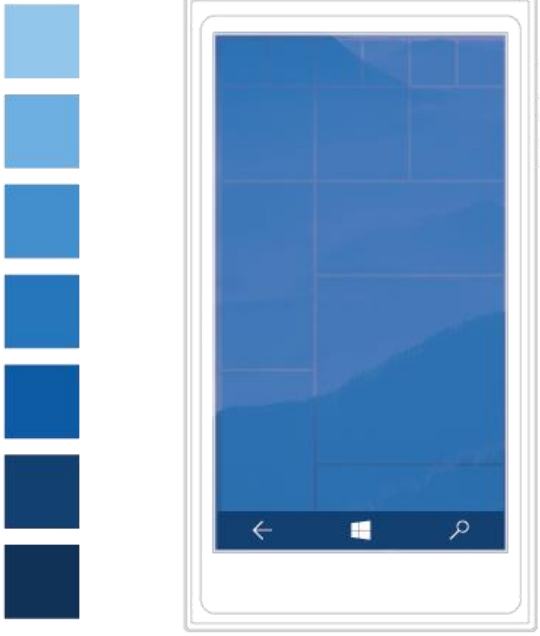


デザインに関するダウンロード

PowerPoint および Adobe Illustrator 用のデザイン テンプレートでプロジェクトをすぐに開始できます。

デザインの基本

これらの記事では、ユニバーサル Windows プラットフォーム (UWP) アプリ (Windows ランタイム API を使って構築される Windows アプリの種類の 1 つ) の設計について紹介します。



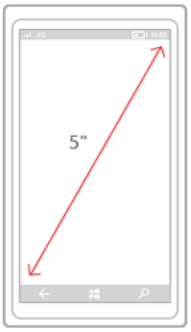
ユニバーサル Windows プラットフォーム (UWP) アプリは、電話からタブレット、PC まで、任意の Windows ベース デバイスで実行できます。ウェアラブルや家電製品などの小型のデバイスで実行する UWP アプリを作成することもできます。UWP アプリをデザインする場合、さまざまなディスプレイ サイズを持つさまざまなデバイスに合ったユーザー インターフェイスを作成する必要があります。簡単にデザインできるように、自動的にすべてのデバイスで適切に動作するユニバーサル コントロールのセットが提供されています。プラットフォームは、デバイス間のテキストとビジュアルをスケールすることで常に見やすくなるように内部処理を行っています。

すべてのデバイスで同じコードとデザインを使うことができ、特定の画面サイズについてユーザー インターフェイスを調整することもできます。たとえば、タブレットや PC で問題なく動作するインターフェイスをデザインして、そのコードの大部分を再利用しつつ、モバイル デバイス用のエクスペリエンスをカスタマイズすることができます。

開始する前に

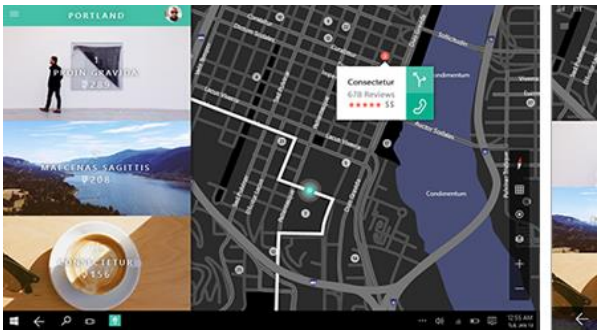
デザイナー向け UWP アプリの概要

この記事では、デザインの観点からユニバーサル Windows プラットフォームの機能と利点について説明します。無料で使用できるプラットフォームの機能とプラットフォームに付属するツールを紹介します



デバイスの基本情報

アプリで実行できるデバイスを紹介します。



アプリの計画

計画プロセスによって時間を節約し、可能な限り最適なアプリを作成できます。

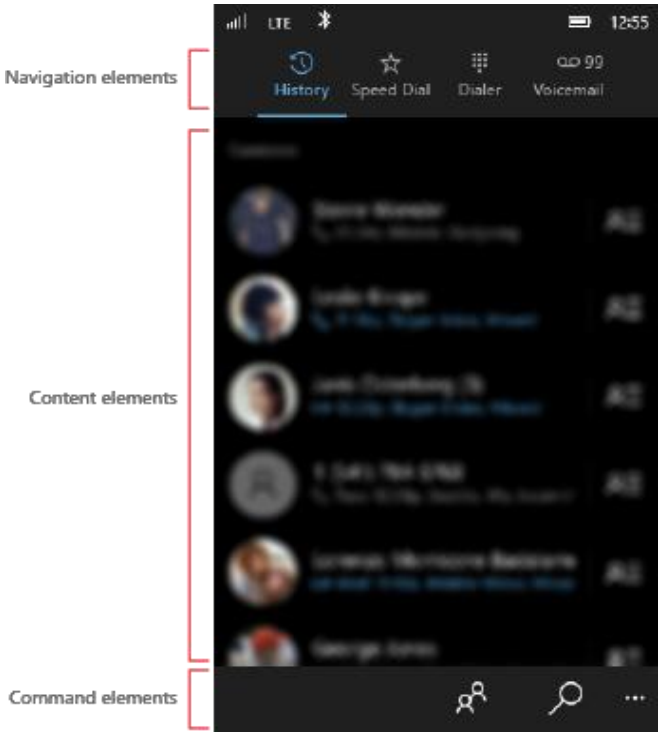
デザインの開始

テンプレートの入手

テンプレートには、UWP アプリのデザインを始めるために必要なものがすべて含まれています。

UI の基本

基本的な UI デザインについて説明し、アプリの UI の出発点となるいくつかの一般的なパターンを確認します。



ナビゲーションの基本

ナビゲーション要素を使用すると、表示するコンテンツをユーザーが見つけやすくなります。3 つの単純なルールに従って、魅力的なナビゲーション エクスペリエンスをデザインする方法について学習します。

コマンドの基本

コマンド要素を使用すると、アプリがインタラクティブになります。コマンド要素と適切なコマンド サーフェスを組み合わせて、優れた操作エクスペリエンスを作成する方法について説明します。

コンテンツの基本

アプリの主な目的は、コンテンツへのアクセスを提供することです。3 つの主なコンテンツ シナリオと、使用する要素が及ぼす影響について説明します

レスポンシブ デザイン 101

さまざまな画面サイズに合わせてアプリを最適化する方法について説明します。

デザイナー向けユニバーサル Windows プラットフォーム (UWP) アプリの概要

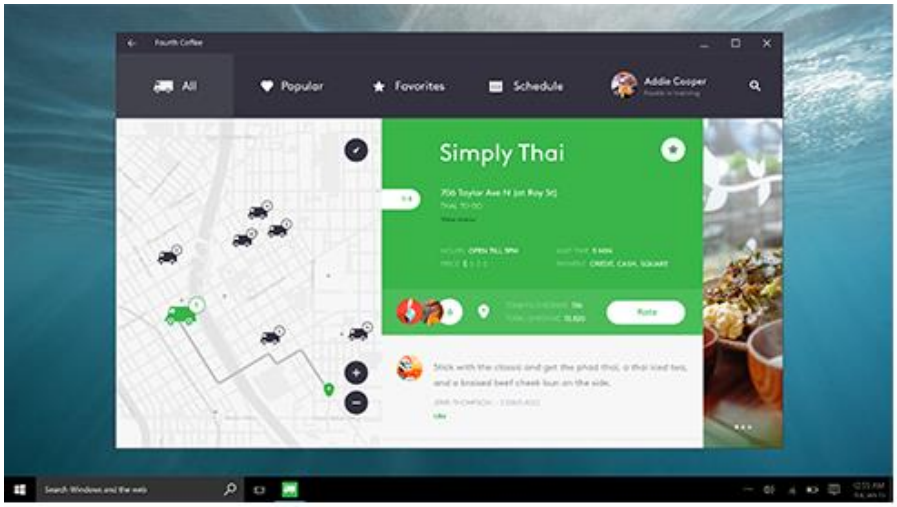
UWP アプリを作成するときは、次のどの Windows デバイスでも実行できるアプリを作成します。

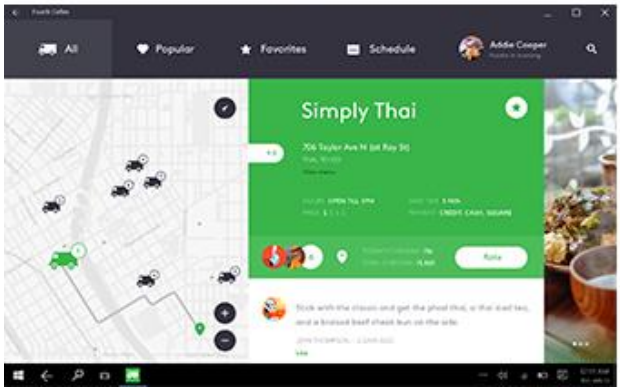
- モバイル デバイス ファミリ: Windows Phone、ファブレット
- デスクトップ デバイス ファミリ: タブレット、ノート PC、PC
- チーム デバイス ファミリ: Surface Hub
- IoT デバイス ファミリ: ウェアラブルや家電製品などの小型のデバイス

アプリを 1 種類のデバイス ファミリ (モバイル デバイス ファミリなど) だけで利用できるように制限できます。また、Windows が実行されているすべてのデバイスでアプリを利用できるようにすることもできます。

すべてのモバイル デバイスでアプリが適切に表示されるようにデザインすることが重要な課題となります。画面サイズや入力方法が大きく異なる複数のデバイスで、優れたユーザー エクスペリエンスを実現するアプリをデザインするためにはどのように取り組めばよいでしょうか。

複数のデバイス ファミリ向けのデザインを行う場合は、追加の考慮事項、計画、設計が必要になりますが、UWP には、一連の組み込み機能とユニバーサルな構成要素が用意されており、これらにより、複数のデバイス向けの優れたユーザー エクスペリエンスを簡単に実現することができます。



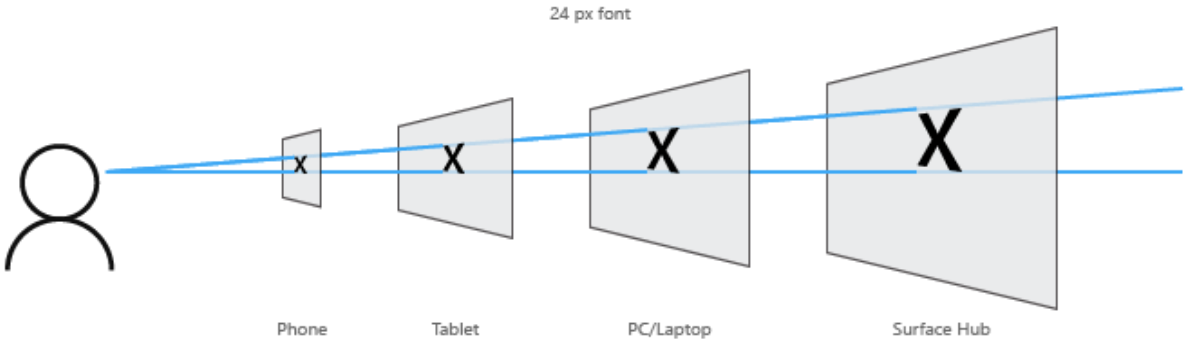


デザイナー向けの組み込み機能

最初に、UWP アプリの作成時に利用できる一部の機能を見てみましょう。何か特別な操作をしなくても、これらの機能によるメリットを得ることができます。これらの機能による効果は自動的に反映されます。

- **効果的なピクセルとプラットフォームのスケーリング**

  Windows デバイスでアプリを実行するとき、システムでは、コントロール、フォント、および他の UI 要素を画面に表示する方法を正規化するアルゴリズムを使います。このスケーリング アルゴリズムでは、視聴距離と画面の密度 (ピクセル/インチ) を考慮して、体感的なサイズを最適化します (物理的なサイズではありません)。スケーリング アルゴリズムによって、10 フィート離れた Surface Hub における 24 ピクセルのフォントが、数インチ離れた 5 インチ サイズの電話における 24 ピクセルのフォントと同じようにユーザーに読みやすい状態で表示されます。



スケーリング システムのしくみのため、UWP アプリをデザインするときには、実際の物理ピクセルではなく、有効ピクセルでデザインすることになります。有効ピクセルを使ったデザインの方法ついて詳しくは、「レスポンシブ デザインの紹介」をご覧ください。

- **ユニバーサル入力とスマート操作**

  特定の入力デバイス向けにデザインすることはできますが、そのようなデザインを行う必要はありません。これは、UWP アプリでは、"スマート" 操作を利用した入力システムを使うためです。つまり、クリックの発生元が実際のマウス クリックであるか、指によるタップであるかどうかを認識しなくても、クリック操作に対応したデザインを行うことができます。

ユニバーサルな構成要素

UWP には、複数のデバイス ファミリに対応したアプリを簡単にデザインできる、便利な構成要素が用意されています。

- **ユニバーサル コントロール**

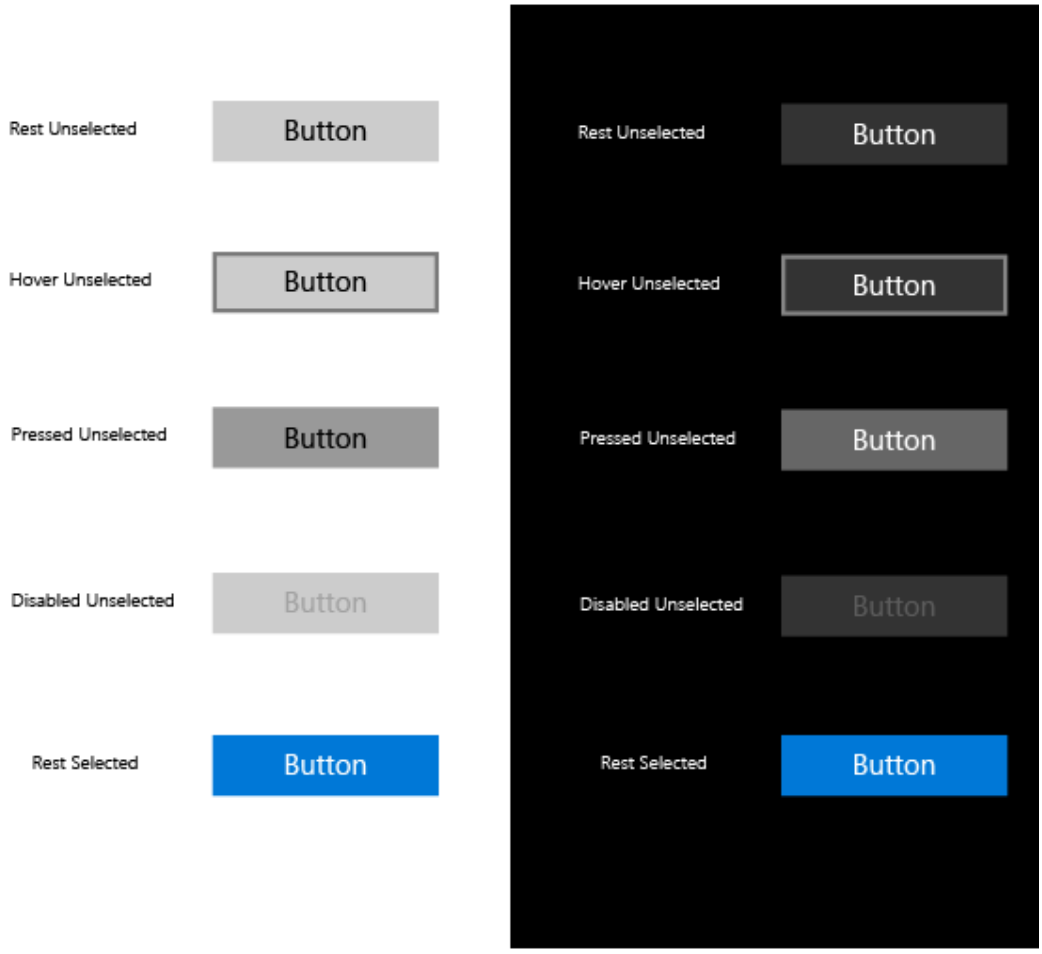
  UWP には、すべての Windows デバイスで適切に動作することが保証されている一連のユニバーサル コントロールが用意されています。この一連のユニバーサル コントロールには、オプション ボタンやテキスト ボックスなどの一般的なフォーム コントロールから、データ ストリームやテンプレートから項目の一覧を生成できるグリッド ビューやリスト ビューなどの高度なコントロールまで、すべてのコントロールが含まれてい

ます。これらのコントロールは入力に対応しており、各デバイス ファミリに適した、一連の入力アフォーダンス、イベント状態、および全体的な機能と共に展開されます。

- **ユニバーサル スタイル**

  UWP アプリは、次の機能を実現できる既定のスタイル セットを自動的に取得します。

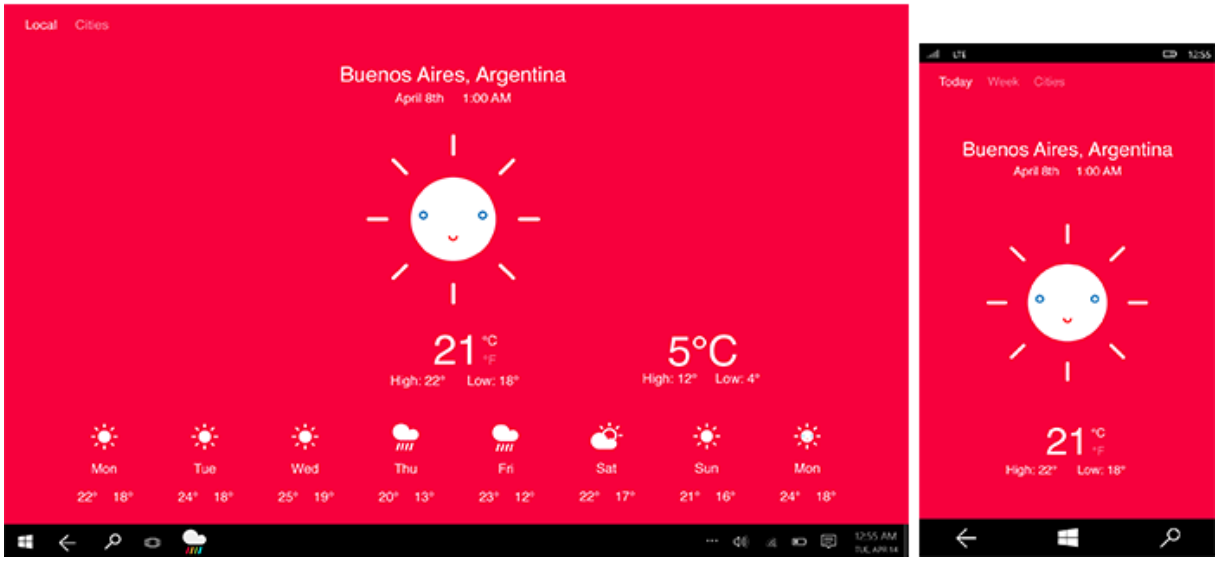
  - 一連のスタイルによって、アプリに淡色や濃色のテーマを自動的に適用したり (ユーザーが選ぶことができます)、ユーザーのアクセント カラーの設定を組み込んだりすることができます。



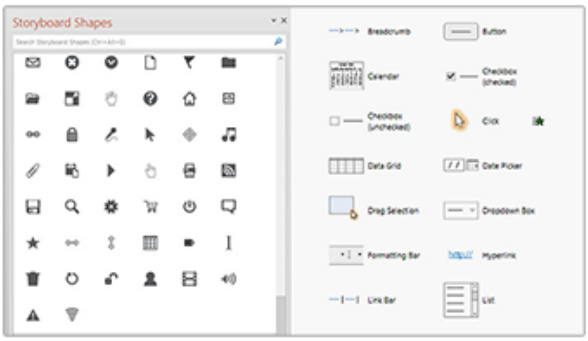
  - すべてのデバイスでアプリのテキストを鮮明に表示する Segoe ベースの書体見本 (type ramp)。
  - インタラクションのための既定のアニメーション。

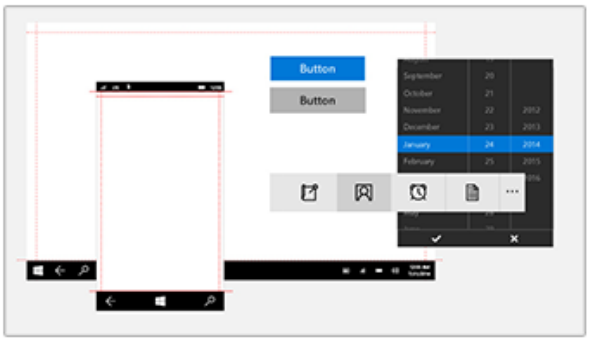
- ハイ コントラスト モードを自動的にサポート。適用するスタイルはハイ コントラストとなるようにデザインされているため、アプリがハイ コントラスト モードのデバイスで実行されているときに正確に表示されます。
- その他の言語を自動的にサポート。既定のスタイルでは、Windows がサポートするすべての言語に対して、自動的に正しいフォントが適用されます。1 つのアプリで複数の言語を使って、正確に表示することもできます。
- RTL の読み取り順序に対する組み込みサポート。

これら既定のスタイルをカスタマイズして、アプリを個性的なものにしたり、既定のスタイルをユーザー固有のスタイルと完全に置き換えて、独自のビジュアル効果を実現したりすることができます。たとえば、独自のビジュアル スタイルを使った天気予報アプリのデザインを次に示します。



- **ユニバーサル テンプレート**





Adobe Illustrator 用テンプレートと Microsoft PowerPoint 用テンプレートが用意されています。これらのテンプレートには、UWP アプリのデザインを始めるために必要なも

のがすべて含まれています。また、これらのテンプレートには、すべてのユニバーサル デバイスの画面サイズに対応したユニバーサル コントロールとユニバーサル レイアウトが含まれています。

- Adobe illustrator 用テンプレートのダウンロード
- PowerPoint 用テンプレートのダウンロード

よく寄せられる質問

- **単一の UI を作成し、その UI をすべてのデバイスで使うことはできますか?**

  はい。単一の UI を作成し、その UI をすべてのデバイスで使うことができます。デバイス ファミリごとにカスタム UI を作成する必要はありません。デザイン ガイドラインを利用することで、すべてのデバイスで適切に動作する単一の UI を作成できます。

  また、アプリが利用できる画面領域を確保するために UI をカスタマイズすることもできます。たとえば、電話サイズのウィンドウでアプリを実行するときに、特定の UI 要素を非表示にすることができます。これにより、多くのスペースをコンテンツに割り当てることができます。特定のサイズに対応するようにアプリをどの程度調整するかは、自由に決めることができます。

- **UWP アプリはすべてのデバイスで実行させる必要がありますか?**

  いいえ、アプリはすべてのデバイスで実行する必要はありません。1 種類のデバイス (携帯電話など) をターゲットとすることはできませんが、アプリを 1 種類のデバイス ファミリ (携帯電話、ファブレット、一部のタブレットが含まれるモバイル デバイス ファミリなど) に制限することはできます。アプリを公開するときに、すべてのデバイス ファミリで利用可能にする、一部のデバイス ファミリで利用可能にする、または 1 つのデバイス ファミリでのみ利用可能することを選ぶことができます。

ユニバーサル Windows プラットフォーム (UWP) アプリ用デバイスの基本情報

UWP アプリをサポートするデバイスを理解することは、各フォーム ファクター向けの最適なユーザー エクスペリエンスを提供するのに役立ちます。特定のデバイス向けのアプリを設計するときは、アプリがデバイスにどのように表示されるか、そのデバイスでアプリがいつどこでどのように使われるか、ユーザーがそのデバイスをどのように操作するかについて、特に考慮する必要があります。

電話とファブレット

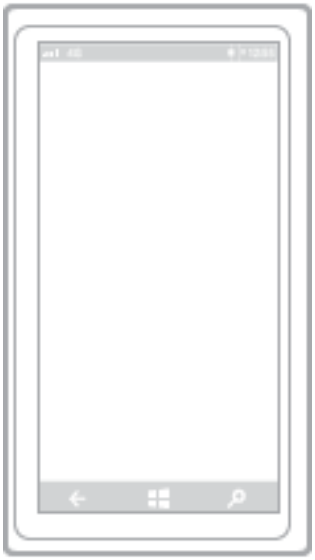
すべてのコンピューティング デバイスの中で最も広く使われている電話では、限られた画面領域と基本的な入力方法を使って、さまざまな操作を行うことができます。 電話にはさまざまなサイズがあり、大きい電話はファブレットと呼ばれます。ファブレットでのアプリのエクスペリエンスは、電話でのエクスペリエンスと似ていますが、画面領域が大きくなることで、コンテンツ操作時に重要な変更が可能になります。

画面サイズ

- 電話の場合、4" ～ 5"
- ファブレットの場合、5.5" ～ 7"

一般的な使い方

- 主に縦方向で使用されます。これはほとんどの場合、片手で電話を持つのが簡単であること、およびその方法で完全に電話を操作できることが理由ですが、写真やビデオの表示、本の閲覧、テキストの作成など、横方向が適切なエクスペリエンスもあります。
- ほとんどの場合、そのデバイスの所有者である 1 人のユーザーによって使われます。
- 常に手近にあり、通常はポケットやバッグに入れられます。
- 短時間使われます。
- ユーザーは、電話を使うときによくマルチタスクを実行します。
- テキストは一気に入力されます。



UI に関する考慮事項

- 電話の画面の小さいサイズでは、横方向でも縦方向でも、一度に 1 つのフレームのみを表示できます。電話のすべての階層型ナビゲーション パターンでは「ドリルダウン」モデルを使い、ユーザーが単一フレームの UI レイヤーを移動するようにします。
- 電話と同じように、縦モードのファブレットには、一度に 1 つのフレームのみを表示できます。ただし、ファブレットで使うことができる画面領域が大きいことから、ユーザーはファブレットを横方向に回転させ

てそのまま保持することで、2 つのアプリ フレームを同時に表示できます。

- 横方向と縦方向の両方で、スクリーン キーボードが表示されているときに、コマンド バーを表示するための十分な画面領域があることを確認します。

入力	• タッチ • 音声
デバイスの標準的な機能	• マイク • カメラ • 移動センサー（複数） • 位置センサー（複数）

タブレット

超軽量のタブレット コンピューターは、タッチスクリーン、カメラ、マイク、および加速度計を備えています。タブレットの画面サイズは、通常は 7 ～ 13.3 インチです。

画面サイズ	• 7" ～ 13.3"
一般的な使い方	• タブレットは、約 80% がその所有者によって使われ、残りの約 20% が共有されています。 • 最も一般的なのは、自宅でテレビを視聴しているときのコンパニオン デバイスとして使うことです。

	• 電話とファブレットよりも長い時間使われます。 • テキストは一気に入力されます。
UI に関する考慮事項	• タブレットは、横方向でも縦方向でも一度に 2 つのフレームを表示できます。 • システムの戻るボタンはナビゲーション バーに配置されます。
入力	• タッチ • スタイラス • 外部キーボード（ときどき） • マウス（ときどき） • 音声 （ときどき）
デバイスの標準的な機能	• マイク • カメラ • 移動センサー（複数） • 位置センサー（複数）

PC とノート PC

Windows PC とノート PC には、多種多様なデバイスと画面サイズがあります。一般に、PC やノート PC は電話やタブレットより多くの情報を表示できます。

画面サイズ	• 13" 以上

一般的な使い方

- デスクトップとノート PC のアプリは共有で使われますが、一度に使うことができるのは 1 人のユーザーだけであり、通常は長時間使われます。

UI に関する考慮事項

- アプリは、ユーザーが決めたウィンドウ表示のサイズにすることができます。ウィンドウのサイズによっては、1 ～ 3 つのフレームを表示できます。大型のモニターでは、アプリは 3 つを超えるフレームを表示できます。
- アプリをデスクトップまたはノート PC で使う場合、ユーザーがアプリのファイルを制御できます。アプリの設計者は、アプリのコンテンツを管理するメカニズムを必ず用意してください。["名前を付けて保存"] や ["最近使ったファイル"] などのコマンドや機能を含めることを検討してください。
- システムの戻るボタンはオプションです。アプリの開発者が表示するように選択した場合、アプリのタイトル バーに表示されます。

入力

- マウス

	• キーボード • タッチ（ノート PC とオール イン デスクトップ） • Xbox コントローラーなどのゲーム パッドが使われることがあります。
デバイスの標準的な機能	• カメラ • マイク

Surface Hub デバイス

Microsoft Surface Hub は、複数のユーザーによる同時使用のために設計された大画面のチーム コラボレーション デバイスです。

画面サイズ	• 55" および 84"
一般的な使い方	• Surface Hub 上のアプリは、会議などで短時間共有して使われます。 • ほとんどの場合、Surface Hub デバイスが固定された状態で使われ、移動することはめったにありません。
UI に関する考慮事項	• Surface Hub 上のアプリは 4 つの状態で表示されます。フィル (利用可能なステージ エリアを使う固定の縦横比 449:236 での表示)、全画面 (標準の縦横比 16:9 での全画面表示)、スナップ (ステージの右端また

	は左端を使う可変の縦横比での表示)、バックグラウンド (アプリの実行中は非表示になり、タスク スイッチャーで利用可能) です。 • スナップ モードまたはフィル モードでは、Skype サイドバーが表示され、アプリは横方向に縮小されます。 • システムの戻るボタンはオプションです。アプリの開発者が表示するように選択した場合、アプリのタイトル バーに表示されます。
入力	• タッチ • ペン • 音声 • キーボード • タッチパッド （リモート）
デバイスの標準的な機能	• カメラ • マイク

Windows IoT デバイス

Windows IoT デバイスは、最新のデバイスであり、主に小型のエレクトロニクス、センサー、接続機能が物理デバイス内に埋め込まれたものです。 通常、これらのデバイスはネッ

トワークまたはインターネット経由で接続され、感知した実際のデータについて報告します。場合によっては、データの処理を実行することもあります。 デバイスには画面がない ("ヘッドレス" デバイスとも呼ばれます) 場合と、一般的に 3.5" 以下の小さい画面に接続される ("ヘッド付き" デバイスとも呼ばれます) 場合があります。

画面サイズ

- 3.5" 以下
- 一部のデバイスには画面がない

一般的な使い方

- 通常、ネットワークまたはインターネット経由で接続されており、感知した実際のデータについて報告します。場合によっては、データに対して処理を実行することもあります。
- これらのデバイスは電話やその他の大型デバイスとは異なり、一度に 1 つのアプリケーションのみ実行できます。
- 常に操作するデバイスではありませんが、必要なときに利用でき、必要でないときに邪魔になりません。
- アプリには専用のバック アフォーダンスはなく、これは開発者の責任です。

UI に関する考慮事項

- "ヘッドレス "デバイスには画面がありません。
- "ヘッド付き" デバイスの画面は最小限で、画面サイズと機能が制限され

	ているため、必要なもののみが表示されます。 • ほとんどの場合、向きはロックされるため、アプリでは 1 つの表示方向を考慮するだけで済みます。
入力	• デバイスに応じて可変
デバイスの標準的な機能	• デバイスに応じて可変

ユニバーサル Windows プラットフォーム (UWP) アプリの計画

Microsoft のデザイン チームのアプリ作成プロセスは、5 つの段階 (コンセプト、構造、インタラクション、ビジュアル、プロトタイプ) で構成されています。同様のプロセスを採用して、だれもが楽しめるエクスペリエンスを実現することをお勧めします。

コンセプト（概念）

アプリに焦点を当てる

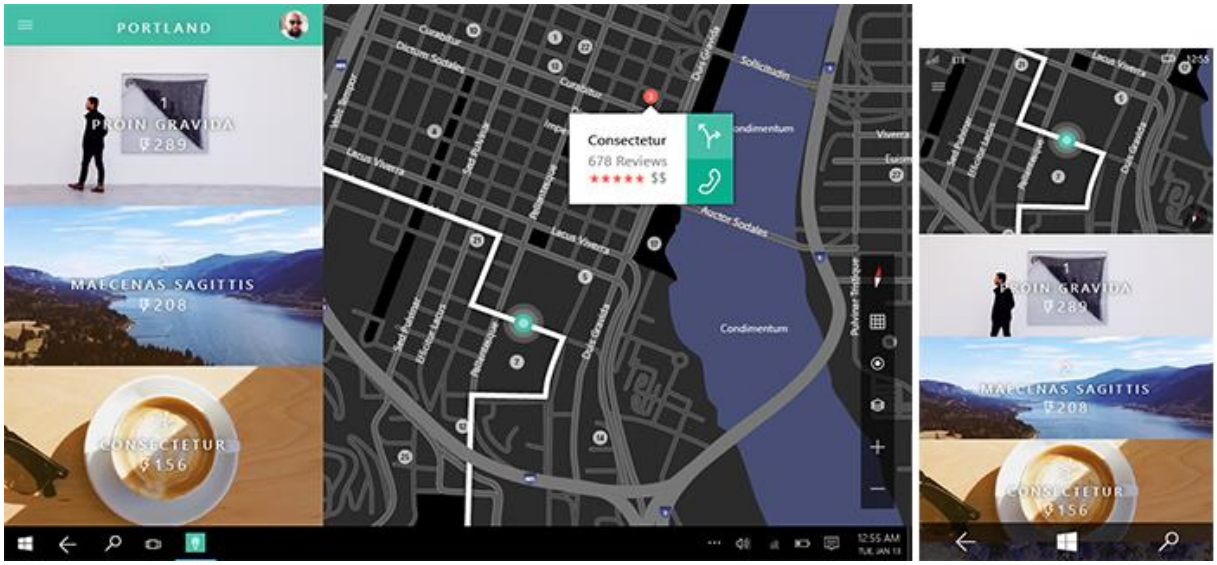
UWP アプリを計画する際は、アプリの内容や対象ユーザーだけでなく、アプリの 1 番の特徴も決定する必要があります。すべての優れたアプリの中心には、しっかりした基盤を提供する強力なコンセプトがあります。

たとえば、写真アプリを作るとします。まず、ユーザーが写真を操作、保存、共有する動機を考えてみます。それは、写真で思い出をたどったり、写真を通じて他の人々と交流したり、写真を安全に保管したりすることであるとわかります。このような用途に適した写真アプリを作ることにします。このユーザー エクスペリエンスの目標を意識しながら、アプリの設計プロセスを進めていきましょう。

何のためのアプリか まず、大まかなコンセプトを決めます。ユーザーがアプリで何をできるようにするか、一覧にまとめてみましょう。

たとえば、旅行の計画に使うアプリを作るとします。次のように、頭に浮かんだアイデアをメモしてみましょう。

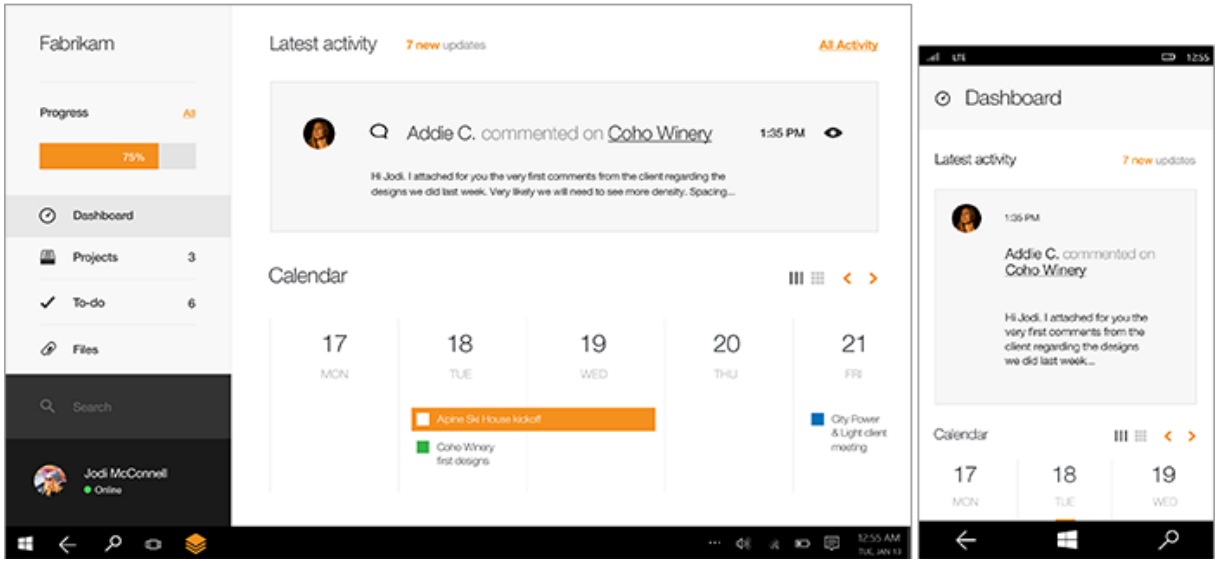
- 旅行計画に含まれている場所の地図をすべて集めて、旅行中もそれを携帯する。
- 旅行先で滞在中に開催されるイベントを見つける。
- 必ずするアクティビティや必ず見る観光名所の一覧を、旅行に行くメンバーが各自で作成し、他のメンバーも見られるようにする。
- 旅行のメンバーが撮ったすべての写真をまとめて、友人や家族と共有する。
- 航空料金に基づいて、お勧めの行き先を選ぶ。
- 目的地周辺のレストラン、店舗、アクティビティが多数掲載されている情報源を見つける。



アプリの 1 番の特徴は何か すべてのアイデアを全体的に見て、特に目立つシナリオがないか考えてみます。数多くのアイデアの中から絞り込んで、これに集中しようと思えるシナリオを 1 つだけ選びます。優れたアイデアをたくさん捨てることになりますが、その 1 つのシナリオを良いものにしあげるには、いさぎよくアイデアを捨てることが肝心です。

シナリオを 1 つ選んだら、アプリの 1 番の特徴を普通の人に 1 行で伝えるにはどうしたらよいかを考えます。 例:

- この旅行アプリを使えば、友人どうしで協力してグループ旅行の計画を作り上げることができます。
- このワークアウト アプリを使うと、友人どうしでトレーニングの経過を記録して、互いに成果を見せることができます。
- この食料雑貨アプリを使うと、家族で毎週の食料雑貨を計画的に買うことができ、買い忘れや重複がなくなります。



このようにアプリの 1 番の特徴を示す説明文を作っておくと、アプリを作るプロセスで、設計上のさまざまな決定事項や妥協点を判断する際に役立ちます。アプリの 1 番の特徴にするユーザー シナリオを中心とした説明文にしますが、単なる機能一覧にならないように注意してください。この説明文では、アプリが実行できる機能を説明するのではなく、ユーザーがこのアプリを使ってできることを説明します。

ファネルのデザイン

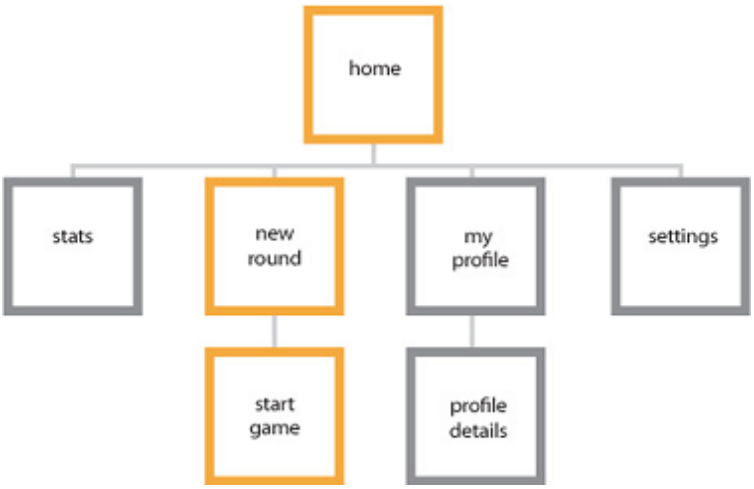
運用環境に取り入れるまでの道のりは長いと思いますが、自由にアイデアを考えて、開発することはとても魅力的です。ただし、仮にそれを実現したとして、新たに別の興味深いアイデアが思い付きます。当然のことながら、2 つのアイデアの優劣にかかわらず、既に投資してきたアイデアにこだわろうとするでしょう。その別のアイデアをプロセスのもっと早い段階で思い付いていればよかったのです。このような場合に用いるファネルのデザインは、できるだけ早く最善のアイデアを発見するのに役立つ手法のことです。

"ファネル"（じょうご、漏斗） という用語は、その形に由来しています。じょうごの幅広の部分では、多くのアイデアが入ると、各アイデアが忠実度の非常に低いデザインの成果物 (スケッチ、テキストの段落) であるとわかります。このように集められたアイデアがじょうごの狭い方に向かって移動するにつれ、アイデアを表す成果物の忠実度が高まると同時にアイデアの数は削減されます。各成果物では、アイデアどうしを比較して判定するため、または "これが使って便利なのか (つまり直感的に使えるのか)" などの特定の質問に答えるために必要な情報のみを抽出する必要があります。各成果物にそれ以上の時間と労力を使わない

でください。アイデアによっては、テスト時に途中で挫折することもありますが、そのアイデアの判定に必要以上に投資しないようにするため、それでかまいません。じょうごの中でさらに残り続けるアイデアは、忠実度の高いものとして扱われます。最終的には、魅力的なアイデアを表す 1 つの成果物が残ることになります。これは、単に最初に思い付いたからではなく、その利点によって残ったアイデアといえます。これで、最適なアプリを設計できるでようになるでしょう。

構造

組織化によってすべてを簡単にする



コンセプトに満足したら、次の段階としてアプリの青写真を作る準備に入ります。情報アーキテクチャ (IA) により、コンテンツには、必要な構造的整合性が提供されます。これは、アプリのナビゲーション モデルを定義し、最終的にアプリの ID を定義するのに役立ちます。コンテンツをどのように整理するか、また、どうすればユーザーがそのコンテンツを見つけられるのかを計画することにより、ユーザーがアプリを使ったときのエクスペリエンスについて理解を深めることができます。

優れた IA は、ユーザーのシナリオを容易にするだけでなく、最初に主な画面を構想するのに役立ちます。たとえば、Audible アプリは、ユーザーのライブラリ、ストア、ニュース、統計情報へのアクセスを提供するハブを直接開始します。エクスペリエンスが重視されているため、ユーザーはオーディオ ブックをすばやく取得して楽しむことができます。アプリのより深いレベルでは、より具体的なタスクに焦点を当てています。

ダイナミクス

アプリのストーリーを伝える

コンセプトの段階でアプリの目的を定義した場合、ダイナミクスの段階ではその目的を遂行するだけです。これを実現するには多くの方法があります。たとえば、ワイヤーフレームを使ってページ フロー (アプリ内のある場所から別の場所に移動する方法や、目的を果たすためにアプリで行う一連の操作) をスケッチしたり、必要とするアプリの音声や言葉について考えたりします。ワイヤーフレームは簡単ですが忠実度の低いツールで、アプリのユーザー フローについて重要な判断を行うのに役立ちます。

アプリ フローは、"1 番の特徴" の説明文と関連していることが必要です。アプリ用に選んだ 1 つのシナリオを実現できるように、フローを作ります。優れたアプリは、覚えやすくて操作が少ないフローを備えています。まずは画面間レベルで考えてみましょう。アプリを初めて使うかのように眺めてください。作るページのユーザー シナリオを正確に示す場合、不要な画面タッチが多くならないように、必要なものだけを提供します。ダイナミクスはアニメーションにも関連しています。適切なアニメーション機能によって、ページ間の滑らかな動作と使いやすさが決まります。

この手順に役立つ一般的な方法:

- **フローの概要**　最初の操作とその次の操作を決める
- **フローのストーリーボード**　フロー完了までの UI 操作の順序を決める
- **プロトタイプ**　簡単なプロトタイプを使ってフローをテストする

ユーザーは何をできるか たとえば、旅行アプリを使うと "友人どうしで協力してグループ旅行の計画を作り上げる" ことができます。必要なフローをリストにまとめましょう。

- 一般的な情報で旅行計画を作る。
- 友人たちを旅行に誘う。
- 友人の旅行に参加する。
- 他の旅行者が勧める旅行計画を見る。
- 目的地とアクティビティを旅行に追加する。
- 友人たちが追加した目的地とアクティビティを編集したり、コメントを書いたりする。
- 友人や家族に見てもらえるように旅行計画を共有する。

ビジュアル

言葉を使わずに伝える

アプリのダイナミクスを確立したら、アプリのビジュアルを洗練してアプリを魅力的なものにすることができます。優れたビジュアル効果として、アプリの外観だけでなく、アニメーションやモーションを使ってライブ感を出す方法を定義します。選んだカラー パレット、アイコン、アートワークは、この目的に合わせたビジュアルの一例です。

どのアプリにも独自の一意な ID があるため、アプリで利用できる視覚的な向きを確かめます。コンテンツによって外観を誘導でるのであって、外観によってコンテンツが決まるのではないことに注意してください。

プロトタイプ

作品を洗練させる

プロトタイプは、ファネルのデザイン (既に説明した手法) における 1 つの段階です。ここでは、アイデアを表す成果物が、スケッチ以上だが、完成したアプリほど複雑ではないレベルに進化します。プロトタイプは、ユーザーに表示される画面を手書きで表したフローである場合があります。テストの実行者は、実行中のアプリをシミュレートするために、さまざまな画面を下に配置したり、ページ上のいくつかの UI をくっつけたり離したりすることで、ユーザーフローに応答する場合があります。また、操作者がスクリプトを最後まで実行して正しいボタンを押す場合、プロトタイプは複数のワークフローをシミュレートする非常

にシンプルなアプリになります。この段階で、アイデアは実際に実現性を帯び始め、これまで費やしてきた作業が本格的にテストされます。動くアプリのプロトタイプを作る場合、時間をかけて、最も必要なコンポーネントを調整して絞り込んでください。

優れたアプリの作成が反復プロセスであるということは、経験の少ない開発者にとっていくら強調してもしすぎることはありません。初期段階から何度もプロトタイプを作ることをお勧めします。創造力を養う努力と同様、最適なアプリは徹底的な試行錯誤による成果です。

1. アプリに含める機能を決める

ユーザーの目的を理解し、その目的を助ける方法もわかったら、次にすることは、それを実現するための機能を探すことです。ユニバーサル Windows プラットフォーム (UWP) を調べて、アプリのニーズに対応する機能を探します。各機能については、「ユニバーサル Windows プラットフォーム (UWP) アプリのガイドライン」に従ってください。

一般的な方法:

- **プラットフォームの調査**　プラットフォームにある機能を確かめて、どのように使えるかを考える。
- **関連付けのダイアグラム**　フローと機能を結び付ける。
- **プロトタイプ**　機能をテストして、必要な働きができるかを確かめる。

アプリ コントラクト 他のアプリとのユーザー フローや他の機能とのユーザー フローを実現するアプリ コントラクトに、アプリを参加させることができます。

- **共有** ユーザーがアプリのコンテンツを他のアプリ経由で他のユーザーと共有できるだけでなく、他のユーザーやアプリの共有コンテンツを受け取ることもできるようにします。
- **リモート再生** オーディオ、ビデオ、または画像をアプリからホーム ネットワーク上の別のデバイスにストリーミングして楽しむことができます。
- **ファイル ピッカーとファイル ピッカーの拡張機能** ユーザーがローカル ファイル システム、接続された記憶装置、ホームグループ、さらに他のアプリからファイルをアプリに読み込み、保存できるようにします。また、ファイル ピッカーの拡張機能を使って、アプリのコンテンツを他のアプリに読み込むこともできます。

詳しくは、アプリ コントラクトと拡張機能についてのページをご覧ください。

さまざまなビュー、フォーム ファクター、ハードウェア構成 Windows では、ユーザーがさまざまな持ち方や向きでデバイスを構えるので、アプリもそれに対応する必要があります。アプリの UI は、ユーザーが使うあらゆるデバイス、入力モード、向き、ハードウェア構成、状況で適切に表示される必要があります。

タッチ優先 Windows では、独自の直感的なタッチ操作を使うことができます。これはマウスと同じように使えるだけではありません。

たとえば、セマンティック ズームは、大きなコンテンツ セットの中を移動するときにタッチ操作のメリットが活かした操作方法です。パンまたはスクロールでコンテンツのカテゴリを移動し、カテゴリを拡大して詳しい情報を見ることができます。タブなどの従来のナビゲーション手法やレイアウト パターンよりも、コンテンツを感覚的、ビジュアルに表現でき、多くの情報を示すことができます。

もちろん、回転、パン、スワイプ、クロススライドなどタッチ操作のさまざまな長所を利用できます。

魅力と新鮮さ アプリが新鮮に感じられ、次のような標準的なエクスペリエンスでユーザーを魅了できることが重要です。

- **アニメーション** アニメーションのライブラリを使って、アプリに軽快で柔軟な印象を加えることができます。コンテキストの変化がわかりやすく、ビジュアルの切り替えがエクスペリエンスに結び付きます。
- **トースト通知** トースト通知によって、即時性が必要なコンテンツや個人的に関係のあるコンテンツをユーザーに知らせ、アプリが非表示の場合は、アプリを表示するようにユーザーにうながします。
- **アプリ タイル** ユーザーと関連性の深い最新の更新情報を知らせて、ユーザーがアプリを開くように誘います。

個人設定

- **設定** アプリ設定を保存することによって、ユーザーは好みのエクスペリエンスを作ることができます。すべての設定を 1 つの場所にまとめて、ユーザーが使い慣れた一般的なメカニズムでアプリを構成できるようにします。
- **ローミング** データのローミングによってデバイス間で連続性のあるエクスペリエンスを実現し、ユーザーが作業を中断したところから再開できるようにすると共に、使われているデバイスに関係なく、ユーザーが最も重視する UX を保持できるようにします。設定と状態をローミングによって維持することで、キッチンで家族が使う PC から仕事用 PC、個人用のタブレット、その他のフォーム ファクターなど、どこにいてもアプリを快適に使えるようにします。
- **ユーザー タイル** ユーザーがアプリを個人設定で使えるようにします。ユーザーの画像を読み込んだり、アプリからのコンテンツを Windows で個人用タイルとして設定できるようにします。

デバイスの機能 最新のデバイスの機能をアプリで十分に活用してください。

- **近接ジェスチャー** ユーザーは、複数のデバイスを物理的に "タップ" することで接続できます (マルチプレイヤー ゲーム)。
- **カメラと外部記憶装置** 内蔵カメラまたは接続されたカメラを利用します。チャットや電話会議、ビデオ ログの記録、プロファイル写真の撮影、日常的な文書作成など、アプリの特徴に応じたさまざまな活動に使うことができます。
- **加速度計やその他のセンサー** 最新のデバイスはさまざまなセンサーを備えています。アプリでは、環境光に応じてディスプレイの明るさを調節したり、ユーザーがディスプレイの向きを変えたときに UI を自動的に再配置したり、物理的な動きに応じて処理を行ったりできます。
- **地理位置情報** 標準的な Web データまたは地理位置情報センサーからの地理位置情報を使って、ユーザーの移動や地図上での位置確認、または近くのユーザー、アクティビティ、目的地に関するユーザー通知を支援します。

旅行アプリの例についてもう一度考えてみましょう。友人どうしで協力してグループ旅行の計画を作り上げるアプリには、次のような機能があるとよいかもしれません。

- **共有:** 次の旅行の計画を複数のソーシャル ネットワークで公開して、旅行前の楽しみを家族や友人と分かち合います。

- **検索**　他の共有された旅行計画や公開された旅行計画を検索して、アクティビティや目的地を探し、自分たちの旅行に取り入れることができます。
- **通知**　旅行の仲間が旅行計画を更新したときに、通知を受け取ります。
- **設定**　通知する旅行、旅行計画の検索用を許可するソーシャル グループなどの設定をユーザーが構成できます。
- **セマンティック ズーム**　ユーザーは旅行計画のタイムラインを閲覧しながら、数多くの計画済みアクティビティを必要に応じて拡大表示し、詳しい内容を見ることができます。
- **ユーザー タイル**　友人と旅行を共有するときに表示する画像を選択できます。

このように、Windows を使って、ユーザーを喜ばせる魅力的なエクスペリエンスを作ることができます。

2.　アプリで収益を得る方法を決定する

さまざまな方法でアプリから収益を得られます。アプリ内の広告や販売を使う場合は、それに対応するように UI を設計します。

3.　アプリの UX を設計する

ここでは、適切な基本構造を作ります。アプリの 1 番の特徴が決まり、サポートするフローも決まったので、次はユーザー エクスペリエンス (UX) 設計の基本構造を検討します。

UI コンテンツをどのようにまとめるか　ほとんどのアプリ コンテンツは、特定の形式のグループまたは階層にまとめることができます。コンテンツの最上位グループの内容は、1 番の特徴の説明文に沿ったものにします。

旅行アプリを例に考えてみましょう。旅行計画をグループ化する方法はいくつかあります。アプリの主な目的が興味深い目的地を見つけることであれば、冒険旅行、リゾート旅行、ロマンチックな休暇旅行など、興味の対象ごとにグループ化できます。しかし、このアプリの主な目的は友人と協力して旅行計画を立てることなので、家族、友人、同僚など、ソーシャル グループごとに旅行計画を整理する方が理にかなっています。

コンテンツのグループ化の方法が決まると、アプリにどのようなページやビューが必要なのかがわかります。Microsoft Visual Studio に備えられているプロジェクト テンプレートでは一般的なアプリ レイアウト パターンが用意されており、ほとんどのコンテンツに対応できます。

UI コンテンツをどのように表示するか　UI をまとめる方法が決まったら、UI がどのように構築され、ユーザーに表示されるかを指定された UX の目標を定義できます。どのシナリオでも、ユーザーがアプリを楽しんで使い続けることができるよう、できるだけ早くユーザーの使いたくなるという気持ちを高める必要があります。これを行うには、最初にユーザーに表示する UI の部分を決めて、その部分を完成させてから、その他の重要でない部分の構築に時間をかける必要があります。

旅行アプリでは、ユーザーはまず具体的な旅行計画を探します。この情報をできるだけ早くユーザーに表示するには、まず ListView コントロールを活用して、旅行の一覧を表示します。

旅行の一覧が表示されると、友人の旅行のニュース フィードなどの機能の読み込みが始まります。

必要な UI サーフェスとコマンドは何か　前の手順で決めたフローをもう一度見てみます。各フローについて、ユーザーが行う手順を大まかに作成します。

たとえば、"友人や家族が旅行計画を確認できるように共有する" フローで考えてみましょう。ユーザーが既に旅行計画を作成しているとします。旅行計画を共有するには、次の手順が必要です。

1. アプリを開き、作成した旅行の一覧を表示します。
2. 共有する旅行をタップします。
3. 旅行の詳しい内容が画面に表示されます。
4. 共有を開始するための UI を操作します。
5. 旅行を共有したい友人のメール アドレスまたは名前を選ぶか、入力します。
6. 共有の操作を完了するための UI を操作します。
7. アプリの旅行の情報が更新され、旅行計画を共有するメンバーの一覧が反映されます。

このプロセスによって、作成する必要のある UI と、さらに作り込みが必要な部分 (アプリをまだ使っていない友人に送るメールの定型文を考えるなど) を把握することができます。さらに、不要な手順を削る作業も始めます。たとえば、旅行の詳しい内容を共有前に実際に見る必要はないかもしれません。フローが整理されると使いやすくなります。

さまざまなサーフェスを使う方法について詳しくは、Windows ユニバーサル アプリのコマンド設計をご覧ください。

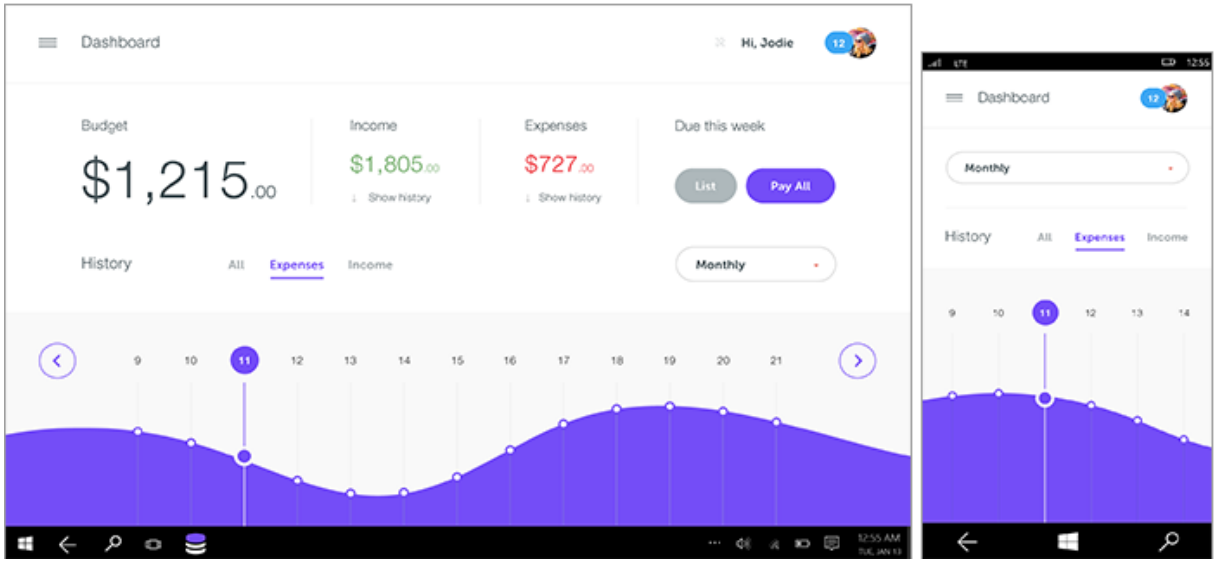
フローをどのようにするか　ユーザーの手順を定義したら、そのフローをパフォーマンスの目標にします。詳しくは、「パフォーマンスの計画」をご覧ください。

コマンドをどのように配置するか 大まかなフロー手順に従って、設計する必要のあるコマンドを決めます。次に、アプリのどこにコマンドを配置するかを考えます。

- **できるだけコンテンツを直接操作できるようにします。** できる限り、コンテンツを操作するコマンドを用意せず、アプリのキャンバス上でユーザーがコンテンツを直接操作できるようにします。たとえば、旅行アプリで旅行計画を編集するときに、リスト内のアクティビティを上下に移動するコマンド ボタンを使うのではなく、キャンバスのリスト上でアクティビティをドラッグ アンド ドロップできるようにします。
- **コンテンツを直接操作できない場合、**コマンドを次の UI サーフェスのいずれかに配置します。

- **コマンド バー**　コマンドはできる限りコマンド バーに配置してください。通常、コマンド バーは非表示になっており、ユーザーの操作で表示されます。
- **アプリのキャンバス上**　目的が 1 つに限られているページまたはビューは、その目的用のコマンドをキャンバス上に直接配置できます。しかし、このようなコマンドはなるべく作らないでください。
- **コンテキスト メニュー**　クリップボードの操作 (切り取り、コピー、貼り付けなど) や、選択できないコンテンツに実行するコマンド (地図の目的の位置にピンを追加するなど) に使うことができます。

各ビューでアプリを準備する方法を決定します。 Windows では横向きと縦向きがサポートされ、全画面表示から最小幅にいたるまで、アプリの幅をあらゆるサイズに変更できます。アプリは、任意のサイズの任意の画面で、どの向きでも適切に表示され、機能することが望まれます。つまり、さまざまなサイズとビューについて、UI 要素のレイアウトを計画する必要があります。そうすることで、ユーザーのニーズと好みに合わせてアプリ UI が柔軟に変化します。



さまざまな画面サイズ用のデザインについては、「ユニバーサル Windows プラットフォーム (UWP) アプリ用レスポンシブ デザイン」をご覧ください。

4.　第一印象を良くする

初めてアプリを起動したユーザーがどのように思ったり感じたりするかを考えましょう。アプリの "1 番の特徴" の説明文を再確認します。アプリの 1 番の特徴をユーザーに直接説明

できなくても、第一印象を演出することでメッセージを伝えることができます。次の機能を利用します。

タイルと通知　タイルはアプリの顔です。ユーザーのスタート画面に表示される数多くのアプリの中から自分のアプリが起動される決め手になるのは、何でしょうか。タイルでアプリのブランドを明確にして、アプリの特徴を示すようにします。アプリで常に最新の役立つ情報が伝えられるようにタイル通知を使うことで、ユーザーがアプリを繰り返し利用するようにします。

スプラッシュ スクリーン　スプラッシュ スクリーンはできるだけ短時間で読み込む必要があります。この画面は、アプリの状態を初期化する必要が生じない限り画面上に維持する必要があります。スプラッシュ スクリーンの表示内容で、アプリの特徴を伝えます。

初めての起動　ユーザーがサービスの新規登録、アカウントへのログイン、独自のコンテンツの追加を行う前に、ユーザーに対して何を表示しますか。ユーザーに情報を求める前に、アプリの価値をアピールします。アプリを使ってもらうためには、ユーザーが自由に試せるサンプル コンテンツを表示して、アプリの目的を知ってもらうのもよいでしょう。

ホーム ページ　ホーム ページとは、ユーザーがアプリを起動するたびに表示されるページです。ここに表示されるコンテンツは、目的をはっきりさせ、アプリの特徴がすぐにわかるようにする必要があります。 このページでは、1 番の特徴だけを強調します。アプリの他の部分はユーザーが探し出してくれることを期待しましょう。 ランディング ページでは、ユーザーがアプリの特徴に気付きやすくするのではなく、散漫な印象を与えないようにしてください。

5.　設計を検証する

設計のやり直しを避けるために、実際の開発があまり進まないうちに、設計内容またはプロトタイプをガイドライン、ユーザーの印象、要件に照らして検証します。それぞれの機能について、アプリの改善に役立つユーザー エクスペリエンス ガイドラインと、Windows ストアでアプリを販売するために必要なストア要件があります。 Windows アプリ認定キットを使って、Windows ストアの技術要件を満たすかどうかのテストを実行できます。また、Visual Studio のパフォーマンス ツールを使って、すべてのシナリオでユーザーに快適なエクスペリエンスを提供しているかどうかを確認できます。

ユニバーサル Windows プラットフォーム (UWP) アプリ用レスポンシブ デザイン 101

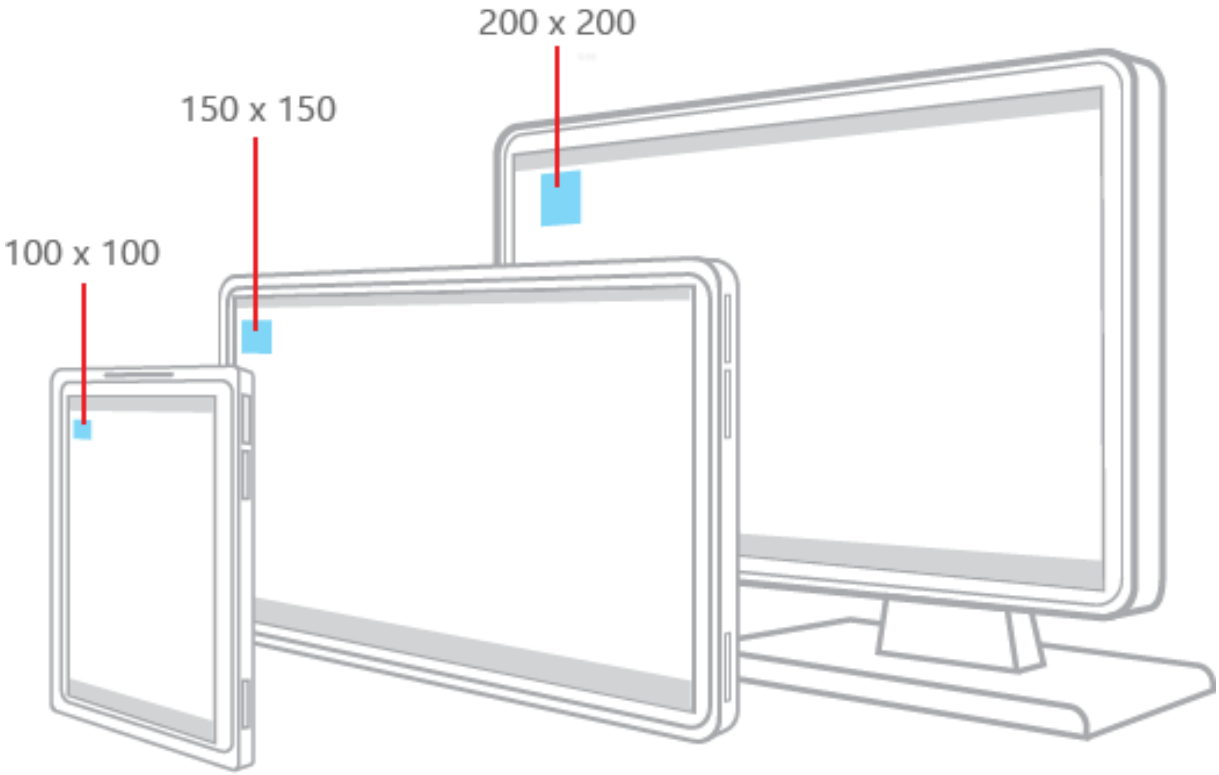
「UWP アプリの紹介」では、さまざまな画面サイズの複数の Windows デバイスで UWP アプリを実行できるようにする方法について説明しました。特定のデバイスや画面サイズに合わせてアプリをカスタマイズする必要はありません。Windows がバックグラウンドでの作業を実行し、すべてのデバイスでユーザー インターフェイスが確実に読みやすくなり、機能するようにします。ただし、特定のデバイスに合わせてアプリを調整することが必要な場合もあります。たとえば、PC またはノート PC でアプリを実行するときに、追加のコンテンツを表示すると、これが、携帯電話のような小型デバイスの画面では煩雑になる場合があります。

特定の画面サイズに合わせてアプリの機能を強化する方法は数多くあります。その中には、迅速で単純な方法もあれば、ある程度の作業が必要なものもあります。

有効なピクセル単位でのデザイン

「デザイナー向け UWP アプリの紹介」では、UWP アプリを設計する方法について説明しました。この設計では、実際の物理ピクセルではなく、有効なピクセル (effective pixels)で設計を行います。

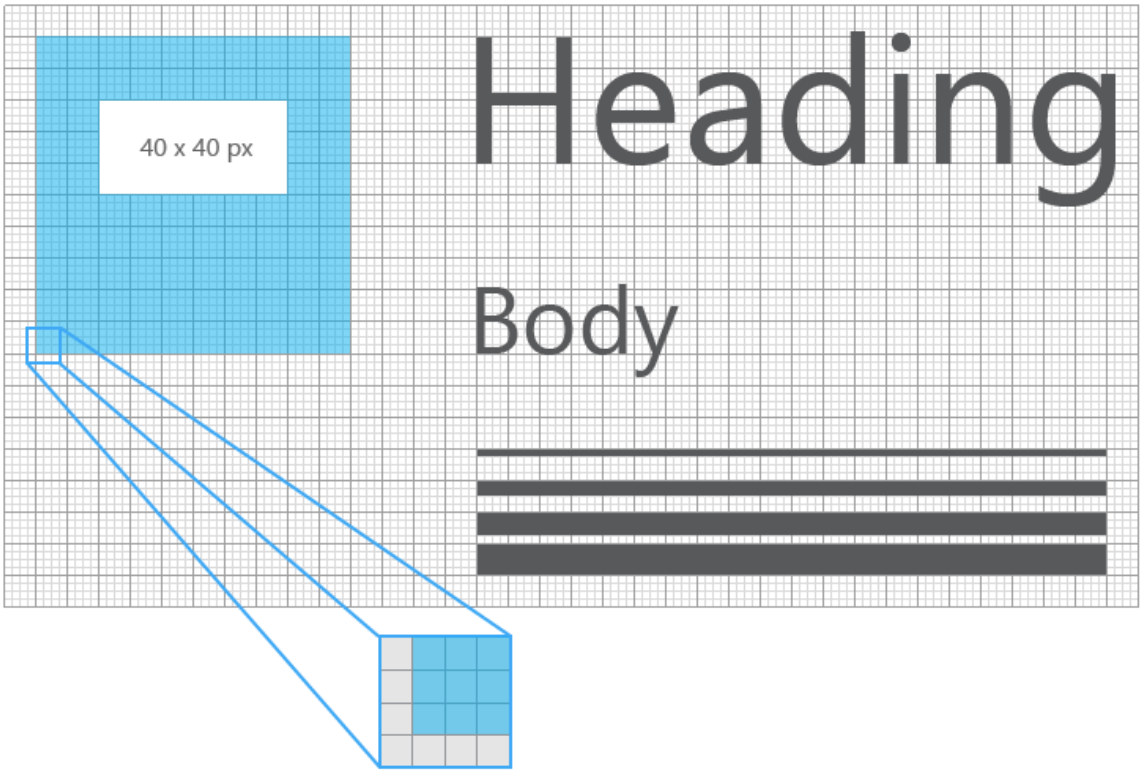
有効なピクセルを使用すると、UI 要素の実際の体感的なサイズに集中することができ、ピクセルの密度や、さまざまなデバイスの視聴距離を気にする必要がなくなります。たとえば、1x1 インチの要素を設計する場合、その要素は、すべてのデバイスで約 1 インチで表示されます。非常に大型で高ピクセル密度の画面では、この要素が 200x200 物理ピクセルになる場合があります。一方、携帯電話などの小型デバイスでは、100x100 物理ピクセルになる可能性があります。



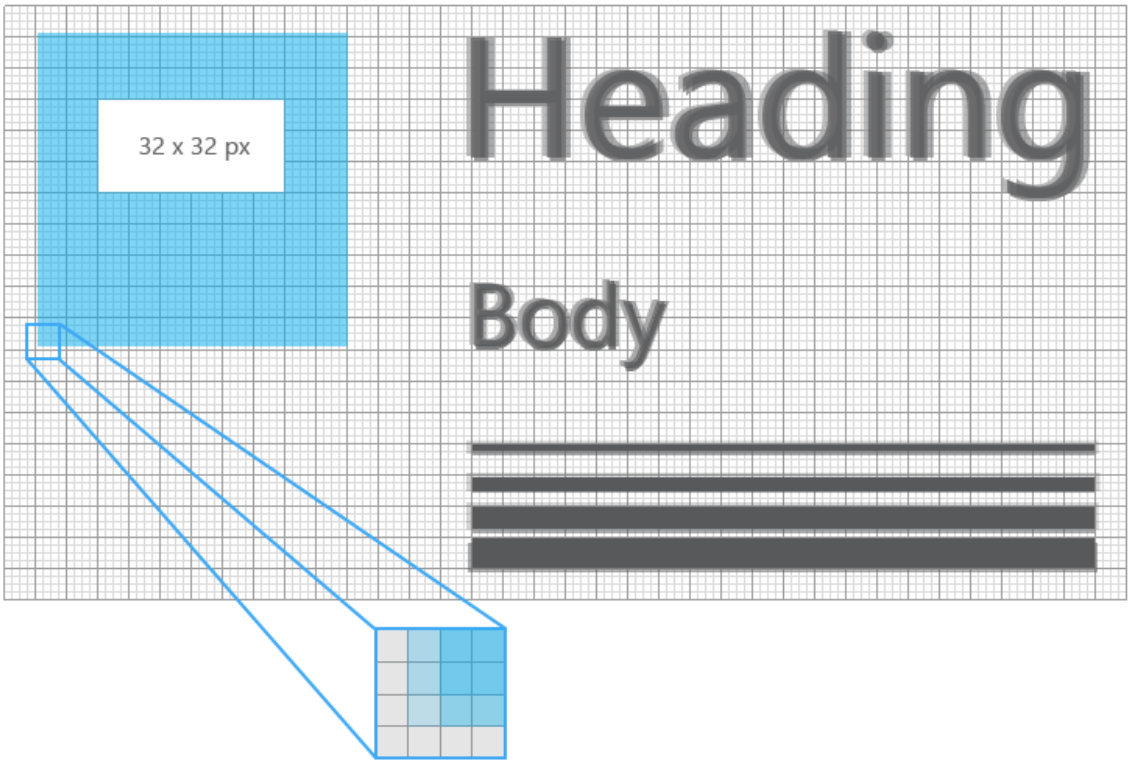
それでは、このことが、アプリの設計方法にどのような影響を及ぼすでしょうか。

- 設計時には、ピクセル密度と実際の画面解像度を無視できます。その代わり、サイズの種類による有効な解像度 (有効ピクセル単位の解像度) が向上するように設計します (サイズの種類は、この記事の後半で定義します)。
- システムによる UI のスケーリングは、4 の倍数単位で行われます。外観が鮮明になるように、4x4 のピクセル グリッドにデザインをスナップします。余白、UI 要素のサイズと位置、およびテキストの位置 (サイズは除きます。テキストのサイズに制限はありません) を、4x 有効ピクセルの倍数にします。

  次の図は、4x4 のピクセル グリッドにマップされるデザイン要素を示しています。デザイン要素には常に、はっきりした鮮明な縁があります。



次の図は、4x4 のピクセル グリッドにマップされないデザイン要素を示しています。これらの設計要素の縁は、デバイスによっては、ぼやけて不鮮明に表示されます。



ヒント イメージ編集プログラムで画面のモックアップを作成するときは、DPI を 72 に設定し、画像サイズを、対象の画面サイズで有効な解像度に設定します(サイズ クラスと有効な解像度の一覧については、後述の「特定の画面サイズの推奨事項」をご覧ください)。

特定のデバイス ファミリと画面サイズに合わせてアプリを調整する理由

前のセクションでは、UWP アプリが有効ピクセルを使って、デザイン要素を見やすくし、すべての Windows デバイスで使用できるようにする方法について説明しました。では、特定のデバイス ファミリ向けにアプリの UI をカスタマイズする理由は何でしょうか。

注 先に進む前に知っておくべきですが、アプリが実行されている特定のデバイスをアプリが検出する手段を、Windows は提供しません。アプリが実行されているデバイス ファミリー (モバイル、デスクトップなど)、有効な解像度、およびアプリが利用できる画面領域の大きさ(アプリのウィンドウのサイズ) を伝えることができます。

- **最も効果的に領域を使用し、移動する必要を減らすには**
  携帯電話など、画面の小さいデバイスで適切に表示するアプリを設計した場合、そのアプリは、はるかに大きいディスプレイを備えた PC でも使用可能ですが、おそらく、無駄な領域があります。画面が特定のサイズを超える場合は、より多くのコンテンツを表示するように、アプリをカスタマイズできます。たとえば、あるショッピング アプリでは、携帯電話には、一度に 1 つのカテゴリの商品が表示されますが、PC またはノート PC には、複数のカテゴリと製品が同時に表示されます。
  より多くのコンテンツを画面に表示すると、ユーザーが実行する必要があるナビゲーションの量が減少します。
- **デバイスの機能を活用するには**
  一部のデバイスでは、特定のデバイス機能がある可能性が高くなります。たとえば、電話では、位置センサーとカメラが付属している可能性が高く、PC では、どちらも付属していない可能性が高くなります。 アプリは、利用できる機能を検出し、その機能の使用を有効にすることができます。

- **入力用に最適化するには**

  ユニバーサル コントロール ライブラリは、すべての入力タイプ (タッチ、ペン、キーボード、マウス) と連携できますが、その場合も、UI 要素を再配置して、特定の入力タイプを最適化することができます。たとえば、画面の下部にナビゲーション要素を配置すると、携帯電話のユーザーにとってはアクセスしやすくなりますが、ほとんどの PC ユーザーは、ナビゲーション要素は画面の上部に表示されると想定しています。

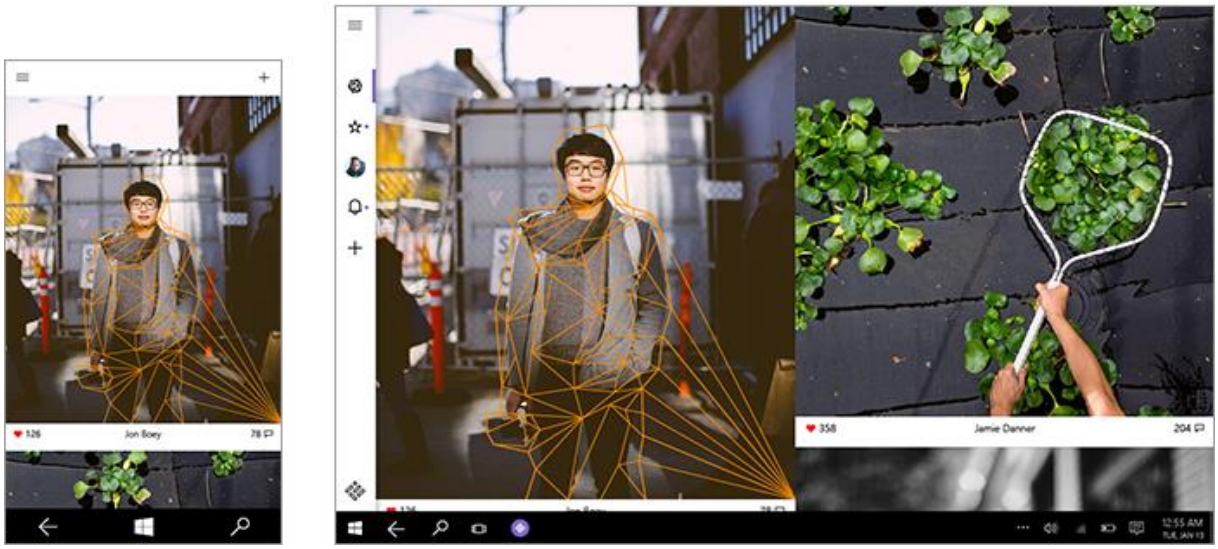
レスポンシブ デザインの手法

アプリの UI を特定の画面の幅に合わせて最適化することは、レスポンシブ デザインの作成と呼ばれます。ここでは、アプリの UI のカスタマイズに使用できる 6 種類のレスポンシブ デザイン手法を紹介します。

位置の変更

アプリの UI 要素の場所と位置を変更して、各デバイスを最大限に活用することができます。この例では、電話やファブレットのビューが縦向きであり、完全なフレームが一度に 1 つだけ表示されるため、スクロール UI が必要です。縦方向か横方向かを問わず、2 つの画面フレームを表示できるデバイスでアプリが使用される場合、フレーム B は、専用の領域を占有できます。グリッドを使用して配置する場合は、UI 要素が再配置されるときに同じグリッドに従うことができます。

写真アプリ向けのこの設計の例では、写真アプリのコンテンツは大きな画面に再配置されます。



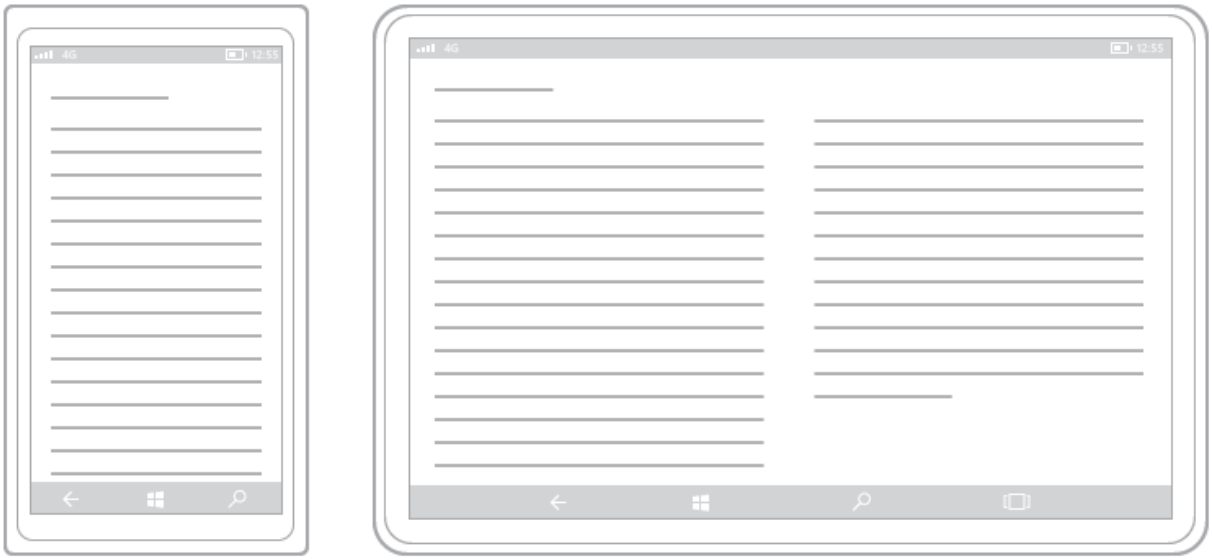
サイズ変更

余白と UI 要素のサイズを調整して、フレーム サイズを最適化できます。次の例が示すように、これにより、コンテンツ フレームを拡大するだけで、より大きな画面で画面を見やすくすることができます。

再配置

デバイスと向きに応じて UI 要素のフローを変更すると、アプリでコンテンツの表示を最適化できます。たとえば、より大きな画面を対象にする場合は、より大きなコンテナーに切り替え、列を追加し、別の方法でリスト項目を生成することが効果的である可能性があります。

次の例は、携帯電話またはファブレットの縦にスクロールするコンテンツの 1 列をより大きな画面に再配置して、2 列のテキストを表示する方法を示しています。



明示

UI は、画面領域に基づいて表示したり、デバイスが追加機能、特定の状況、または推奨される画面の向きをサポートする場合に表示できます。

タブを示す次の例では、カメラ アイコンがある中央のタブが携帯電話またはファブレットのアプリに固有であり、大型のデバイスには適用されません。この理由から、右側のデバイスでは表示されています。UI の表示/非表示のもう 1 つの一般的な例は、メディア プレーヤーのコントロールです。この場合、ボタン セットは、小型のデバイスでは縮小され、使用できる画面領域が大きい大型デバイスでは展開されます。たとえば、PC では、メディア プレーヤーが処理できる画面上の機能は、携帯電話での機能よりもはるかに多くなっています。

明示/非表示の手法の一部には、より多くのメタデータを表示するタイミングの選択が含まれます。携帯電話やファブレットなど、領域が貴重である場合は、表示するメタデータの量を最小限にすることをお勧めします。ノート PC やデスクトップ PC では、かなりの量のメタデータを表示できます。メタデータの表示または非表示を処理する方法の例は、次のとおりです。

- メール アプリでは、ユーザーのアバターを表示できます。
- 音楽アプリでは、アルバムやアーティストの詳細情報を表示できます。
- ビデオ アプリでは、キャストとスタッフの詳細表示など、映画やショーの詳細情報を表示できます。
- どのアプリでも、列を分割して、さらに詳細な情報を表示できます。
- どのアプリでも、縦に並べられたものを横に並べて配置することができます。 携帯電話やファブレットから大型デバイスに移行する場合、縦に並べられたリスト項目は、リスト項目の行とメタデータの列の表示に変更できます。

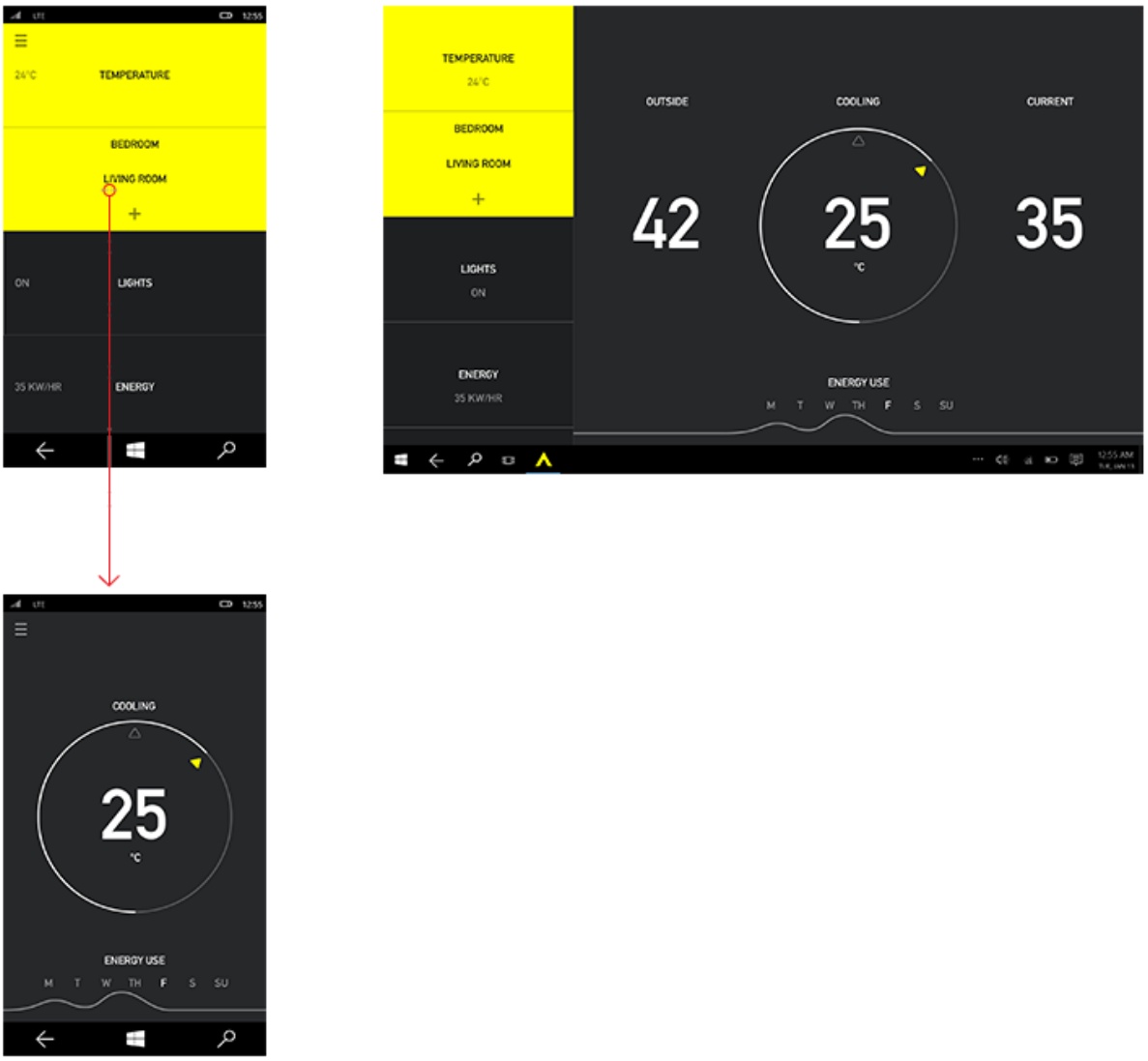
置換

この手法では、特定のデバイスの画面サイズまたは向きにユーザー インターフェイスを切り替えることができます。 次の例では、ナビゲーション ウィンドウとそのコンパクトで一時的な UI は小型デバイスに適していますが、大型デバイスには、タブの方が適しています。

再構築

アプリのアーキテクチャを折りたたんだり、分岐させたりして、対象となる特定のデバイスをより明確にすることができます。次の例では、左側のデバイスから右側のデバイスに移動すると、ページが結合されます。

ここでは、スマート ホーム アプリの設計に適用される、この手法の例を示します。



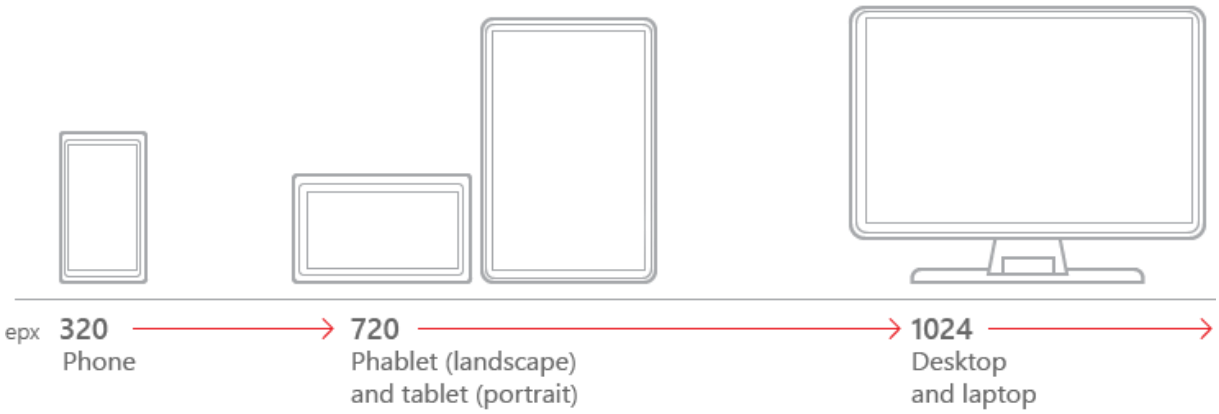
特定サイズ に対するデザインのブレーク ポイント

対象デバイスと、Windows 10 エコシステム全体での画面サイズの数はあまりに多いため、それぞれのために UI を最適化しても意味がありません。その代わり、320、720、および 1024 epx という 3 種類の主要なキー幅 ("ブレークポイント" とも呼ばれます) を設計することをお勧めします。

ヒント 特定のブレークポイント向けに設計するときは、アプリ (アプリのウィンドウ) で使用できる画面領域の大きさ向けに設計します。 アプリが全画面表示で実行されているとき

は、アプリ ウィンドウが画面と同じサイズですが、それ以外の状況では、画面より小さいサイズです。

次の表は、さまざまなサイズ クラスを説明し、これらのサイズ クラスを調整するため一般的な推奨事項を示します。



サイズの種類	**小**	**中**	**大**
有効なピクセル単位の幅	320	720	1024
一般的なサイズ	4" ～ 6"	6+" ～ 12"	13 インチ以上
一般的なデバイス	電話	タブレット、 画面の大きい電話	PC、ノート PC、 Surface Hub
一般的な推奨事項	• ユーザーが親指で簡単にアクセスできるように、画面の下部にナビゲーションおよびコマンド要素を配置すると、ハンドヘルド デバイスでのナビゲーションと操作が容易になります。	• タブ要素を左揃えにします。 • ウィンドウの左右の余白を 24 ピクセルに設定します。アプリ ウィンドウの左右の端の	• ナビゲーションおよびコマンド要素をアプリ ウィンドウの上部に配置します。 • タブ要素を左揃えにします。

- タブ要素を中央に配置します。
- ウィンドウの左右の余白を 12 ピクセルに設定して、アプリ ウィンドウの左右の端の間でビジュアルな区切りを作成します。
- 一度に 1 列/地域を使います。
- 検索を表すアイコンを使います (検索ボックスを表示しない)。
- ナビゲーション ウィンドウをオーバーレイ モードにして画面領域を節約します。
- マスター/詳細要素を使用している場合は、上下に並べる表示モードを使用して画面領域を節約します。

間でビジュアルな区切りを作成することをお勧めします。
- 最大 2 列/領域
- 検索ボックスを表示します。
- 常に表示されるようにナビゲーション ウィンドウを固定モードにします。

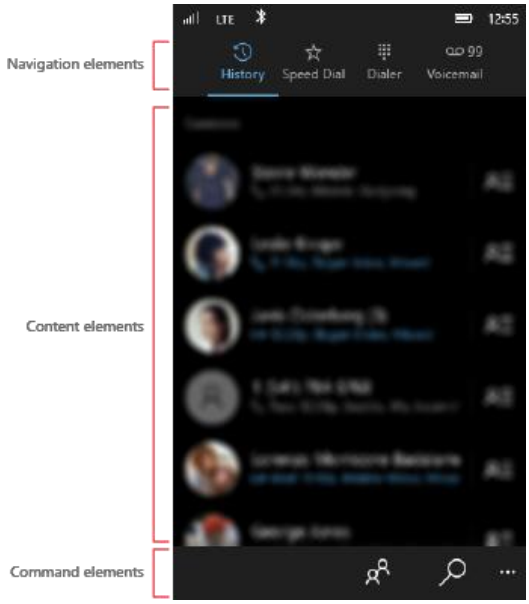
- ウィンドウの左右の余白を 24 ピクセルに設定します。
アプリ ウィンドウの左右の端の間でビジュアルな区切りを作成することをお勧めします。
- 最大 3 列/領域
- 検索ボックスを表示します。
- 常に表示されるようにナビゲーション ウィンドウを固定モードにします。

ユニバーサル Windows プラットフォーム (UWP) アプリの UI の基本

最新のユーザー インターフェイスは複雑であり、テキスト、形状、色、アニメーションで構成されますが、それらは最終的には使っているデバイスの画面の個々のピクセルで構成されます。ユーザー インターフェイスの設計を開始すると、選択肢が多すぎて圧倒されることがあります。

シンプルに考えることができるように、アプリの構造をデザインの観点から定義してみましょう。アプリが画面とページで構成されるとしましょう。各ページには、3 種類の UI 要素 (ナビゲーション、コマンド実行、コンテンツ) で構成されるユーザー インターフェイスがあります。

アプリの構造



ナビゲーション要素

ナビゲーション要素は、表示するコンテンツをユーザーが選択できるようにします。ナビゲーション要素の例には、タブとピボット、ハイパーリンク、ナビゲーション ウィンドウなどがあります。

コマンド要素

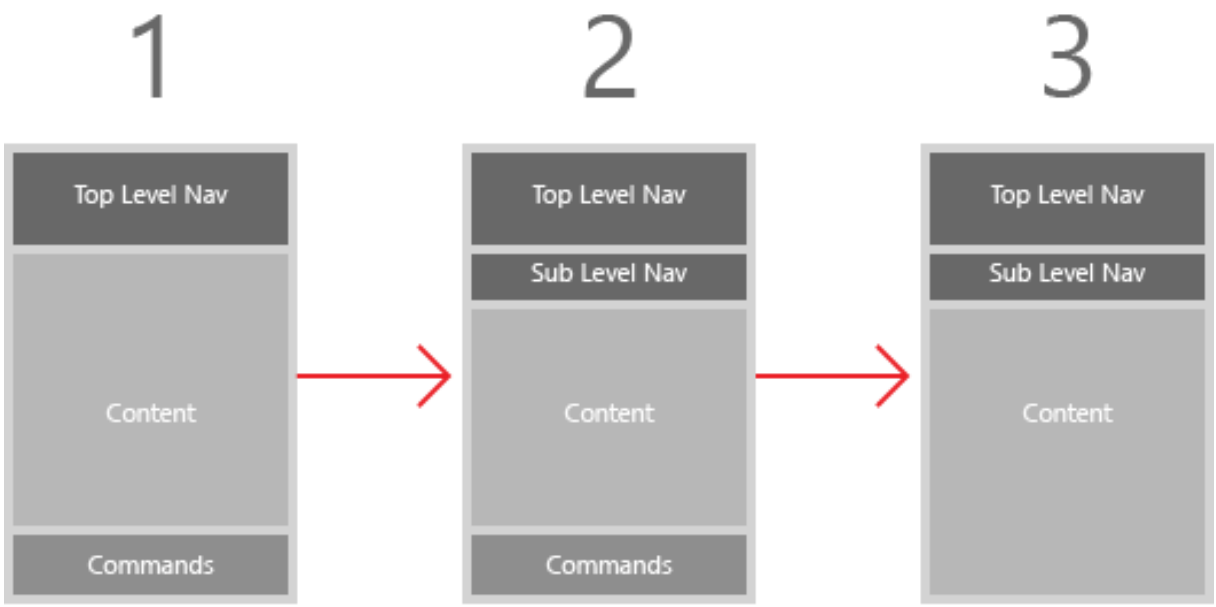
コマンド要素は、操作、保存、コンテンツの共有などの操作を開始します。コマンド要素の例には、ボタンやコマンド バーがあります。コマンド要素には、実際には画面に表示されないキーボード ショートカットも含めることができます。

コンテンツ要素

コンテンツ要素は、アプリのコンテンツを表示します。ペイント アプリでは、コンテンツは描画になり、ニュース アプリでは、コンテンツはニュース記事になります。

少なくとも、アプリは、スプラッシュ スクリーンと、ユーザー インターフェイスを定義するホーム ページで構成されます。一般的なアプリは複数のページと画面で構成され、ナビゲーション要素、コマンド要素、コンテンツ要素はページごとに変化します。

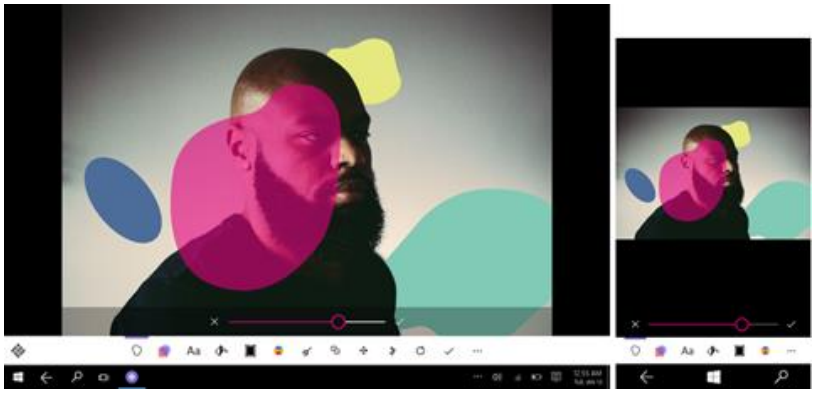
次の図は、さまざまなページを持つ架空のアプリ構造を示しています。各ページには、さまざまなナビゲーション、コマンド、およびコンテンツ要素があります。



ナビゲーション、コマンド、およびコンテンツ要素を結合するための一般的ないくつかの UI パターンを見てみましょう。

一般的な UI パターン

1 つ以上の要素を組み合わせると、パターンが作られます。これらの 5 つの UI パターンは、ナビゲーション、コマンド、およびコンテンツを提供するためによく使われます (ほとんどのアプリは少なくとも 1 つを使います)。 これらの UI 要素は、アプリの UI を構築する際の基礎として使うことができます。 また、UI パターンを組み合わせ、さらに複雑な機能を実現することもできます。

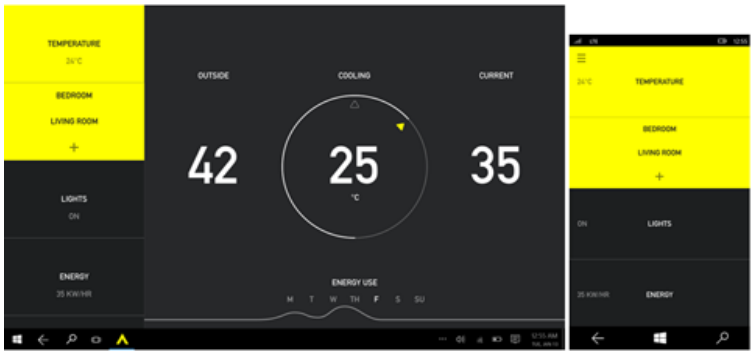
UI 要素	説明
アクティブなキャンバス 	**要素の種類:** コマンド + コンテンツ アクティブなキャンバスは、単一ビュー アプリやモーダル エクスペリエンス (ブラウザー、ドキュメント ビューアー、写真ビューアー/エディター、描画アプリ、自由スクロール メイン ビューを利用する他のアプリなど) に使います。最上位のアクションやページ レベルのアクションを含めることができます。 次に示す写真編集アプリの設計は、アクティブなキャンバスの使用方法を示しています。 
マスター/詳細 	**要素の種類:** ナビゲーション + コンテンツ マスター/詳細パターンでは、マスター リストと、現在選択されている項目の詳細が表示されます。このパターンは、メールや連絡先一覧/アドレス帳によく使用されます。 次に示す株式を追跡するアプリの設計では、マスター/詳細パターンを使います。 

ナビゲーション ウィンドウ

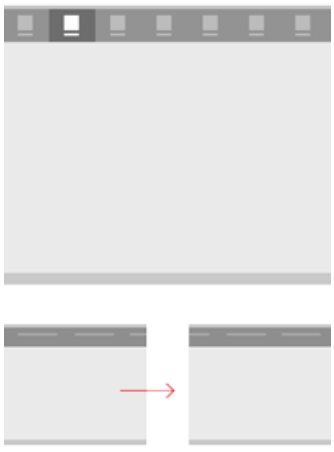
要素の種類: ナビゲーション + コンテンツ

ナビゲーション ウィンドウでは、さまざまなトップレベルのナビゲーション項目を使うことができます。これにより、画面領域を節約することができます。ナビゲーション ウィンドウは、ボタン、ウィンドウ、コンテンツ領域という 3 つの主要なコンポーネントで構成されています。ボタンは、ウィンドウを開閉できるようにする UI 要素です。ウィンドウは、ナビゲーション要素のコンテナーです。コンテンツ領域に、選んだナビゲーション項目からの情報が表示されます。また、ナビゲーション ウィンドウはドッキングされた状態で表示することもできます。この場合、ウィンドウは常に表示されます。

次に示すスマート ホーム アプリ機能の設計では、ナビゲーション ウィンドウ パターンを使います。



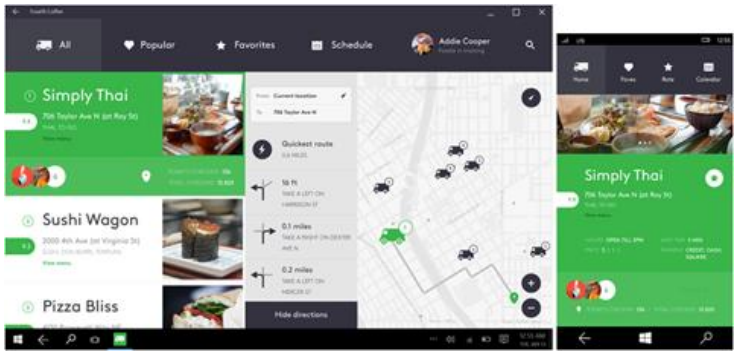
タブとピボット



要素の種類: ナビゲーション + コンテンツ

ページへのリンクの永続的な一覧が表示され、通常は同じデータセット内の異なるピボット (ビューまたはフィルター) 間で迅速な移動手段が提供されます。

次に示すスマート ホーム アプリの設計では、タブ/ピボット パターンを使います。



独自のパターンの作成

これで、アプリの基本的な構造について理解し、いくつかの一般的な UI パターンについて調べて、ナビゲーション、コマンド、コンテンツに関する設計ガイドラインを確認することができました。これらの記事は、どの UI 要素とパターンをいつ使うかを理解するのに役立ちます。

- **ナビゲーション**
  3 つの単純なルールに従って、魅力的なナビゲーション エクスペリエンスを設計する方法について学習します。
- **コマンド**
  コマンド要素と、それらを使って優れた操作エクスペリエンスを作成する方法について学習します。
- **コンテンツ**
  コンテンツはアプリの最も重要な部分です。それは、ユーザーがアプリを使っている間、ほぼ常に注目する領域です。この記事では、さまざまな種類のコンテンツとコンテンツ使用エクスペリエンスのための推奨事項について説明します。

ユニバーサル Windows プラットフォーム (UWP) アプリの ナビゲーション デザインの基本

UWP アプリのナビゲーションは、ナビゲーション構造、ナビゲーション要素、システム レベルの機能から成る柔軟なモデルに基づいています。これら全体で、アプリ、ページ、コンテンツ間での移動の際にさまざまな直感的なユーザー エクスペリエンスを実現します。

場合によっては、アプリのコンテンツと機能のすべてを単一ページに配置して、ユーザーがパンするだけでそのコンテンツ内を移動できるようにすることができます。ただし、ほとんどのアプリは通常、複数ページのコンテンツと、それをユーザーが利用するための機能を備えています。アプリに複数のページがある場合は、適切なナビゲーション エクスペリエンスを提供する必要があります。

UWP アプリにおいて効果的で便利なマルチページ ナビゲーション エクスペリエンスを実現するには、以下の要素が必要です (詳細は後述)。

- **適切なナビゲーション構造**
  わかりやすいナビゲーション エクスペリエンスを作成するためには、ユーザーにとって意味のあるナビゲーション構造を構築することが重要です。
- **互換性のあるナビゲーション要素** (選択した構造をサポートする要素)
  ナビゲーション要素を使うと、ユーザーは必要なコンテンツにアクセスすることができ、アプリ内のどの位置にいるかを把握することもできます。ただし、ナビゲーション要素は、コンテンツやコマンド実行要素用に使うことができる領域を占有します。そのため、アプリの構造に最適なナビゲーション要素を使うことが重要になります。
- **システム レベルのナビゲーション機能("戻る" など) に対する適切な応答**
  直感的な操作性を感じさせる一貫したエクスペリエンスを実現するには、ユーザーが想定している方法で、システム レベルのナビゲーション機能に応答します。

適切なナビゲーション構造を構築する

ページのグループから構成されるコレクションとしてアプリを見てみましょう。各ページには、コンテンツや機能の固有のセットが含まれます。たとえば、写真アプリには、写真を撮影するためのページ、画像を編集するためのページ、画像ライブラリを管理するためのページが含まれている場合があります。 これらのページをグループに配置する方法によって、

アプリのナビゲーション構造が定義されます。ページのグループを配置する一般的な方法には、次の 2 つがあります。

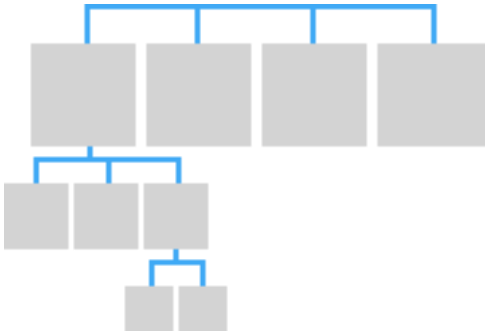
階層に配置する

ページは、ツリー状の構造に配置されます。それぞれの子ページの親は 1 つだけですが、親ページは 1 つ以上の子ページを持つことができます。子ページにアクセスするには、親ページを経由します。

ピアとして配置する

ページは、並列の関係で配置されます。どのような順序でもページ間を移動できます。

一般的なアプリでは両方の配置方法を使います。たとえば、一部のページはピアとして配置され、他のページは階層に配置されます。



では、どのような場合にページを階層に配置し、どのような場合にページをピアとして配置すればよいでしょうか。この質問に答えるには、グループ内のページ数、特定の順序でページ間を移動する必要があるかどうか、およびページ間の関係を考慮する必要があります。一般に、構造がフラットであれば、構造を理解しやすくなり、すばやい移動が可能になります。ただし、深い階層が適している場合もあります。

次の場合は、階層関係を使うことをお勧めします。

- ユーザーが特定の順序でページ間を移動する必要がある場合。その順序を強制的に適用するように階層を配置します。

次の場合は、ピアの関係を使うことをお勧めします。

- ページをどのような順序でも表示できる場合。
- 各ページはそれぞれ異なるページであり、明確な親/子関係はありません。

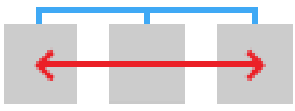
- グループ内の各ページ間には明確な親子関係があります。
- グループ内のページ数が 7 ページを超える場合。
  グループ内のページ数が 7 ページを超える場合、ページが一意であるかどうかを判断したり、グループ内の現在の位置を把握したりするのが難しくなる場合があります。このことがアプリでは問題にはならないと考えられる場合は、ページをピアとして作成します。このことが問題となる可能性がある場合は、階層構造を使って、ページを 2 つ以上の小さなグループに分割することを検討してください (ハブ コントロールを使うと、ページをカテゴリにグループ化できます)。
- グループ内のページ数が 8 ページよりも少ない場合。
  グループ内のページ数が 7 ページを超える場合、ページが一意であるかどうかを判断したり、グループ内の現在の位置を把握したりするのが難しくなる場合があります。このことがアプリでは問題にはならないと考えられる場合は、ページをピアとして作成します。このことが問題となる可能性がある場合は、階層構造を使って、ページを 2 つ以上の小さなグループに分割することを検討してください (ハブ コントロールを使うと、ページをカテゴリにグループ化できます)。

適切なナビゲーション要素の使用

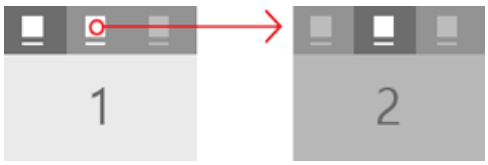
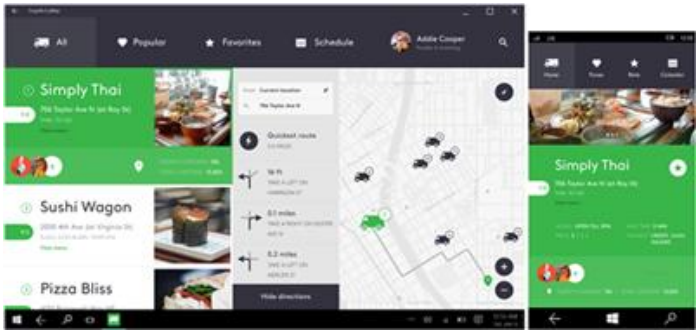
ナビゲーション要素は 2 つのサービスを提供できます。つまり、これらの要素を使うと、ユーザーは必要なコンテンツにアクセスすることができ、要素によっては、アプリ内のどの位置にいるかを把握することができます。ただし、ナビゲーション要素は、コンテンツやコマンド実行要素用にアプリで使うことができる領域を占有します。そのため、アプリの構造に最適なナビゲーション要素を使うことが重要になります。

ピア ツー ピアのナビゲーション要素

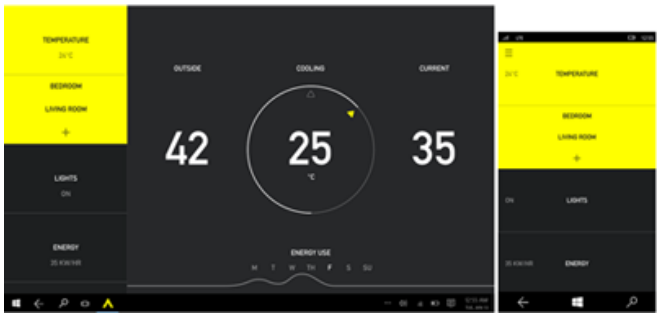
ピア ツー ピアのナビゲーション要素によって、同じサブツリーの同じレベルにあるページ間のナビゲーションが有効になります。



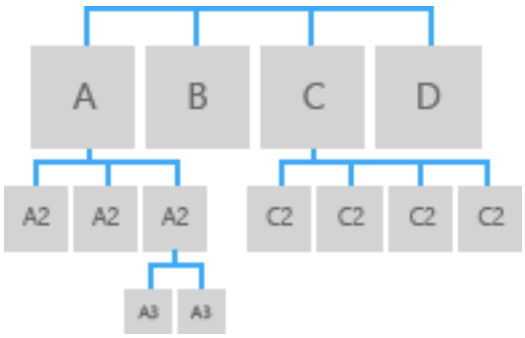
ピア ツー ピアのナビゲーションでは、タブまたはナビゲーション ウィンドウを使うことをお勧めします。

ナビゲーション要素	説明
タブとピボット 	同じレベルにあるページへのリンクの永続的な一覧を表示します。 タブ/ピボットを使う場合 • ページ数が 2 ～ 5 ページの場合 5 ページを超える場合でもタブ/ピボットを使うことはできますが、すべてのタブ/ピボットを画面に収めることが難しくなることがあります。 • ユーザーが頻繁にページ間を切り替えることを前提としている場合。 レストラン検索アプリの、タブ/ピボットを使います。 
ナビゲーション ウィンドウ 	トップレベルのページへのリンクの一覧を表示します。 ナビゲーション ウィンドウを使う場合 • ユーザーが頻繁にページ間を切り替えることを前提としていない場合。 • ナビゲーション操作の速度を低下させて、領域を節約する必要がある場合。 • ページがトップレベルに存在する場合。

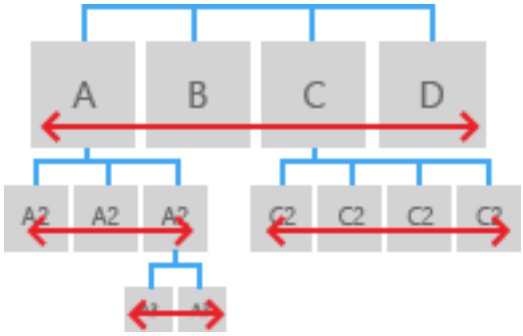
スマート ホーム アプリのこの設計では、ナビゲーション ウィンドウを使います。



ナビゲーション構造に複数のレベルがある場合は、現在のサブツリー内のピアにのみリンクするピア ツー ピアのナビゲーション要素を使うことをお勧めします。3 つのレベルを持つナビゲーション構造を示す、次の図について考えてみましょう。

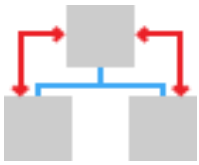


- レベル 1 では、ピア ツー ピアのナビゲーション要素によって、ページ A、B、C、および D へのアクセスが提供されます。
- レベル 2 では、A2 ページのピア ツー ピアのナビゲーション要素は、他の A2 ページにのみリンクしています。これらのナビゲーション要素は、C サブツリー内にあるレベル 2 のページにはリンクしていません。



階層型ナビゲーション要素

階層型ナビゲーション要素は、親ページとその子ページ間のナビゲーションを実現します。



ナビゲーション要素	説明
ハブ 	ハブは、子ページのプレビュー/概要を提供する特殊な種類のナビゲーション コントロールです。ナビゲーション ウィンドウやタブとは異なり、ページ自体に埋め込まれているリンクやセクション ヘッダーを使って、子ページへのナビゲーションを実現します。 ハブを使う場合 • ユーザーが、各子ページへ移動せずに、子ページのコンテンツの一部を表示することを前提としている場合。 ハブによって検出や調査がサポートされ、メディア、ニュース リーダー、ショッピング アプリに適した検出や調査の実施が可能になります。
マスター/詳細 	項目の概要の一覧 (マスター ビュー) が表示されます。項目を選ぶと、対応する項目ページが詳細セクションに表示されます。 マスター/詳細要素を使う場合 • ユーザーが頻繁に子項目間を切り替えることを前提としている場合。 • ユーザーが個々の項目や項目のグループに対して高いレベルの操作 (削除や並べ替えなど) を実行できるようにする場合、およびユーザーが各項目の詳細情報の表示や更新を実行できるようにする場合。 マスター/詳細要素は、メールの受信トレイ、連絡先リスト、データ入力に適しています。

株式を追跡するアプリのこの設計では、マスター/詳細パターンを使います。

履歴ナビゲーション要素

ナビゲーション要素	説明
戻る	ユーザーは、アプリ内のナビゲーション履歴や、デバイスによってはアプリ間を移動できます。詳細は、後述する「システム レベルのナビゲーションシステム機能がアプリで正しく動作するようにする」セクションを参照してください。

コンテンツに埋め込まれたナビゲーション要素

ナビゲーション要素	説明
コンテンツに埋め込まれたナビゲーション要素	コンテンツに埋め込まれたナビゲーション要素は、ページのコンテンツに表示されます。他のナビゲーション要素はページのグループやサブツリー全体で一貫性がありますが、それとは異なり、コンテンツに埋め込まれたナビゲーション要素はそれぞれのページに固有のナビゲーション要素です。

ナビゲーション要素の連結

ナビゲーション要素を連結してアプリに最適なナビゲーション エクスペリエンスを作成することができます。たとえばアプリでナビゲーション ウィンドウを使って、トップ レベルのページにアクセスできるようにしたり、第 2 レベルのページにアクセスできるタブを提供することができます。

システム レベルのナビゲーションシステム機能がアプリで正しく動作するようにする

Web の場合、個々の Web サイトには独自のナビゲーション システム (目次、ボタン、メニュー、リンクの簡単な一覧など) が用意されています。ナビゲーション エクスペリエンスは、Web サイトによっては大幅に異なる場合があります。ただし、一貫して同じナビゲーション エクスペリエンスが 1 つあります。それは "戻る" 操作です。ほとんどのブラウザーには、表示している Web サイトに関係なく同じように動作する戻るボタンがあります。

同様の理由から、Windows プラットフォーム (UWP) では、アプリのユーザーのナビゲーションの履歴内の移動や、デバイスによってはアプリ間の移動について、一貫性のある "戻る" ナビゲーション システムを提供します。

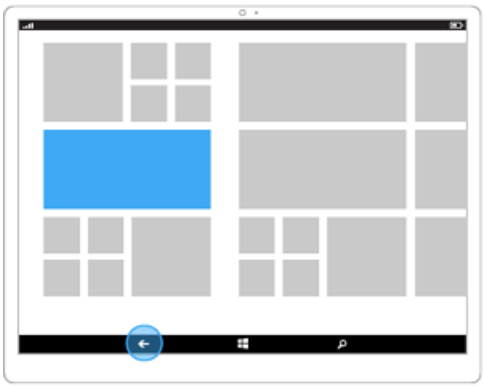
システムの戻るボタンの UI は、フォーム ファクターや入力デバイスの種類ごとに最適化されますが、ナビゲーション エクスペリエンスはグローバルであり、デバイスや UWP アプリで一貫しています。

主なフォーム ファクターでの戻るボタン UI を次に示します。

電話

- 常に表示されます。
- デバイスの下部にあるソフトウェアまたはハードウェア ボタン。
- アプリ内部やアプリ間で、戻るナビゲーションを実現します。

タブレット

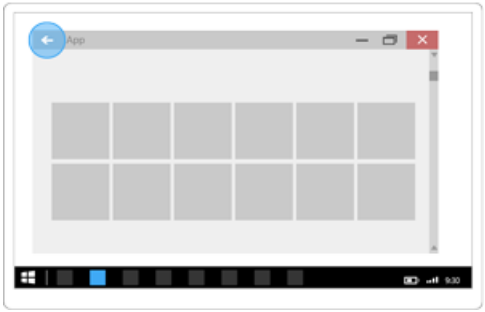


- タブレット モードでは、常に表示されます。
  デスクトップ モードでは利用できません。代わりに、タイトル バーの戻るボタンを有効にすることができます。
  ユーザーは、**[設定]**、**[システム]**、**[タブレット モード]** の順

に選択するか、**[デバイスをタブレットとして使用すると、Windows のタッチ機能がより使いやすくなります]** をオンにすることによって、タブレット モードでの実行とデスクトップ モードでの実行を切り替えることができます。

- デバイスの下部のナビゲーション バーにあるソフトウェア ボタン 。
- アプリ内部やアプリ間で、グローバルな戻るナビゲーションを実現します。

PC、ノート PC



- デスクトップ モードはオプションです。
  タブレット モードでは利用できません。
  既定では無効になっています。有効にすることをオプトインする必要があります。
  ユーザーは、**[設定]、[システム]、[タブレット モード]** の順に選択するか、**[デバイスをタブレットとして使用すると、Windows のタッチ機能がより使いやすくなります]** をオンにすることによって、タブレット モードでの実行とデスクトップ モードでの実行を切り替えることができます。

- アプリのタイトル バーに表示されます。
- アプリ内部のみで、戻るナビゲーションを実現します。アプリ間のナビゲーションはサポートされません。

Surface Hub

- 常に表示されます。
- デバイスの下部にあるソフトウェア ボタン。
- アプリ内部やアプリ間での戻るナビゲーション。

ここでは、戻るボタンの UI に依存しないが、まったく同じ機能を提供する代替入力の種類をいくつか示します。

入力デバイス

キーボード  Windows キー + Backspace

Cortana  "コルタナさん、戻る" と話す。

アプリが電話、タブレット、PC、ノート PC で実行され、システムの戻るボタンが有効になっていると、戻るボタンが押されたときに、システムからアプリに通知されます。 ユーザーは、戻るボタンによって、アプリのナビゲーション履歴における前の場所に戻ることを想定しています。ナビゲーション履歴に追加するナビゲーション操作の種類、および戻るボタンが押されたときの応答方法は、自由に決めることができます。

システムの "戻る" ナビゲーションの動作

有効にした場合、"戻る" ナビゲーションは常にアプリ間およびアプリ内で動作する。

ここでは、さまざまなフォーム ファクターや入力デバイスで一般的な、システムの "戻る" ナビゲーション動作をいくつか示します。

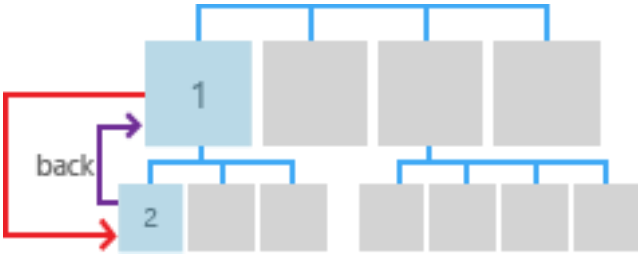
動作	説明
システムの戻るボタン	デバイス全体でサポートされます。 デバイスやモードの設定に応じて、ハードウェア ボタン、またはタスク バーやアプリのタイトル バーに表示されるソフトウェア ボタン。
デスクトップ モードからタブレット モード	タブレット モードのバック スタックは、デスクトップ モードの現在のタスク ビューにある、開いているアプリの順序に基づいています。
タブレット モードからデスクトップ モード	デスクトップ モードでは、アプリはバック スタックを持ちません。タブレット モードのバック スタックは破棄されます。デスクトップ モードから、タブレット、デスクトップと移動した場合、タブレット モードで実行されたすべてのナビゲーションを考慮して、システムは開いているすべてのアプリを以前の状態に復元しようとします。
アプリのナビゲーション構造	"戻る" ナビゲーションは、タブやピボットなど、アプリで定義されたナビゲーション要素では動作しません。マスター/詳細ナビゲーション モデルでは、詳細画面からの "戻る" ナビゲーションは常にホームに戻ります。
フォーカス	"戻る" ナビゲーションは、常に、フォーカスのある UI に対して機能します。それが、アプリ ベースの UI であるか、ポップアップなどのシェル ベースの UI であるかは関係ありません。
閉じる	"戻る" ナビゲーションでは、アプリは終了されません。アプリはバック スタックから削除される可能性はありますが、タスク バー、タスク ビュー、および Alt + Tab キーによる実行中のアプリの一覧を使って利用できます。 アプリを終了すると、アプリはバック スタックから削除されます。

再開	(スタート画面、タスク バー、タスク ビュー、または Alt + Tab キーで) アプリを再開すると、アプリは常にバック スタックの一番上に移動します。
仮想デスクトップ	バック スタックは、常に現在のデスクトップにローカルです。
スナップされたビュー	各アプリには独自のバック スタックがあり、システムの戻るボタンはフォーカスがあるアプリでのみ動作します。 分割画面の区切り線を使用して画面からアプリが押し出されると (削除)、アプリはバック スタックから削除されます。
一時的な UI	一時的な UI (タッチ キーボード、ダイアログ) は閉じられます。
ディープ リンク	"戻る" ナビゲーションは、(デバイスがデスクトップ モードでない限り) 呼び出し元アプリに戻ります。
選択項目	"戻る" ナビゲーションでは、選択した項目を元に戻し、変更されていない一覧に戻します。

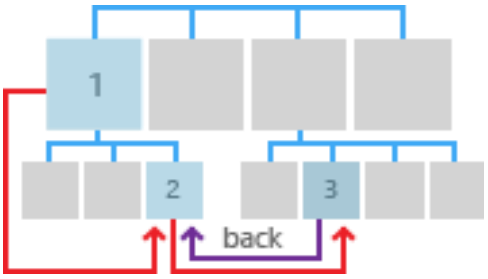
カスタムの "戻る" ナビゲーションの動作

バック スタック ナビゲーションを提供する場合、エクスペリエンスが他のアプリと一貫している必要があります。ナビゲーション操作では、次のパターンに従うことをお勧めします。

ナビゲーション操作	**ナビゲーション 履歴へ追加**
ページ間、異なるピア グループ	○ この図では、ユーザーはピア グループを経由して、アプリのレベル 1 からにレベル 2 に移動します。そのため、このナビゲーションはナビゲーション履歴に追加されます。



次の図では、ユーザーは同じレベルにある 2 つピア グループの間を移動し、この場合もピア グループを経由します。そのため、このナビゲーションはナビゲーション履歴に追加されます。

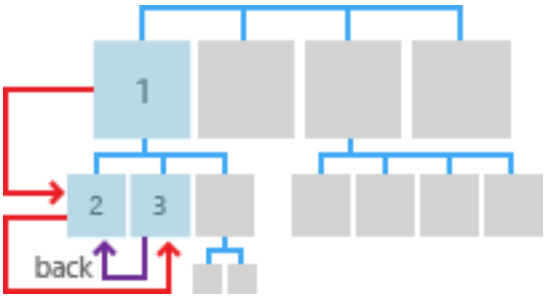


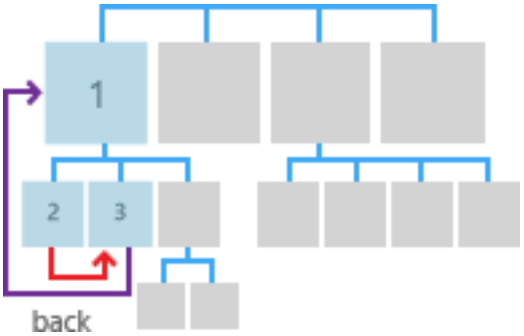
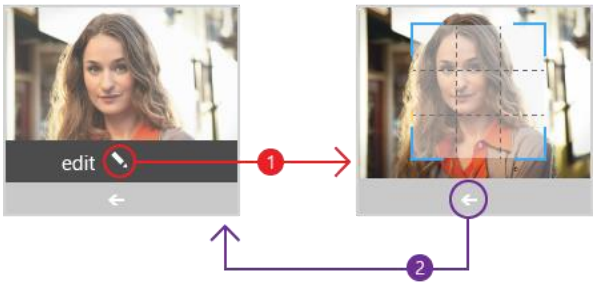
ページ間、同じピア グループ、ナビゲーション要素は画面上に表示されない

ユーザーは、同じピア グループでページ間を移動します。両方のページを対象とした直接的なナビゲーションを実現するナビゲーション要素 (タブ/ピボットや、ドッキングされたナビゲーション ウィンドウなど) は画面に表示されません。

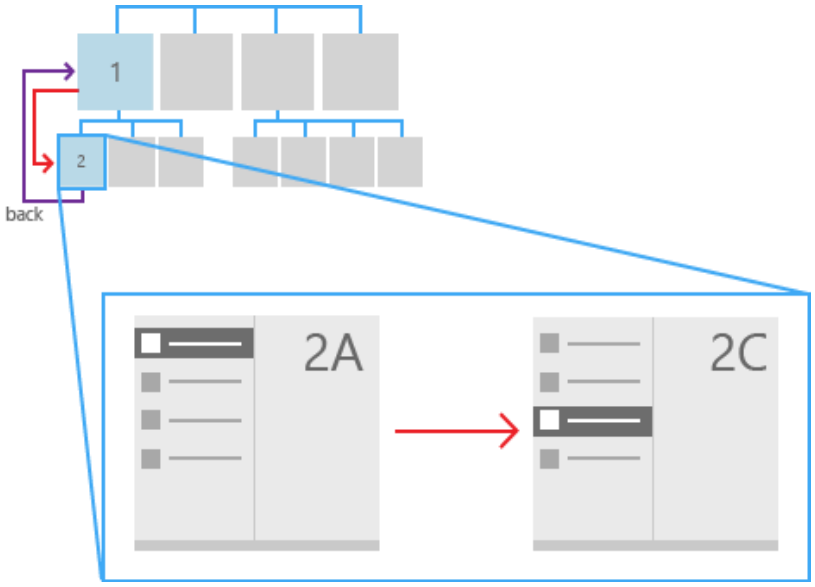
○

次の図では、ユーザーは同じピア グループ内の 2 つのページ間を移動します。ページでは、タブやドッキングされたナビゲーション ウィンドウは使われていません。そのため、このナビゲーションはナビゲーション履歴に追加されます。



ページ間、同じピア グループ、画面上に表示されるナビゲーション要素を使う ユーザーは、同じピア グループ内のページ間を移動します。 両方のページは同じナビゲーション要素に表示されます。たとえば、両方のページで同じタブ/ピボット要素を使っていたり、両方のページがドッキングされたナビゲーション ウィンドウに表示されるとします。	× ユーザーが戻るボタンを押すと、現在のピア グループに移動する前に表示していた最後のページに戻ります。 
一時的な UI の表示 アプリは、ダイアログ、スプラッシュ スクリーン、スクリーン キーボードなどのフライアウト ウィンドウや子ウィンドウを表示します。または、アプリが複数選択モードなどの特別なモードに移行します。	× ユーザーが戻るボタンを押すと、一時的な UI が閉じられ (スクリーン キーボードが非表示になる、ダイアログがキャンセルされるなど)、一時的な UI を生成したページに戻ります。 
項目の列挙 アプリが、マスター/詳細リストで選んだ項目	×

の詳細など、画面上の項目のコンテンツを表示します。	項目の列挙は、ピア グループ内の移動に似ています。ユーザーが戻るボタンを押すと、項目の列挙が表示されている現在のページの前のページに移動されます。



再開

ユーザーが別のアプリに切り替えた後で、元のアプリに戻った場合は、ナビゲーション履歴にある最後のページに戻すことをお勧めします。

次のステップ

他の 2 種類の UI 要素に関する設計ガイドラインをご覧ください。

- コマンド
  コマンド要素と、それらを使って優れた操作エクスペリエンスを作成する方法について学習します。
- コンテンツ
  コンテンツはアプリの最も重要な部分です。それは、ユーザーがアプリを使っている間、ほぼ常に注目する領域です。この記事では、さまざまな種類のコンテンツとコンテンツ使用エクスペリエンスのための推奨事項について説明します。

ユニバーサル Windows プラットフォーム (UWP) アプリの コマンド デザインの基本

UWP アプリのコマンド要素は、ユーザーがメール送信、項目の削除、フォームの送信などの操作を実行できる対話型の UI 要素です。この記事では、ボタンやチェック ボックスなどのコマンド要素、それらの要素でサポートされる操作、それらの要素をホストするコマンド サーフェス (コマンド バーやコンテキスト メニューなど) について説明します。

適切な種類の操作の提供

コマンド インターフェイスを設計する際、最も重要な決定はユーザーが何を実行できる必要があるかという点です。たとえば、フォト アプリを作成している場合、ユーザーには写真を編集するツールが必要です。ただし、写真を表示できるソーシャル メディア アプリを作成している場合は、イメージ編集の優先度が高くない可能性があるので、スペース節約のために編集ツールを省略できます。ユーザーが達成する目的を決定して、それに役立つツールを提供します。

アプリの適切なインターフェイスを計画する際の推奨事項については、「アプリの計画」をご覧ください。

操作に適切なコマンド要素を使用します

適切な操作に適切な要素を使うことは、直感的に使うことができるアプリとなるか、使いにくくてややこしいアプリとなるかの分かれ目になります。ユニバーサル Windows プラットフォーム (UWP) には、アプリで使うことができる多くのコマンド要素がコントロールの形で用意されています。最も一般的ないくつかのコントロールの一覧と、それによって可能になる操作の概要を以下に示します。

カテゴリ	要素	操作
ボタン	**ボタン**	メール送信、ダイアログでのアクションの確認、フォーム データの送信など、即座にアクションをトリガーします。
日付/時刻選択ツール	**CalendarDatePicker、CalendarView、DatePicker、TimePicker**	クレジット カードの有効期限を入力したり、アラームを設定したりするときなどに、ユーザーが日時情報を表示して変更できるようにします。

リスト	**ドロップダウン リスト、リスト ボックス、リスト ビューとグリッド ビュー**	対話型のリストまたはグリッド内に項目を表示します。これらの要素を使うと、ユーザーは新着の一覧からムービーを選んだり、在庫を管理したりすることができます。
テキスト予測入力	**オート サジェスト ボックス**	入力候補を表示して、ユーザーがデータを入力したりクエリを実行したりする時間を節約できるようにします。
選択コントロール	**チェック ボックス**、**ラジオ ボタン**、**トグル スイッチ**	アンケートに入力するときや、アプリ設定を構成するときなど、ユーザーがさまざまなオプションを選ぶことができるようにします。

注　コントロールの中から抜粋しており、すべてのコントロールを網羅していないことに注意してください。

適切なサーフェイスへコマンドを配置します

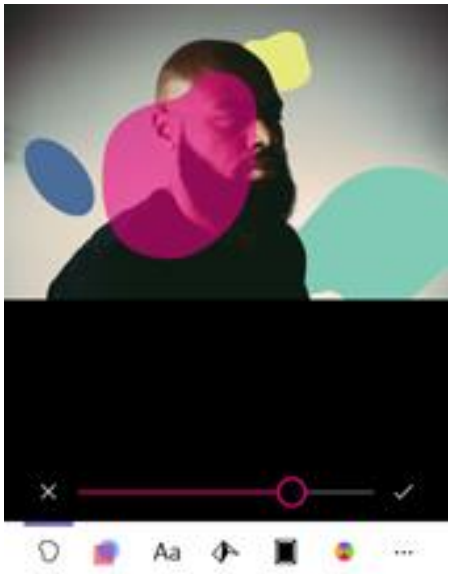
アプリのキャンバス (アプリのコンテンツ領域) や、コマンド バー、メニュー、ダイアログ、ポップアップなどコマンド コンテナーとして機能する特殊なコマンド要素を含む、アプリの多くのサーフェスにコマンド要素を配置できます。コマンドを配置する際の一般的な推奨事項は次のとおりです。

- できる限り、コンテンツを操作するコマンドを用意せず、アプリのキャンバス上でユーザーがコンテンツを直接操作できるようにします。たとえば、旅行アプリで旅行計画を編集するときに、リスト内のアクティビティを上下に移動するコマンド ボタンを使うのではなく、キャンバスのリスト上でアクティビティをドラッグ アンド ドロップできるようにします。
- ユーザーが直接コンテンツを操作できない場合は、コマンドを次の UI サーフェスのいずれかに配置します。
  - **コマンド バー:** ほとんどのコマンドはコマンド バーに配置してください。コマンドを整理しやすくなり、アクセスしやすくなります。
  - **アプリのキャンバス上:** 目的が 1 つに限られているページまたはビューは、その目的用のコマンドをキャンバス上に直接配置できます。
    しかし、このようなコマンドはなるべく作らないでください。

- **コンテキスト メニュー:** クリップボードの操作 (切り取り、コピー、貼り付けなど) や、選択できないコンテンツに実行するコマンド (地図の目的の位置にピンを追加するなど) に使うことができます。

Windows に用意されたコマンド サーフェスの一覧と、それらを使用する際の推奨事項を以下に示します。

サーフェス	説明
アプリのキャンバス（コンテンツ領域） 	あるコマンドが重要で、ユーザーが中心的なシナリオを完了するうえで常に必要な場合は、そのコマンドをキャンバス (アプリのコンテンツ領域) に配置できます。 コマンドは影響を与えるオブジェクトの近く (またはその上) に配置できるため、キャンバスにコマンドを配置すると使い方がわかりやすくなります。 ただし、キャンバスに配置するコマンドは慎重に選んでください。アプリのキャンバスにコマンドが多すぎると、貴重な画面のスペースがなくなり、ユーザーを困惑させる可能性があります。それほど頻繁に使わないコマンドの場合、メニューやコマンド バーの [その他] 領域など、別のコマンド サーフェスに配置することを検討してください。
コマンド バー 	コマンド バーを使うと、ユーザーはアクションに簡単にアクセスできます。コマンド バーは、ユーザーのコンテキストに固有のコマンドまたはオプション (写真の選択や描画モードなど) を表示するためにも使うことができます。 コマンド バーは画面の上部または画面の下部、あるいは画面の上部と下部の両方に配置できます。次に示す写真編集アプリの設計は、コンテンツ領域とコマンド バーを示しています。



メニューと**コンテキスト メニュー**

複数のコマンドをコマンド メニューにグループ化することで効率性が高まる場合があります。 メニューを使うと、より狭い場所により多くのオプションを表示できます。メニューには対話的なコントロールを含めることができます。

コンテキスト メニューは、よく使うアクションへのショートカットを提供し、特定のコンテキストにのみ関連するセカンダリ コマンドにアクセスできるようにします。

コンテキスト メニューは、次の種類のコマンドとコマンド シナリオを対象としています。

- 選んだテキストに対する状況依存の操作 (コピー、切り取り、貼り付け、スペル チェックなど)。
- 操作する必要があるものの、選択することも、他の方法で指定することもできないオブジェクトのためのコマンド。
- クリップボード コマンドの表示。
- カスタム コマンド。

次に示す例は、ショートカット メニューを使って経路の変更、経路のブックマーク登録、別の電車の選択を行う地下鉄アプリのデザインを示しています。

ダイアログ コントロール

ダイアログは、状況依存のアプリ情報を表示するモーダル UI オーバーレイです。ほとんどの場合、ダイアログは明示的に閉じられまでアプリ ウィンドウの操作を妨げます。また、多くの場合、ユーザーに操作を要求します。

ダイアログは、煩わしく感じることがあるため、特定の状況でのみ使用してください。詳しくは、「アクションを確認、または元に戻すタイミング」をご覧ください。

フライアウト

ユーザーが現在操作している内容に関係する UI を表示する軽量な状況依存のポップアップです。フライアウトは、次の目的で使います。

- メニューを表示する。
- 項目の詳細を表示する。
- アプリの操作をブロックしないでユーザーにアクションの確認を求める。

フライアウトは、その外側をタップするかクリックすることで閉じることができます。

アクションを確認、または元に戻るタイミング

適切に設計されたユーザー インターフェイスであっても、ユーザーがどれほど慎重に作業したとしても、すべてのユーザーは必ず意図しないアクションを実行します。ユーザーにアクションの確認を求めたり、最近のアクションを元に戻す方法を用意することによって、アプリでこのような状況に対処することができます。

- 元に戻すことができず、実行結果が重大な操作の場合は、確認ダイアログ ボックスの使用をお勧めします。このような操作の例は、次のとおりです。
  - ファイルを上書きする
  - ファイルを保存せずに終了する
  - ファイルやデータを完全に削除することを確認する
  - 購入する (確認メッセージを表示しないことをユーザーが選択した場合を除く)
  - 何かへのサインアップなどのフォームを送信する
- 元に戻すことができる操作の場合は、通常、単純な "元に戻す" コマンドを提供すれば十分です。このような操作の例は、次のとおりです。
  - ファイルを削除する
  - メールを削除する (完全には削除しない)
  - コンテンツを変更する、またはテキストを編集する
  - ファイル名を変更する

ヒント アプリで使う確認ダイアログの量に注意してください。ユーザーが間違えたときはとても役に立ちますが、ユーザーが意図的にアクションを実行しようとしているときは邪魔になります。

特定の入力に対する最適化

UWP アプリでは、"スマートな" 操作の入力システムが使われるため、アプリが受け取る入力の種類について心配する必要がありません。たとえば、クリックの発生元が実際のマウス クリックであるか、指によるタップであるかどうかを認識しなくても、クリック操作に対応したデザインを行うことができます。

それでも、特定の入力を便利に活用するためににアプリをカスタマイズする場合もあります。

タッチ

タッチ操作では、(マウスやペンのように) 代替の入力方法として、または相補的な入力方法として (汚れ、ペンで描くひと筆などの他の入力を変更)、UI 要素の直接的な操作 (パン、回転、サイズ変更、移動など) をエミュレートするために、1 つまたは複数の指から物理的な

ジェスチャを使用できます。このような触覚的なエクスペリエンスは、ユーザーが画面の要素を操作するときに、より自然で現実的で感覚を提供します。

注 デバイスによって、さまざまなレベルのタッチ サポートを提供できます。デバイスによって、1 か所の接触のみ、またはマルチタッチ操作 (2 か所以上の接触) をサポートする場合や、タッチ操作をまったくサポートしない場合があります。

ペン

ペン (またはスタイラス) は、マウスのようにピクセル単位のポインティング デバイスとして使うことができます。また、デジタル インク入力にとっては、最適なデバイスです。

Windows のインク プラットフォームでペンを使うと、自然な形で手書きノート、描画、コメントを作れます。このプラットフォームは、デジタイザー入力からのインク データのキャプチャ、インク データの生成、出力デバイスへのひと筆としてのデータのレンダリング、インク データの管理、手書き認識の実行をサポートします。 ユーザーが書いたり描画したりするときのペンの空間移動のキャプチャに加えて、アプリで筆圧、形状、色、不透明度などの情報を収集して、紙の上でペン、鉛筆、ブラシを使っているときに近いユーザー エクスペリエンスを実現することもできます。

ペン入力とタッチ入力が異なるのは、タッチでは画面上の UI 要素に対する物理的なジェスチャ (スワイプ、スライド、ドラッグ、回転など) を通じて、それらのオブジェクトへの直接の操作をエミュレートする機能があることです。このような操作をサポートするには、ペン固有の UI コマンド、またはアフォーダンスを提供します。たとえば、"前へ" ボタンと "次へ" ボタン (または "+" ボタンと "-" ボタン) を使ってコンテンツのページをフリップしたり、オブジェクトを回転、サイズ変更、ズームしたりできるようにします。

注 ペン デバイスには、アクティブとパッシブの 2 種類があります。ここでペン デバイスと言う場合、高度な入力データを提供し、主に正確なインク操作やポイント操作に使われるアクティブなペンを指します。パッシブなペンは同様の高度な入力データを提供せず、基本的にタッチ入力をエミュレートします。

マウス

マウスは、生産性アプリや、ターゲット設定とコマンド実行にピクセル精度を必要とするユーザー操作に最適です。または、高密度 UI をサポートする必要があります。

マウス入力は、キーボードのさまざまなキー (Ctrl、Shift、Alt キーなど) を追加することで動作を変更できます。これらのキーは、マウスの左ボタンや右ボタン、ホイール ボタン、X ボタンと組み合わせて、マウスに最適化した拡張コマンド セットを作成できます。

ペンと同様に、マウス入力とタッチ入力が異なるのは、タッチでは画面上の UI 要素に対する物理的なジェスチャ (スワイプ、スライド、ドラッグ、回転など) を通じて、それらのオブジェクトへの直接の操作をエミュレートする機能があることです。このような操作をサポートするには、マウス固有の UI コマンド、またはアフォーダンスを提供します。たとえば、"前へ" ボタンと "次へ" ボタン (または "+" ボタンと "-" ボタン) を使ってコンテンツのページをフリップしたり、オブジェクトを回転、サイズ変更、ズームしたりできるようにします。

タッチパッド

タッチパッドは、間接的なマルチタッチ入力と、マウスのようなポインティング デバイスの精密入力を組み合わせたものです。この組み合わせにより、タッチパッドはタッチに最適化された UI にも、生産性アプリのより小さいターゲットにも適しています。

通常、タッチパッドは、オブジェクトと UI の直接的な操作のタッチと同様のサポートを提供するタッチ ジェスチャのセットをサポートします。

タッチパッドでサポートされているインタラクションのエクスペリエンスは複合的であるため、単にタッチ入力のサポートに依存するのではなく、マウス スタイル UI コマンドまたはアフォーダンスも提供することをお勧めします。このような操作をサポートするには、タッチパッド固有の UI コマンド、またはアフォーダンスを提供します。たとえば、"前へ" ボタンと "次へ" ボタン (または "+" ボタンと "-" ボタン) を使ってコンテンツのページをフリップしたり、オブジェクトを回転、サイズ変更、ズームしたりできるようにします。

キーボード

キーボードはテキスト用の主要な入力デバイスであり、多くの場合、特定の障碍のあるユーザーや、キーボードを使った方がアプリをすばやく効率よく操作できると考えるユーザーにとって欠かせません。

ユーザーはハードウェア キーボードと 2 つのソフトウェア キーボード (スクリーン キーボード (OSK) およびタッチ キーボード) を通じて、ユニバーサル アプリを操作できます。

OSK は、物理的なキーボードの代わりに使うことができるビジュアルなソフトウェア キーボードです。タッチ、マウス、ペン/スタイラス、またはその他のポインティング デバイスを通じてデータを入力します (タッチ スクリーンは必須ではありません)。OSK は、物理的なキーボードが存在しないシステムや、運動障碍により一般的な物理入力デバイスを使うことができないユーザーのために用意されています。OSK は、ハードウェア キーボードの機能のすべて、または少なくともほとんどをエミュレートします。

タッチ キーボードは、タッチ入力でのテキスト入力に使われる、ビジュアルなソフトウェア キーボードです。タッチ キーボードはテキスト入力専用であり (ハードウェア キーボードをエミュレートしません)、テキスト フィールドや編集可能なテキスト コントロールにフォーカスがあるときにだけ表示されるので、OSK の代わりになるものではありません。タッチ キーボードはアプリまたはシステム コマンドをサポートしていませんい。

注 OSK の方がタッチ キーボードより優先され、OSK が表示されている場合はタッチ キーボードは表示されません。

音声認識

Windows 10 では、**Cortana** の機能を拡張することで、音声コマンドを処理してアプリケーションを起動し、単一のアクションを実行できるようにすることができます。

音声コマンドは、1 つの言葉を声に出すことであり、音声コマンド定義 (VCD) ファイルで定義されています。**Cortana** を通じてインストール済みアプリに指示が伝えられます。アプリは、操作のレベルと複雑さに応じて、フォアグラウンドまたはバックグラウンドで起動することができます。たとえば、追加のコンテキストやユーザー入力が必要な音声コマンドはフ

ォアグラウンドで処理するのが最適ですが、基本的なコマンドはバックグラウンドで処理できます。

アプリの基本的な機能を統合して、ユーザーが直接アプリを開かずにほとんどのタスクを実行できる中心的エントリ ポイントを提供することで、**Cortana** はアプリとユーザーの仲介役となります。多くの場合、これによってユーザーの時間と労力を大幅に減らすことができます。

あるいは、Windows 10 には、アプリに直接統合できる 2 つの音声コンポーネントとして、音声認識と音声合成 (TTS: text-to-speech) が用意されています。詳しくは、「音声認識の設計ガイドライン」をご覧ください。

ジェスチャ

ジェスチャは、アプリケーションを制御または操作するための入力として認識される、なんらかの形式のユーザーとのインタラクションです。ジェスチャには、手を使って画面上の何かをターゲットにするだけの単純なものから、特定の学習されたパターンの動きをターゲットにしたり、体全体を使った連続的な動きの長いストレッチまで、さまざまな形式があります。 カスタム ジェスチャを設計するときは、その意味がロケールやカルチャによって異なる場合があるため、注意が必要です。

ゲームパッド/コントローラー

ゲームパッド/コントローラーは、通常はゲーム専用の高度に専門化されたデバイスです。ただし、基本的なキーボード入力をエミュレートするためにも使われ、キーボードとよく似た UI ナビゲーション エクスペリエンスを提供します。

複数の入力

人がお互いにコミュニケーションをとる際に音声とジェスチャを組み合わせて使うように、アプリの操作では、複数の種類とモードの入力を使用すると便利な場合があります。ただし、これら複合的な操作は混乱を招くこともあるため、できる限り直感的で自然である必要があります。

できるだけ多くのユーザーやデバイスに対応し、可能な限り多くの入力の種類 (ジェスチャ、音声、タッチ、タッチパッド、マウス、キーボード) と連携するようにアプリを設計すると、最大の柔軟性、操作性、ユーザーに対する支援が得られます。

ユニバーサル Windows プラットフォーム (UWP) アプリの コンテンツ デザインの基本

どのようなアプリでも、主な目的はコンテンツへのアクセスを提供することです。たとえば、写真編集アプリでは写真がコンテンツであり、旅行アプリでは地図と旅行の目的地に関する情報がコンテンツです。 ナビゲーション要素はコンテンツへのアクセスを提供します。コマンド要素はユーザーがコンテンツを操作できるようにし、コンテンツ要素は実際のコンテンツを表示します。

この記事では、3 つのコンテンツ シナリオでのコンテンツの設計に関する推奨事項を示します。

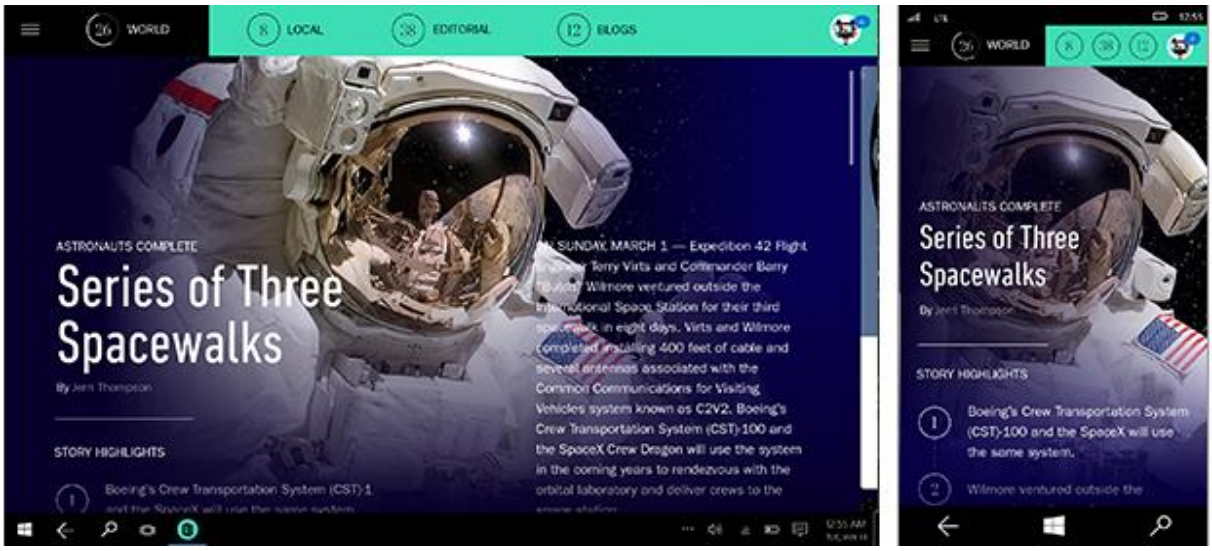
適切なコンテンツ シナリオの設計

次の 3 つの主要なコンテンツ シナリオがあります。

- **消費:** コンテンツを消費する、主に一方向のエクスペリエンス。消費は、読む、音楽を聴く、ビデオを見る、写真や画像を表示するなどのタスクです。
- **創造:** 新しいコンテンツの作成が焦点となる、主に一方向のエクスペリエンス。創造は、写真やビデオの撮影のように何かをゼロから作る、描画アプリで新しい画像を作る、新しいドキュメントを開くなどに分かれます。
- **インタラクティブ:** コンテンツの消費、作成、修正を含む、双方向のコンテンツ エクスペリエンス。

「消費」を中心としたアプリ

消費を中心としたアプリでは、コンテンツ要素の優先順位が最も高く、ユーザーが目的のコンテンツを探すために必要なナビゲーション要素が次に続きます。消費を中心としたアプリの例として、ムービー プレーヤー、読書アプリ、音楽アプリ、フォト ビューアーなどがあります。



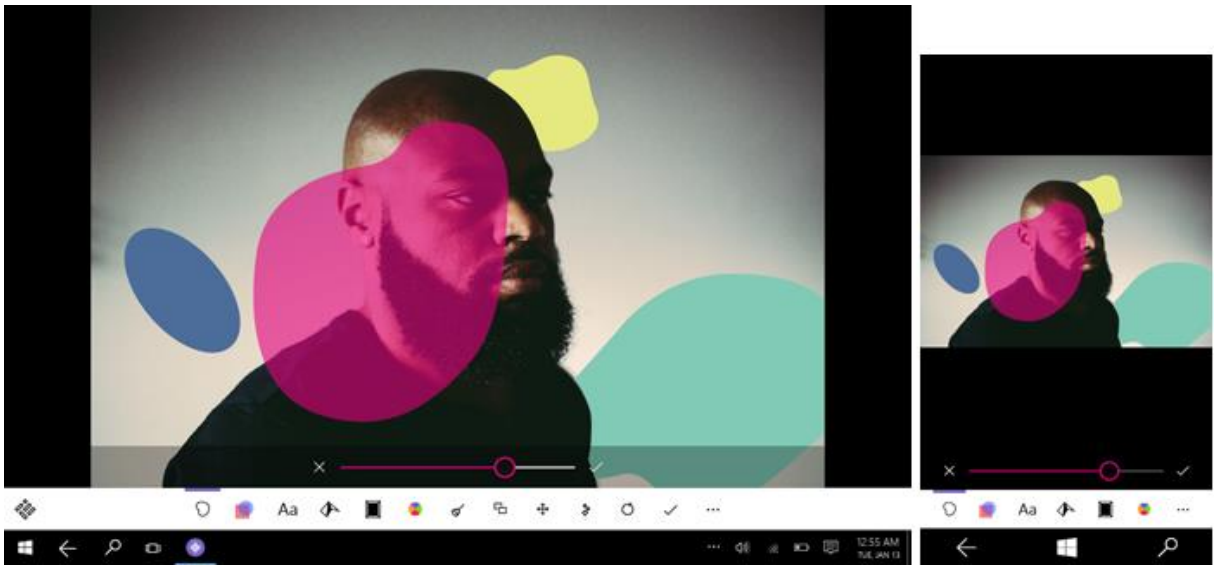
消費を中心としたアプリに関する一般的な推奨事項:

- 専用のナビゲーション ページとコンテンツ表示ページを作成し、ユーザーが探していたコンテンツを見つけたときに、無駄な情報がない専用のページでそのコンテンツを表示できるようにすることを検討します。
- コンテンツを画面全体に拡張し、その他のすべての UI 要素を非表示にする、全画面表示のオプションを作成することを検討します。

「創造」を中心としたアプリ

コンテンツとコマンド要素は、創造を中心としたアプリでは最も重要な UI 要素です。コマンド要素により、ユーザーは新しいコンテンツを作成することができます。この例には、ペイント アプリ、写真編集アプリ、ビデオ編集アプリ、ワード プロセッシング アプリがあります。

例として、コマンド バーを使ってツールや写真操作オプションにアクセスできるようにするフォト アプリのデザインを以下に示します。すべてのコマンドがコマンド バーにあるため、アプリは画面のスペースのほとんどをそのコンテンツ (編集する写真) に充てることができます。

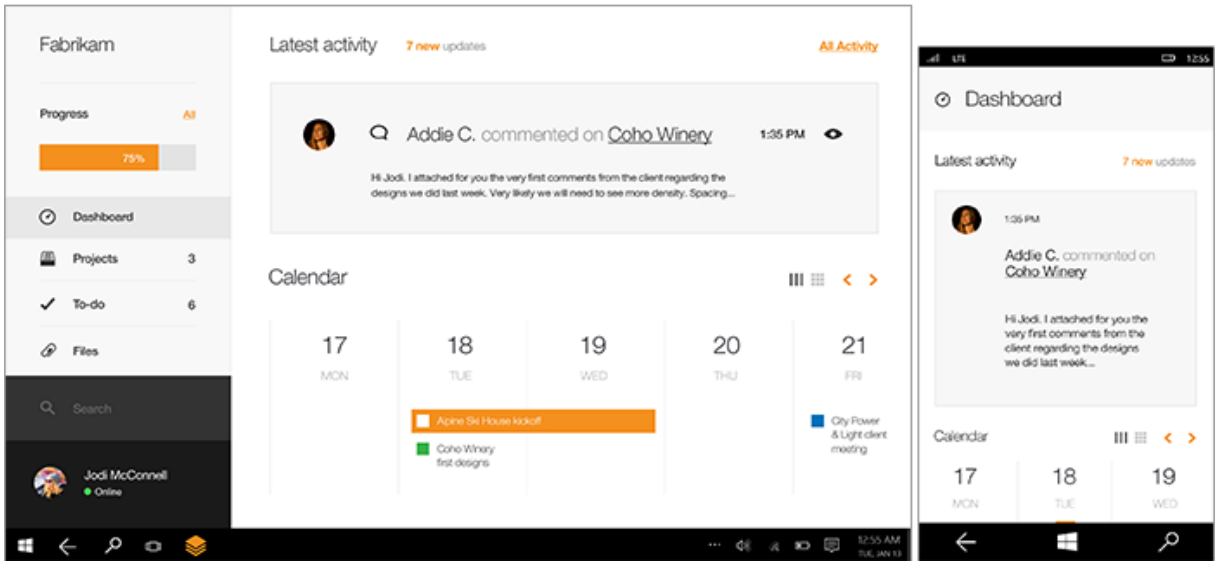


創造を中心としたアプリに関する一般的な推奨事項:

- ナビゲーション要素の使用を最小限に抑えます。
- コマンド要素は、作成に重点を置いたアプリで特に重要です。ユーザーは多くのコマンドを実行するため、コマンド履歴/元に戻す機能を提供することをお勧めします。

「インタラクティブ」なコンテンツを提供するアプリ

インタラクティブなコンテンツを提供するアプリでは、ユーザーはコンテンツを作成、表示、編集します。多くのアプリはこのカテゴリに分類されます。これらの種類のアプリの例には、基幹業務アプリ、在庫管理アプリ、ユーザーがレシピを作成または変更できる料理のアプリなどがあります。



これらの種類のアプリでは、次の 3 つすべての UI 要素のバランスを取る必要があります。

- ナビゲーション要素を使用すると、ユーザーがコンテンツを見つけて表示しやすくなります。コンテンツの表示と検索が最も重要なシナリオである場合は、ナビゲーション要素、フィルター処理と並べ替え、検索を優先します。
- コマンド要素により、ユーザーはコンテンツを作成、編集、操作することができます。

- インタラクティブなコンテンツを提供するアプリに関する一般的な推奨事項:3 つすべてが重要であるときに、ナビゲーション、コンテンツ、およびコマンド要素のバランスを取ることは困難である場合があります。可能であれば、コンテンツの閲覧、作成、編集用の別の画面を作成するか、モード スイッチを提供することを検討します。

よく使われるコンテンツ要素

一般的に使われるコンテンツ上の UI 要素を扱います (たとえば、UI 要素のリストなどえす。詳しくは、コントロールとパターンのガイドラインを参照してください)。

カテゴリ	要素	説明
オーディオとビデオ	メディア トランスポート コントロール **MediaTransportControls(XAML)**、**audio(HTML)**、**video(HTML)**	オーディオとビデオを再生します。
画像ビューアー	**FileView**、**Image(XAML)**、**img(HTML)**	画像を表示します。FlipView は、コレクション内の画像 (アルバム内の写真や製品の詳細ページ内の項目など) を一度に 1 つずつ表示します。
リスト	**ドロップダウン リスト、リスト ボックス、リスト ビューとグリッド ビュー**	対話型のリストまたはグリッド内に項目を表示します。これらの要素を使うと、ユーザーは新着の一覧からムービーを選んだり、在庫を管理したりすることができます。
テキストとテキスト入力	XAML: **TextBloxk**、**TextBox**、**RichEditBox** HTML: ほぼすべての HTML 要素をテキストの表示、または編集に使うことができます。	テキストを表示します。 一部の要素を使うと、ユーザーがテキストを編集することができます。詳しくは、「テキストとテキスト入力のガイドライン」をご覧ください。

UX ガイドライン

この索引で示されているガイドラインは、優れた UWP アプリのデザインに役立ちます。 優れた UWP アプリについてまだ十分に明確でない場合は、「UWP アプリとは」を先にお読みください。

アニメーション

目的がはっきりし、適切にデザインされたアニメーションは、アプリを生き生きとさせ、精巧で洗練された印象を与えます。ビジュアルな切り替えとコンテキストの変更がユーザーのエクスペリエンスに結びつきます。

アニメーションの利点

アニメーションは、単にものを動かすだけではありません。アニメーションはユーザーがアプリに入り込み、タッチを通じて操作するための物理的なエコシステムを作るツールです。エクスペリエンスの品質は、ユーザーに対するアプリの反応の良さと、UI が伝える個性の種類によって変わります。

アニメーションがアプリで目的を果たしていることを確認します。優れたユニバーサル Windows プラットフォーム (UWP) アプリでは、アニメーションを使って UI を生き生きさせています。アニメーションは次の条件を満たす必要があります。

- ユーザーの動作に基づいてフィードバックを提供する。
- UI を操作する方法をユーザーに伝える。
- 前または次のビューに移動する方法を示す。

ユーザーがアプリ内で費やす時間が増えたり、アプリのタスクが高度になると、高品質なアニメーションの重要性が増します。ユーザーが認知的負荷とアプリの使いやすさを感じる方法を変えるために使うことができます。アニメーションには、他にも多くの直接的なメリットがあります。

- アニメーションは、操作のヒントを与えます。

アニメーションには方向があり、前後に動いたり、コンテンツの内外に動いたりし、ユーザーが現在のビューまでどのように到達したかに関する最小限の "階層リンク" のような手掛かりを残します。

- アニメーションは、パフォーマンスが向上した印象を与えます。
  ネットワークが遅延したり、システムが動作を一時停止したとき、アニメーションによってユーザーが感じる待ち時間を短くできます。
- アニメーションは、個性を加えます。
  よく考えられた Windows Phone UI では、アニメーションを使って、アプリが今この場に関心を持っているという印象が生み出され、ユーザーが忘れられたり後回しにされているという印象を和らげています。
- アニメーションは、一貫性を高めます。
  切り替えによって、ユーザーは既によく知っているタスクとの類似性を引き出して新しいアプリケーションの操作方法を学習できます。
- アニメーションは、洗練された印象を高めます。
  アニメーションを使うと、ユーザーは電話がフリーズしているのではなく処理中であることを知ることができ、ユーザーが関心を持つ可能性のある新しい情報を受動的に表示することができます。

このセクションの内容

トピック	説明
追加と削除	リスト アニメーションを使うと、写真のアルバムや検索結果の一覧などのコレクションに対して任意の数の項目を挿入または削除できます。
コンテンツ切り替え	コンテンツ切り替えアニメーションを使うと、コンテナーや背景はそのままに、画面のある領域のコンテンツを変更できます。新しいコンテンツはフェード インします。既にあるコンテンツを差し替える場合、そのコンテンツはフェード アウトします。
ドラッグ	ドラッグ アンド ドロップ アニメーションは、リスト内で項目を移動するときや、特定の項目を別の項目上にドロップするときなど、オブジェクトを移動する際に使います。
エッジに基づく UI アニメーション	エッジに基づく UI アニメーションでは、画面の端を起点とする UI の表示と非表示を切り替えられます。

フェード	フェード アニメーションは、項目を画面に表示したり、項目を画面から非表示にするときに使います。フェード アニメーションには、フェード イン、フェード アウト、クロスフェードの 3 種類があります。
ページ切り替え	ページ切り替えアニメーションを使うと、新たに起動したアプリの最初のページを表示したり、アプリ内でページを切り替えたりできます。
ポインター クリック	ポインター アニメーションを使って、項目のタップまたはクリックに対するビジュアルなフィードバックをユーザーに提供します。ユーザーが項目をタップまたはクリックすると、ポインター ダウン アニメーションが再生されます。このアニメーションでは、項目が若干縮小され、押されていることを示します。クリックまたはタップが解放されると、ポインター アップ アニメーションが再生されます。このアニメーションでは、項目が元のサイズに戻り、解放されたことを示します。
ポップアップ	ポップアップ アニメーションを使って、コンテキスト メニューやフライアウトなど、ポップアップ UI の表示と非表示を切り替えます。ポップアップ要素とは、アプリのコンテンツの上に表示されるコンテナーのことで、ユーザーがポップアップ要素の外部をタップまたはクリックすると消えます。
位置変更	位置の変更アニメーションを使って、1 つまたは複数の要素を新しい位置に移動します。

追加と削除のアニメーションのガイドライン

リスト アニメーションを使うと、写真のアルバムや検索結果の一覧などのコレクションに対して任意の数の項目を挿入または削除できます。

重要な API

AddDeleteThemeTransition クラス (XAML)	createAddToListAnimation 関数 (WinJS)
	createDeleteFromListAnimation 関数 (WinJS)

推奨と非推奨

- リスト アニメーションは、既にある一連の項目に新しい項目を 1 つ追加するときに使います。たとえば、新しい電子メールを受け取ったときや、既にあるセットに新しい写真をインポートするときに使います。
- リスト アニメーションは、一連の項目に対して複数の項目を一度に追加するときに使います。たとえば、一連の新しい写真を既にあるコレクションにインポートするときに使います。複数項目の追加と削除は、個々のオブジェクトの処理に間が生じることなく同時に実行されます。
- 追加と削除のリスト アニメーションは、ペアで使います。一方のアニメーションを使った場合は、逆の操作として対応するもう一方のアニメーションを使うようにしてください。
- リスト アニメーションは、要素または要素のグループを一度に追加または削除できる項目リストに使います。
- コンテナーを表示したり非表示にしたりする目的でリスト アニメーションを使うのは避けてください。リスト アニメーションは、既に表示されているコレクションまたはセットのメンバーに対して使います。アプリ サーフェス上に一時的なコンテナーを表示したり非表示にしたりするには、フライアウト アニメーションを使います。アプリ サーフェスの一部となっているコンテナーを表示したり置き換えたりするには、コンテンツ切り替えアニメーションを使います。
- 項目のセット全体に対してリスト アニメーションを使うのは避けてください。コンテナー内のコレクション全体を追加したり削除したりするには、コンテンツ切り替えアニメーションを使います。

コンテンツ切り替えアニメーションのガイドライン

コンテンツ切り替えアニメーションを使うと、コンテナーや背景はそのままに、画面のある領域のコンテンツを変更できます。新しいコンテンツはフェード インします。既にあるコンテンツを差し替える場合、そのコンテンツはフェード アウトします。

重要な API

ContentThemeTransition クラス (XAML)	enterContent 関数 (WinJS)
	exitContent 関数 (WinJS)

推奨と非推奨

- 開始アニメーションは、空のコンテナーに一連の新しい項目を流し込むときに使います。たとえば、アプリの初期読み込みの直後は、アプリのコンテンツの一部が表示に間に合わない場合があります。このような場合、コンテンツを表示する準備が整った段階で、コンテンツ切り替えアニメーションを使い、遅れてコンテンツが表示されるようにします。
- あるコンテンツの組み合わせを、画面内の同じコンテナー内に既に存在する別のコンテンツの組み合わせに置き換えるときに、コンテンツ切り替えを使います。
- 新しいコンテンツを画面に表示するときには、一般的なページのフロー (読む方向) とは逆の方向にコンテンツをスライドさせます。たとえば、左から右に読むドキュメントに対して新しいコンテンツを流し込むアニメーションの場合、新しいコンテンツは右から左に流し込むのが自然です。
- 新しいコンテンツは論理的な流れで配置します。たとえば、最も重要なコンテンツを最後にします。
- 更新対象のコンテンツを含んだコンテナーが複数存在する場合、切り替え効果アニメーションは、間を置かずにすべて同時にトリガーします。
- コンテンツ切り替えアニメーションは、ページの全体が変化する場合には使わないでください。この場合には、ページ切り替えアニメーションを使います。
- コンテンツの更新のみであれば、コンテンツ切り替えアニメーションは使わないでください。コンテンツ切り替えアニメーションは、動きを表現するために使います。更新には、フェード アニメーションを使ってください。

ドラッグ アニメーションのガイドライン

ドラッグ アンド ドロップ アニメーションは、リスト内で項目を移動するときや、特定の項目を別の項目上にドロップするときなど、オブジェクトを移動する際に使います。

重要な API

DragItemThemeAnimation クラス (XAML)

dragSourceEnd 関数 (WinJS)

dragSourceStart 関数 (WinJS)

推奨と非推奨

ドラッグ開始アニメーション

- ドラッグの開始アニメーションは、ユーザーがオブジェクトを動かし始めるときに使います。
- ドラッグ アンド ドロップ操作の影響を受けるオブジェクトが他に存在する場合に限り、それらのオブジェクトをアニメーションに含めるようにします。
- ドラッグの開始アニメーションによって始まったアニメーションのシーケンスの終了には、ドラッグの終了アニメーションを使います。ドラッグの終了アニメーションにより、ドラッグの開始アニメーションで変化したドラッグされたオブジェクトのサイズが元に戻ります。

ドラッグの終了アニメーション

- ドラッグの終了アニメーションは、ドラッグされたオブジェクトをドロップするときに使います。
- ドラッグの終了アニメーションは、リストの追加および削除アニメーションと組み合わせて使います。
- ドラッグの開始アニメーションに影響を受けるオブジェクトが存在する場合に限り、それらのオブジェクトをドラッグの終了アニメーションに含めるようにします。

- ドラッグの終了アニメーションは、ドラッグの開始アニメーションよりも先に使わないでください。ドラッグ シーケンスの完了後にオブジェクトを元のサイズに戻すためには、両方のアニメーションを使う必要があります。

項目間でのドラッグの開始アニメーション

- 項目間へのドラッグの開始アニメーションは、2 つのオブジェクトの間のドロップ可能な場所にドラッグ ソースをドラッグするときに使います。
- 適度な大きさのドロップ ターゲット領域を選んでください。この領域が小さすぎると、ドラッグ ソースをドロップする際に重ね合わせるのが難しくなるため、好ましくありません。
- ドロップ可能な場所を示すために影響を受けるオブジェクトが移動するときには、互いにまっすぐに引き離すことをお勧めします。移動方向が上下になるか、左右になるかは、影響を受けるオブジェクトが並ぶ向きによって異なります。
- ドラッグ ソースを領域内にドロップできない場合、項目間でのドラッグの開始アニメーションは使わないでください。項目間へのドラッグの開始アニメーションは、影響を受けるオブジェクトの間にドラッグ ソースをドラッグできることをユーザーに知らせるためのものです。

項目間でのドラッグの中止アニメーション

- 項目間へのドラッグの中止アニメーションは、ユーザーがオブジェクトをドラッグして 2 つのオブジェクトの間のドロップ可能な領域から出すときに使います。
- 項目間でのドラッグの開始アニメーションよりも先に、項目間でのドラッグの中止アニメーションを使わないでください。

エッジに基づく UI アニメーションのガイドライン

エッジに基づく UI のアニメーションでは、画面のエッジ (端) を起点とする UI の表示と非表示を切り替えられます。 この表示と非表示のアクションは、ユーザーが開始することも、アプリから開始することもできます。UI は、アプリの手前に表示するか、メイン アプリ サーフェスの一部として表示することができます。UI をアプリ サーフェスの一部として

表示する場合は、UI を表示できるようにアプリの残りの部分のサイズを調整する必要があります。

重要な API

EdgeUIThemeTransition クラス (XAML)

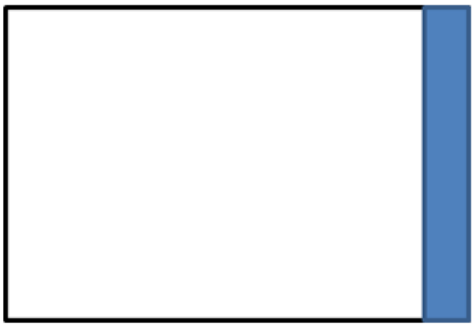
showEdgeUI 関数 (WinJS)

hideEdgeUI 関数 (WinJS)

推奨と非推奨

- 画面領域をあまり占有しないカスタム メッセージ バーやエラー バーを表示または非表示にするには、エッジ (端) UI アニメーションを使います。
- 作業ウィンドウやカスタム ソフト キーボードなど、画面内側にスライドして領域を大きく確保する UI を表示するには、パネル アニメーションを使います。
- UI を開くには、それが関連付けられているエッジ (端) から画面内側にスライドします。
- UI を閉じるには、画面内側から、開いたときと同じエッジ (端) に向かってスライドします。
- UI のスライド操作に応じてアプリのコンテンツ サイズを変更する必要がある場合は、フェード アニメーションを使ってサイズを変更します。
    - UI を画面内側に向かってスライドする場合は、エッジ (端) UI アニメーションまたはパネル アニメーションの後にフェード アニメーションを使います。
    - UI を画面外側に向かってスライドする場合は、エッジ (端) UI アニメーションまたはパネル アニメーションと同時にフェード アニメーションを使います。

- 通知には、このアニメーションを適用しないでください。エッジに基づく UI に通知を格納することはお勧めしません。
- 画面のエッジ (端)にない UI コンテナーやコントロールには、エッジ (端) UI アニメーションとパネル アニメーションを適用しないでください。このアニメーションは、画面のエッジ (端)にある UI の開閉とサイズ変更にのみ使います。他のタイプの UI を移動するには、位置変更アニメーションを使います。



Use edge UI animations

Use reposition animation

フェード アニメーションのガイドライン

フェード アニメーションは、項目を画面に表示したり、項目を画面から非表示にするときに使います。フェード アニメーションには、フェード イン、フェード アウトの 2 種類があります。

重要な API

FadeInThemeAnimation クラス (XAML)　fadeIn 関数 (WinJS)

FadeOutThemeAnimation クラス (XAML)

推奨と非推奨

- アプリで互いに関係のない要素や、テキストの多い要素を切り替えるときには、フェード アウトとフェード インを使います。そうすることで、差し替え前のオブジェクトが完全に消えてから差し替え後のオブジェクトを表示させることができます。
- 差し替える 2 つの要素のサイズが一定であり、ユーザーに同じ項目を見ているような印象を与えたいときには、差し替え後の要素を差し替え前の要素の上にフェード インさせます。フェード インが完了したら、差し替え前の項目は消すことができます。これは、差し替え後の項目が差し替え前の項目を完全に覆い隠せる場合にのみ可能な方法です。

- リストの項目を追加または削除する目的でフェード アニメーションを使うのは避けてください。そのような目的には、専用に作成したリスト アニメーションを使います。
- フェード アニメーションは、ページの全コンテンツを変化させるときには使わないようにしてください。そのような目的には、専用に作成したページ切り替えアニメーションを使います。
- フェード アウトは要素を削除するための繊細な方法です。

ページ切り替えアニメーションのガイドライン

ページ切り替えアニメーションを使うと、新たに起動したアプリの最初のページを表示したり、アプリ内でページを切り替えたりできます。

注 スクリーンの一部の領域を占めるだけのコンテンツを切り替えるときには、ページ切り替えアニメーションではなく、コンテンツ切り替えアニメーションを使います。

重要な API

EntranceThemeTransition クラス (XAML)

enterPage 関数 (WinJS)

exitPage 関数 (WinJS)

推奨と非推奨

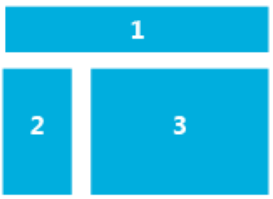
- ページは、自然な境界に沿って 2 ～ 5 の領域セットに分割してください。各領域に適用するタイミングをずらすと、領域が一度に全部ではなく、順番に表示されます。一般的な分割方法と分割の表示の順番については、「その他の使い方のガイダンス」をご覧ください。
- 差し替え前のページと差し替え後のページに共通するコンテンツがそのままの位置にとどまり、そのコンテンツにアニメーションが適用されていないことを確認します。たとえば、差し替え前のページと差し替え後のページのどちらにも **[戻る]** ボタンがある場合には、そのボタンは切り替えアニメーションの対象にしません。

- 差し替え前のページに **[戻る]** ボタンがなく (アプリの最初のページなど)、差し替え後のページにはあるという場合には、その **[戻る]** ボタンは別の領域に指定し、アニメーションを適用して他の領域よりも先に表示します。
- 差し替え前のページと差し替え後のページとの間で背景が異なる場合には、フェード アニメーションを使って新しい背景を表示します。ページ切り替えアニメーションと同時にフェード イン アニメーションを開始します。
- 差し替え後のページの一部のコンテンツが表示にすぐに表示できる状態にない場合には、ページ切り替えアニメーションを使うとその時点で準備ができているコンテンツが流し込まれます。その間に必要があれば、残りのコンテンツを準備している間に [プログレス コントロール] を表示します。残りのコンテンツの表示の準備が整ったら、コンテンツ領域に基づいてアニメーションを使って流し込みます。コンテンツ領域が大きい場合には、コンテンツ切り替えアニメーションを使います。コンテンツ領域が小さい場合や、連続性のないコンテンツの場合には、フェード イン アニメーションを使います。
- 画面に表示するページは、一般的なページのフロー (読む方向) とは逆の方向にスライドさせます。たとえば、差し替え後のページのコンテンツを読む方向が左から右の場合、新しいページは右から左にスライドするのが自然です。右から左に読むアプリの場合、新しいページは左から右にスライドするのが自然です。同じように、下の図に示すように 1 つのページを分割する場合、分割した部分を表示する順番は、読む方向と逆にするのが自然です。
- ユーザーがアプリ ウィンドウのサイズを変更するときには、ページ切り替えアニメーションを実行しないでください。ページ切り替えアニメーションは、特定のビューを使っているときにページ間で移動する場合に限られます。ビューが変わるときには、システムがレイアウトの切り替えのアニメーションを処理します。

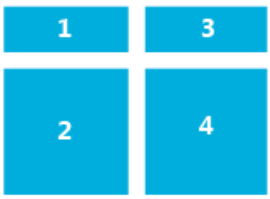
その他の使い方のガイダンス

ここでは、最もよくみられるページの分割方法について、表示の順番も含めて説明します。

2 stages: Header, content

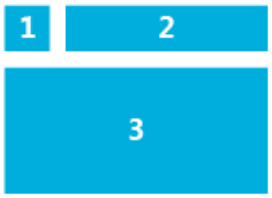


3 stages: Header, left content, right content

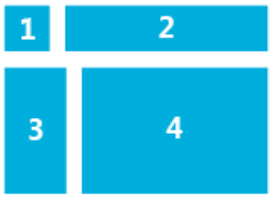


4 stages: Left header, left content, right header, right content

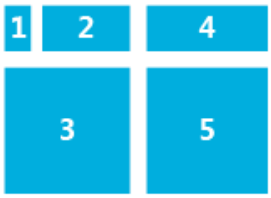
[戻る] ボタンのある一般的なページ分割を次に示します。差し替え前のページに **[戻る]** ボタンがなく (アプリの最初のページなど)、差し替え後のページにはあるという場合には、その **[戻る]** ボタンは別の領域に指定し、アニメーションを適用して他の領域よりも先に表示します。



3 stages: Back button, header, content



4 stages: Back button, header, left content, right content



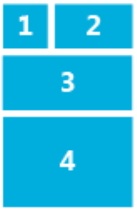
5 stages: Back button, left header, left content, right header, right content

ここでは、狭い幅のビューまたは縦向きビューで表示されるアプリに使うページ分割方法のうち最もよく見られるものについて、分割の表示の順番も含めて説明します。差し替え前のページに既に **[戻る]** ボタンがある場合、そのボタンは切り替えアニメーションの対象にせず、そのままの位置を維持します。

2 stages: Header, content

3 stages: Header, top content, bottom content

3 stages: Back button, header, content



4 stages: Back button, header, top content, bottom content

ポインター クリック アニメーションのガイドライン

ポインター アニメーションを使って、項目のタップまたはクリックに対するビジュアルなフィードバックをユーザーに提供します。ポインター ダウン アニメーション (押された項目を若干縮小して傾ける) は、項目が最初にタップされたときに再生されます。 ポインター アップ アニメーション (項目を元の位置に復元する) は、ユーザーがポインターから指を離したときに再生されます

重要な API

PointerUpThemeAnimation クラス (XAML)　PointerDownThemeAnimation クラス (XAML)

推奨と非推奨

- ポインター アップ アニメーションを使うときには、タップまたはクリックして指を離した直後にアニメーションを開始するようにします。これにより、タップまたはクリックによってトリガーされたアクション (新しいページへの移動など) の応答が遅れたとしても、ユーザーの操作が認識されたというフィードバックを即座に返すことができます。

ポップアップ UI アニメーションのガイドライン

ポップアップ アニメーションを使って、コンテキスト メニューやフライアウトなど、ポップアップ UI の表示と非表示を切り替えます。ポップアップ要素とは、アプリのコンテンツの上に表示されるコンテナーのことで、ユーザーがポップアップ要素の外部をタップまたはクリックすると消えます。

重要な API

PopInThemeTransition クラス (XAML)　showPopup 関数 (WinJS)

PopupThemeTransition クラス (XAML)

推奨事項

- ポップアップ アニメーションは、アプリのページに含まれない、コンテキスト メニュー、フライアウト、その他のコンテキスト対応 UI などのカスタム ポップアップ UI 要素を表示または非表示にするときに使います。 Windows で用意されているコモン コントロールには、既にこのアニメーションが組み込まれています。
- ツールチップやダイアログにポップアップ アニメーションを使わないでください。
    - アプリのメイン コンテンツの UI を表示または非表示にするときにはポップアップ アニメーションを使わないでください。ポップアップ アニメーションは、メイン アプリ コンテンツの上に表示するポップアップ コンテナーの表示と非表示を切り替えるときにのみ使います。

位置変更 アニメーションのガイドライン

位置の変更アニメーションを使って、1 つまたは複数の要素を新しい位置に移動します。

重要な API

RepositionThemeTransition クラス (XAML)　　createRepositionAnimation 関数 (WinJS)

RepositionThemeAnimation クラス (XAML)

推奨と非推奨

- 位置変更アニメーションは、エッジに基づく UI を表示したり非表示にしたりするときには使わないでください。エッジに基づく UI とは、画面のエッジ （端）の 1 つに固定された要素またはコンテナーを指します。

アプリ設定とアプリ データのガイドライン

このセクションでは、アプリ設定を示し、その設定をアプリ データとして格納するためのユーザー エクスペリエンス ガイドラインを取り上げます。

アプリ設定は、ユニバーサル Windows プラットフォーム (UWP) アプリの中で、ユーザーによるカスタマイズが可能な部分です。たとえば、ニュース リーダー アプリで、表示するニュース ソースや画面に表示する記事の数をユーザーが指定できるようにすることが考えられます。

アプリ データは、アプリ自体が作成して管理するデータです。アプリ データには、ランタイム状態、アプリ設定、参照コンテンツ (たとえば、辞書アプリの辞書定義)、その他の設定が含まれます。アプリ データはアプリの存在に関連付けられ、そのアプリに対して意味を持ちます。

このセクションの内容

トピック	説明
アプリの設定	このトピックでは、アプリ設定を作成し表示する際のベスト プラクティスについて説明します。
アプリ データのローミング	ローミング アプリ データが含まれるように Windows アプリを設計するときは、次のガイドラインに従ってください。

アプリ設定のガイドライン

アプリ設定は、アプリの中でユーザーによるカスタマイズが可能な部分です。たとえば、ニュース リーダー アプリで、表示するニュース ソースや画面に表示する記事の数をユーザーが指定できるようにすることが考えられます。この記事では、アプリ設定を作成し表示する際のベスト プラクティスについて説明します。

アプリにアプリ設定を含めるかどうか

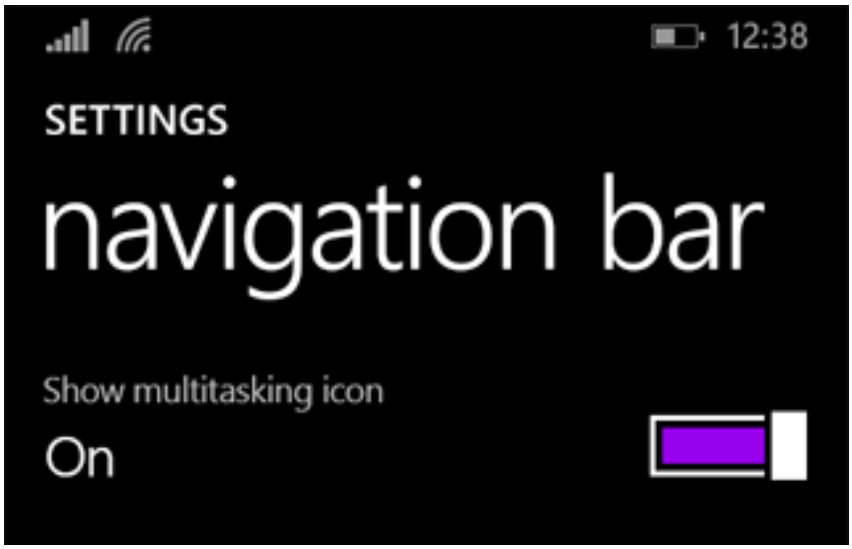
設定ページに含める設定には、次のようなものがあります。

- アプリの動作に影響するが、頻繁な再調整を必要としない構成オプション。たとえば、天気予報アプリで温度の既定の単位として摂氏または華氏を選択する機能、メール アプリでアカウント設定を変更する機能、アクセシビリティ オプションなどです。
- 音楽、効果音、配色テーマなど、ユーザーの設定に基づくオプション。
- プライバシー ポリシー、ヘルプ、アプリのバージョン、著作権情報など、頻繁にはアクセスされないアプリ情報。

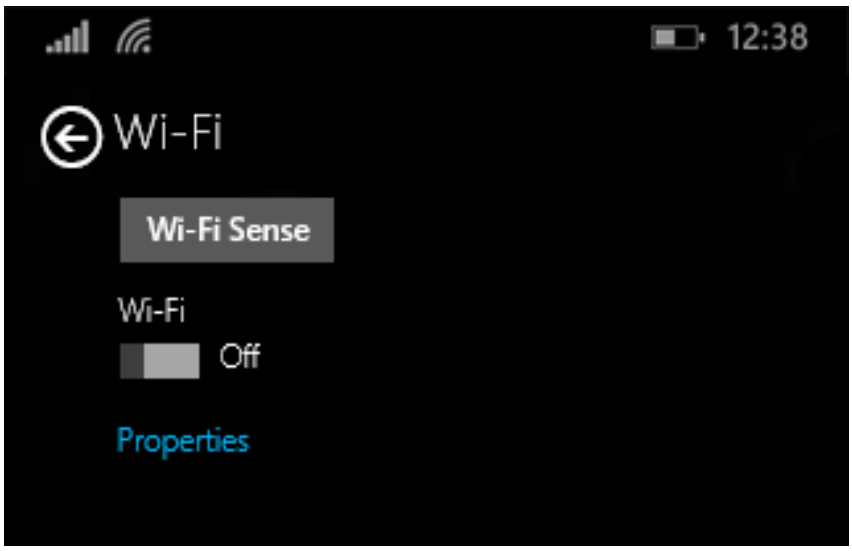
アプリの通常のワークフローに含まれるコマンド (お絵かきアプリでのブラシ色の変更など) は設定ページに含めません。コマンド配置について詳しくは、「コマンド デザインの基本」をご覧ください。

例

設定ページは簡潔にし、バイナリ (オン/オフ) コントロールを利用します。トグル スイッチを使った例を次に示します。



設定ページの例をもう 1 つ示します。この場合は、より詳細な設定へのリンク ("プロパティ") を使っています。



推奨と非推奨

一般的な原則

- アプリ設定ページに、すべてのアプリ設定のエントリ ポイントを作成します。
- 設定はシンプルにします。適切な既定値を定義し、設定の数は最小限にします。
- ユーザーが設定を変更したときは、アプリにすぐに変更が反映されるようにします。
- お絵かきアプリでのブラシ色の変更など、アプリの通常のワークフローに含まれるコマンドはアプリ設定に含めないでください (詳しくは、「コマンド デザインの基本」をご覧ください) 。

エントリ ポイント

エントリ ポイントとは、設定のページの上部に表示されるラベルです。含める設定の一覧ができたら、エントリ ポイントに関する次のガイドラインを考慮してください。

- 1 つのエントリ ポイントに類似したオプションや関連するオプションをまとめます。
- 設定ウィンドウに 4 つを超えるエントリ ポイントの追加は避けます。

- アプリのコンテキストに関係なく、同じエントリ ポイントを表示します。いくつかの設定が特定のコンテキストに適合しない場合は、アプリの設定フライアウトでそれらの設定を無効にします。
- エントリ ポイントのラベルは、できればわかりやすい 1 単語にします。たとえば、アカウント関連の設定の場合は、エントリの名前を "アカウント設定" ではなく "アカウント" にします。設定のエントリ ポイントが 1 つだけ必要だが、設定のわかりやすいラベルが思い付かない場合は、"オプション" または "既定" を使います。
- エントリ ポイントがフライアウトではなく直接 Web に移動する場合は、ハイパーリンクとしてスタイルを設定した "ヘルプ (オンライン)" や "Web フォーラム" など、ユーザーにビジュアルなヒントを与えます。Web への複数のリンクは、1 つのエントリ ポイントを使ってフライアウトにまとめることを検討してください。たとえば、"バージョン情報" エントリ ポイントでは、使用条件、プライバシーに関する声明、アプリのサポートへのリンクを含むフライアウトが開くようにします。
- 使用頻度の高い設定にそれぞれ独自のエントリ ポイントを割り当てられるように、使用頻度の低い設定は 1 つのエントリ ポイントにまとめます。主に情報提供のみを行うコンテンツやリンクは、"アプリについて" エントリ ポイントに配置します。
    - [アクセス許可] ウィンドウの機能と重複しないようにします。 このウィンドウは既定で用意されており、その内容を変更することはできません。

設定フライアウトへの設定コンテンツの追加

- コンテンツは 1 列で上から下へ表示し、必要に応じてスクロールできるようにします。スクロールの長さは画面の高さの 2 倍までに抑えます。
- アプリ設定では次のコントロールを使うことを検討します。
    - トグル スイッチ: ユーザーが値をオンまたはオフに設定できるようにする場合。
    - ラジオ ボタン : ユーザーが相互排他的な関連するオプション (5 個まで) の中から 1 つの項目を選択できるようにする場合。

- テキスト入力ボックス: ユーザーがテキストを入力できるようにする場合。ユーザーから取得するテキストの種類 (メール、パスワードなど) に応じた種類のテキスト入力ボックスを使います。
- ハイパーリンク: アプリ内の別のページや外部 Web サイトに移動する場合。ユーザーがハイパー リンクをクリックすると、現在の設定 UI は閉じられます。
- ボタン: ユーザーが現在の設定フライアウトを閉じることなく即座に操作を開始できるようにする場合。

- 使用できないコントロールがある場合は、説明用のメッセージを追加します。使用できないコントロールの上に、このメッセージを配置します。
- 設定フライアウトとヘッダーがアニメーション化された後で、コンテンツとコントロールを単一のブロックとしてアニメーション化します。**enterPage** または **EntranceThemeTransition** アニメーションを使って、100 ピクセル左のオフセットでコンテンツをアニメーション化します。
- 必要に応じて、コンテンツの整理と明確化の助けになるように、セクション ヘッダー、段落、ラベルを使います。
- 設定を繰り返し表示する必要がある場合は、UI の階層を追加するか、展開/折りたたみモデルを使います。階層の深さは 2 階層までに抑えます。たとえば、天気予報アプリの都市別の設定では、都市の一覧を表示し、ユーザーが都市をタップしたときに、新しいフライアウトを開くか、展開して設定オプションを表示できるようにします。
- コントロールや Web コンテンツの読み込みに時間がかかる場合は、進行状況不定コントロールを使ってユーザーに読み込み中であることを示します。詳しくは、「プログレス コントロールのガイドライン」をご覧ください。
- 移動や変更を確定するためのボタンは使いません。別のページに移動するにはハイパーリンクを使います。また、ボタンを使って変更をコミットする代わりに、ユーザーが設定ポップアップを閉じたときにアプリ設定の変更を自動的に保存します。

アプリ データのローミングのガイドライン

ローミング **ApplicationData** API を使ってアプリのデータを保存すると、Windows によって、このデータはクラウドにレプリケートされ、アプリがインストールされているその他のすべてのユーザー デバイスに同期されます。ローミング アプリ データが含まれるように Windows アプリを設計するときは、次のガイドラインに従ってください。

重要な API

ApplicationData クラス

RoamingFolder プロパティ

RoamingSettings プロパティ

アプリでローミング データを使うかどうか

ローミング データを使ってユーザーの設定、基本設定、セッション情報を保存し、複数のデバイス間で統一感のあるアプリのエクスペリエンスを提供します。ローミング データがユーザーの Microsoft アカウントに関連付けられていることに注意してください。ローミング データが同期されるのは、ユーザーが同じ Microsoft アカウントを使ってデバイスにログインし、複数のデバイスにアプリをインストールしている場合のみです。

たとえば、ユーザーが別のデバイスで使っていたアプリを新しいデバイスにインストールした場合、最初のデバイスで使っていた設定や基本設定がすべて新しいデバイスに自動的に適用されます (ユーザーが同じ Microsoft アカウントを使って両方のデバイスにログインすることを前提としています)。それらの設定や基本設定に対する以降の変更もすべて自動的に移行されるため、デバイスに関係なく一貫したエクスペリエンスを得ることができます。セッション情報をローミング データとして保存すると、ユーザーは、あるデバイスで閉じたり中止したりしたアプリ セッションを、別のデバイスに切り替えた後に引き続き使うことができます。

注 **RoamingSettings**、**RoamingFolder**、**RoamingStorageQuota** の各プロパティは、Windows Phone 用には実装されていません。

注 このようなファイルは、**RoamingFolder** に配置されていてもローミングされません。

- フォルダーのように振る舞うファイルの種類。たとえば、拡張子が .zip や .cab のファイル

- 名前の先頭に空白のあるファイル
- 名前に次の Unicode 文字を含むファイル

  e794、e795、e796、e7c7、e816、e817、e818、e81e、e826、e82b、e82c、e831、e832、e83b、e843、e854、e855、e864、e7e2、e7e3、e7f3
- ファイル パス (ファイル名 + 拡張子) が 256 文字を超えるファイル
- 空のフォルダー
- 開いているハンドルがあるファイル

推奨事項

- ユーザーの基本設定やカスタマイズ、リンク、小さなデータ ファイルにローミングを使います。たとえば、ローミングを使って、ユーザーの背景色の基本設定をすべてのデバイスで保持します。
- ユーザーがデバイス間で作業を続けられるようにローミングを使います。たとえば、下書きしたメールの内容やリーダー アプリで最近表示したページなどのアプリ データをローミングします。
- アプリ データを更新して、**DataChanged** イベントを処理します。このイベントは、クラウドからのアプリ データの同期が完了したときに発生します。
- 生データではなくコンテンツへの参照をローミングします。たとえば、オンライン記事のコンテンツではなく URL をローミングします。
- タイム クリティカルな重要な設定に対しては、**RoamingSettings** に関連付けられた HighPriority 設定を使います。
- デバイス固有のアプリ データをローミングしないでください。ローカルにあるファイル リソースのパス名など、ローカルのみに関連した情報もあります。ローカル情報をローミングする場合は、その情報が別のデバイスで無効なときにアプリを回復できることを確認してください。
- 大量のアプリ データをローミングしないでください。アプリでローミングできるアプリ データの量には制限があります。この最大値を取得するには、**RoamingStorageQuota** プロパティを使ってください。 この制限に達した場合、アプリ データのサイズが制限を下回るまで、データはローミングできません。アプリを設計する際は、この制限を超えないようにサイズの大きいデータをどのように制限するかを検討してください。たとえば、ゲームの状態を保存するのにそれぞ

れ 10 KB 必要になる場合は、ユーザーによる保存を 10 ゲームまでに制限したりすると効果的です。

- 即時同期に依存するデータにローミングを使わないでください。Windows では即時同期が保証されません。ユーザーがオフラインであったり、待ち時間の長いネットワークを使っている場合、ローミングはかなり遅れる可能性があります。UI が即時同期に依存しないことを確認してください。
- 頻繁に変更されるデータにローミングを使わないでください。たとえば、再生中の曲の秒刻みの位置など、頻繁に変更される情報を追跡する場合は、この情報をローミング アプリ データとして保存しないでください。代わりに、現在再生中の曲など、変更の頻度が少なく、ユーザー エクスペリエンスも損なわないような情報を利用します。

その他の使い方のガイダンス

ローミングの前提条件

アプリ データのローミングは、Microsoft アカウントを使ってデバイスにログインするすべてのユーザーに利点をもたらします。ただし、いつでもデバイスでアプリ データのローミングを切り替えることができるのは、ユーザーとグループ ポリシーの管理者です。 ユーザーが Microsoft アカウントを使わない場合やデータのローミング機能を無効にする場合、ユーザーは引き続きアプリを使用できますが、アプリ データは各デバイスに対してローカルのままになります。

PasswordVault に格納されているデータは、ユーザーが "信頼" しているデバイスにしか移行されません。デバイスが信頼されていない場合、この資格情報コンテナーのセキュリティで確保されているデータはローミングされません。

競合の解決

アプリ データのローミングは、複数のデバイスでの同時使用を想定していません。 2 台のデバイスで特定のデータ単位が変更されたことが原因で同期中に競合が発生した場合、最後に書き込まれた値が常に優先されます。これにより、アプリで最新の情報が利用されます。

データ単位が設定コンポジットの場合、競合の解決は設定の単位で行われ、最新の変更を含むコンポジットが同期されます。

データを書き込むタイミング

想定される設定の有効期間に応じて、データを書き込むタイミングを変える必要があります。変更の頻度が低いアプリ データや変更間隔の長いアプリ データは、変更されたらすぐに書き込むようにします。 ただし、頻繁に変更されるアプリ データは、アプリが中断されたとき以外は、一定の間隔 (5 分に 1 回など) でのみ書き込むようにします。 たとえば、音楽アプリでは、"現在の曲" の設定は新しい曲の再生が始まるたびに書き込みますが、曲の途中の実際の位置は中断したときにのみ書き込みます。

使いすぎに対する保護

リソースの不適切な使用を防止するために、システムにはさまざまな保護メカニズムが備わっています。アプリ データが想定どおりに移行されない場合は、デバイスが一時的に制限されていることが考えられます。通常、この状況はしばらくすると自動的に解決されるため、操作は必要ありません。

バージョン

アプリ データは、バージョンに基づいてデータ構造をアップグレードできます。バージョン番号は、アプリのバージョンとは別の番号で、自由に設定することができます。強制ではありませんが、バージョン番号は新しいデータほど大きくすることを強くお勧めします。新しいデータを表すバージョン番号が小さくなると、データ損失などの望ましくない問題が発生する可能性があります。

アプリ データのローミングは、バージョン番号が同じインストールされたアプリの間でしか行われません。たとえば、どちらもバージョン 2 のデバイスの間やどちらもバージョン 3 のデバイスの間ではデータが移行されますが、バージョン 2 を実行中のデバイスとバージョン 3 を実行中のデバイスの間ではローミングは行われません。 他のデバイスでさまざまなバージョン番号を利用していたアプリを新たにインストールする場合、新たにインストールしたアプリは、最も大きいバージョン番号と関連付けられているアプリ データを同期します。

テストとツール

開発者は、ローミング アプリ データの同期をトリガーするためにデバイスをロックできます。 一定の期間にわたってアプリ データが移行されていない場合は、次の点を確認してください。

- ローミング データの最大サイズを超えていないこと (詳しくは、「**RoamingStorageQuota**」をご覧ください)。
- ファイルが閉じていて、適切に解放されていること。
- 同じバージョンのアプリを実行しているデバイスが 2 台以上あること。

コントロールとパターン

このセクションでは、ユニバーサル Windows プラットフォーム (UWP) アプリ コントロールとパターンの設計ガイドラインを、すばやく簡単に参照できるように 1 か所で示します。

- コントロールは、開発者用に作成された UI 要素です。それらの要素はそのまま利用できます。または、その外観をアプリのスタイルに合わせて調整することができます。用意されているコントロールを使うと、一般的なインタラクションを作成する際に時間を短縮できます。コントロールは、すべての種類の入力でそのまま利用できます。開発者と共同で作業する場合は、設計でこれらのコントロールを参照することによって認識を共有できます。コントロールに関する開発者情報については、「コントロールの一覧」を参照してください。
- パターンは、1 つまたは複数のコントロールを使って特定の種類の機能を提供するレシピです。

注　ここで紹介するすべてのコントロールは、Adobe Photoshop および Adobe Illustrator 用のテンプレートに含まれています。テンプレートの入手方法については、ダウンロードに関するページをご覧ください。

このセクションの内容

トピック	説明
アクティブなキャンバス	単一ビュー アプリやモーダル エクスペリエンス (フォト ビューアー/エディター、ドキュメント ビューアー、マップ、描画、自由スクロール ビューを利用する他のアプリなど) のコンテンツ領域とコマンド領域を持つパターンです。
オート サジェスト ボックス	ユーザーが入力するときに、検索候補を表示するテキスト入力ボックスです。
戻るボタン	戻るボタンは、バック スタックまたはユーザーのナビゲーション履歴を使って "戻る" ナビゲーションを実現する、システムの UI アフォーダンスです。
ボタン	ボタン (コマンド ボタン) は、特定の操作を直ちに実行する方法をユーザーに与えます。

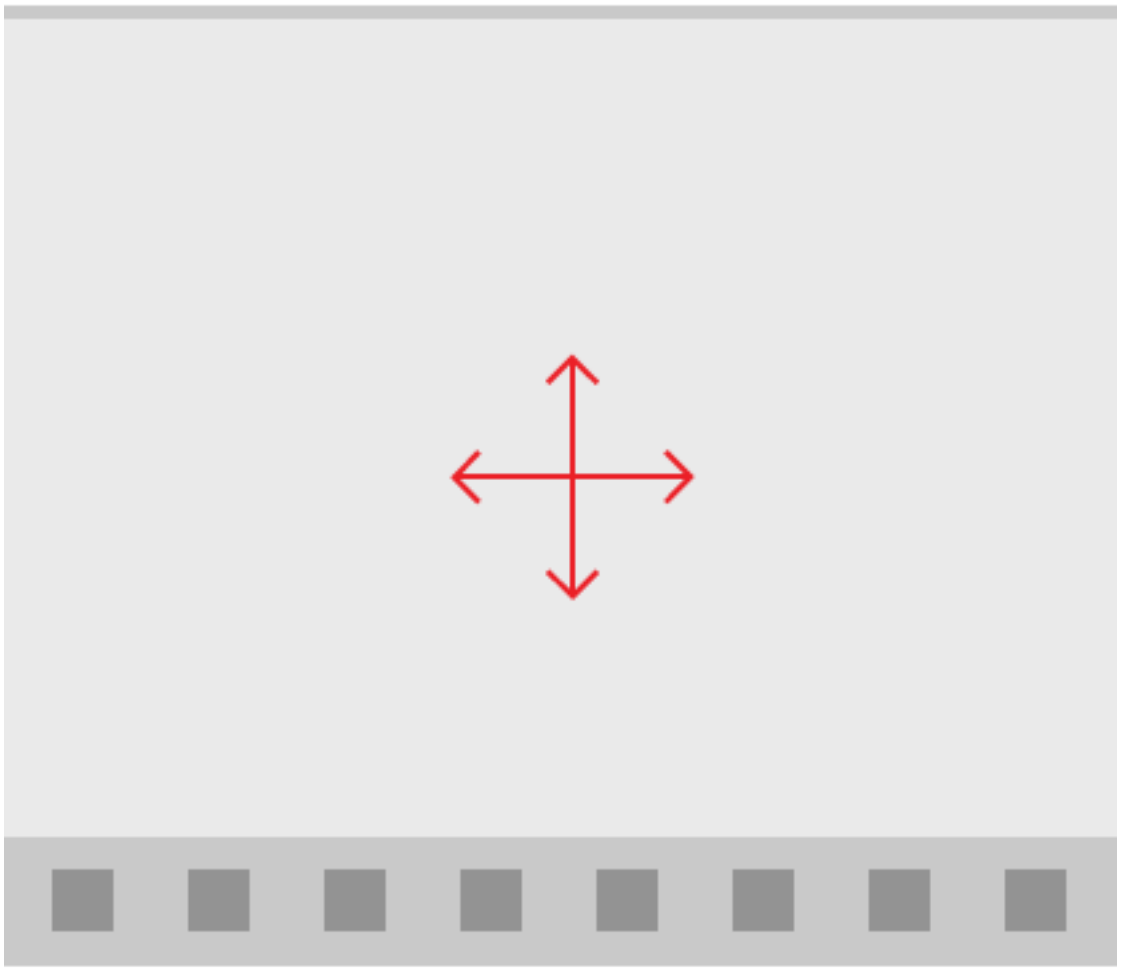
カメラ UI	カメラ キャプチャ UI ダイアログでは、カメラが内蔵または接続されたデバイスで組み込みカメラ UI を使って、ユーザーが写真やビデオをキャプチャすることができます。
チェックボックス	アクション項目の選択や選択解除を行うときに使います。単一のリスト項目や複数のリスト項目に対して使うことができます。
コマンドバー	コマンド バーを使うと、ユーザーは必要なコマンドに簡単にアクセスできます。コマンド バーは、ユーザーのコンテキストに固有のコマンドまたはオプション (写真の選択や描画モードなど) を表示するためにも使うことができます。
コンテキストメニュー	コンテキスト メニューには、ユーザーが現在のコンテキストで使うことのできるコマンドとオプションの一覧を表示します。
日付と時刻コントロール	日付と時刻のコントロールでは、日付と時刻を表示および設定できます。この記事では設計ガイドラインを示し、適切なコントロールを選ぶのに役立ちます。
ダイアログコントロール	ダイアログは、ユーザーに操作を求めるモーダル UI オーバーレイです。
フィルターと並び替え	フィルター コマンドと並べ替えコマンドを使うと、コンテンツの表示方法を変更できます。
FlipView	コレクション内の画像 (アルバム内の写真や製品の詳細ページ内の項目など) を一度に 1 つずつ表示します。
フライアウト	フライアウト (flyout) は、ユーザーが現在操作している内容に関係する UI を一時的に表示するために使われる軽量なポップアップです。
ハブ	ハブ コントロールは、階層型ナビゲーション パターンを使って、リレーショナル情報アーキテクチャを使ったアプリをサポートします。
ハイパーリンク	ハイパーリンクはユーザーを、アプリの別の部分、別のアプリ、または別のブラウザー アプリを使って呼び出した URI (Uniform Resource Identifier) に誘導します。
ラベル	ラベルは、コントロールまたは関連するコントロールのグループの名前やタイトルです。ラベルには機能は実装されません。
リスト	リストでは、コレクション ベースのアプリ コンテンツを、タッチに最適化された一貫した方法で表示して操作できます。

マスター/詳細	マスター/詳細パターンでは、マスター リストと、現在選択されている項目の詳細が表示されます。このパターンは、メールや連絡先一覧/アドレス帳によく使用されます。
メディア プレーヤー	ビデオ、オーディオ、および画像を表示したり聴いたりするには、メディア プレーヤーを使います。
ナビゲーション ウィンドウ	トップレベルのナビゲーションを提供して、画面領域を節約します。
プログレス	プログレス コントロールは、時間のかかる操作が進行中であることを示すフィードバックをユーザーに返します。
ラジオ ボタン	ラジオ ボタンでは、ユーザーは 2 つ以上の選択肢から 1 つのオプションを選ぶことができます。
評価	評価コントロールは、評価を示すアイコンをクリックすることで、ユーザーが何かを評価できるようにします。評価には、平均評価、暫定評価、ユーザー評価の 3 種類があります。
スクロール バー	パンとスクロールを行うと、画面の境界外のコンテンツを拡張表示することができます。
検索	検索は、ユーザーがアプリでコンテンツを見つけることができる 2 つの方法のうちの 1 つです。このガイダンスでは、検索エクスペリエンスの構成要素、検索スコープ、実装、コンテキストでの検索の例について説明します。
セマンティック ズーム	セマンティック ズーム コントロールを使うと、ユーザーは同じデータ セットの 2 つの異なるセマンティック表示間でズームを実行できるようになります。
スライダー	ユーザーが有効な範囲から値を設定できます。
SplitView	分割ビュー コントロールには、展開/折りたたみ可能なウィンドウとコンテンツ領域があります。コンテンツ領域は常に表示されます。ウィンドウは展開/折りたたみを行うことも、開いた状態のままにすることもでき、アプリ ウィンドウの右側または左側から表示できます。このウィンドウには 3 つのモードがあります。

タブとピボット	タブとピボットを使うと、アクセス頻度の高いコンテンツ間を移動できます。
トグル スイッチ	トグル スイッチは、ユーザーが項目をオンまたはオフに切り替えることができる物理的なスイッチを表します。
ツールチップ	ユーザーに操作の実行を指示する前に、ツールチップを使ってコントロールに関する詳しい情報を表示します。
Web ビュー	Web ビュー コントロールは、Internet Explorer のように動作するビューをアプリに組み込みます。また Web ビュー コントロールでは、ハイパーリンクの表示と動作が可能です。

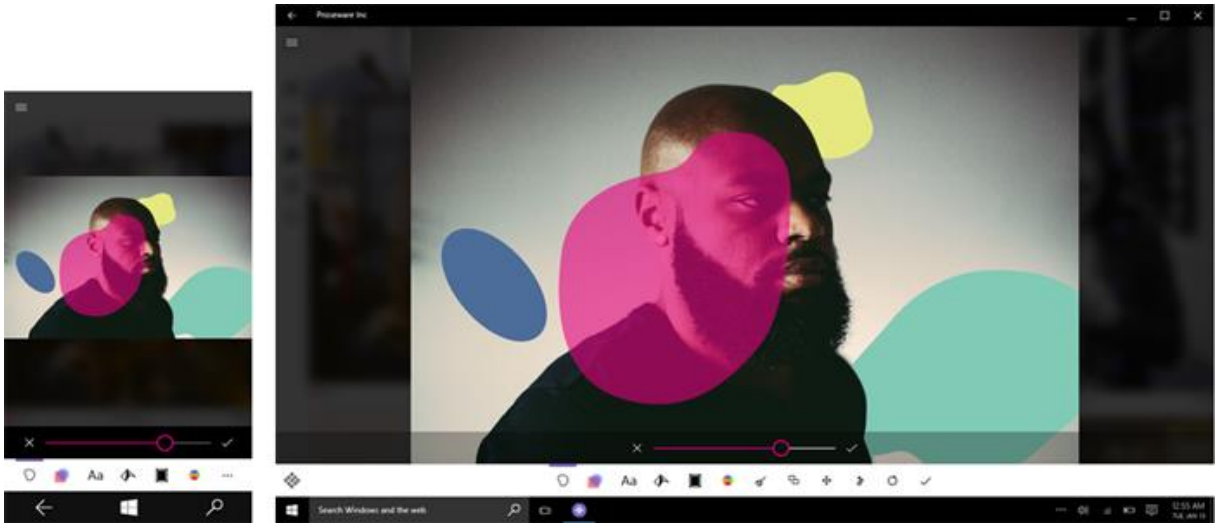
アクティブなキャンバス パターンに関するガイドライン

アクティブなキャンバスは、コンテンツ領域とコマンド領域を持つパターンです。これは、単一ビュー アプリやモーダル エクスペリエンス (フォト ビューアー/エディター、ドキュメント ビューアー、マップ、描画、自由スクロール ビューを利用する他のアプリなど) のパターンです。アクティブなキャンバスは操作を行うために、必要な操作の数と種類に応じて、コマンド バー、またはボタンのみとペアを構成できます。

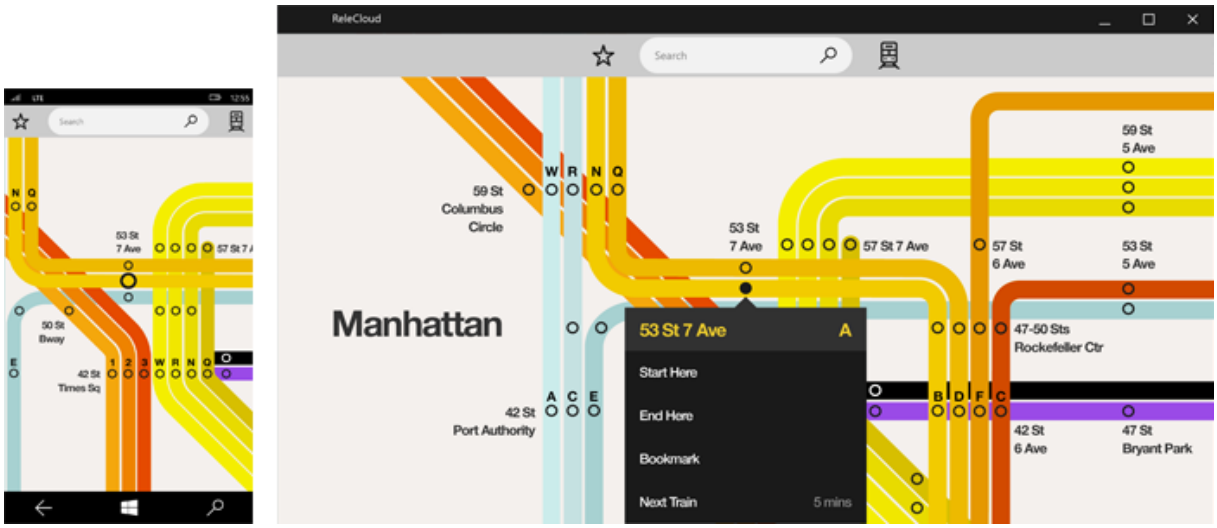


例

写真編集アプリのこのデザインはアクティブなキャンバス パターンが特徴となっています。左側にモバイルの例、右側にデスクトップの例を示します。イメージ編集サーフェスがキャンバスであり、下部にあるコマンド バーにはアプリのコンテキスト依存アクションがすべて含まれています。



地下鉄路線図アプリのこの設計では、2 つの操作と検索ボックスのみを含むシンプルな UI が上部にある、アクティブなキャンバスを利用しています。右の画像に示すように、状況依存の操作がコンテキスト メニューに表示されます。



このパターンの実装

アクティブなキャンバスは、コンテンツ領域とコマンド領域で構成されます。

コンテンツ領域。 コンテンツ領域は、通常は自由スクロール キャンバスです。複数のコンテンツ領域がアプリ内に存在できます。

コマンド領域。 多くのコマンドを配置する場合は、画面サイズに応じて反応するコマンド バーの使用が適切である可能性があります。それほど多くのコマンドを配置せず、応答性が高い UI が重要でない場合は、省スペース型ボタンを利用できます。

オート サジェスト ボックスのガイドライン

オート サジェスト ボックスは、基本的な検索候補の一覧をトリガーするテキスト入力ボックスです。候補となる用語は、一般的な検索用語とユーザーが入力した履歴の組み合わせから取得できます。

重要な API

AutoSuggestBox クラス (XAML)	AutoSuggestBox オブジェクト (HTML)

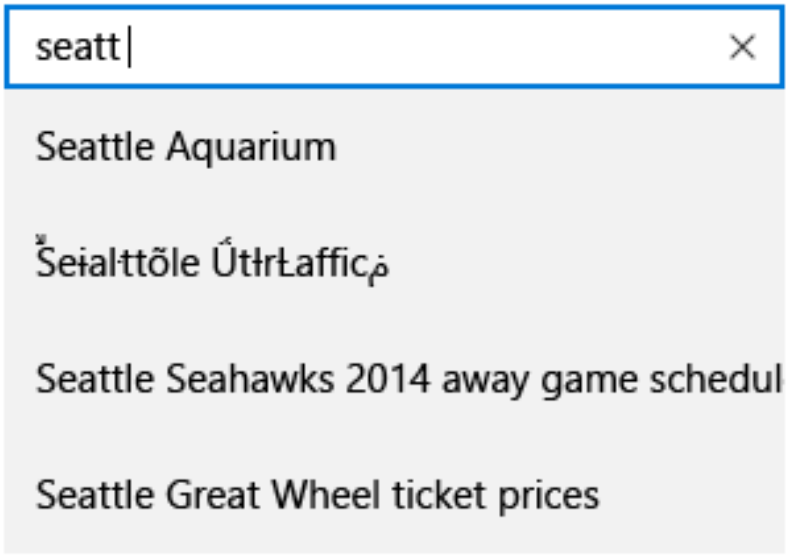
適切なコントロールの選択

候補の一覧を使ってテキストを検索できる、シンプルでカスタマイズ可能なコントロールが必要な場合に使います。

例

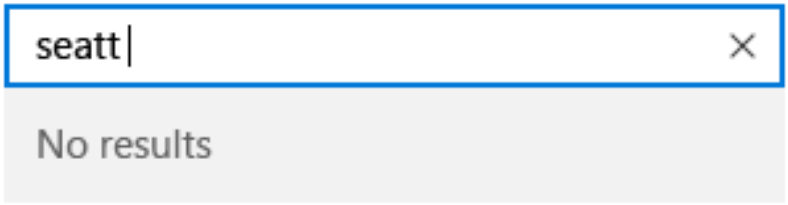
オート サジェスト ボックスの入力ポイントは、ヘッダー (XAML のみ) とヒント テキスト付きのボックスで構成されます。

自動的に提案した結果の一覧は、ユーザーがテキストの入力を開始すると自動的に内容が入力されます。結果の一覧は、テキスト入力ボックスの上または下に表示されます。[すべてクリア] ボタンも表示されます。



推奨事項

- 自動提案ボックスを使って検索を実行したときに、入力したテキストに対応する検索結果が存在しなかった場合は、"検索結果が見つかりませんでした" という 1 行を表示します。これにより、ユーザーは検索要求が実行されたことがわかります。



戻るボタンのガイドライン

戻るボタンは、バック スタックまたはユーザーのナビゲーション履歴を使って "戻る" ナビゲーションを実現する、システムの UI アフォーダンスです。

ナビゲーション履歴のスコープ (アプリ内かグローバルか) はデバイスとデバイス モードによって決まります。

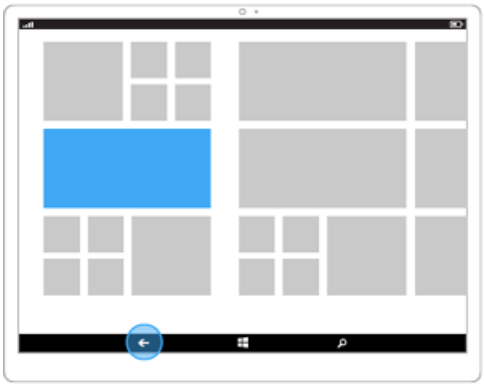
例

システムの戻るボタンの UI は、デバイスや入力の種類ごとに最適化されますが、ナビゲーション エクスペリエンスはグローバルであり、デバイスやユニバーサル Windows プラットフォーム (UWP) アプリで一貫しています。 これらの異なるエクスペリエンスには次のものがあります。

電話

- 常に表示されます。
- デバイスの下部にあるソフトウェアまたはハードウェア ボタン。
- アプリ内部やアプリ間で、戻るナビゲーションを実現します。

タブレット

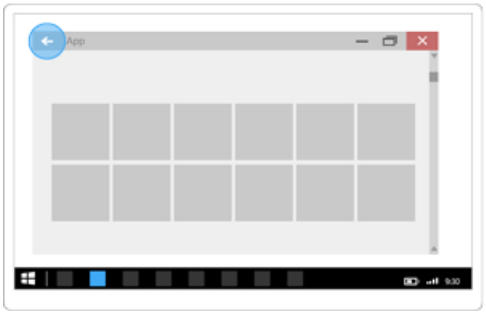


- タブレット モードでは、常に表示されます。
  デスクトップ モードでは利用できません。代わりに、タイトル バーの戻るボタンを有効にすることができます。
  ユーザーは、**[設定]**、**[システム]**、**[タブレット モード]** の順に選択するか、**[デバイスをタブレットとして使用すると、Windows のタッチ機能がより使いやすくなります]** をオンに

することによって、タブレット モードでの実行とデスクトップ モードでの実行を切り替えることができます。

- デバイスの下部のナビゲーション バーにあるソフトウェア ボタン 。
- アプリ内部やアプリ間で、グローバルな戻るナビゲーションを実現します。

PC、ノート PC



- デスクトップ モードはオプションです。
  タブレット モードでは利用できません。
  既定では無効になっています。有効にすることをオプトインする必要があります。
  ユーザーは、**[設定]、[システム]、[タブレット モード]** の順に選択するか、**[デバイスをタブレットとして使用すると、Windows のタッチ機能がより使いやすくなります]** をオンにすることによって、タブレット モードでの実行とデスクトップ モードでの実行を切り替えることができます。
- アプリのタイトル バーに表示されます。
- アプリ内部のみで、戻るナビゲーションを実現します。ア

プリ間のナビゲーションはサポートされません。

Surface Hub

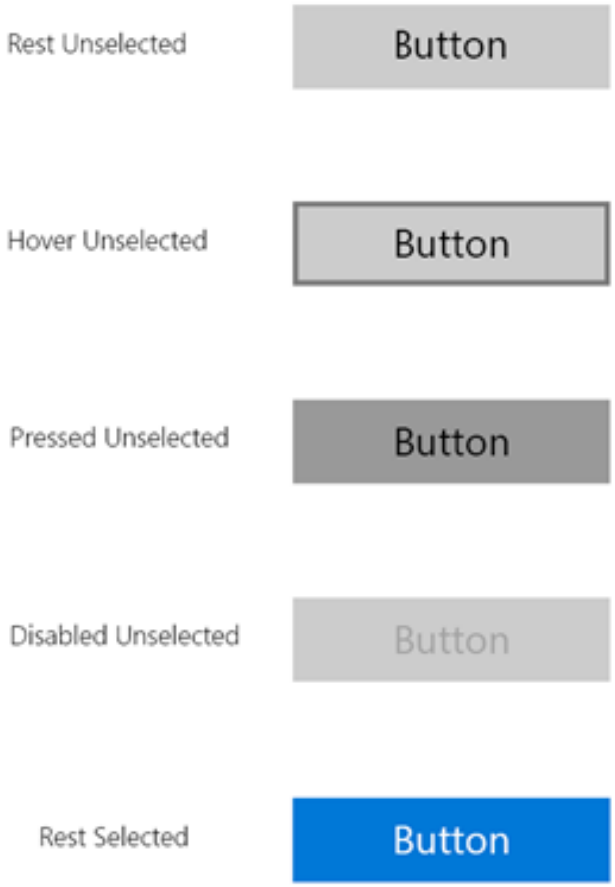
- 常に表示されます。
- デバイスの下部にあるソフトウェア ボタン。
- アプリ内部やアプリ間での戻るナビゲーション。

推奨事項

- "戻る" ナビゲーションを有効にします。
  "戻る" ナビゲーションが有効でない場合は、アプリはグローバルなバック スタックに含まれますが、アプリ内のページ ナビゲーション履歴は保持されません。
- デスクトップ モードではタイトル バーの戻るボタンを有効にします。
  アプリ内のページ ナビゲーション履歴は保持され、アプリ間の "戻る" ナビゲーションはサポートされていません。
  注 タブレット モードでは、ユーザーがデバイスの上部から下へスワイプするか、デバイスの上部付近にマウス ポインターを動かしたときに、タイトル バーが表示されます。重複や混乱を避けるため、タイトル バーの戻るボタンは、タブレット モードでは表示されません。
- アプリ内のナビゲーション履歴が使い果たされた場合や利用できない場合は、デスクトップ モードでタイトル バーの戻るボタンを非表示にするか、無効にします。
  可能な限り "戻る" ナビゲーションを行ったことをユーザーに明確に示します。
- 各戻るコマンドでは、バック スタック内の 1 ページ前、または、デスクトップ モードでない場合は、直前のアプリに戻る必要があります。
  "戻る" ナビゲーションが、直感的でない場合、一貫性がない場合、予測不可能な場合、ユーザーは混乱する可能性があります。

ボタンのガイドライン

ボタン (コマンド ボタン) は、特定の操作を直ちに実行する方法をユーザーに与えます。



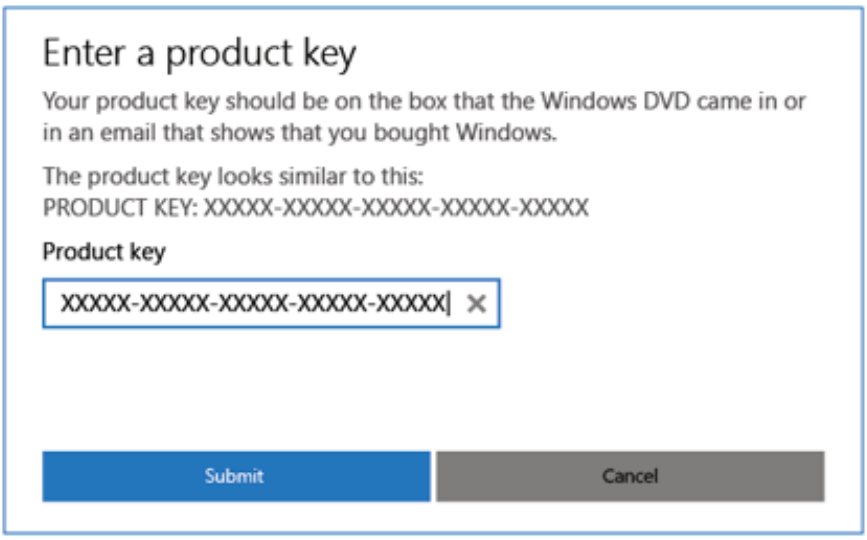
重要な API

Button クラス (XAML)

button 要素 (HTML)

input type="submit" 要素 (HTML)

例



適切なコントロールの選択

ボタンを使うと、ユーザーは直ちに操作を開始できます (フォームの送信など)。

他のページに移動する操作では、ボタンは使わず、リンクを使います。 例外: ウィザードでのページの移動には、[戻る] と [次へ] というラベルのボタンを使います。他の種類の前に戻る移動や上位レベルへの移動では、戻るボタンを使います。

推奨と非推奨

- ボタンの用途と状態をユーザーがはっきりと理解できるようにします。
- ボタンによって行われる操作を明確に説明する、簡潔で具体的でわかりやすいテキストを使います。通常、ボタンのテキスト コンテンツは、1 語の動詞です。
- ボタンのテキスト コンテンツが動的な場合 (ローカライズされる場合など) は、ボタンのサイズがどのように変化し、その周囲のコントロールに何が起こるかを考えます。
- テキスト コンテンツの付いたコマンド ボタンの場合は、最小のボタン幅を使います。
- テキスト コンテンツの付いた幅が狭い横長または縦長のコマンド ボタンは使わないようにします。
- ブランドのガイドラインで別のフォントが指示されていない限り、既定のフォントを使います。

- ある操作をアプリの複数のページで実行できるようにするには、各ページでボタンを使うのではなく、下部のコマンド バーを使うことを検討します。
- ユーザーに対して表示するボタンは、1 つまたは 2 つにします (例: [承諾] と [キャンセル])。3 つ以上の操作をユーザーに示す必要がある場合は、操作を開始する 1 つのコマンド ボタンをユーザーが選択できるチェックボックスまたはラジオ ボタンを、それらの操作を開始するための 1 つのコマンド ボタンと共に使うことを検討します。
- 最も一般的な操作や推奨される操作を示す既定のコマンド ボタンを使います。
- ボタンをカスタマイズすることを検討します。ボタンの形は既定では四角形ですが、ボタンの外観を構成するビジュアル効果をカスタマイズできます。ボタンのコンテンツは、通常はテキスト (一例: [承諾] や [キャンセル]ー) ですが、アイコンに置き換えるか、アイコンとテキストを使うことができます。
- ユーザーがボタンを操作したとき、ボタンの状態と外観を変更して、ユーザーにフィードバックを返します。ボタンの状態には、Normal、Pressed、Disabled などがあります。
- ユーザーがボタンをタップまたはクリックしたときに、ボタンのアクションを開始します。通常、アクションは、ユーザーがボタンを離したときに開始されますが、指がボタンを押したときにボタンのアクションを開始するように設定することもできます。
- コマンド ボタンは、状態の設定には使わないでください。
- アプリの実行中、ボタンのテキストは変更しないでください。たとえば、"次へ" というボタンのテキストは "続行" というテキストには変更しないでください。
- 送信、リセット、標準ボタンの既定のスタイルを取り替えないでください。
- ボタン内に多すぎるコンテンツを配置しないでください。表やチェック ボックスなど、他の大半の HTML 要素を含めることができますが、多すぎるコンテンツはユーザーを混乱させることになります。ボタン内のコンテンツは、簡潔でわかりやすくします (画像と少しのテキストのみにします)。

カメラ キャプチャの UI のガイドライン

カメラ キャプチャ UI ダイアログでは、カメラが内蔵または接続されたデバイスで組み込みカメラ UI を使って、ユーザーが写真やビデオをキャプチャすることができます。 キャプチャのエクスペリエンスの一部として、呼び出し元のアプリケーションに戻す前に、ユーザーはキャプチャした写真やビデオをトリミングできます。さらに、写真やビデオをキャプチャする前に、明るさ、コントラスト、露出など、カメラの一部の設定を調整することもできます。

重要な API

CameraCaptureUI.CaptureFilesAsync メソッド

カメラ UI の提供箇所

カメラ UI は、カメラが内蔵または接続されているデバイスで利用できます。ほとんどの携帯電話、タブレット、ノート PC には、カメラが内蔵されています。PC やその他のデバイスにカメラが内蔵されていることは、それほど一般的ではありません。

適切な UI の選択

- カメラ キャプチャ UI は、アプリから最小限のコードで写真またはビデオをキャプチャする場合に使います。
- リアルタイムのフィードバックを得る必要がある場合や、キャプチャ中の画像を制御する必要がある場合は、カメラ キャプチャ UI を使わないでください。たとえば、バーコード リーダー アプリで、カメラを使ってバーコードをスキャンする際に、ユーザーにリアルタイムのフィードバックを提供し、それによりユーザーはバーコードが読み取り可能かどうかを知ることができます。この場合カメラ ダイアログは、キャプチャされたビデオ ストリームを直接制御できないため、適切なオプションではありません。代わりに **MediaCapture** API を使ってください。
- ユーザー インターフェイスにカスタムのコントロールを追加する必要がある場合、カメラ キャプチャ UI は使わないでください。 カメラ ダイアログで提供される機能を超えて UI のカスタマイズを追加する必要がある場合は、代わりに **MediaCapture** API を使ってください。

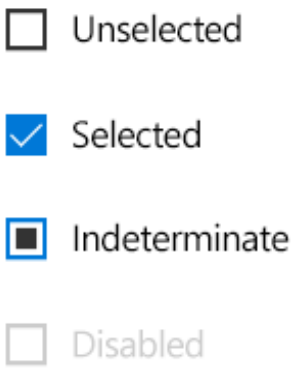
- プログラムによるキャプチャ デバイス コントロール (フォーカス、フラッシュ、手ブレ補正など) を含めて、キャプチャ プロセスに対する低レベルの制御を必要とする場合は、カメラ キャプチャ UI を使わないでください。代わりに **MediaCapture** API を使ってください。

推奨と非推奨

- トリミングがアプリケーションで提供される場合には、カメラ ダイアログでトリミングをオフにしてください。 ビデオや写真の編集アプリケーションの場合、または写真やビデオの編集機能を提供する場合、トリミングをオフにしてカメラ ダイアログを使う必要があります。そうすると、アプリケーションのトリミング機能はカメラ ダイアログが提供する機能と重複しません。

チェック ボックス コントロールのガイドライン

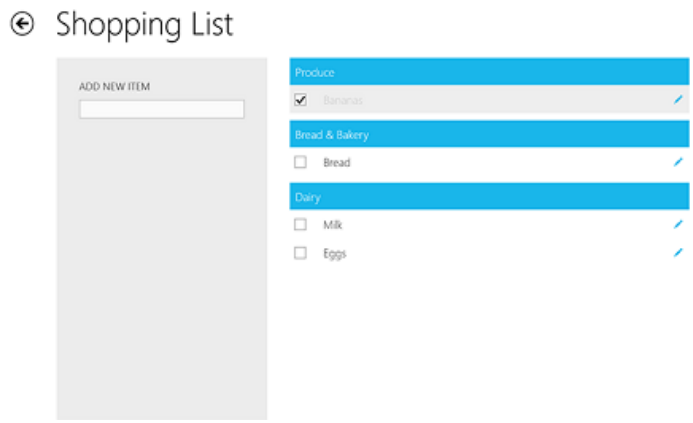
チェック ボックスは、アクション項目の選択や選択解除を行うときに使います。また、チェック ボックスは単一のリスト項目や複数のリスト項目に対して使うことができます。コントロールには 3 つの選択状態 (選択されていない、選択されている、不確定) があります。不確定状態は、選択されていない状態と選択されている状態の両方がサブ選択肢のコレクションに含まれている場合に示されます。



重要な API

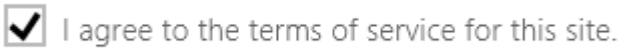
CheckBox クラス (XAML)	input type="checkbox" 要素 (HTML)

例



適切なコントロールの選択

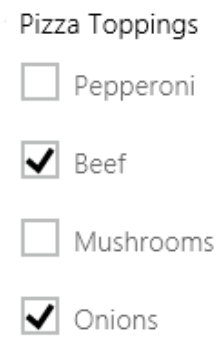
"このアカウントを記憶する" ログイン シナリオや、サービス契約の条項など、はい/いいえの二者択一の選択肢に対しては、1 つのチェック ボックスを使います。



複数選択シナリオの場合 (ユーザーが相互排他的でない選択肢のグループから 1 つ以上の項目を選ぶ場合) は、複数のチェック ボックスを使います。

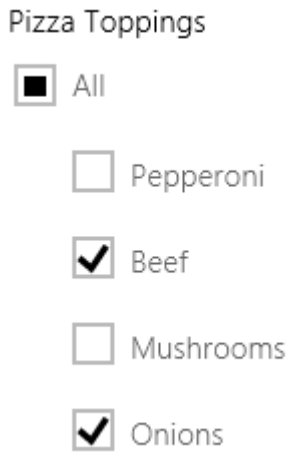
二者択一の場合、チェック ボックスとトグル スイッチとの主な違いは、チェック ボックスが状態を管理し、トグル スイッチが動作を管理する点です。 チェック ボックスによる操作はコミットを遅らせることができますが (たとえばフォームの送信の一部として)、トグル スイッチによる操作は直ちにコミットしなければなりません。また、複数の選択ができるのは、チェック ボックスだけです。

ユーザーがオプションの任意の組み合わせを選べる場合は、チェック ボックスのグループを作成します。



ラジオ ボタンのグループが 1 つの選択を表すラジオ ボタンとは異なり、グループ内の各チェック ボックスが個別の独立した選択を表します。1 つ以上のオプションがあっても、選択できるのが 1 つだけの場合は、代わりにラジオ ボタンを使います。

オプションが複数のオブジェクトに適用される場合は、チェック ボックスを使って、オプションの適用対象がすべてのオブジェクトか、一部のオブジェクトか、いずれにも適用されないかを示すことができます。オプションの適用対象がすべてのオブジェクトではなく、一部のオブジェクトである場合は、混在する選択を表すために、チェック ボックスの中間的な状態を使います。 混在する選択のチェック ボックスの 1 つの例は、すべてでなく一部のサブ項目をユーザーが選んだ場合に中間的な状態になる [すべて選択] チェック ボックスです。



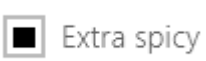
詳しくは、次のトピックをご覧ください。

- **indeterminate property (HTML)**
- **Indeterminate event (XAML)**

推奨と非推奨

- チェック ボックスの用途と現在の状態が明確であることを確認します。
- チェック ボックスのテキスト コンテンツは 2 行以内にします。
- チェック マークをオンにすると true、チェック マークをオフにすると false に設定されるようにチェック ボックスのラベルを記述します。
- ブランドのガイドラインで別のフォントが指示されていない限り、既定のフォントを使います。
- 提示する選択が複数ある場合は、レイアウト パネルが組み込まれたスクロール ビューアー コントロールを使うことを検討してください。
- すべてではなく一部の子オブジェクトのためにオプションが設定されていることを示すために、不確定の状態を使います。
- 不確定の状態を使う場合は、下位のチェック ボックスを使って、どのオプションが選択され、どのオプションが選択されていないかがわかるようにします。ユーザーにサブ選択肢がわかりやすいように UI を設計します。
- テキスト コンテンツが動的な場合、コントロールのサイズがどのように変わり、周囲のビジュアル効果にどのような影響が生じるかを検討してください。
- 相互排他的な複数のオプションから選ぶ場合は、オプション ボタンを使うことを検討してください。
- 2 つのチェック ボックスのグループを並べて配置しないようにします。グループを分けるには、グループ ラベルを使います。
- オン/オフの制御やコマンドの実行にはチェック ボックスを使わず、代わりにトグル スイッチを使います。
- ダイアログ ボックスなどの他のコントロールを表示するためにチェック ボックスを使わないでください。
- 不確定の状態を、第 3 の状態を示すために使わないでください。不確定の状態は、すべてでなく一部の子オブジェクトのためにオプションが設定されていることを示す目的で使います。そのため、ユーザーが不確定の状態を直接設定できないようにします。

  良くない例を示すために、次のチェック ボックスでは不確定の状態を使って "中辛" を示しています。

このような場合は、[Not spicy]、[Spicy]、[Extra spicy] という 3 つのオプションがあるラジオ ボタン グループを使います。

- (HTML のみ) ラベル内にチェック ボックスを配置して、ラベルをクリックするとチェック ボックスが切り替わるようにします。こうすることで、選択領域のサイズが広がり、タッチ ユーザーがチェック ボックスを使いやすくなります。

コマンド バーのガイドライン

コマンド バーを使うと、ユーザーはアクションに簡単にアクセスできます。コマンド バーは、ユーザーのコンテキストに固有のコマンドやオプション (写真の選択や描画モードなど) を表示するために使うことができます。また、アプリのページやセクション間のナビゲーションにも使うことができます。コマンド バーは、ナビゲーション パターンと一緒に使うことができます。

コマンド バーは、表示したままにするコマンドまたはナビゲーション項目を配置するためのアクション領域と、コマンド バーに省略記号 [•••] として表示される [その他] 領域の 2 つのコンポーネントで構成されます。[その他] 領域では、使う頻度が高くないコマンドやナビゲーション項目のドロップダウン リスト表示メニューが開きます。[•••] ボタンを選ぶと、メニューが開き、アクション領域の各項目のテキスト ラベルが表示されます。[その他] に項目がない場合、ドロップダウンは開きませんが、アクション領域にある項目のテキスト ラベルはその場合も表示されます。

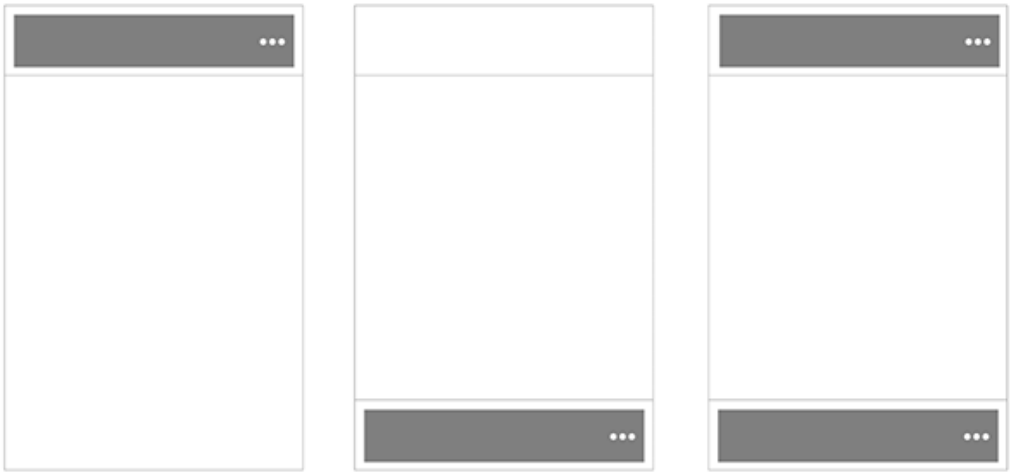
重要な API

CommandBar クラス (XAML)

Toolbar (WinJS)
AppBar (WinJS)

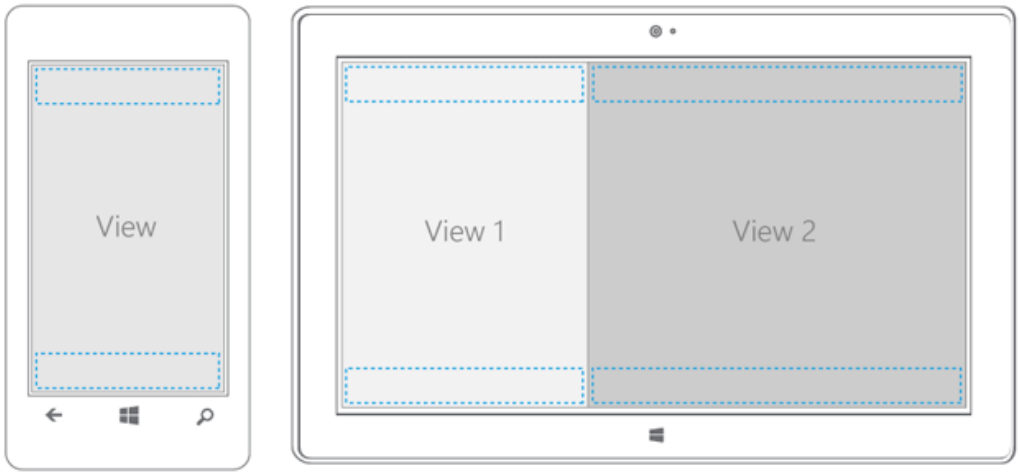
コマンド バーの配置

コマンド バーは画面の上部または画面の下部、画面の上部と下部の両方、またはインラインに配置できます。



- モバイル デバイスでは、コマンド バーをアプリに 1 つだけ配置する場合、手に届きやすいように画面の下部に配置します。アプリの下部にタブがある場合、UI が下部に集中しすぎないように上部にコマンド バーを配置することを検討してください。
- 大きな画面では、コマンド バーを 1 つだけ配置する場合、画面の上部に配置することをお勧めします。
- ユーザーがコンテキスト依存アクションに使用できるように、コマンド バーをインラインで配置することもできます。

コマンド バーは、単一ビュー画面 (左側の例) と複数ビュー画面 (右側の例) の次の画面領域に配置できます。インラインのコマンド バーは、操作領域の任意の場所に配置できます。



アクションの配置

コマンド バーに配置するアクションには、見やすさに基づいて優先順位を付けます。

- バーに常に表示する最も重要なコマンドは、アクション領域の最初の数スロットに配置します。最小画面 (幅 320 epx) では、他の画面 UI に応じて 2 ～ 4 項目がコマンド バーのアクション領域に収まります。
- 重要度の低いコマンドは、バーのアクション領域の後方に配置するか、[その他] 領域の最初の数スロット内に配置します。これらのコマンドは、バーに十分な画面領域がある場合は表示されますが、十分な領域がない場合は [その他] 領域のドロップダウン メニューに収まります。
- 最も重要度の低いコマンドは、[詳細] 領域内に配置します。これらのコマンドは、常にドロップダウン メニューに表示されます。

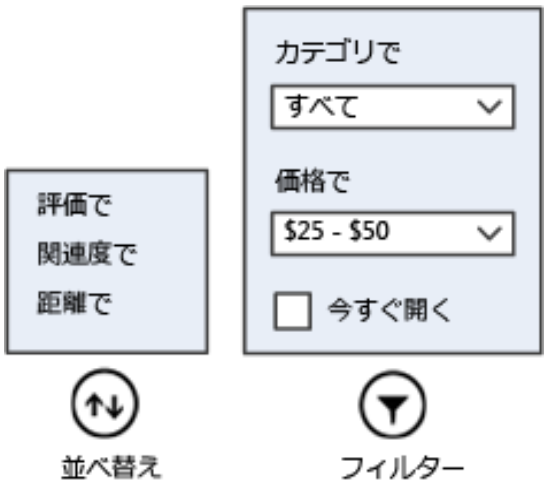
アクション領域の項目は、アイコンまたはボタンのいずれかを使って視覚化できます。アイコンのみを使う場合は、テキスト ラベルを含めます。テキスト ラベルは、[•••] が選択されたときにアイコンの下に表示されます。

複数のページで一貫して表示されるコマンドがある場合は、一貫した場所にそのコマンドを配置することをお勧めします。また、[Accept] (承諾)、[Yes] (はい)、[OK] (OK) コマンドは、[Reject] (拒否)、[No] (いいえ)、[Cancel] (キャンセル) コマンドの左に配置することをお勧めします。一貫性があることで、ユーザーは安心してシステム内を移動でき、アプリのナビゲーションに関する知識をさまざまなアプリで利用することができます。

すべてのアクションを [その他] ドロップダウン メニュー内に配置して、コマンド バーに [•••] だけが表示されるようにすることができます。ただし、すべてのアクションを非表示にすると、ユーザーが混乱する可能性があることに注意してください。

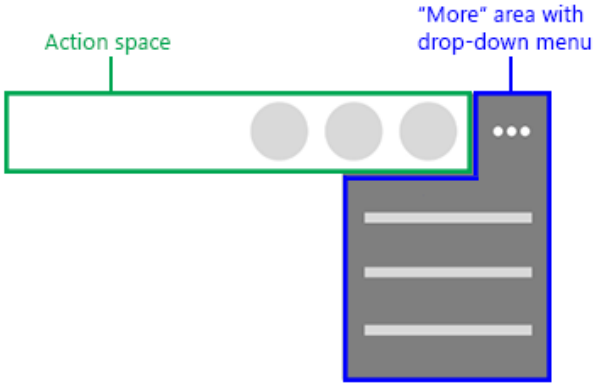
コマンド バーのフライアウトとヒント

コマンドは論理的にグループ化することを検討します。たとえば、[返信]、[全員に返信]、[転送] を [応答] メニューに配置します。



[•••] を選択しない限り、テキスト ラベルはコマンド バーの操作に対して非表示になるため、アクション アイコンのヒントを使用することを検討します。

[その他]領域



- [その他] のアフォーダンス [•••] は、メニューの可視エントリ ポイントです。ツール バー右端のプライマリ アクションの隣に配置されます。
- プライマリ操作領域の各操作は、アイコンで表されます。オーバーフロー メニューを選ぶと、プライマリ操作領域の各操作のテキスト ラベルが表示されます。
- オーバーフロー メニューの領域は、使用頻度が少ない操作に割り当てられます。
- 操作は、ブレークポイントを境としてプライマリ操作領域とオーバーフロー メニューの間で移動させることができます。画面またはアプリのウィンドウ サイズに関係なく、常にアクションをプライマリ操作領域内に維持するように指定することもできます。
- 使用頻度の低いアクションは、より大きな画面でアプリ バーを展開したときに、オーバーフロー メニュー内に残しておくことができます。

応答性のガイダンス

- アプリ バー内のアクションと同じ数のアクションが、縦向きと横向きの両方の向きで表示される必要があります。これにより、ユーザーが認識する負荷が軽減されます。利用可能なアクションの数は、縦向きでのデバイスの幅によって決まります。
- ブレークポイントをターゲットにすると、メニューのサイズやアプリ ウィンドウのサイズの変化に応じてアクションをメニューの内外に移動できます。

コンテキスト メニューのガイドライン

コンテキスト メニューは、ユーザーがすぐに操作できる軽量のメニューです。カスタム コマンドを含めることができます。 コンテキスト メニューは、その外側をタップまたはクリックして閉じることができます。

重要な API

MenuFlyout クラス (XAML)	コンテキスト メニューの追加 (HTML)

適切なコントロールの選択

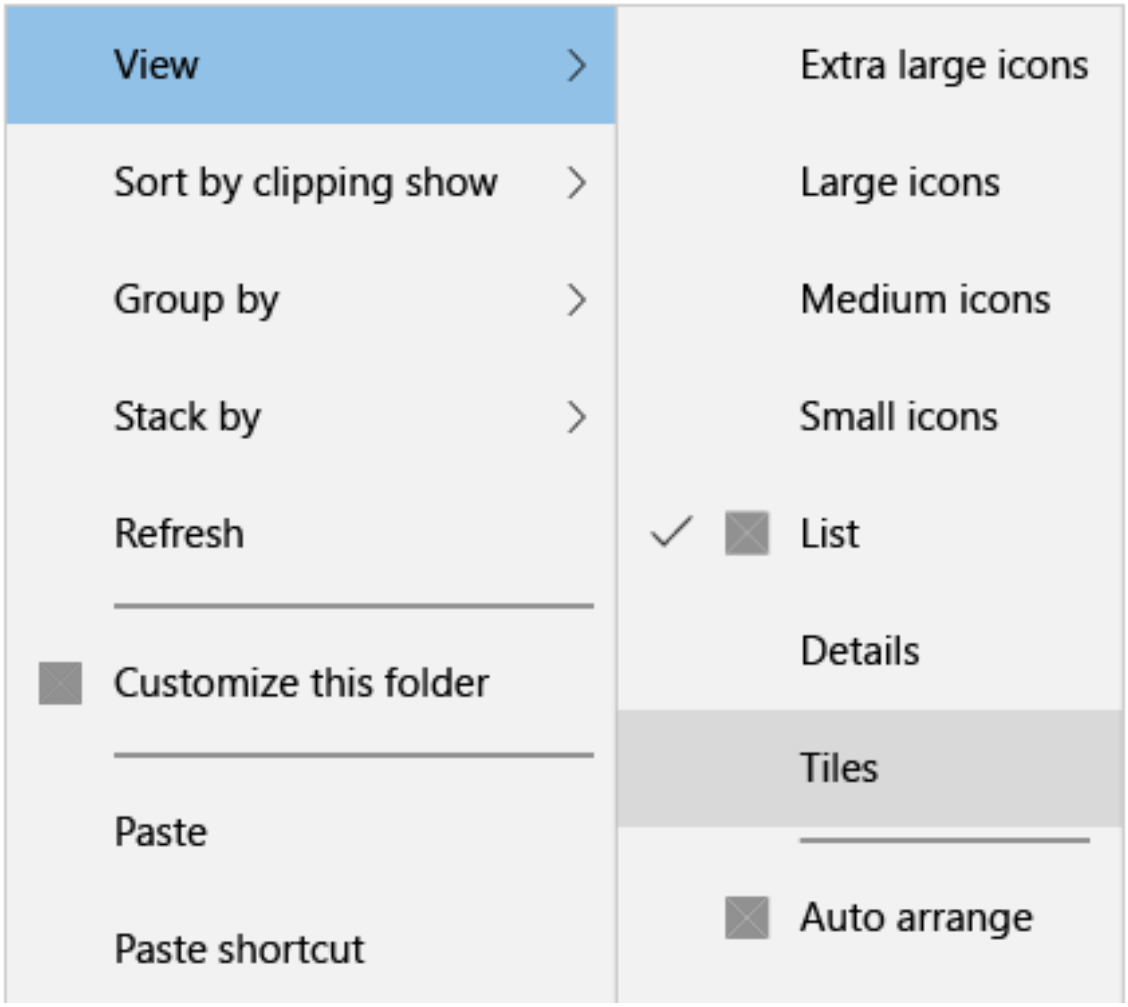
コンテキスト メニューを使用して、次の操作を実行できます。

- 選んだテキストに対する状況依存の操作 (コピー、切り取り、貼り付け、スペル チェックなど)。
- クリップボード コマンド。
- カスタム コマンド。
- 操作する必要があるものの、選択することも、他の方法で指定することもできないオブジェクトのためのコマンド。

例

次に、一般的な単一ペインのコンテキスト メニューを示します。これは、少数の単純なコマンドに使用されます。必要に応じて区切り記号を使い、類似のコマンドをグループ化します。

より包括的なコマンドの集合には、カスケード コンテキスト メニューが使われます。ポップアップ レベルが 1 つで、スクロール可能です。必要に応じて区切り記号を使い、類似のコマンドをグループ化します。



使い方のガイダンス

- コンテキスト メニューでコマンドのグループ間の区切り記号を使用するのは、次の場合です。
  - 関連するコマンドのグループを区別する場合。
  - コマンドのセットをグループ化する場合。
  - クリップボード コマンド (切り取り/コピー/貼り付け) などの一般的なコマンドのセットを、アプリまたはビューに固有のコマンドと区別する場合。
- ノート PC とデスクトップ PC では、コンテキスト メニューとヒントはアプリケーションのウィンドウに限定されず、外側に描画できます。アプリが完全にウィンドウの外側にショートカット メニューをレンダリングしようとすると、例外がスローされます。

推奨事項

- コンテキスト メニュー コマンドは、長くならないようにします。コマンドが長い場合は、その末尾が切り捨てられます。
- 各コマンド名には、文としての大文字表記を使います。
- すべてのコンテキスト メニューで、表示するコマンドの数を最小限にします。
- UI 要素を直接操作できる場合は、そのコマンドをコンテキスト メニューに入れないようにします。コンテキスト メニューは、それ以外では画面に表示されない状況依存のコマンドに限定する必要があります。

日付と時刻コントロールのガイドライン

日付と時刻のコントロールでは、日付と時刻を表示および設定できます。この記事では設計ガイドラインを示し、適切なコントロールを選ぶのに役立ちます。

重要な API

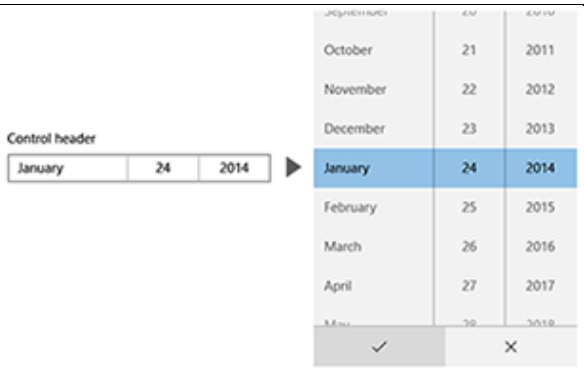
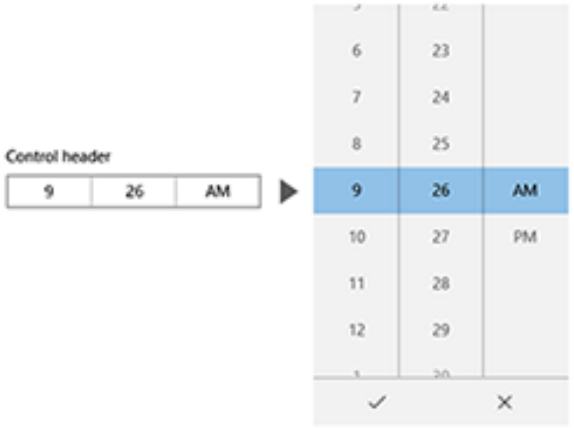
[CalendarView クラス (XAML)]

[DatePicker クラス (XAML)] [CalendarDatePicker クラス (XAML)]

日付、または時刻コントロールの選択

選択できる日付と時刻のコントロールは 4 つあります。

コントロール		機能
カレンダー ビュー		月、年、または 10 年の形式で単一の日付または日付の範囲を表示します。選択ツール サーフェスはありません。
CalendarDatePicker コントロール		カレンダー ビューの機能に加えて、選択ツール サーフェスが含まれており、単一の日付または日付の範囲を選択できます。

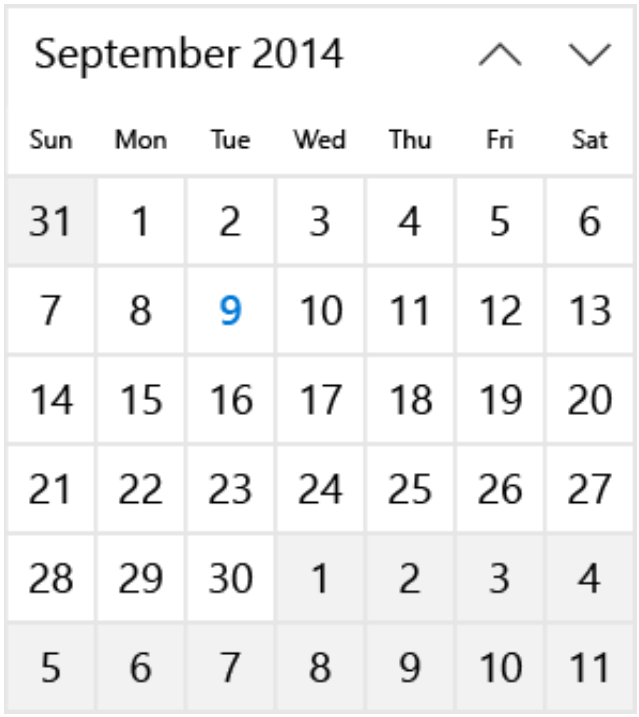
DatePicker コントロール		カレンダーの日付選択ツールのコンパクトな代替コントロール。フォームで使用することも、領域が不足するときにだけ使用することもできます。
TimePicker コントロール		単一の時刻を選択できるコンパクトな選択ツール。現在の時刻は表示されません。

CalendarView コントロール

カレンダー ビューでは、月、年、または 10 年の形式で単一の日付または日付の範囲を表示できます。このコントロールは読み取り専用であり、日付入力のための選択ツール サーフェスはありません。選択ツール サーフェスを持つ類似のコントロールが必要な場合は、CalendarDatePicker コントロールを参照してください。

カレンダー ビューには、月ビュー、年ビュー、10 年ビューという 3 つの拡張ビューがあります。

月ビュー、標準の月ごとのカレンダー。



年ビュー、各月を含む暦年。

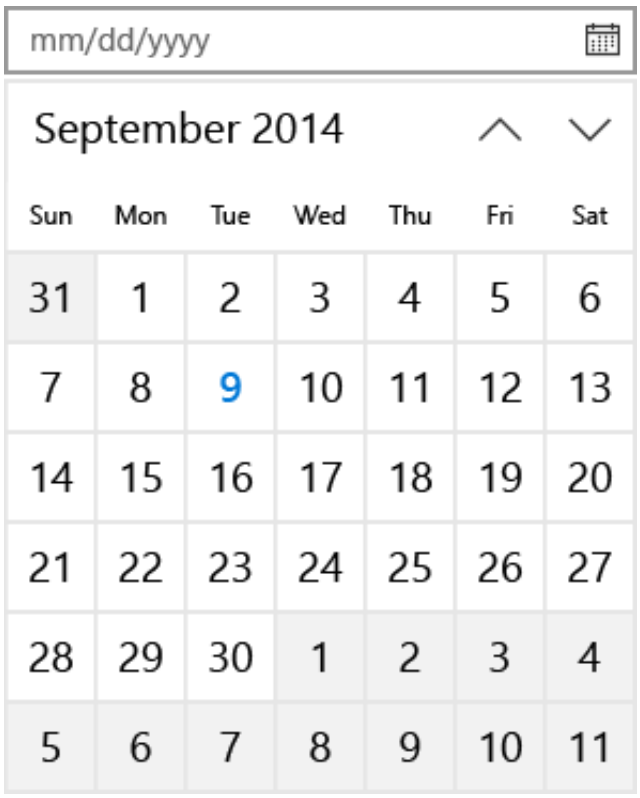
10 年ビュー、10 年の範囲。

CalendarDatePicker コントロール

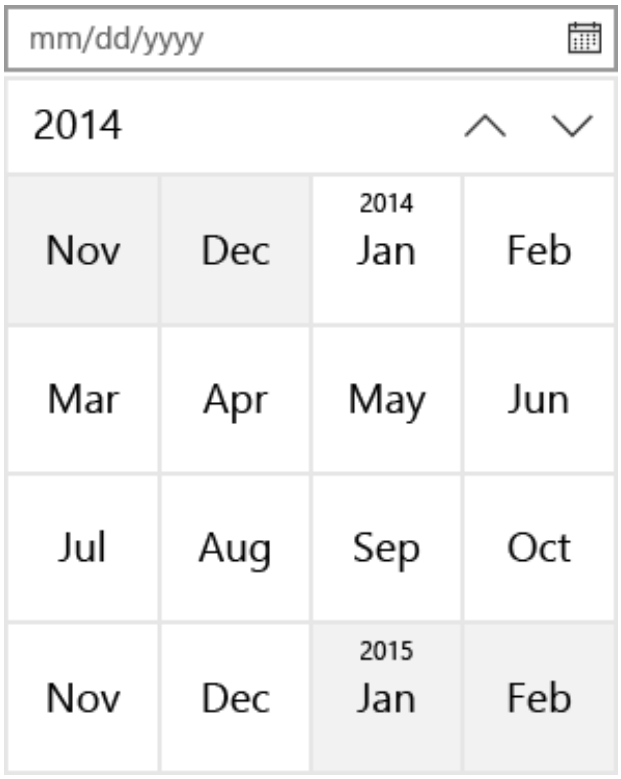
CalendarDatePicker は、カレンダー ビューと同じですが、カレンダーの上に日付の入力フィールドがあります。月、年、または 10 年の形式で単一の日付または日付の範囲を選択できます。日付が設定されていない場合、エントリ ポイントにはプレースホルダー テキストが表示されます。

3 つの拡張ビューには、月、年、10 年があります。各拡張ビューの上部には、日付の入力フィールドが含まれます。

月ビュー、標準の月ごとのカレンダー。



年ビュー、各月を含む暦年。



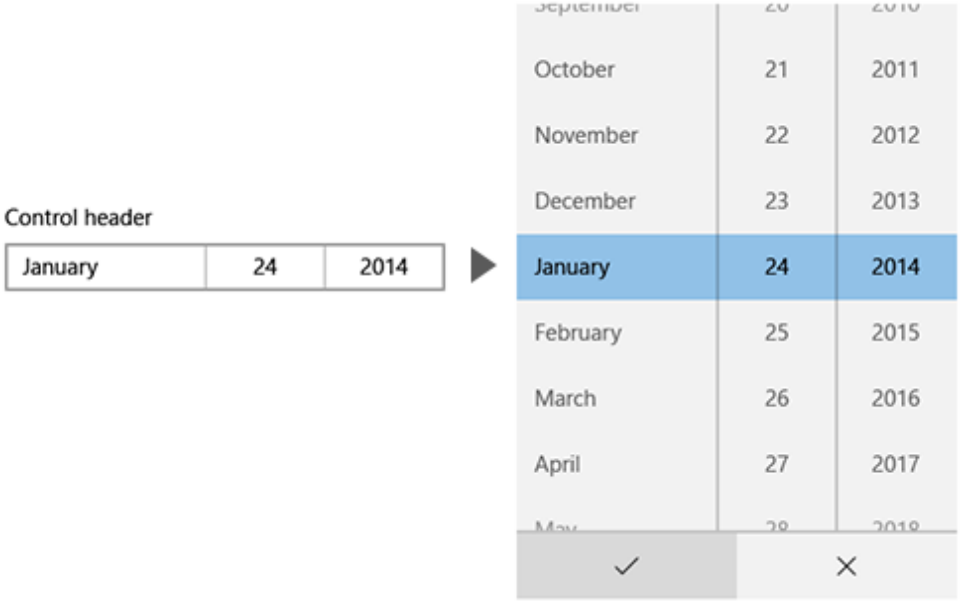
10 年ビュー、10 年の範囲。

DatePicker コントロール

DatePicker コントロールによって、標準化された方法で特定の日付を選ぶことができます。エントリ ポイントには、選んだ日付が表示されます。エントリ ポイントを選ぶと、ユーザーによる選択を可能にする選択ツール サーフェスが展開されて、日付を選べるようになります。 DatePicker は他の UI をオーバーレイし、他の UI を別の位置に移動させることはありません。

エントリ ポイントをタップすると、ユーザーが選択できる選択ツール サーフェスが展開されて、日付を選べるようになります。

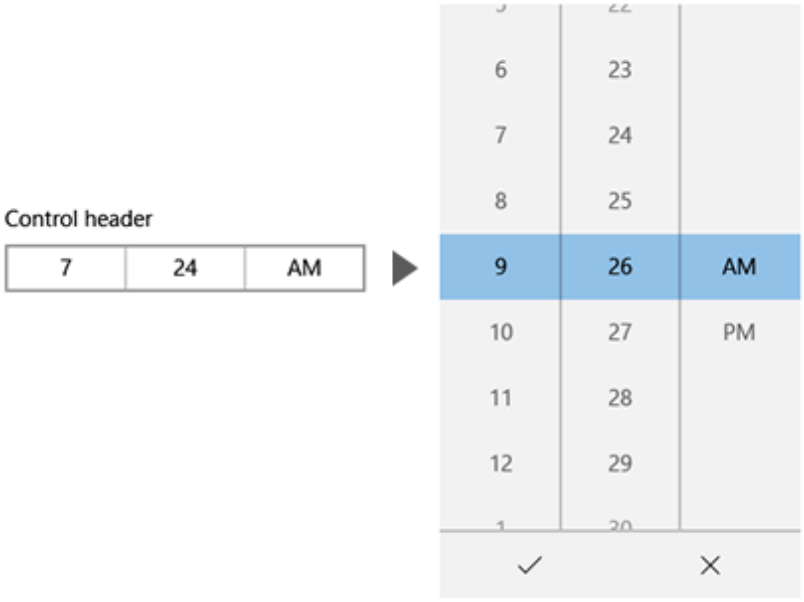
コントロールのエントリ ポイントを選ぶと、中央から縦方向に展開されます。



TimePicker コントロール

TimePicker コントロールは、予定などのための 1 つの時刻を選択するために使用されます。時刻の選択は、ユーザーによって設定される静的な表示であり、現在の時刻は表示されません。 エントリ ポイントには選んだ時刻が表示されます。エントリ ポイントを選ぶと、ユーザーによる選択を可能にする選択ツール サーフェスが展開されて、時刻を選べるようになります。TimePicker は他の UI をオーバーレイし、他の UI を別の位置に移動させることはありません。

コントロールのエントリ ポイントをタップすると、中央から縦方向に展開されます。



ダイアログ コントロールのガイドライン

ダイアログは、状況依存のアプリ情報を表示するモーダルな UI オーバーレイです。ほとんどの場合、ダイアログは明示的に閉じられまでアプリ ウィンドウの操作を妨げます。また、多くの場合、ユーザーに操作を要求します。

重要な API

ContentDialig クラス (XAML)　　ContentDialig オブジェクト (HTML)

適切なコントロールの選択

ユーザーが操作を中断して、読んだり認識したりする必要がある重要な情報を伝えるには、ダイアログ コントロールを使います。

ダイアログ コントロールは、ユーザーの明確な操作を求める場合、またはユーザーが認識する必要がある重要なメッセージを伝える場合に使います。次のようなシナリオが考えられます。

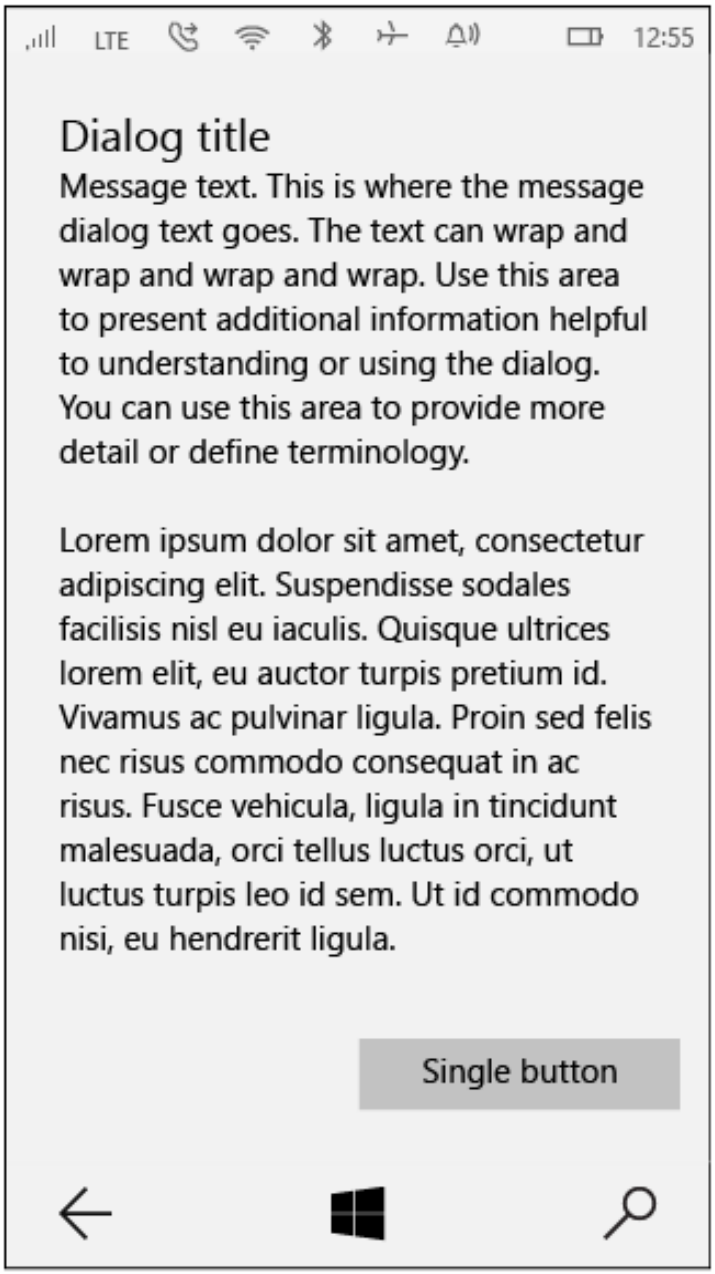
- ユーザーのセキュリティが侵害される可能性がある場合
- ユーザーが重要な資産に永続的な変更を加えようとしている場合
- ユーザーが重要な資産を削除しようとしている場合
- アプリ内購入を確認するには

接続エラーなど、エラーがアプリ全体の状況に適用される場合は、インライン エラーではなくエラー ダイアログを使います。

アプリからユーザーにブロック質問を行う必要がある場合 (アプリで自動的に選ぶことができない場合など) は質問ダイアログを使います。ブロック質問は無視したり先送りにしたりできない質問です。この質問では、ユーザーに明確な選択肢を提示する必要があります。

例

これは、ボタンを 1 つ備えた確認ダイアログの全画面表示の例です。この種類のダイアログ ボックスでは、ボタンを押して次に進む前に読まれることを想定した、多くの情報が表示されます。



ボタンを 2 つ備えたダイアログの例を次に示します。このダイアログでは二者択一の選択肢が表示されます。一般に、このダイアログに表示される情報は簡潔にまとめられています。

一般的な推奨事項

- ダイアログ内のテキストの 1 行目で、問題やユーザーの目的 (実行する内容) を明確に示す必要があります。
- ダイアログのタイトルは主な説明で、省略可能です。
  - 簡潔なタイトルを使って、ユーザーがそのダイアログで行う必要がある操作を説明します。タイトルが長い場合は、折り返されず省略されます。
  - ダイアログを使って、簡単なメッセージ、エラー、または質問を表示する場合は、タイトルを省略することもできます。主な情報はコンテンツのテキストを使って伝えます。
  - タイトルは、ボタンの選択に直接関連するものにします。
- ダイアログ コンテンツには説明のテキストを含め、これは必須です。
  - メッセージ、エラー、または操作をブロック質問をできる限り簡潔に示します。
  - ダイアログ タイトルを使う場合は、コンテンツ領域を詳しい情報の提示や用語の定義に使います。タイトルの言葉づかいを変えただけの文を繰り返さないようにします。
- 少なくとも 1 つのダイアログ ボタンを表示する必要があります。
  - テキストを指定したボタンを使って、主な説明またはコンテンツに対する応答を示します。たとえば、主な説明が "使っているコンピューターへの AppName からの

アクセスを許可しますか?" の場合、"許可" ボタンと "ブロック" ボタンを使います。具体的な応答の言葉はすばやく理解できるので、効率的に判断できます。

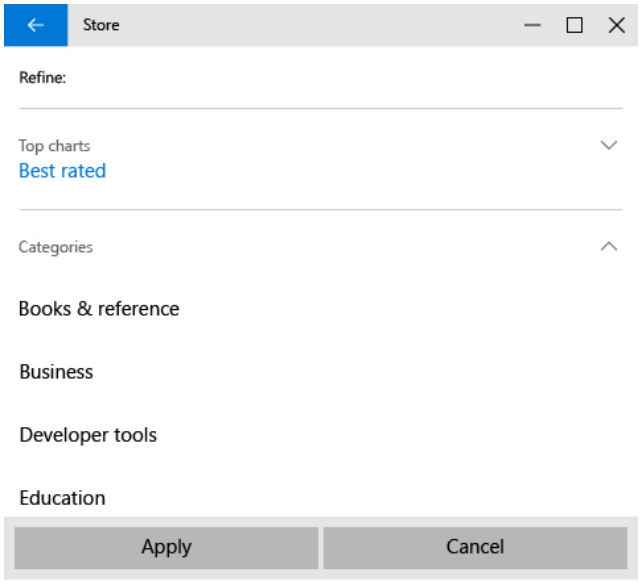
- エラー ダイアログでは、ダイアログ ボックスにエラー メッセージと関連情報 (ある場合) を表示します。エラー ダイアログで使う唯一のボタンは "閉じる" かこれに似た操作である必要があります。
- ページの特定の場所 (たとえばパスワード フィールド) に関連するエラー (検証エラーなど) の場合、ダイアログは使わずに、アプリのキャンバス自体を使ってインライン エラーを表示します。

フィルターと並び替えのガイドライン

フィルター コマンドと並べ替えコマンドを使うと、コンテンツをカスタムの配列にまとめて表示できます。

フィルター

フィルター コマンドは、一定の基準に基づいて、一覧内のコンテンツを非表示にします。たとえば、"高評価" に基づいてストア アプリを表示することができます。



フィルターが推奨されるケース: フィルターは項目数が多い一覧に役立ちます。スクロールな必要な大きさの一覧に最も大きなメリットがあります。

フィルターを提供するには、いくつかの方法があります。

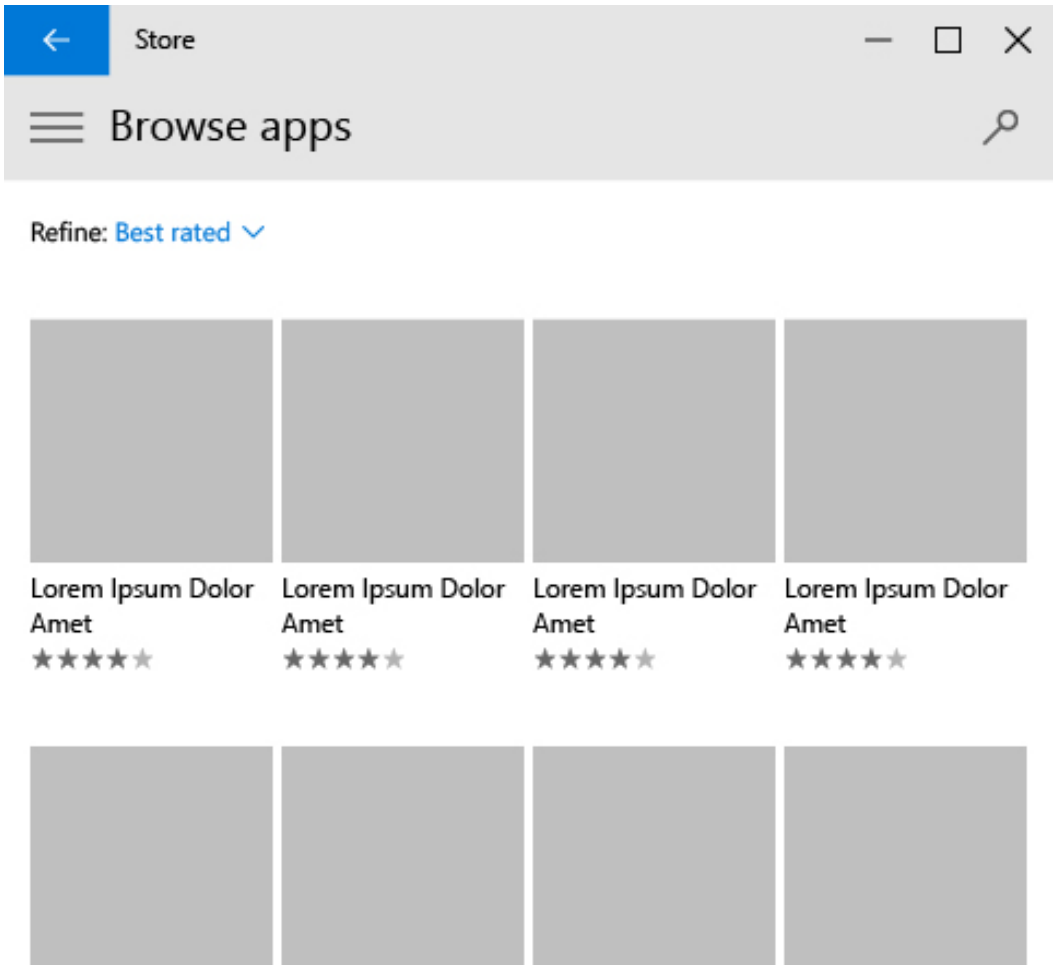
- 自由形式: オート サジェスト ボックスを使って、ユーザーが任意のテキストを入力できるようにすることができます。アプリには、フィルター テキストに一致する要素を持つ項目を表示し、それ以外のすべての項目を非表示にできます。自由形式フィルターは、表示している項目の内容をよく知っていて、特定の項目を探しているユーザーに適しています。オート サジェスト ボックスがあれば自由形式フィルターを実装できるため、それほど多くの画面領域を使いません。
- 定義済みのフィルター オプション: 一覧内の項目に固有の限定的なフィルター オプション セットをユーザーに表示できます。たとえば、ショッピング アプリでは、ユーザーが価格帯、納期、またはユーザーの評価でフィルター処理できるようにすることができます。 定義済みのフィルター オプションのメリットは、何を探しているのかがはっきりわかっていないユーザーに役立ち、自由形式フィルターよりも複雑なフィルター処理を行うことができるという点です。定義済みフィルターのデメリットは、提供するオプションの数によっては、画面領域を多く消費するという点です。

推奨事項

- フィルターがアクティブなときは必ずユーザーに通知してください。通知しないと、ユーザーは一部の項目が非表示になっていることを気付かない可能性があります。
- 常にフィルターを簡単に解除できるようにしてください。一般的なユーザーは、さまざまなフィルター オプションを試します。フィルターを簡単に解除できるメカニズムを用意すると、ユーザーはさまざまなオプションを試すようになります。
- フィルター オプションが排他的に適用されるのか、追加で適用されるのかをはっきりさせて、期待される動作をユーザーが知ることができるようにしてください。

並び替え

並べ替えコマンドは、一覧内のコンテンツの表示順序を変更します。たとえば、トラベルアプリでユーザーは旅行先を人気順に並べ替えることができます。



フィルターとは異なり、並べ替えでは項目が非表示になりません。順序が変わるだけです。

並べ替えが推奨されるケース: 並べ替えオプションは項目数が多い一覧に役立ちます。スクロールな必要な大きさの一覧に最も大きなメリットがあります。

推奨事項

項目がどのように並べ替えられているかをユーザーに通知してください。追加のオプションを用意しない場合でも、項目がどのように並べ替えられているかをユーザーに知らせると、どのように項目を参照したらよいかを理解することができます。

FlipView コントロールのガイドライン

コレクション内の画像 (アルバム内の写真や製品の詳細ページ内の項目など) を一度に 1 つずつ表示するためは、FlipView コントロールを使います。タッチ デバイスでは、項目をスワイプしてコレクション内を移動します。マウスでは、マウスをホバーするとナビゲーション ボタンが表示されます。キーボードでは、方向キーを使ってコレクション内を移動します。

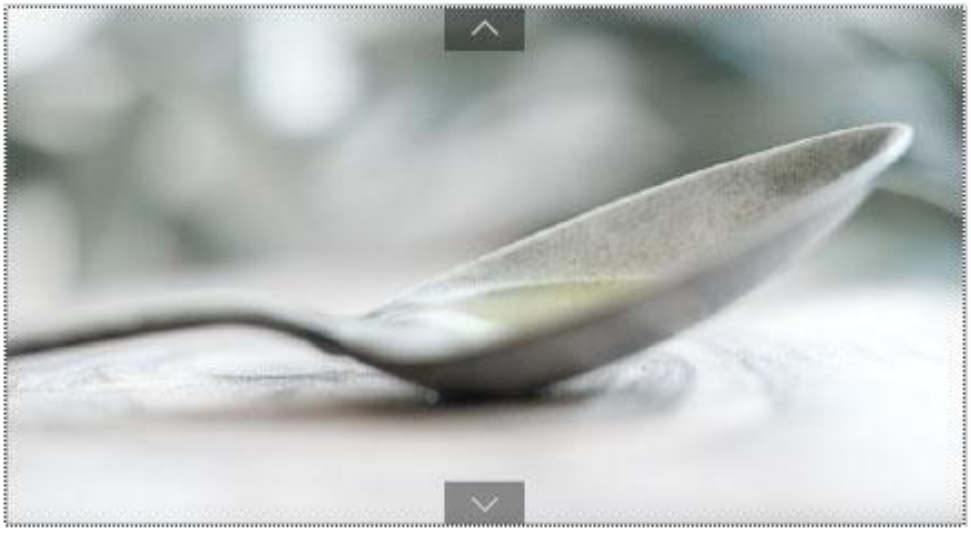
適切なコントロールの選択

FlipView は、小規模から中規模のコレクション (最大約 25 個の項目を含むコレクション) の画像を参照する場合に最適です。このようなコレクションの例として、製品の詳細ページ内の項目やフォト アルバム内の写真などがあります。多くの場合、大規模なコレクションで FlipView を使うことはお勧めしませんが、このコントロールは、フォト アルバム内の個々の画像を表示する際に役立ちます。

例

水平方向の閲覧、左端の項目から開始し、右にフリップするのが、FlipView の一般的なレイアウトです。このレイアウトは、すべてのデバイス上で縦方向でも横方向でも正常に動作します。

Flipview は、垂直方向にも閲覧できます。



コンテキスト インジケーターの追加

FlipView 内のコンテキスト インジケーターによって、便利な基準点を設けることができます。標準的なコンテキスト インジケーターに含まれるドットは、対話型です。この例に示されているように、最適な配置は通常、中央のギャラリーの下です。

比較的大きなコレクション (10 ～ 25 個の項目) の場合、より多くのコンテキストを提供するインジケーター (縮小表示のフィルムストリップなど) を使用することを検討します。 単純なドットを使用するコンテキスト インジケーターとは異なり、フィルム ストリップ内の各サムネイルは対応する画像の小さいバージョンであり、選択可能にする必要があります。

推奨事項

- FlipView は最大 25 個程度の項目のコレクションに最適です。
- 大規模なコレクションでは FlipView コントロールを使わないでください。これは、項目ごとにフリップ操作を繰り返す必要があり、ユーザーの負担になるためです。フォト アルバムは例外です。フォト アルバムには数百または数千の画像が含まれている場合があります。ほとんどの場合、フォト アルバムでは、グリッド ビューのレイアウトで写真を選ぶと、フリップ ビューに切り替わります。他の大きいコレクションについては、リスト ビューまたはグリッド ビューを検討してください。
- コンテキスト インジケーターの場合:
  - ドット (または選択した任意の視覚的マーカー) の順序は、中央に配置され、水平方向にパンするギャラリーの下にあるときに最適に動作します。
  - 垂直方向にパンするギャラリーでコンテキスト インジケーターを使う場合は、中央に配置され、画像の右側にあるときに最適な動作になります。
  - 強調表示されているドットは現在の項目を示します。通常、強調表示されているドットは白で、その他のドットは灰色で表されます。
  - ドットの数は変更できますが、多すぎるとユーザーは現在の場所を把握することが難しくなります。通常、表示するドットの最大数は 10 個です。

フライアウトのガイドライン

フライアウト (flyout) は、ユーザーが現在操作している内容に関係する UI を一時的に表示するために使われる軽量なポップアップです。このフライアウトは、メニューを表示する場合、非表示コントロールを表示する場合、項目の詳細を表示する場合、またはユーザーに操作の確認を求める場合に使うことができます。フライアウトは、ユーザーのタップまたはクリックのみに応じて表示する必要があります。フライアウトは、ユーザーがフライアウトの外側をタップしたときには必ず閉じます。

重要な API

Flyout クラス (XAML)　　Flyout オブジェクト (WinJS)

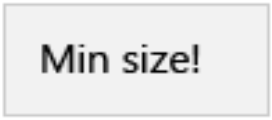
適切なコントロールの選択

フライアウトは次の目的で使うことができます。

- 一時的な状況依存の UI。
- 警告と確認 (被害が発生する可能性がある操作に関連する警告や確認など)。
- より詳細な情報の表示 (画面上の項目に関する詳しい情報など)。
- 第 2 レベルのメニュー。
- カスタム コマンド。

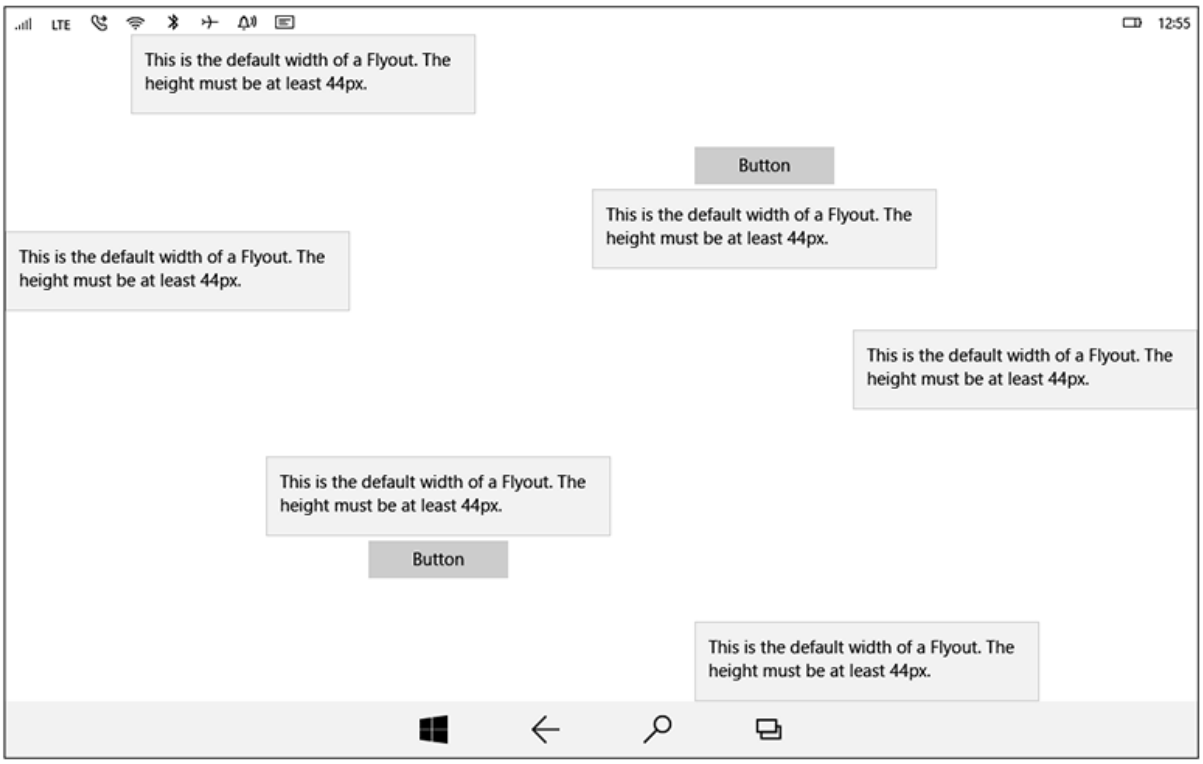
例

最小サイズのフライアウトには、ごく基本的なメッセージのみを含めることができますが、ボタンを含めることはできません。



次の例は、既定のフライアウトの幅を示しています。テキストは折り返されます。テキストの量がコンテナーのサイズを超えると、スクロール バーが表示されます。

この例は、フライアウトの画面上の配置とボタンの配置を示しています。



推奨事項

- 他の状況依存 UI と同様に、フライアウトは、呼び出した位置の横に配置します。
  - フライアウトの固定先とするオブジェクトと、そのオブジェクトのどの側面にフライアウトを表示するかを指定します。
  - フライアウトは、重要な UI の妨げにならないように配置してください。
- フライアウトの内容が選ばれたらすぐに、フライアウトは消えるようにします。

- フライアウト メニューは、1 レベルにするのが最適です。フライアウト メニューのレベルが複数であると、ナビゲーションが難しくなり、ユーザー エクスペリエンスが低下します。

ハブ コントロールのガイドライン

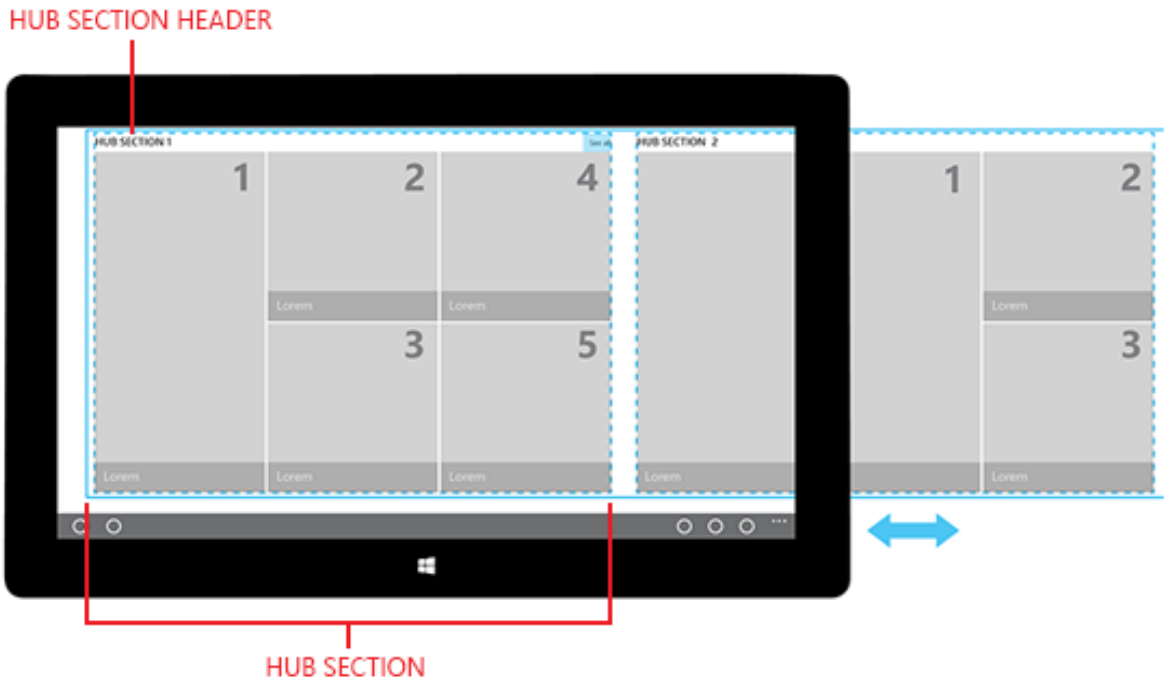
ハブ コントロールを使うと、アプリのコンテンツを、関連した別個のセクションやカテゴリに整理できます。ハブのセクションは、優先順に走査するためのものであり、さらに細かいエクスペリエンスを目的とした出発点として使うことができます。

重要な API

Hub クラス (XAML)

Hub オブジェクト (WinJS)

HubSection クラス (XAML)



ハブのコンテンツはパンする堅牢なビューに表示でき、新しい情報、入手可能な項目、関連する項目がひとめでわかります。ハブには通常ページ ヘッダーがありますが、複数の各コンテンツ セクションにはセクション ヘッダーがあります。

ハブ コントロールには、コンテンツのナビゲーション パターンを構築するのに適したいくつかの機能があります。

- **ビジュアルなナビゲーション**
  ハブを使うと、多様性のある簡潔なスキャンしやすい配列にコンテンツを表示できます。
- **分類**
  各ハブ セクションでは、コンテンツを論理的な順序に配置できます。
- **従来のサポート**
  ハブは、将来のデバイス ターゲットにもうまく変換されます。
- **混在したコンテンツの種類**
  コンテンツの種類が混在する場合、資産の可変サイズと比率は共通です。ハブを使うと、コンテンツの各種類を一意なものにし、各ハブ セクションに整然と配置できます。
- **ページとコンテンツの可変幅**
  パノラマ モデルであるハブでは、セクション幅を調整できます。これは異なる深度のあるコンテンツに役立ち、また項目数が少なくても多くても同様に適切に書式設定できます。
- **柔軟なアーキテクチャ**
  アプリのアーキテクチャを浅く維持する場合、すべてのチャネル コンテンツをハブ セクションの概要に収めることができます。

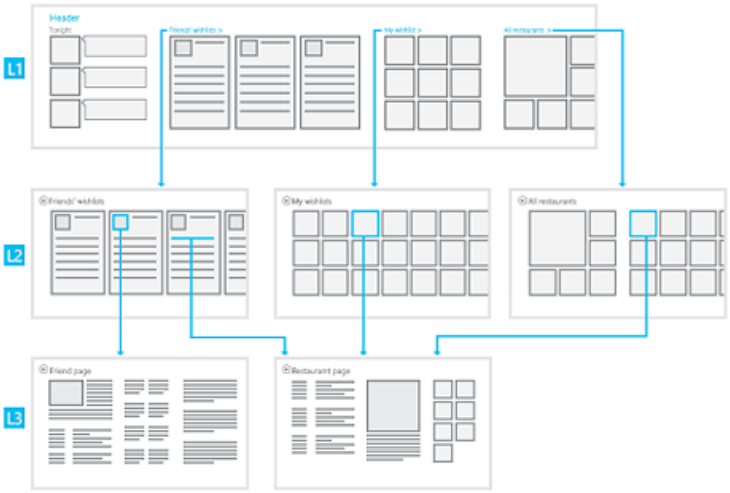
適切なコントロールの選択

ハブ コントロールは、コンテンツの階層関係を調整して表示するために使われます。ハブは、ストアやメディア コレクションにおいて有益なコンテンツや新しいコンテンツを見つけやすくすることを優先します。

ハブは、コンテンツに対するナビゲーションとしても使われます。ナビゲーション パターンについては、ナビゲーション デザインを参照してください。

ハブのアーキテクチャ

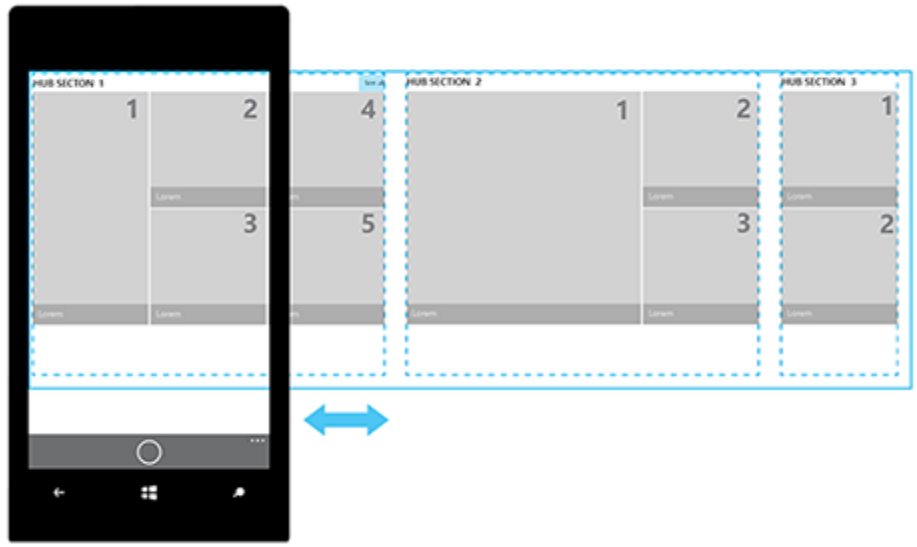
ハブ コントロールには、リレーショナル情報アーキテクチャを使ったアプリをサポートする階層型ナビゲーション パターンがあります。ハブはさまざまなカテゴリのコンテンツで構成され、それぞれがアプリのセクション ページに対応付けられています。セクション ページは、シナリオとセクションに含まれるコンテンツが最適に表現されるような形で表示できます。



レイアウトとパン / スクロール

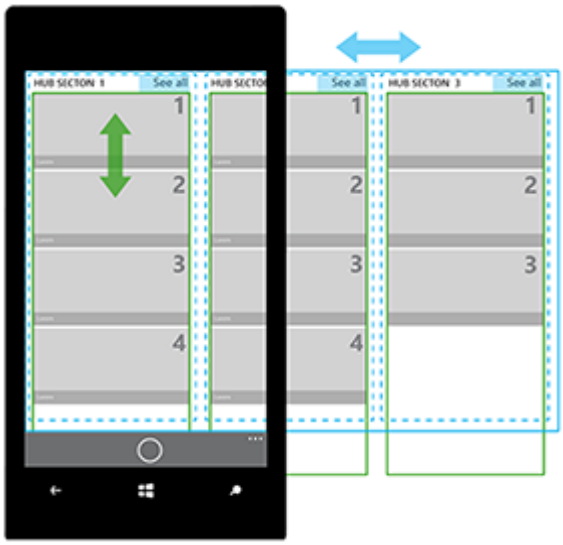
コンテンツをハブに配置して移動する方法はたくさんあります。ハブのコンテンツ リストが常にハブのスクロール方向に対して垂直にパンされる点だけ確認してください。

水平方向のパン

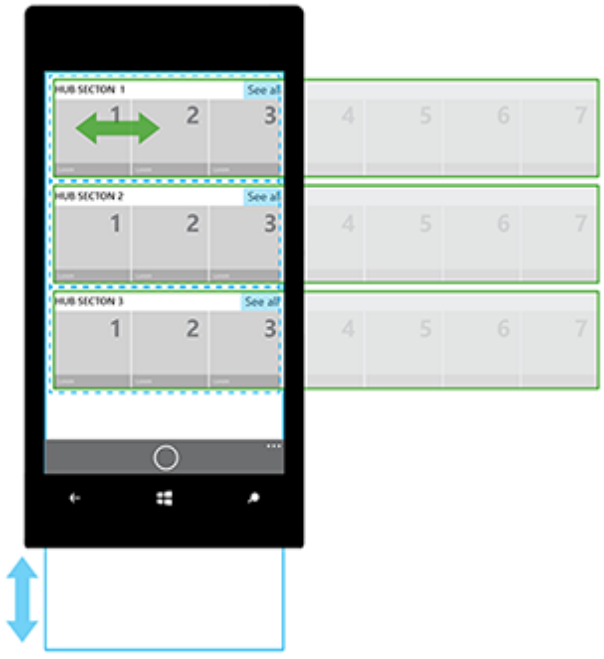


垂直方向のパン

垂直方向のリストとグリッドのスクロールによる水平方向のパン



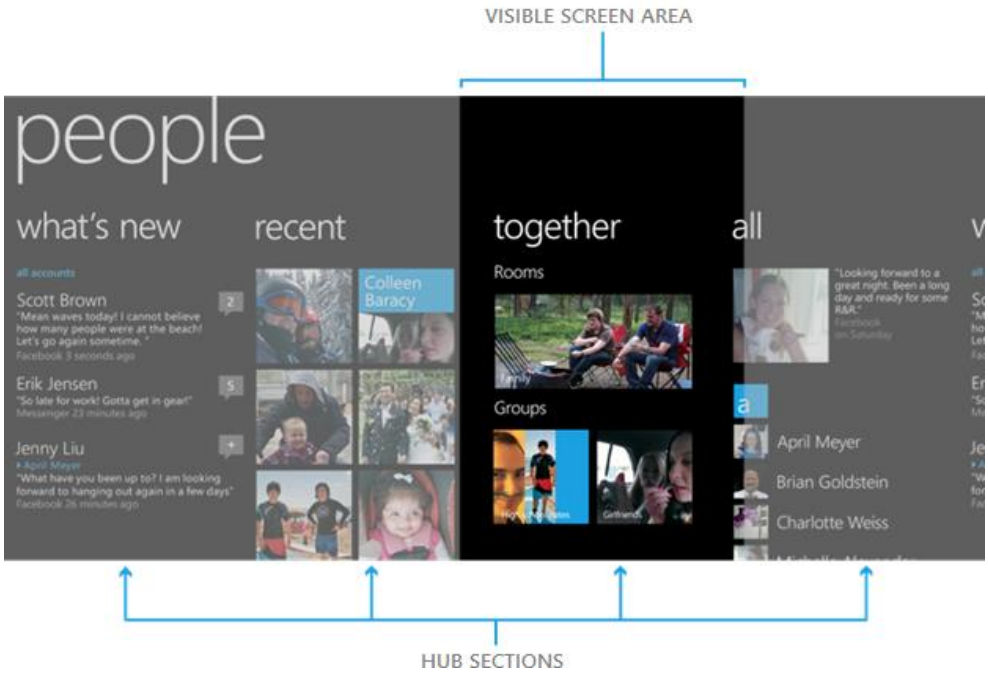
水平方向のリストとグリッドのスクロールによる垂直方向のパン



例

ハブは、設計上の大きな柔軟性を備えています。そのため、魅力的で視覚に訴えるさまざまなエクスペリエンスを提供するアプリを設計できます。最初のグループには、ヒーロー画像やコンテンツ セクションを使うことができます。ヒーローには、最も強調したい内容を失うことなく垂直方向と水平方向にトリミングできる大きい画像を使ってください。次の例は、1 つのヒーロー画像と、その画像を横長、縦長、狭い幅で表示するためにトリミングする方法を示しています。

モバイル デバイスでは、一度に 1 つのハブ セクションを表示できます。



推奨事項

- ハブ セクションに他のコンテンツがあることがユーザーにわかるように、一定量のコンテンツが見えるようにコンテンツをクリッピングすることをお勧めします。
- アプリの必要に基づいて、ハブ コントロールに複数のハブ セクションを追加し、各セクションに独自の機能目的を持たせることができます。たとえば、あるセクションには一連のリンクとコントロールを含め、別のセクションはサムネイルのリポジトリとすることができます。ユーザーは、ハブ コントロールに組み込まれているジェスチャのサポートを使って、これらのセクション間をパンすることができます。
- さまざまなウィンドウ サイズに対応できるように、コンテンツを動的に再配置するのが最適です。
- 多くのハブ セクションがある場合は、ハブにセマンティック ズームを追加することを検討します。これには、アプリが狭い幅に合わせてサイズ変更されたときにセクションを見つけやすくなるという利点もあります。
- ハブ セクション内の項目から別のハブに移動することはお勧めしません。代わりに、対話型ヘッダーを使って別のセクションまたはページに移動します。
- ハブ テンプレートを基盤として使って、アプリの特長を反映したレイアウトになるようにカスタマイズします。ハブ コントロールの次の機能をカスタマイズできます。
    - セクションの数
    - 各セクションのコンテンツの種類
    - セクションの配置と順序
    - セクションのサイズ
    - セクションとセクションの間隔
    - セクションとハブの上端または下端の間隔
    - ヘッダーとコンテンツのテキストのスタイルとサイズ
    - 背景、セクション、セクション ヘッダー、セクション コンテンツの色

ハイパー リンクのガイドライン

ユーザーがハイパーリンクを選ぶと、アプリの別の部分、別のアプリ、または別のブラウザー アプリを使って呼び出した URI (Uniform Resource Identifier) に移動します。ハイパーリンクは主に 2 種類あります。それらは、動的テキストとしてインラインで表示されるテキスト コントロールと、マークアップ テキストとして表示されるボタンです。

重要な API

Hyperlink クラス (XAML)

HyperlinkButton クラス (XAML)

a 要素 | a オブジェクト (HTML)

適切なコントロールの選択

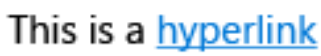
ユーザーがテキストを選び、そのテキストに関する詳しい情報が表示される場所に移動するとき、この操作に応答するテキストが必要となる場合に、ハイパーリンクを使います。

必要に応じて適切な種類のハイパーリンクを選んでください。

- 自動改行を必要とするが、大きいサイズのヒット ターゲットを必要としない場合は、テキスト ハイパーリンクを使います。多くの場合、ハイパーリンク テキストのサイズは小さく、ターゲットとして使うのが難しくなることがあります (特にタッチ操作の場合)。
- 自動改行を必要としない場合や、大きいサイズのヒット ターゲットのみを必要とする場合は、ボタンを使います。ボタンの高さは 44 epx です。

例

動的テキストでインライン表示されるテキスト要素としてのハイパーリンクの例を次に示します。



ボタンとしてのハイパーリンクの例を次に示します。これは、マークアップ テキストとして表示され、レイアウト コンテナー内に配置できます。

推奨事項

- ハイパーリンクを使う場合は、移動のみを目的としてください。他の操作のためにハイパーリンクは使わないでください。
- テキスト ベースのハイパーリンクには、書体見本の本文スタイルを使います。
- 個々のハイパーリンクの間には十分な間隔を空けます。これにより、それぞれのハイパーリンクを区別することができ、ハイパーリンクを間違えずに選ぶことができます。
- ユーザーの移動先を示すヒントをハイパーリンクに追加します。ユーザーが外部サイトに移動する場合は、ヒント内にトップレベルのドメイン名を入れ、補助的なフォント色を使ってそのテキストのスタイルを指定します。

ラベルのガイドライン

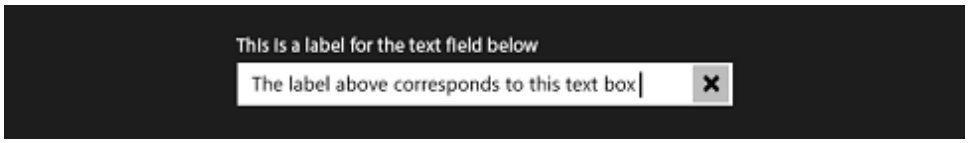
ラベルは、コントロールまたは関連するコントロールのグループの名前やタイトルです。

XAML では、多くのコントロールに組み込みの Header プロパティがあり、これを使ってラベルを表示します。Header プロパティがないコントロールの場合、またはコントロールのグループにラベルを付ける場合は、代わりに **TextBlock** を使います。

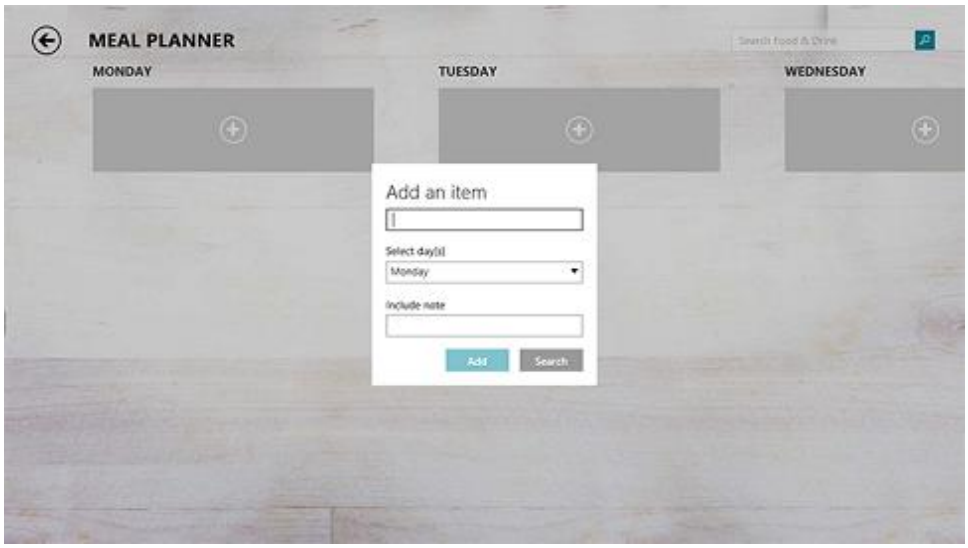
HTML では、**label element** を使います。

重要な API

label (HTML)



例



推奨事項

- 隣接するコントロールに入力する必要がある内容が明確ではない場合には、ユーザーへの説明のためにラベルを使います。また、関連するコントロールのグループに

ラベルを付けることや、関連するコントロールのグループの近くに説明テキストを表示することができます。

- コントロールにラベルを付ける場合、説明テキストの文ではなく、名詞や簡潔な名詞句のラベルを入力します。コロン、その他の句読点は使わないでください。
- ラベルにに説明テキストを入力するときは、テキスト文字列を長くすることができ、句読点も使うことができます。

リストのガイドライン

リストでは、コレクション ベースのアプリ コンテンツを、タッチに最適化された一貫した方法で表示して操作できます。この記事では、次の示す 4 種類のリスト パターンを説明します。

- リスト ビュー：主にテキストの多いコンテンツのコレクションを表示するために使います。
- グリッド ビュー：主に画像の多いコンテンツ コレクションを表示するために使います。
- ドロップダウン リスト：拡張可能なリストから、ユーザーが 1 つの項目を選択できます。
- リスト ボックス：スクロール可能なボックスから、ユーザーが 1 つまたは複数の項目を選択できます。

ここでは、各リスト パターンについて、設計のガイドライン、特徴、例を示します。

重要な API

ListView クラス (XAML)　　GridView クラス (XAML)

ComboBox クラス (XAML)

リスト ビュー

リスト ビューは、項目の分類、グループ ヘッダーの割り当て、項目のドラッグ アンド ドロップ、コンテンツの管理、項目の順序の変更を行うことができます。

適切なコントロールの選択

リスト ビューは、次の用途で使います。

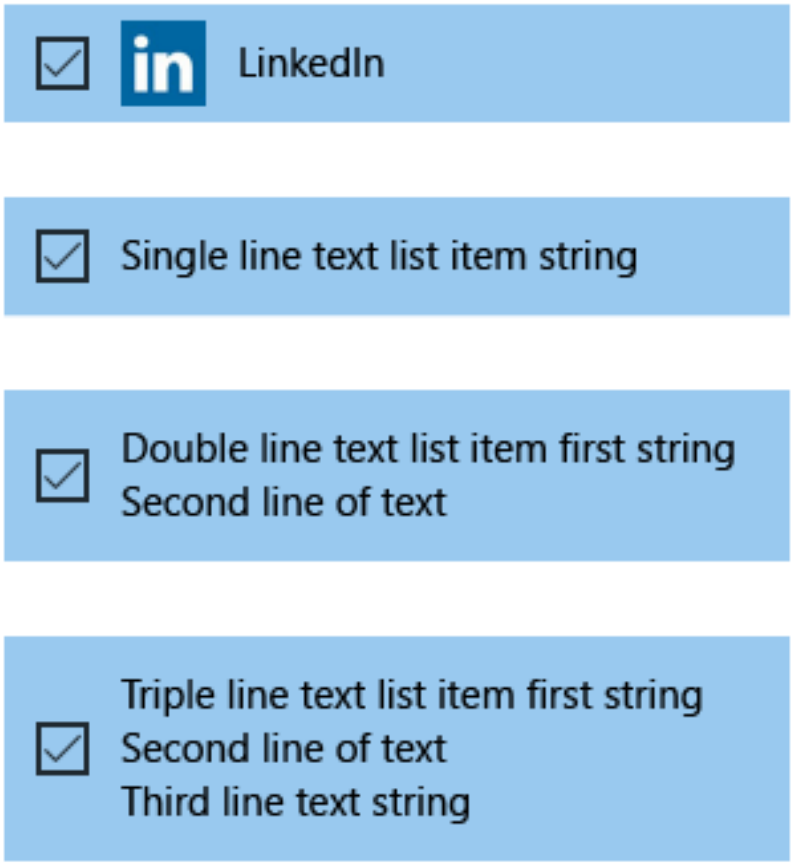
- 主にテキストで構成されるコンテンツのコレクションを表示する。
- コンテンツの単一のコレクションまたはカテゴリ別コレクションをナビゲートする。
- マスター/詳細パターンにマスター ウィンドウを作成する。マスター/詳細パターンは、メール アプリによく使われます。このパターンでは、選択できる項目の一覧を一方のウィンドウ (マスター) に表示し、選択された項目の詳細ビューをもう一方のウィンドウ (詳細) に表示します。

例

マスター/詳細パターンを使う場合、リスト ビューを使ってマスター ウィンドウを整理できます。マスター ウィンドウには、選択できる項目の一覧が表示されます。ユーザーがマスター ウィンドウで項目を選ぶと、選んだ項目の追加情報が詳細ウィンドウに表示されます。詳細ウィンドウには、よくグリッド ビューが含まれます。

複数のリストを組み合わせて、マスター/詳細の複雑な階層を作成することができます。詳しくは、「マスター/詳細パターン」をご覧ください。

この例のリスト レイアウトには、グループ ヘッダーがあり、1 列で表示されています。



推奨事項

- 同じリストに含まれる各項目の動作は同じにする必要があります。
- リストがグループに分割されている場合、グループ化されたコンテンツ内をユーザーが移動しやすくなるように、セマンティック ズームを使うことができます。

グリッド ビュー

グリッド ビューは、画像ベースのコンテンツのコレクションを配置および閲覧する場合に適しています。グリッド ビュー レイアウトでは、スクロールが垂直方向、パンが水平方向で行われます。項目は、左から右、上から下の読み取り順序で配置されます。

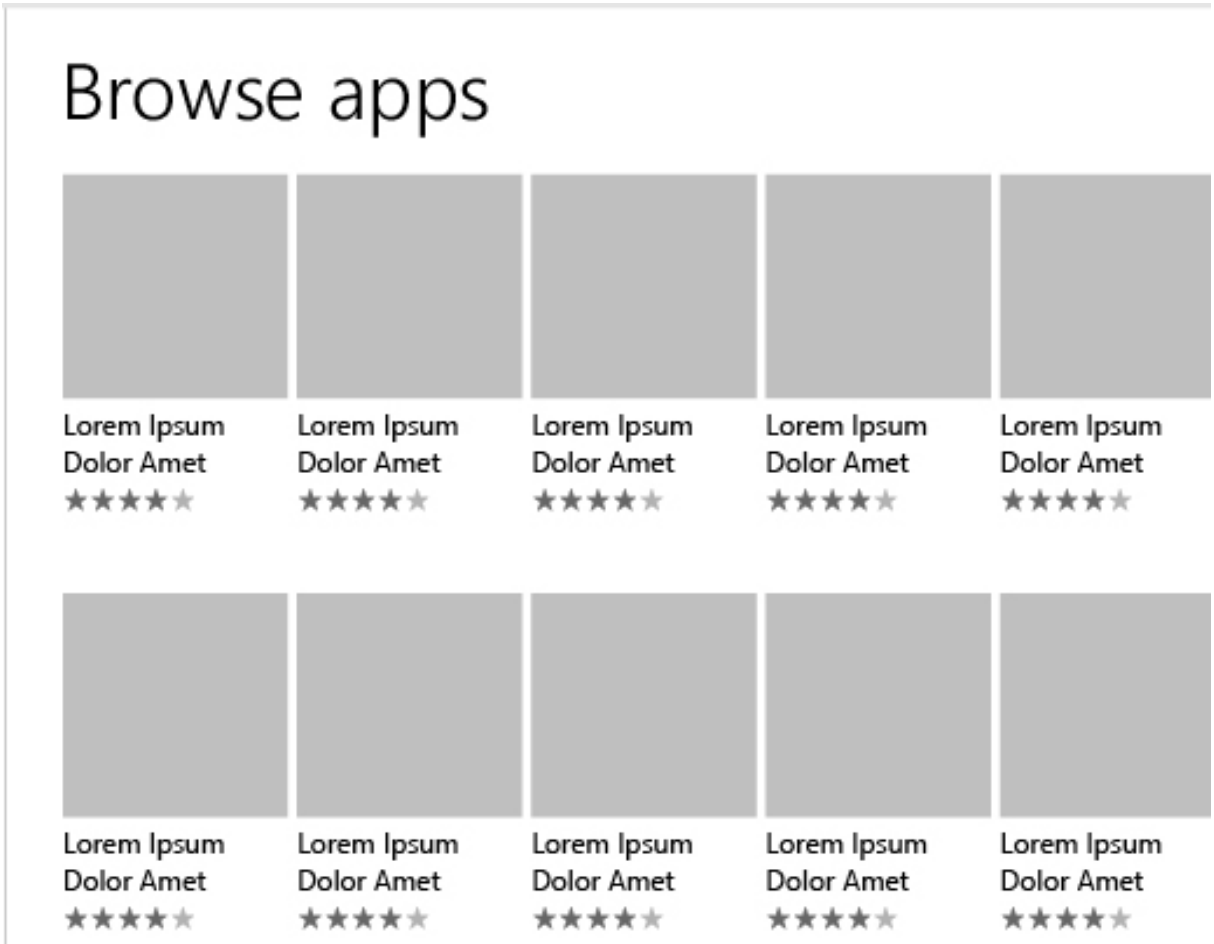
適切なコントロールの選択

グリッド ビューは次の用途で使います。

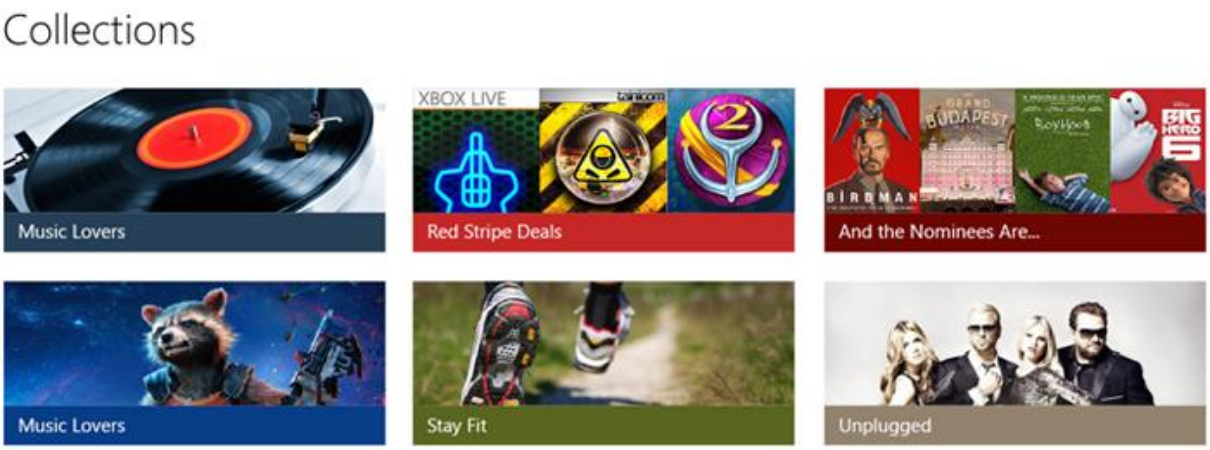
- 主に画像で構成されるコンテンツ コレクションを表示する。
- コンテンツ ライブラリを表示する。
- セマンティック ズームに関連付けられた 2 つのコンテンツ ビューの形式を設定する。

例

ここでは、アプリの参照用を例として、標準的なグリッド ビューのレイアウトを示します。グリッド ビュー項目のメタデータは通常、数行のテキストと項目の評価に制限されます。



グリッド ビューは、写真やビデオなどのメディアを表示するためによく使用される、コンテンツ ライブラリに最適なソリューションです。コンテンツ ライブラリでは、ユーザーが項目をタップして動作を開始します。



推奨事項

- 同じリストに含まれる各項目の動作は同じにする必要があります。
- リストがグループに分割されている場合、グループ化されたコンテンツ内をユーザーが移動しやすくなるように、セマンティック ズームを使うことができます。

ドロップ ダウン リスト

ドロップダウン リストはコンボ ボックスとも呼ばれます。最初はコンパクトな状態ですが、拡張して、選択可能な項目の一覧を表示することができます。ドロップダウン リストでは、単一項目の選択または複数選択がサポートされます。選択された項目は常に表示されます。表示されていない項目は、選択されている項目をタップすると表示されます。

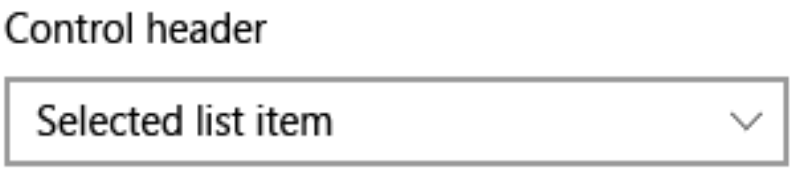
適切なコントロールの選択

- 1 行のテキストで十分に表すことができる項目のセットから単一の値をユーザーが選択できるようにするには、ドロップダウン リストを使います。
- 項目に複数行のテキストまたは画像が含まれる場合は、ドロップダウン リストではなくリスト ビューまたはグリッド ビューを使います。
- 項目が 5 個より少ない場合は、ラジオ ボタン (1 つの項目だけを選べる場合) またはチェック ボックス (複数の項目を選べる場合) を使用を検討します。
- 選択項目がアプリのフローにおいて二次的な重要性しか持たない場合に、ドロップダウン リストを使います。 ほとんどのユーザーのほとんどの状況で既定のオプシ

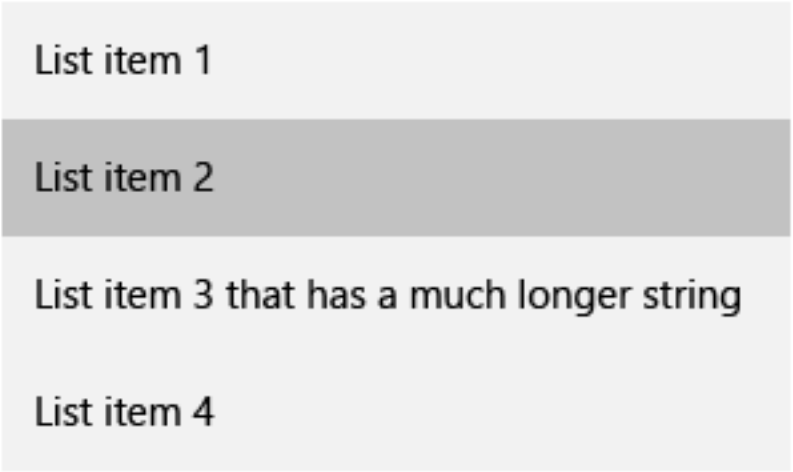
ョンがお勧めされている場合は、リスト ボックスを使ってすべての項目を表示すると、オプションに必要以上の注意を引いてしまう可能性があります。ドロップダウン リストを使うことで、領域を節約し、無駄な情報を最小限にすることができます。

例

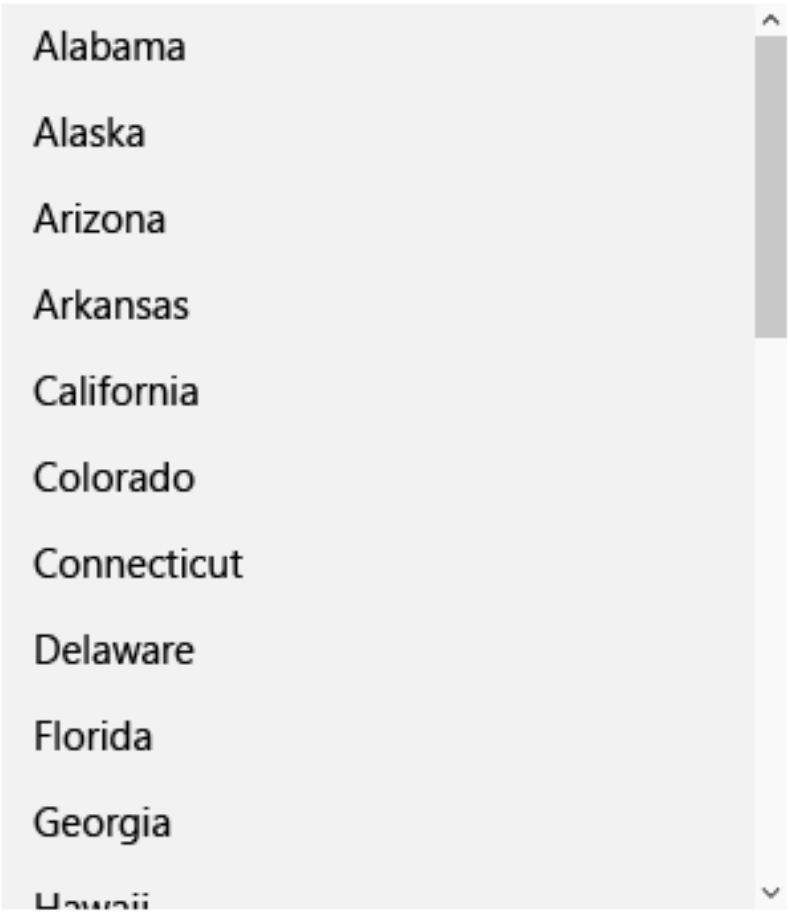
コンパクト状態のドロップ ダウン リストには、ヘッダーを表示します。



ドロップダウン リストは、長い文字列の幅をサポートするために拡張できますが、読みにくくなるような長すぎる文字列は避けます。



ドロップダウン リスト内のコレクションが一定の長さに達すると、対応できるようにスクロール バーが表示されます。リスト内の項目は論理的にグループ化します。



推奨

- ドロップダウン リスト項目のテキストのコンテンツは、単一行に制限します。
- ドロップダウン リスト内の項目は、最も論理的な順序に並べ替えます。関連するオプションをグループ化し、最も一般的なオプションを先頭に配置して、項目をアルファベット順に配置します。名前はアルファベット順、数値は数値順、日付は時系列順に並べ替えます。

リスト ボックス

リスト ボックスを使うと、ユーザーはコレクションから 1 つまたは複数の項目を選択できます。リスト ボックスはドロップダウン リストと似ていますが、常に開いている点がドロ

ップダウン リストと異なります。リスト ボックスには、コンパクトな (展開されていない) 状態がありません。すべての項目を表示する領域がない場合には、リスト内の項目をスクロールできます。

適切なコントロールの選択

- リスト ボックスは、リスト内の項目が重要であるため目立つように表示する場合や、項目一式を表示するための十分な画面領域がある場合に便利です。
- リスト ボックスでは、重要な選択で完全な代替セットにユーザーの注意を向ける必要があります。ドロップダウン リストの場合はまず、選択した項目にユーザーの注意を引き付けます。
- 次のような場合はリスト ボックスの使用を避けてください。
  - リスト内の項目が非常に少ない場合。単一選択のリスト ボックスで常に同じ 2 つのオプションを提示するのであれば、ラジオ ボタンの方が適している可能性があります。3 ～ 4 個の静的な項目を提示する場合もラジオ ボタンの使用を検討してください。
  - リスト ボックスが単一選択であり、リスト内のオプションが常に同じ 2 項目で、その一方が他方の否定を意味している場合 ("オン" と "オフ" など)。このような場合は、単一のチェック ボックスまたはトグル スイッチを使用してください。
  - 項目数が非常に多い場合。長いリストには、グリッド ビューまたはリスト ビューの方が適しています。グループ化されたデータの非常に長いリストの場合はセマンティック ズームの使用をお勧めします。
  - 項目が連続する数値である場合。このような場合は、スライダーの使用を検討してください。
  - 選択項目がアプリのフローで二次的な重要性しか持たないか、または大半の状況で大半のユーザーに既定のオプションが推奨される場合。このような場合は、ドロップダウン リストを使用してください。

推奨事項

- リスト ボックス内の項目数の最適な範囲は 3 ～ 9 です。

- リスト ボックスは、項目が動的に変化する可能性がある場合に適しています。
- 可能であれば、項目のリストのパンまたはスクロールが必要にならないように、リスト ボックスのサイズを設定します。
- リスト ボックスの目的、および現在選択されている項目が明確であることを確認します。
- タッチ フィードバックおよび項目の選択状態のビジュアル効果とアニメーションを予約します。
- リスト ボックス項目のテキストのコンテンツは、単一行に制限します。項目がビジュアルである場合、サイズをカスタマイズできます。項目に複数行のテキストかイメージが含まれる場合は、グリッド ビューまたはリスト ビューを使用してください。
- ブランドのガイドラインで別のフォントが指示されていない限り、既定のフォントを使います。
- コマンドの実行または他のコントロールの動的な表示と非表示の切り替えのためにリスト ボックスを使わないでください。

選択モード

選択モードでは、単一の項目または複数の項目を選択して、それらの項目に対して操作を実行できます。選択モードは、コンテキスト メニュー、[Crtl] キーまたは [Shift] キーを押しながらの項目のクリック、またはギャラリー ビューでの項目に対するターゲットのロールオーバーによって起動できます。選択モードがアクティブであるとき、各リスト項目の横にチェック ボックスを表示し、画面の上部または下部に操作を表示できます。

選択モードには、次の 3 つがあります。

- 単一 - ユーザーは同時に 1 つの項目だけを選ぶことができます。
- 複数 - ユーザーは修飾子を使わずに複数の項目を選ぶことができます。
- 拡張 - ユーザーは、[Shift] キーを押すなどの修飾子を使って複数の項目を選ぶことができます。

項目の任意の場所をタップすると、項目が選ばれます。コマンド バーの操作をタップすると、選択したすべての項目に影響します。項目が選ばれていない場合、コマンド バーの操作は [すべて選択] を除いて非アクティブになります。

選択モードには簡易非表示モデルがありません。選択モードがアクティブなフレームの外側をタップしても、モードを取り消すことはできません。これにより、モードが誤って非アクティブ化されることを防止できます。戻るボタンをクリックすることで、複数選択モードが終了します。

操作が選択されているときは、確認できるようにビジュアルに示します。特定の操作に対して (特に破棄を伴う削除などの操作に対して)、確認ダイアログを表示することを検討します。

選択モードは、選択モードをアクティブにしたページに限定され、そのページ以外の項目に影響を与えることはできません。

選択モードへのエントリ ポイントは、そのモードが影響を与えるコンテンツに対して並置する必要があります。

コマンド バーの推奨事項については、「コマンド バーのガイドライン」をご覧ください。

マスター/詳細パターンのガイドライン

マスター/詳細パターンには、コンテンツのマスター ウィンドウ (通常はリスト ビューも表示されます) と詳細ウィンドウがあります。マスター リストの項目を選ぶと、詳細ウィンドウが更新されます。このパターンは、メールやアドレス帳によく使われます。

適切なパターンの選択

次の場合は、マスター/詳細パターンが適しています。

- メール アプリ、アドレス帳、またはリスト/詳細レイアウトをベースとするアプリを構築する。
- 大きいコンテンツ コレクションを特定して優先順位を設定する。
- コンテキスト間を前後に移動しながら、リストから項目をすばやく追加および削除できるようにする。

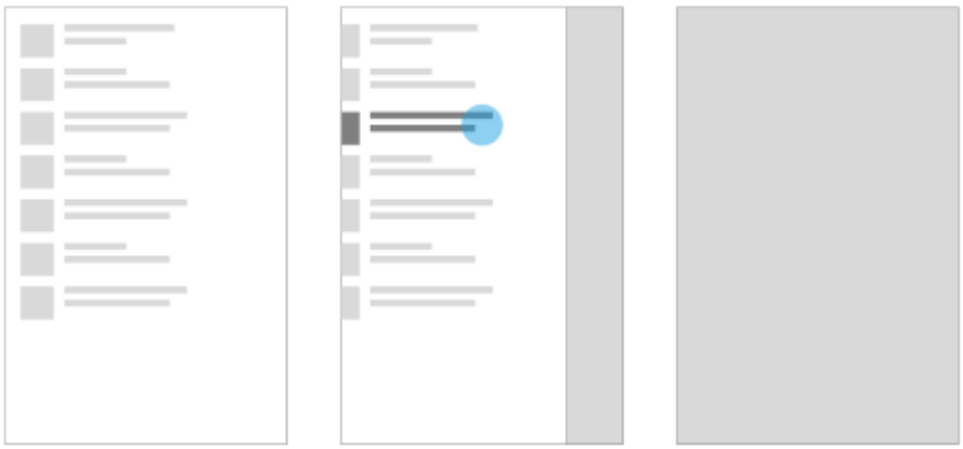
適切なスタイルの選択

マスター/詳細パターンを実装するとき、利用可能な画面領域の量に応じて、スタック スタイルまたは左右に並べるスタイルを使うことをお勧めします。

利用可能なウィンドウの幅	推奨スタイル
320 epx ～ 719 epx	スタック
720 epx 以上	左右に並べる

スタック スタイル

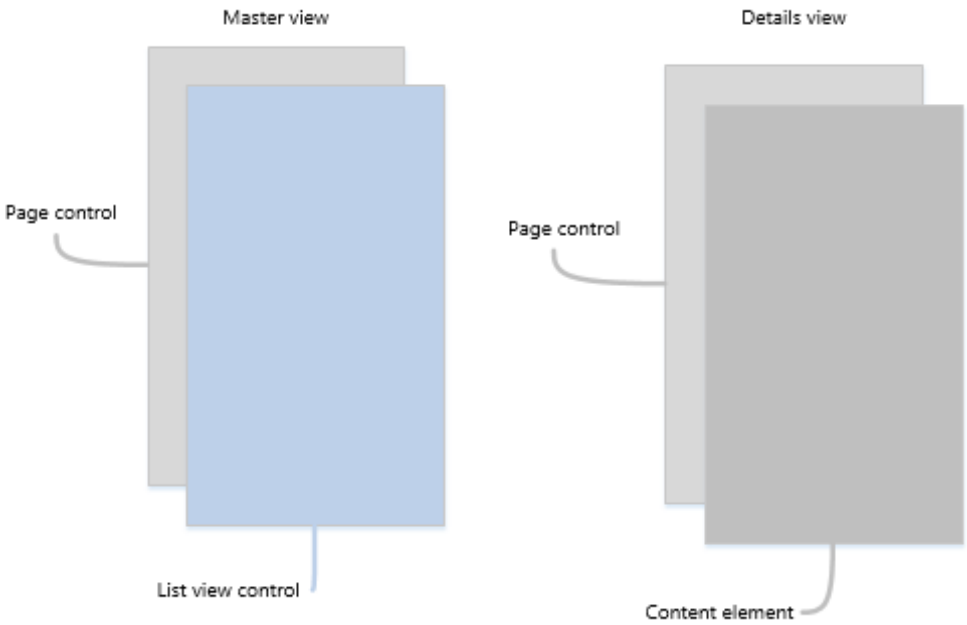
スタック スタイルでは、マスター ウィンドウと詳細ウィンドウのうち 1 つのウィンドウだけが一度に表示されます。



ユーザーがマスター ウィンドウで作業を始め、マスター リストで項目を選んで詳細ウィンドウに "ドリルダウン" します。ユーザーから見ると、マスター ビューと詳細ビューが別々の 2 つのページに存在するように表示されます。

スタック マスター/詳細パターンの作成

スタック マスター/詳細パターンを作る方法の 1 つは、マスター ウィンドウと詳細ウィンドウにそれぞれ別のページを使うことです。マスター リストを表示するリスト ビューを 1 つのページに配置し、詳細ウィンドウのコンテンツ要素を別のページに配置します。



マスター ウィンドウでは、イメージとテキストが含まれるリストを表示するのにリスト ビュー コントロールが適しています。

詳細ウィンドウの場合、最も意味のあるコンテンツ要素を使います。多くの個別フィールドがある場合は、グリッド レイアウトを使って要素をフォームに配置することを検討します。

左右に並べるスタイル

横に並べるスタイルでは、マスター ウィンドウと詳細ウィンドウを同時に表示できます。

マスター ウィンドウのリストは、現在選択されている項目を示すために選択ビジュアルを使用します。マスター リストで新しい項目を選ぶと、詳細ウィンドウが更新されます。

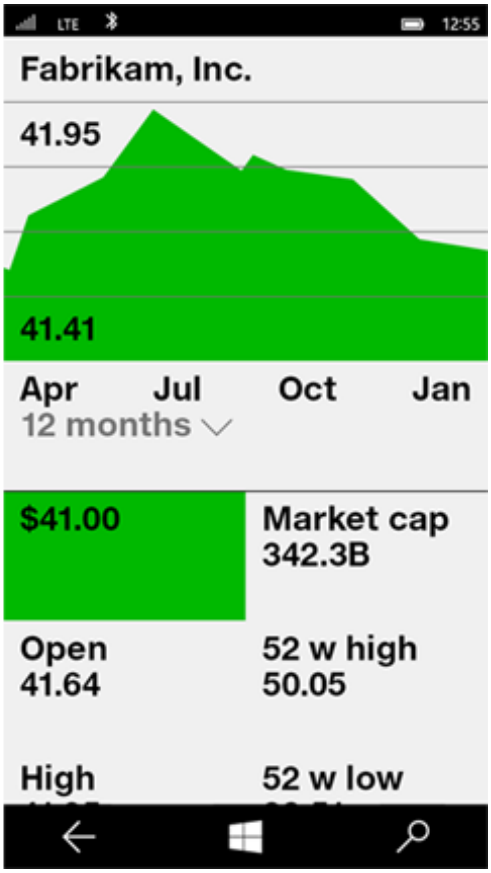
左右に並べるマスター/詳細パターンの作成

マスター ウィンドウでは、イメージとテキストが含まれるリストを表示するのにリスト ビュー コントロールが適しています。

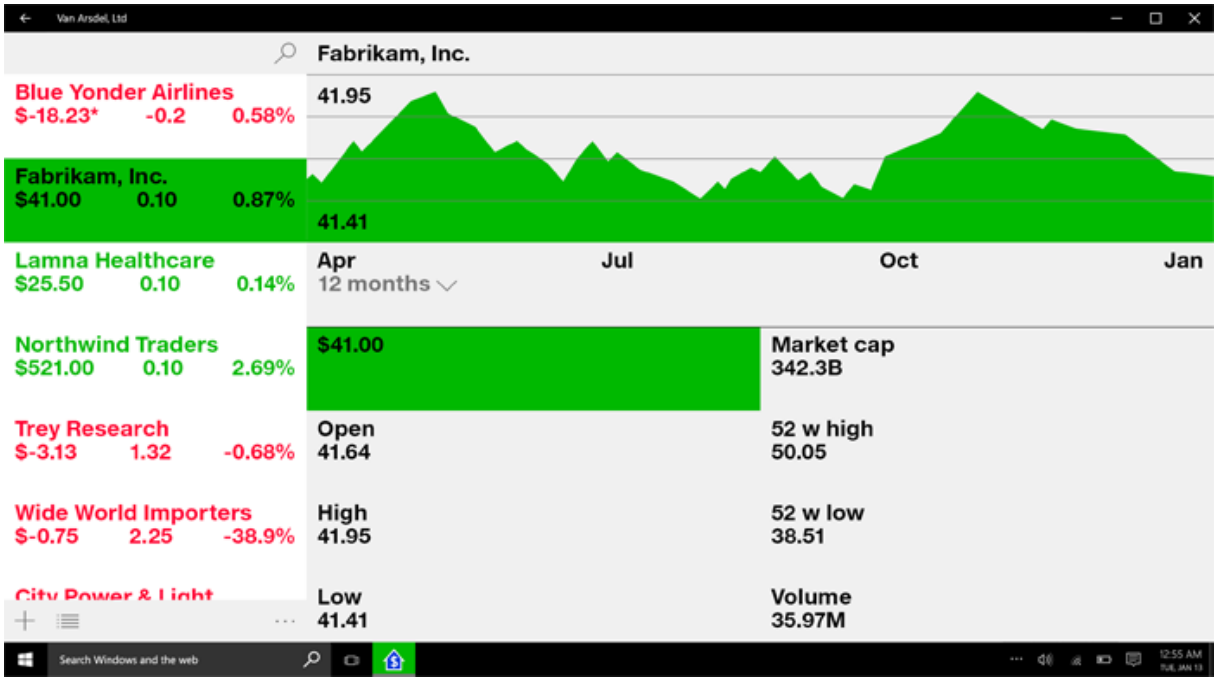
詳細ウィンドウの場合、最も意味のあるコンテンツ要素を使います。多くの個別フィールドがある場合は、グリッド レイアウトを使って要素をフォームに配置することを検討します。

例

株式市場を追跡するアプリの設計では、マスター/詳細パターンを使います。次のアプリの例は、携帯電話に表示されているため、左側にマスター ウィンドウ/リストが、右側に詳細ウィンドウが表示されています。



株式市場を追跡するアプリの設計では、マスター/詳細パターンを使います。次のアプリの例は、デスクトップ PC に表示されているため、マスター ウィンドウ/リストと詳細ウィンドウがどちらも全画面に表示されています。マスター ウィンドウの上部には検索ボックスがあり、下部にはコマンド バーがあります。



メディア プレーヤーのガイドライン

ビデオ、オーディオ、および画像を表示したり聴いたりするには、メディア プレーヤーを使います。メディアはインラインで (ページに埋め込むか、その他のコントロールのグループを使う) 再生するか、専用の全画面表示で再生できます。

重要な API

MediaTransportControls クラス (XAML)

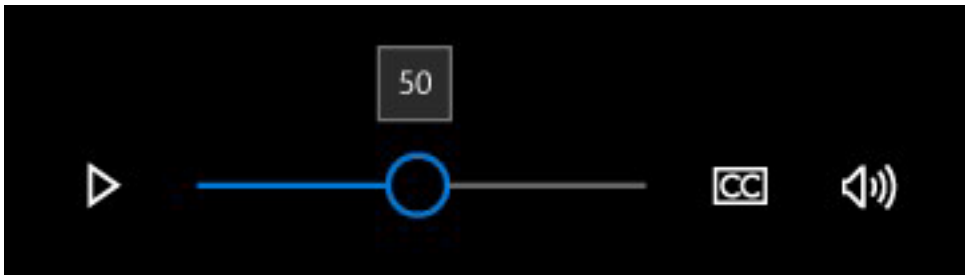
audio (HTML)
video (HTML)

適切なコントロールの選択

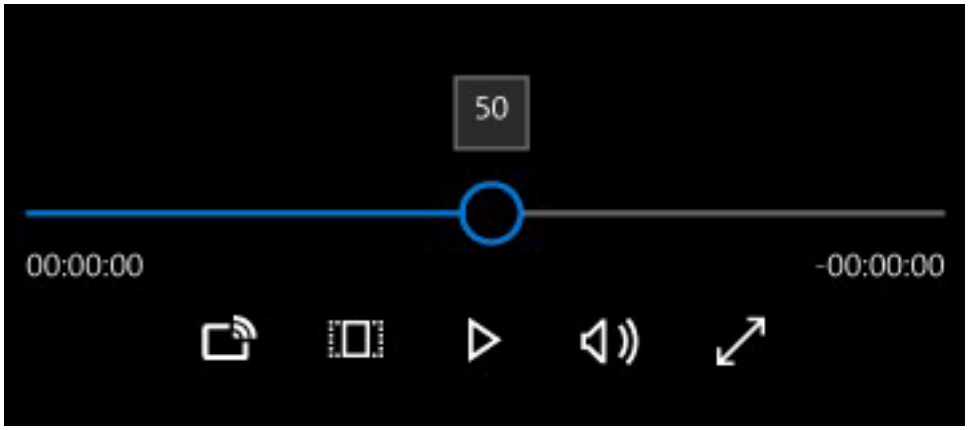
アプリでオーディオまたはビデオを再生する場合は、メディア プレーヤーを使います。画像のコレクションを表示する場合は、FlipView を使います。

例

メディア プレーヤーは 1 行および 2 行のコントロールのレイアウトをサポートします。最初の例は、メディアのタイムラインの左側に再生/一時停止ボタンを配置した 1 行のレイアウトです。このレイアウトは、コンパクトな画面に適しています。



ほとんどの使用シナリオ (特に大きな画面) では、2 行のコントロールのレイアウト (次の図) をお勧めします。このレイアウトでは、コントロールの領域がより多く確保されており、ユーザーが簡単にタイムラインを操作できます。



既定のボタンとオプション

メディア プレイヤーには標準的なボタンのセットが用意されており、ボタンが表示されるかどうかはデバイスによって異なります。

カスタマイズ

プレーヤーのボタン セットやコントロール バーの背景を変更したり、必要に応じてレイアウトを整理したりできます。ユーザーが必要とするのは基本的なコントロール セット (再生/一時停止、巻き戻し、早送り) です。

省略可能なレイアウト

プレーヤーには、標準的なメディア再生だけでなく、次の目的で使う 3 つの代替レイアウトが用意されています。

- 早送りおよび巻き戻し中にフレームを表示する
- メディア コンテンツの再生が完了した後に表示される "次の再生コンテンツ" 画面

推奨事項

メディア プレーヤーには濃色テーマと淡色テーマがありますが、ほとんどの場合は濃色テーマを選びます。暗い背景を使うと、(特に高感度条件では) コントラストが強調され、表示エクスペリエンスに影響を及ぼすコントロール バーが制限されます。

インライン モードで全画面表示モードのレベルを上げて、専用の表示エクスペリエンスを使うことをお勧めします。全画面表示エクスペリエンスが最適であり、インライン モードではオプションが制限されます。

画面領域がある場合は、2 行のレイアウトを採用します。このレイアウトでは、コンパクトな 1 行のレイアウトよりもコントロールの領域が多く確保されます。

アプリに最適なエクスペリエンスを実現するには、以下の点に注意して、必要なカスタム オプションをメディア プレーヤーに追加します。

- メディア再生エクスペリエンス用に最適化されている既定のコントロールのカスタマイズを制限します。
- 携帯電話およびその他のモバイル デバイスでは、デバイスのクロムは黒のままですが、ノート PC やデスクトップ PC では、デバイスのクロムはユーザーのテーマ カラーを継承します。
- 多数のオプションを含むコントロール バーをオーバーロードしないでください。
- メディアのタイムラインを既定の最小サイズよりも縮小しないでください。この操作を行うと、タイムラインの効果が大きく制限されます。

ナビゲーション ウィンドウのガイドライン

ナビゲーション ウィンドウは、さまざまなトップレベルのナビゲーション項目を使うことができるパターンです。これにより、画面領域を節約することができます。ナビゲーション ウィンドウはモバイル アプリに広く使われていますが、大きい画面でも適切に機能します。オーバーレイとして使うと、ユーザーがボタンを押すまでウィンドウは折りたたまれたままで邪魔にならないため、小さい画面で便利です。固定モードで使うと、ウィンドウは開いたままであるため、十分な画面領域がある場合に便利です。

重要な API

SplitView クラス (XAML)

SplitView オブジェクト (WinJS)

適切なパターンの選択

ナビゲーション ウィンドウでは、次の場合に適してします。

- 同じクラス内にトップレベルのナビゲーション項目が多数あるアプリ (フットボール、野球、バスケットボール、サッカーといったカテゴリがあるスポーツ アプリなど)。
- アプリ間で一貫性のあるナビゲーション エクスペリエンスを提供する場合 (ただし、ウィンドウ内にナビゲーション要素だけを配置する場合に限る)。
- トップレベルのナビゲーションのカテゴリの数が中程度から多い (5 ～ 10 以上) 場合。
- 画面領域を節約する場合 (オーバーレイとして)。
- ナビゲーション項目のアクセス頻度が低い場合 (オーバーレイとして)。
- ドラッグ アンド ドロップのシナリオ (固定の場合)。

ナビゲーション ウィンドウの構築

ナビゲーション ウィンドウ パターンは、ボタン、ナビゲーションのカテゴリ用のウィンドウ、およびコンテンツ領域で構成されます。ナビゲーション ウィンドウを構築する最も簡単な方法は、SplitView コントロールを使う方法です。このコントロールには、空のウィンドウとコンテンツ領域 (常に表示) が用意されています。ウィンドウは表示することも非表示にすることもでき、アプリ ウィンドウの右側または左側から表示できます。

SplitView コントロールを使わずにナビゲーション ウィンドウを構築する場合、3 つの主要なコンポーネント (ボタン、ウィンドウ、コンテンツ領域) が必要です。ボタンは、ウィンドウを開閉できるようにします。ウィンドウは、ナビゲーション要素のコンテナーです。コンテンツ領域に、選んだナビゲーション項目からの情報が表示されます。また、ナビゲーション ウィンドウは固定モードで表示することもできます。この場合、ウィンドウは常に表示され、ボタンは必要ありません。

ボタン

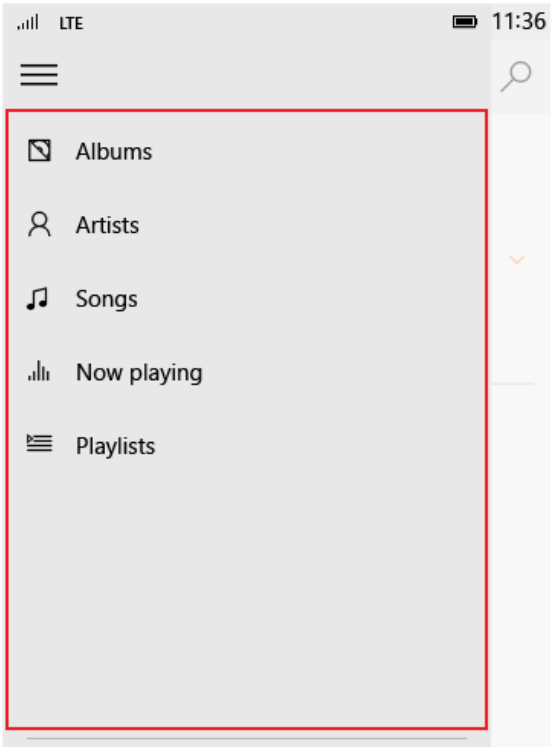
既定では、ナビゲーション ウィンドウのボタンは積み重ねられた 3 本の横線として表示されます。このボタンは、一般的には "ハンバーガー" ボタンと呼ばれます。ナビゲーション ウィンドウのボタンを使うと、必要なときにウィンドウを開いたり閉じたりすることができます。また、このボタンはウィンドウと共に移動されません。ナビゲーション ウィンドウ

のボタンはアプリの左上隅に配置することをお勧めします。ナビゲーション ウィンドウのボタンは、ウィンドウと共に移動されません。

通常、ナビゲーション ウィンドウのボタンはテキスト文字列に関連付けられています。アプリの最上部では、ナビゲーション ウィンドウのボタンの横にアプリのタイトルを表示できます。アプリの下部では、テキスト文字列をユーザーが現在表示しているページに関連付けることができます。

ウィンドウ

ナビゲーションのカテゴリのヘッダーがウィンドウに移動します。アプリ設定とアカウント管理へのエントリ ポイント (該当する場合) もウィンドウに移動します。ナビゲーション ヘッダーは、トップレベルであるか、入れ子になったトップレベル/第 2 レベルです。



コンテンツ領域

コンテンツ領域は、選んだナビゲーション位置の情報が表示される場所です。個々の要素やその他のサブレベル ナビゲーションを含めることができます。

ナビゲーション ウィンドウのバリエーション

ナビゲーション ウィンドウには、オーバーレイと固定の 2 つのバリエーションがあります。オーバーレイは必要に応じて折りたたまれたり展開されたりします。固定ウィンドウは既定で開いたままです。

オーバーレイ

- オーバーレイは、画面サイズに関係なく、縦方向でも横方向でも使うことができます。既定の (折りたたまれた) 状態では、オーバーレイは領域を消費せず、ボタンだけが表示されます。
- 画面領域を節約するオンデマンドのナビゲーションを提供します。携帯電話やファブレット上のアプリに最適です。
- ウィンドウは既定で非表示になり、ボタンだけが表示されます。
- ナビゲーション ウィンドウのボタンを押すと、オーバーレイの開閉を切り替えることができます。
- 展開された状態は一時的なものであるため、選択が行った場合、[戻る] ボタンを使った場合、またはユーザーがウィンドウの外部でタップした場合に消滅します。
- オーバーレイは、コンテンツの上に描画されます。コンテンツを再配置するわけではありません。

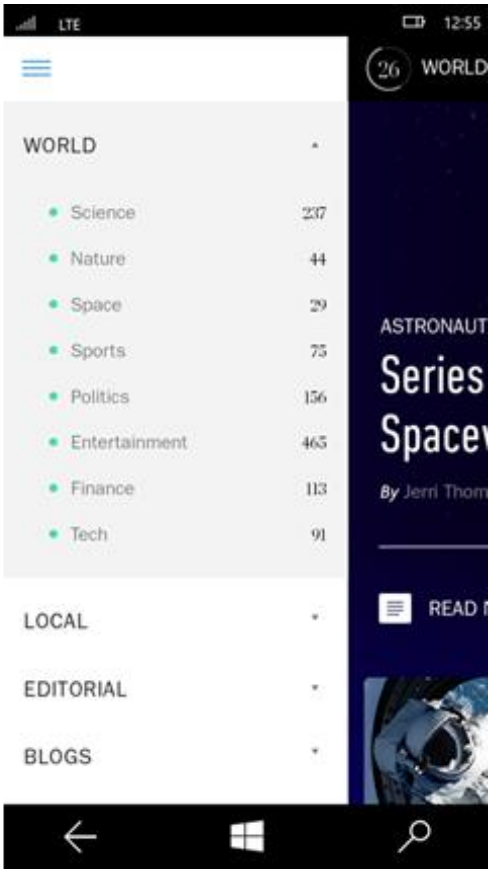
固定

- ナビゲーション ウィンドウは開いたままです。このモードは、目安としてタブレット以上のサイズの大きい画面に適しています。
- 横方向の場合、固定状態を有効活用できる最小画面幅は 720 epx です。このサイズで固定状態にするは、コンテンツの拡大/縮小に特別な注意が必要です。
- ウィンドウ間でのドラッグ アンド ドロップ シナリオがサポートされます。

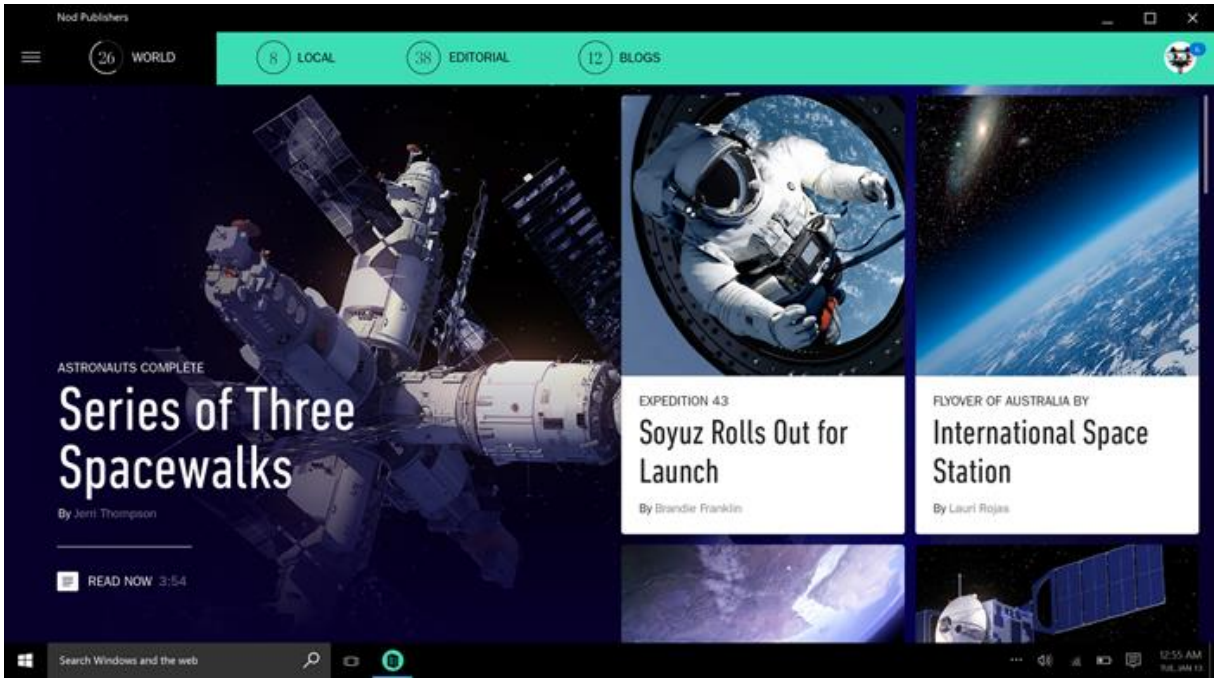
- ナビゲーション ウィンドウのボタンは、この状態には必要ありません。ボタンを使った場合、コンテンツ領域が押し出され、その領域内のコンテンツが再配置されます。
- ナビゲーション ツリーにおけるユーザーの位置を強調表示するため、選択内容を一覧の項目に表示する必要があります。
- デバイスの幅が狭すぎて縦方向では固定ウィンドウを表示できない場合、デバイスの回転時に次の動作をお勧めします。
    - 横向きから縦向き: ウィンドウが折りたたまれて、オーバーレイ状態または最小化状態になります。
    - 縦向きから横向き: ウィンドウが再び表示されます。

例

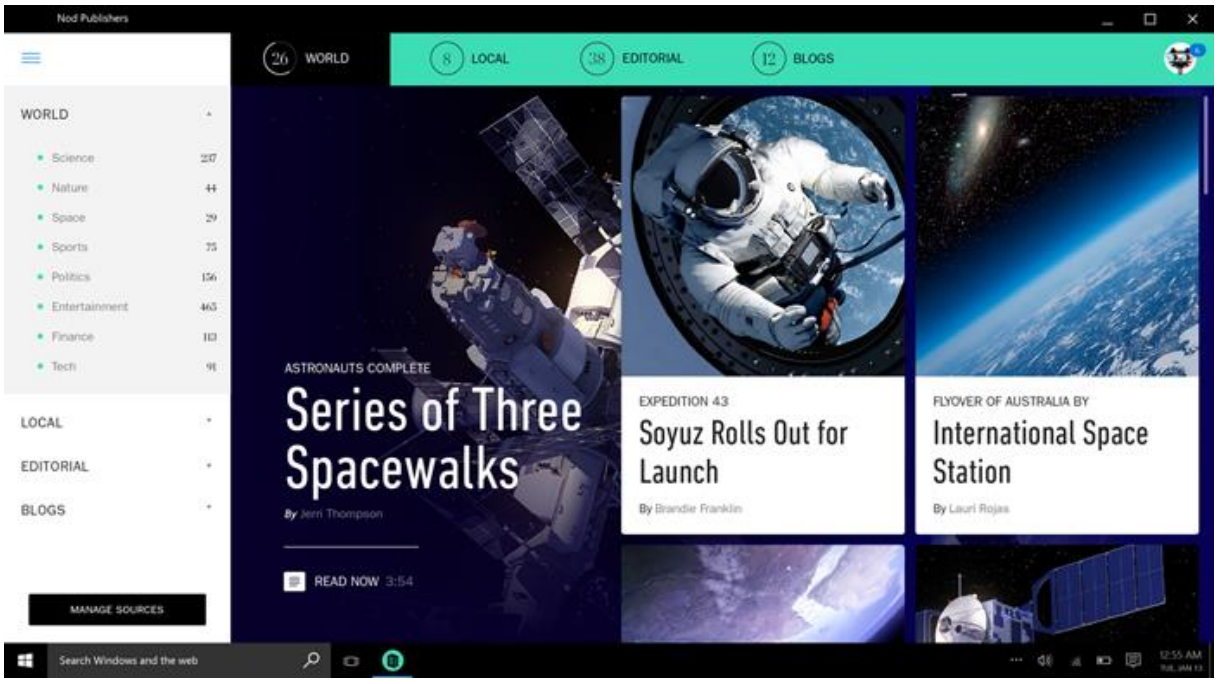
このニュース アプリのデザインでは、ナビゲーション ウィンドウが折りたたまれている場合 (左) と展開されている場合 (右) があります。展開されたウィンドウの場合、リスト ビュー内の各コンテンツ カテゴリ (World、Local など) には、より詳細なナビゲーションのためのサブカテゴリがあります。



次に示す例は、ノート PC やデスクトップ PC に表示される同じニュース アプリのデザインです。この例では、ナビゲーション ウィンドウが折りたたまれており、画面領域を消費しません。

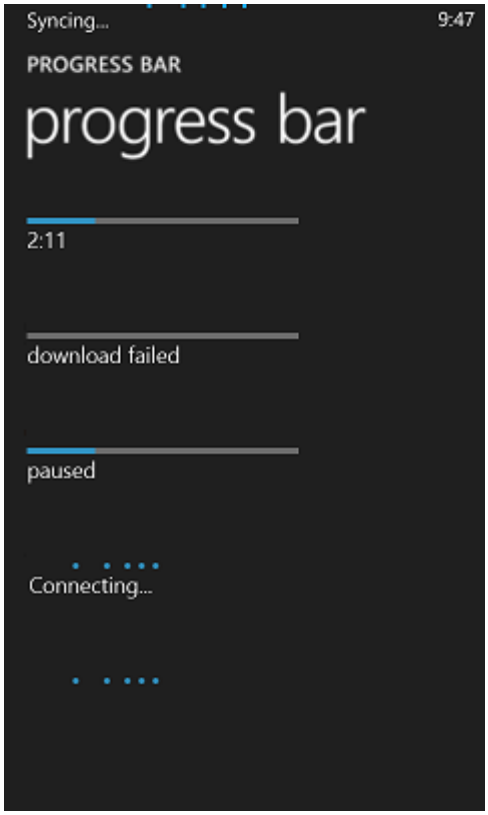


この例では、ナビゲーション ウィンドウが展開され、コンテンツが右側に押される形になっています。このビューでは、ウィンドウの下部に [Manage resources] ボタンが表示されています。ここには、アプリの設定に移動するためのボタンや定期的にアクセスしない類似のページを配置できます。



プログレス コントロールのガイドライン

Windows アプリ：進行状況不定バー、進行状況リング、進行状況確定バー



Windows Phone アプリ：ステータス バーの進行状況インジケーターと進行状況バー

重要な API

[ProgressBar クラス (XAML)](#)

[ProgressRing クラス (XAML)](#)

[progress (HTML)](#)

説明

プログレス コントロールは、時間のかかる操作が進行中であることを示すフィードバックをユーザーに返します。進行状況確定バーは、操作の完了割合を示します。進行状況不定バーと進行状況リングは、操作が進行中であることを示します。

プログレス コントロールは読み取り専用です。対話型ではありません。

例

スプラッシュ スクリーンの進行状況不定リング コントロールの例を次に示します。

進行状況バーは、状態や位置を示す優れたインジケーターでもあります。音楽トラックで使われる進行状況バーは、曲のタイムラインに対応しています。バーの値は、曲の位置を示します。一時停止状態は、再生が一時停止されていることを示します。

適切なコントロールの選択

常にプログレス コントロールを示す必要があるわけではありません。タスクが進行中であることが十分に明白であったり、タスクがすぐに完了するのでプログレス コントロールを表示するとわずらわしかったりする場合もあります。プログレス コントロールを表示するかどうかを判断するときには、次のような点を考慮します。

- **操作を完了するまでに 2 秒より長くかかるか**

そうである場合は、操作を開始したらすぐに、プログレス コントロールを表示します。操作の完了までに 2 秒より長い時間が通常かかるとしても、2 秒未満で完了することもある場合は、ちらつきを避けるために、500 ミリ秒待機してからコントロールを表示します。

- **操作は、ユーザーがタスクを完了するのを待っているか?**
  そうである場合は、プログレス バーを使いません。プログレス バーは、ユーザーの作業ではなく、コンピューターの作業の進行状況を示すものです。
- **何かが行われていることをユーザーが知る必要があるか?**
  たとえば、アプリがバックグラウンドで何かをダウンロードしていて、ダウンロードを開始したのがユーザーでない場合、ユーザーはそのことを知る必要がありません。
- **操作が、ユーザーのアクティビティをブロックしないバックグラウンド アクティビティであり、ユーザーにはほとんど関与しない (少しだけ関与する) か?**
  アプリが、常に見えている必要はないものの、進行状況を表示する必要はあるタスクを実行している場合は、テキストと省略記号を使います。

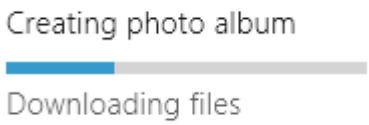
  Sharing in progress...

  タスクが進行中であることを示すために、省略記号を使います。複数のタスクまたは項目がある場合は、残りのタスクの数を示すことができます。すべてのタスクが完了したら、インジケーターを消します。
- **進行状況をビジュアル化するために、操作からのコンテンツを使えるか?**
  使える場合は、プログレス コントロールを表示しません。 たとえば、ディスクから読み込まれる画像を表示する場合は、画像が読み込まれるたびに、画面に 1 つずつ画像が表示されます。プログレス コントロールを表示しても、余分な UI が増えるだけで、何の利点もありません。
- **操作の進行中に、作業が全体に対してどの程度完了したかを決定できるか?**
  その場合は進行状況確定バーを使います。ユーザーをブロックする作業の場合は特に使ってください。それ以外の場合は、進行状況不定バーまたは進行状況リングを使います。ユーザーは何かが行われていることを知るだけですが、それでも十分に有用です。

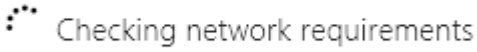
推奨と非推奨

プログレス コントロールには、次の 3 つのスタイルがあります。

- タスクが確定的である場合、つまり、継続時間が明確に定義されていたり、終了が予測可能だったりする場合は、進行状況確定バーを使います。たとえば、時間単位、バイト単位、ファイル単位などの定量化できる測定単位で残りの作業量を推定できる場合は、進行状況確定バーを使います。確定的なタスクには、次のような例があります。
    - アプリが 500 k の写真をダウンロードしていて、これまでに 100 k を受信した。
    - アプリが 15 秒の広告を表示していて、2 秒が経過した。



- ユーザーの操作をブロックするモーダルな確定的でないタスクでは、進行状況不定バーを使います。



- ユーザーの操作をブロックしない非モーダルな確定的でないタスクでは、進行状況不定バーを使います。

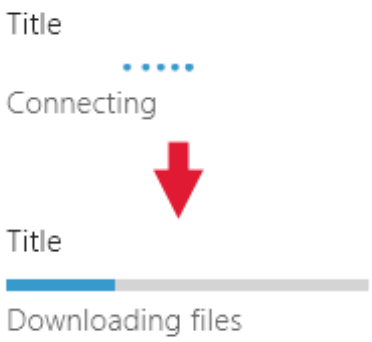
- 一部モーダルなタスクで、モーダルな状態が 2 秒未満である場合は、非モーダルなタスクとして扱います。 一部のタスクは、処理がある程度進むまで操作をブロックし、その後はユーザーがアプリの操作を始められるようになります。 たとえば、ユーザーが検索クエリを実行した場合、最初の結果が表示されるまでは操作がブロックされます。このようなタスクは、モーダルな状態が 2 秒未満である場合は非モーダルとして扱い、進行状況不定バー スタイルを使います。 モーダルな状態が 2 秒より長く続く場合は、タスクのモーダルなフェーズでは進行状況不定リングを使い、非モーダルなフェーズでは進行状況不定バーを使います。

- 進行中の操作をキャンセルするか一時停止するための方法を用意することを検討します。操作が完了するまでユーザーがブロックされるときに、操作の残りの実行時間が明らかな場合は、特に考慮してください。
- アクティビティを示すために、"待機カーソル" は使いません。システムの操作にタッチを使っているユーザーには表示されず、マウスを使っているユーザーにはアクティビティをビジュアル化する方法が 2 つ (カーソルとプログレス コントロール) は必要ないためです。
- 複数のアクティブな関連するタスクには、1 つのプログレス コントロールを表示します。 画面に複数の関連する項目があり、すべてがなんらかのアクティビティを同時に行う場合でも、複数のプログレス コントロールは表示しません。代わりに、最後のタスクの完了時に終了する 1 つのプログレス コントロールを表示します。たとえば、アプリが複数の写真をダウンロードする場合は、写真ごとにプログレス コントロールを表示するのではなく、1 つだけを表示します。
- タスクの実行中は、プログレス コントロールの場所やサイズを変更しません。

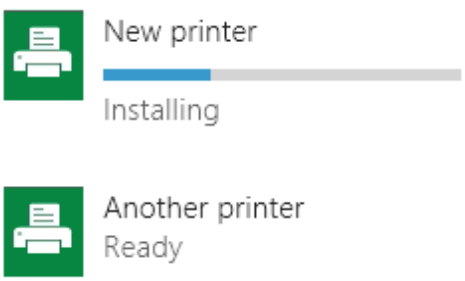
確定的タスクのガイドライン

- 操作がモーダルで (ユーザーの操作がブロックされ)、10 秒より長い時間がかかる場合は、操作を取り消す方法を用意します。操作を取り消すためのオプションは、操作の開始と同時に使えるようにします。
- 進行状況は均等に更新します。 進行状況が 80% を超えた後で長い間停止するような状況にならないようにします。進行が終わりに近づくにつれて、スピードが下がるのではなく、上がることが望まれます。0% から 90% に飛躍するようなことも、ないようにします。
- 進行状況が 100% になった後、進行状況確定バーがアニメーションを終了するまで待機してから、バーを非表示にします。
- タスクはユーザーまたは外部の条件によって停止したが、ユーザーがタスクを再開できる場合は、進行が一時停止されていることをはっきりと示します。JavaScript アプリでは、win-paused CSS スタイルを使います。C#/C++/VB アプリでは、ShowPaused プロパティを true に設定します。進行状況バーの下に、何が起こっているかを示す状態テキストを示します。

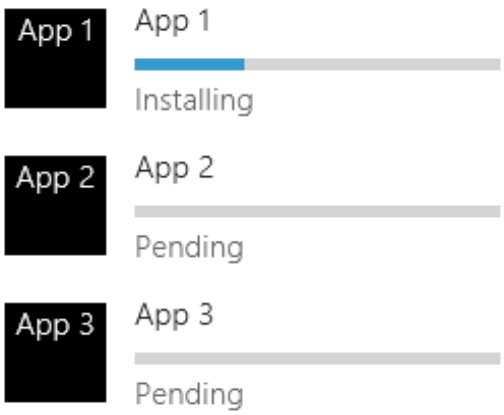
- タスクが停止し、再開できないか始めからやり直す必要がある場合は、エラーが発生したことをはっきりと示します。 JavaScript アプリでは、win-error CSS スタイルを使います。C#/C++/VB アプリでは、ShowError プロパティを true に設定します。バーの下の状態テキストを、何が起きてどのように問題に対処すればよいか (可能な場合) をユーザーに知らせるメッセージに置き換えます。
- 進行状況確定バーを表示する前に、多少の時間 (またはなんらかのアクション) が必要な場合は、まず進行状況不定バーを使い、その後で進行状況確定バーに切り替えます。たとえば、ダウンロード タスクの最初の手順がサーバーへの接続である場合、接続にかかる時間は推定できません。接続の確立後に、進行状況確定バーに切り替えて、ダウンロードの進行状況を表示します。切り替え後も、進行状況バーは正確に同じ位置、同じサイズになるようにします。



- プリンター一覧のような項目の一覧があるときに、一覧の項目に対してなんらかの操作 (いずれかのプリンター用のドライバーのインストールなど) を開始できる場合は、項目の横に進行状況確定バーを表示します。
  進行状況バーの上にタスクの表題 (ラベル)、下に状態を表示します。何が起きているかが明白な場合は、状態テキストは表示しません。 タスクの完了後は、進行状況バーを非表示にします。状態テキストは、項目の新しい状態を知らせるために使います。



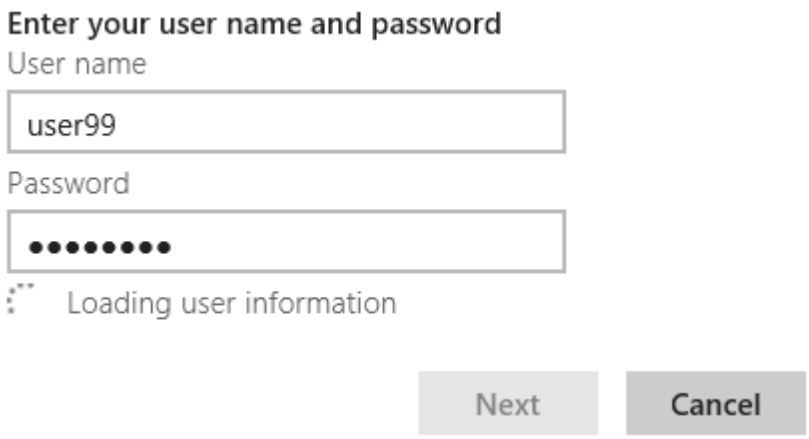
- タスク一覧を表示するには、コンテンツをグリッド内で整列させ、ユーザーが状態をひとめで見られるようにします。保留中の項目も含めて、すべての項目の進行状況バーを表示します。
  この一覧の目的は進行中の操作を示すことなので、完了した操作は一覧から削除します。



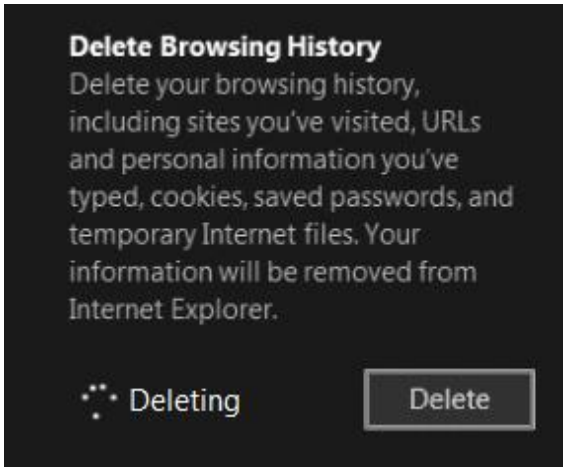
- ユーザーがアプリ バーからタスクを開始し、ユーザーの操作がブロックされた場合は、アプリ バーにプログレス コントロールを表示します。
  進行状況バーが何の進行状況を示しているかが明白な場合は、進行状況バーをアプリ バーの上部に配置して、ラベルと状態は省略できます。そうでない場合は、ラベルと状態テキストを表示します。
  アプリ バーのコントロールを無効にし、コンテンツ領域への入力を無視して、タスクの間は操作を無効にします。
- 進行状況を後退させないでください。 進行状況の値は常に増やします。 操作を元に戻さなければならない場合は、他の操作の進行状況を示すのと同じように、元に戻す処理の進行状況を示します。
- 現在の手順またはタスクが最後のものでないことがユーザーに明白でない場合は、進行状況を再開始 (100% から 0% に) しないようにします。たとえば、データのダウンロードと、そのデータの処理および表示という 2 つの部分がタスクにあるとします。ダウンロードが完了した後、進行状況バーを 0% にリセットし、データ処理の進行状況の表示を始めます。 タスクに複数の手順が含まれていることがユーザーに明白でない場合は、タスクを 1 つの 0 ～ 100% の尺度にまとめて、1 つのタスクから次のタスクに移る際に状態テキストを更新します。

進行状況リングを使う不定的なモーダル タスクのガイドライン

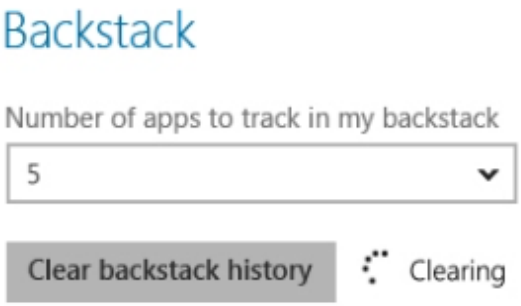
- 操作のコンテキスト内に進行状況リングを表示します。つまり、ユーザーが操作を開始した場所または結果のデータが表示される場所の近くにリングを表示します。
- 進行状況リングの右側に状態テキストを示します。
- 進行状況リングの色を状態テキストの色と同じにします。
- タスクの実行中にユーザーが操作してはいけないコントロールを無効にします。
- タスクの結果がエラーになった場合は、進行状況インジケーターと状態テキストを非表示にして、その場所にエラー メッセージを表示します。
- ダイアログでは、次の画面に移動する前に操作を完了する必要がある場合は、進行状況リングをボタン領域のすぐ上に、ダイアログのコンテンツと左揃えで配置します。



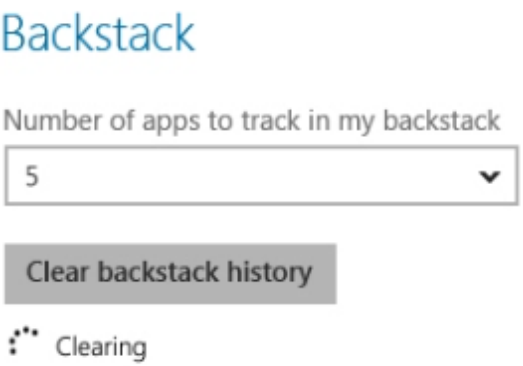
- 右揃えのコントロールがあるアプリ ウィンドウでは、進行状況リングを、処理を引き起こしたコントロールの左またはすぐ上に配置します。進行状況リングを、関連するコンテンツと左揃えで配置します。



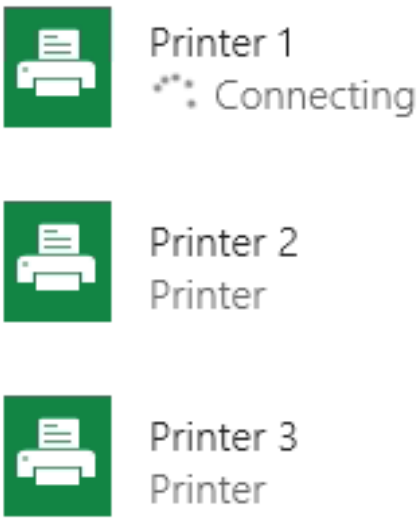
- 左揃えのコントロールがあるアプリ ウィンドウでは、進行状況リングを、処理を引き起こしたコントロールの右またはすぐ下に配置します。



- または –

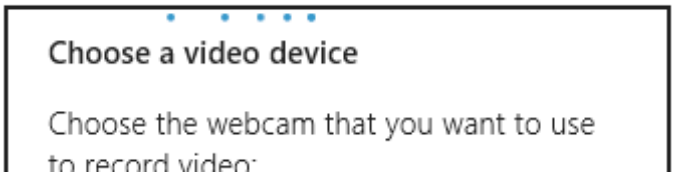


- 複数の項目を表示している場合は、項目のタイトルの下に、進行状況リングと状態テキストを配置します。エラーが発生した場合は、進行状況リングと状態をエラーテキストに置き換えます。

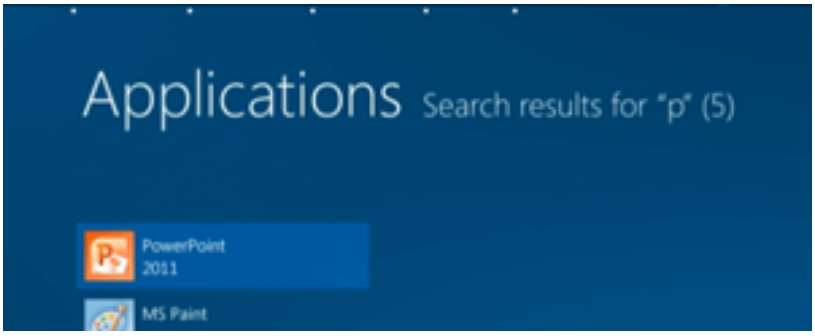


進行状況リングを使う不定的な非モーダル タスクのガイドライン

- フライアウトの中に進行状況を表示する場合は、フライアウトの上部に進行状況不定バーを配置し、幅はフライアウト全体にわたるように設定します。 このように配置すると、不必要に目立つことなく、進行中のアクティビティを示すことができます。 フライアウトにタイトルを表示すると、上部に進行状況バーを配置できなくなるので、タイトルは表示しません。



- アプリ ウィンドウに進行状況を表示する場合は、アプリ ウィンドウの最上部に、ウィンドウ全体にわたるように進行状況不定バーを配置します。



状態テキストのガイドライン

- 進行状況確定バーを使う場合は、状態テキストに進行状況の割合を表示しません。コントロール自体に、その情報が含まれています。
- プログレス コントロールなしでアクティビティを示すためにテキストを使う場合は、アクティビティが進行中であることを表すために省略記号を使います。
- プログレス コントロールを使う場合は、操作が進行中であることをプログレス コントロール自体が示しているので、状態テキストでは省略記号を使いません。

外観とレイアウトのガイドライン

- 進行状況確定バーは、背景が灰色のバーの中を徐々に埋めていく色付きのバーとして表示されます。色付きのバーの全長部分が、操作がどの程度完了したかを相対的に示します。

- 進行状況不定バーまたは進行状況リングは、常に移動する色付きの点で構成されます。
- プログレス コントロールをどの程度目立つ位置に配置するかどうかは、その重要性に基づいて決定します。

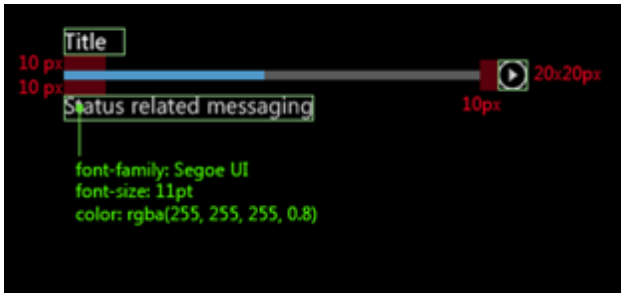
  重要なプログレス コントロールは特定の行動を促す印として使うことができ、システムが作業を完了した後で特定の操作が再開されることをユーザーに伝えます。

  組み込みの Windows Phone アプリの一部では、重要度に応じて、画面の上部にステータス バーの進行状況インジケーターを使っています。同じようにすることができ、確定または不定になるように構成できます。

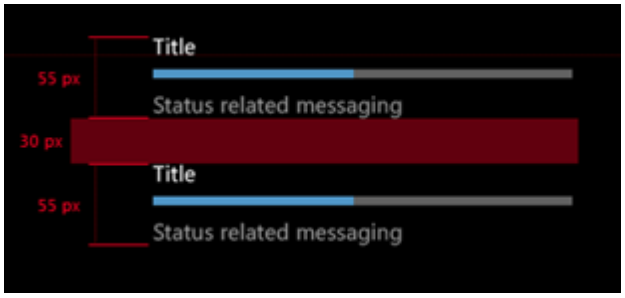
  重要度が低い場合 (ダウンロード中など) は、小さいサイズのコントロールを 1 つのビューに限定して表示します。
- ラベルを使って、進行状況の値を表示するか、進行中の処理を説明するか、処理が中断されていることを示します。ラベルはオプションですが、使うことを強くお勧めします。

  進行中の処理を説明する場合は、動名詞を使います (例: '接続中'、'ダウンロード中'、'送信中')。

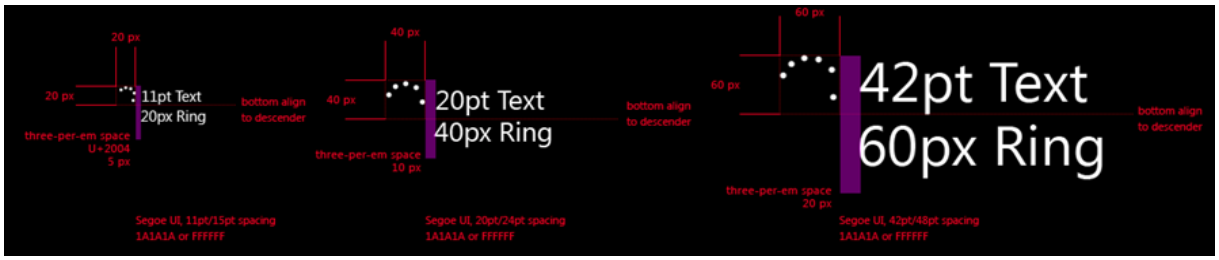
  進行が一時停止したり例外が発生しりした場合は、過去形を使います (例: '一時停止しました'、'ダウンロードは失敗しました'、'取り消されました')。
- ラベルと状態付きの進行状況確定バー



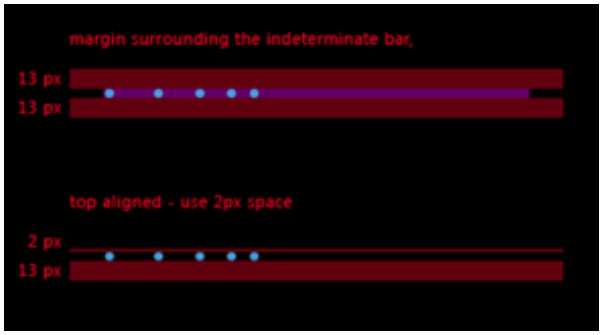
- 複数の進行状況バー



- 状態テキスト付きの進行状況不定リング



- 進行状況不定バー



その他の使い方のガイダンス

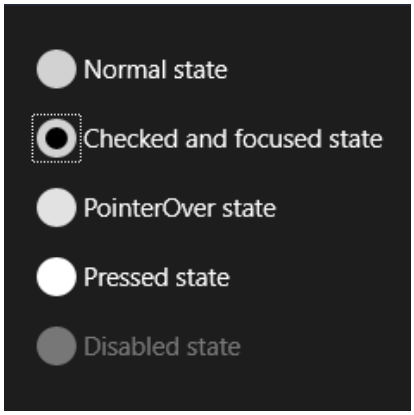
進行状況のスタイルを選ぶためのデシジョン ツリー

- **何かが行われていることをユーザーが知る必要があるか?**
  必要がない場合は、プログレス コントロールを表示しません。
- **タスクの完了に要する時間に関する情報があるか?**
  - **ある場合:** タスクを完了するまでに 2 秒より長くかかるか?
    - **かかる場合:** 進行状況確定バーを使います。10 秒より長くかかるタスクの場合は、タスクを取り消す方法を用意します。
    - **かからない場合:** プログレス コントロールを表示しません。
  - **ない場合:** タスクが完了するまでユーザーは UI を操作できないか?
    - **できない場合:** このタスクは、操作の特定の詳細をユーザーが認識する必要がある複数手順の一部か?
      - **そうである場合:** 画面の中央に水平に配置された状態テキストがある進行状況不定リングを使います。
      - **そうではない場合:** 画面の中央にテキストのない進行状況不定リングを使います。
    - **できる場合:** これは主要アクティビティか?
      - **そうである場合:** 進行状況は UI の特定の 1 つの要素に関連しているか?
        - **関連している場合:** その関連する UI 要素の横に状態テキストがあるインラインの進行状況不定リングを使います。
        - **関連していない場合:** 大量のデータが一覧に読み込まれているか?
          - **読み込まれている場合:** 受信したコンテンツを表すプレースホルダーの上部の進行状況不定バーを使います。
          - **読み込まれていない場合:** 画面またはサーフェイスの上部の進行状況不定バーを使います。
      - **そうではない場合:** 画面の上隅の状態テキストを使います。

ラジオ ボタンのガイドライン

ラジオ ボタンでは、ユーザーは 2 つ以上の選択肢から 1 つのオプションを選ぶことができます。各オプションは、1 つのラジオ ボタンによって表されます。ユーザーは、ラジオ ボタン グループの中から、1 つのラジオ ボタンだけを選ぶことができます。

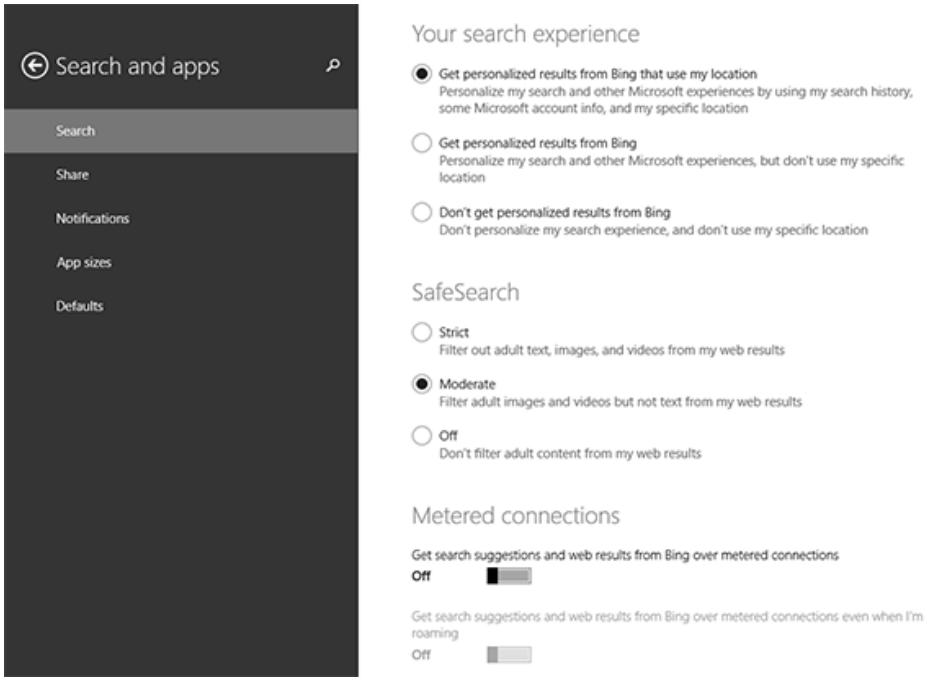
ラジオ ボタンという名称は、ラジオのチャンネル プリセットのボタンから付けられました。



重要な API

RadioButton クラス (XAML)　　input type="radio" 要素 (HTML)

例



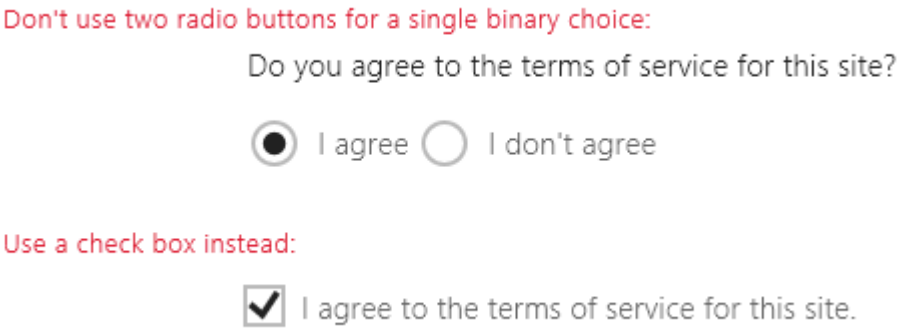
適切なコントロールの選択

ユーザーに 2 つ以上の相互排他的なオプションを提示するには、次のようにラジオ ボタンを使います。

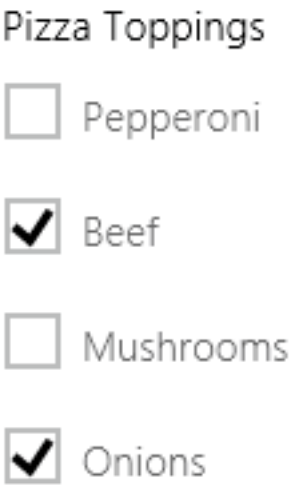
ラジオ ボタンはわかりやすく、アプリで重要なオプションを目立つように表示します。ラジオ ボタンは、広い画面領域を使うに値する重要なオプションを表示する場合であって、選択肢が明確なためにわかりやすいオプション表示が必要な場合に使用してます。

ラジオ ボタンはすべてのオプションを均等に強調するため、必要以上に注目される可能性があります。ユーザーの特別な注目を引く必要がない場合は、他のコントロールを使うことを検討してください。たとえば、 ほとんどの状況でほとんどのユーザーに既定のオプションが適切な場合は、代わりにドロップダウン リストを使います。

2 つだけの相互排他的なオプションの場合は、1 つのチェック ボックスまたはトグル スイッチにまとめます。たとえば、"I agree" と "I don't agree" という 2 つのラジオ ボタンではなく、"I agree" のチェック ボックスを使います。



ユーザーが複数のオプションを選択できる場合は、代わりにチェック ボックスまたはリスト ボックス コントロールを使います。



オプションが 10、20、30 のように固定間隔の数値である場合は、ラジオ ボタンを使いません。代わりに、スライダー コントロールを使います。

オプションが 8 個より多い場合は、ドロップダウン リスト、単一選択のリスト ボックス、またはリスト ビューを使います。

オプションがアプリの現在のコンテキストに基づいて表示される場合や、その他の方法で動的に変化する場合は、単一選択のリスト ボックスを使います。

注 キーボード経由でアクセスした場合、ラジオ ボタンのグループは、1 つのコントロールのように動作します。Tab キーを使うと選んだオプションにのみアクセスできますが、方向キーを使ってグループを切り替えることができます。

推奨と非推奨

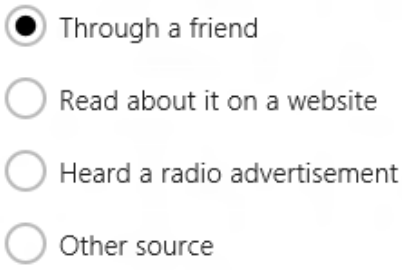
- 一連のラジオ ボタンの用途と現在の状態が明確に表示されていることを確認します。
- ユーザーがラジオ ボタンをタップしたときには、必ずビジュアルなフィードバックを返します。
- ユーザーのラジオ ボタンの操作に合わせて、ビジュアルなフィードバックを返します。ラジオ ボタンの状態には、たとえば、標準、押された状態、オンの状態、オフの状態があります。 ユーザーは、ラジオ ボタンをタップして関連のオプションをアクティブ化します。アクティブなオプションをもう一度タップしても非アク

ティブにはなりませんが、別のオプションをタップすると、そのオプションにアクティブ化の状態が移ります。

- タッチに対するフィードバック用とオンの状態用にビジュアル効果やアニメーションを予約します。オフの状態のラジオ ボタン コントロールは、使われていない、または非アクティブなコントロールとして表示します (無効なコントロールとして表示しないでください)。
- ラジオ ボタンのテキスト コンテンツは、1 行に収まるように作成します。ラジオ ボタンのビジュアル効果をカスタマイズして、メインのテキスト行の下に小さいフォント サイズでオプションの説明を表示することができます。
- テキスト コンテンツが動的な場合、ボタンのサイズがどのように変わり、周囲のビジュアル効果にどのような影響が生じるかを検討してください。
- ブランドのガイドラインで別のフォントが指示されていない限り、既定のフォントを使います。
- ラベルをタップするとラジオ ボタンが選択されるように、ラベル要素でラジオ ボタンを囲みます。
- ラベル テキストは、ラジオ ボタン コントロールの前や上ではなく、後に配置します。
- ラジオ ボタンをカスタマイズすることを検討してください。既定のラジオ ボタンは、2 つの同心円 (内側の円は塗りつぶされ、オンの場合に表示。外側の円はストロークのみ) とテキスト コンテンツで構成されています。しかし、工夫しだいで使いやすさが向上します。アプリのコンテンツを直接操作できれば、ユーザーが理解しやすくなります。たとえば、グラフィックやさりげないテキストのトグル ボタンを使って、提供する実際のコンテンツを表示することもできます。
- ラジオ ボタン グループには、8 個以上のオプションを含めないでください。それより多くのオプションを提示する必要がある場合は、代わりにドロップダウン リスト、リスト ボックス、リスト ビューなどを使います。
- 2 つのラジオ ボタン グループを並べて配置しないようにします。 2 つのラジオ ボタン グループが並んでいると、どのボタンがどのグループに属しているかがわかりにくくなります。グループを分けるには、グループ ラベルを使います。

その他の使い方のガイダンス

この図は、適切な位置と間隔で配置したラジオ ボタンを示しています。



評価コントロールのガイドライン

評価コントロールは、評価を示すアイコンをクリックすることで、ユーザーが何かを評価できるようにします。 評価には、平均評価、暫定評価、ユーザー評価の 3 種類があります。

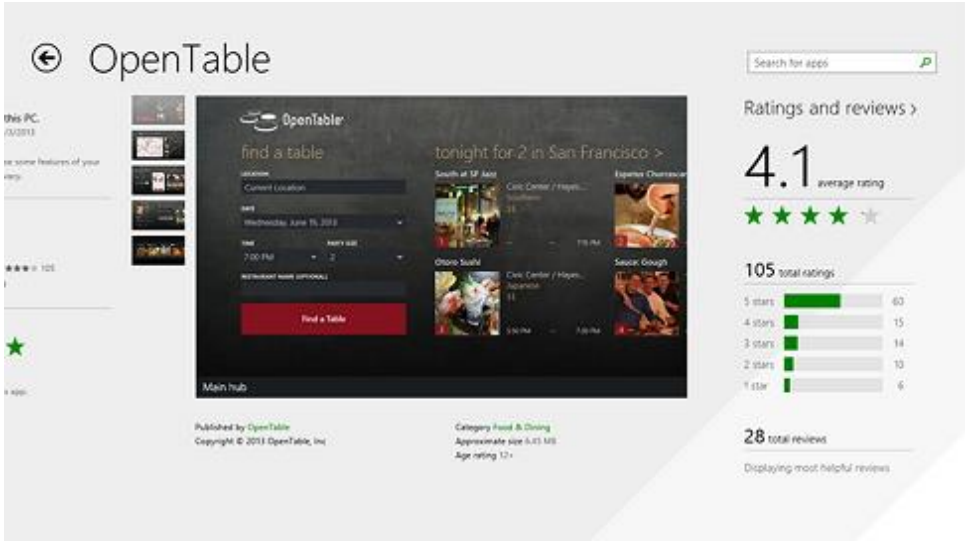
注 評価コントロールは HTML でのみ利用できます。XAML の評価コントロールはありません。

重要な API

Rating オブジェクト (WinJS)

例

次に評価の例を示します。



適切なコントロールの選択

項目またはサービスに対するユーザーの好感度や満足度を表示するには、評価コントロールを使います。たとえば、評価コントロールを使って、ユーザーが映画を評価できるようにします。 画面の明るさなどのように、連続的な範囲を持つ他の種類のデータには使いません (代わりにスライダーを使います)。

評価コントロールをフィルター コントロールとしては使いません。たとえば、ユーザーが検索結果をフィルター処理して、5 つ星のレストランを表示できるようにする場合に、評価コントロールをフィルターとして実装しないようにします。評価コントロールを使うと、ユーザーがレストランに新しい評価を設定しているのだと誤解する可能性があります。このような場合、代わりにドロップダウン リストを使用できます。

1 つ星評価コントロールを、好き/嫌いを表すコントロールとして使いません。代わりに、チェック ボックスを使います。評価コントロールは、二肢選択評価用には設計されていません。たとえば、コントロールをタップしても、星のオンとオフを切り替えることはできません。

推奨事項

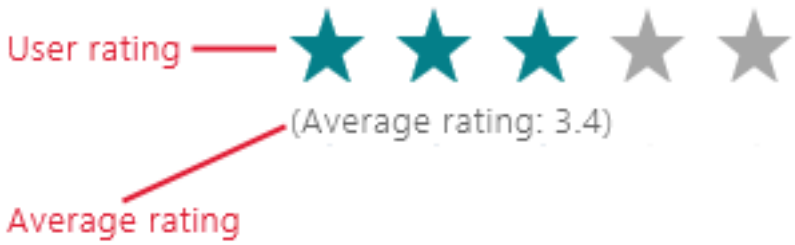
- わかりやすくするための情報をユーザーに提供するには、ツールチップを使います。ツールチップをカスタマイズして、次の図のように、"すばらしい"、"とても良い"、"まあまあ" など、各星の意味を表示できます。

- ユーザーが評価を追加または変更しないようにする場合は、評価コントロールを無効化します。 無効化された評価コントロールには引き続き評価が表示されますが (設定されている場合)、ユーザーは評価の追加や変更ができなくなります。たとえば、評価をログイン ユーザーだけに限定するとします。その場合は、評価コントロールを無効化して、ユーザーがコントロールをタップしたら、ログイン ページが表示されるようにします。

  コントロールを有効化できない場合 (アプリで常に読み取り専用である場合) は、コントロールのホスト要素の class 属性を "win-small" に設定して、他の評価コントロールよりも小さくします。 コントロールを小さくすると、他のコントロールと区別しやすくなり、操作対象でないこともわかりやすくなります。

- 平均評価とユーザー評価を同時に表示します。 ユーザーが評価を指定すると、評価コントロールは平均評価ではなく、ユーザーの評価を表示します。ユーザーにとって意味がある場合は常に、ユーザーの評価の他に平均ユーザー評価も表示します。平均評価を表示するには、次の 2 つの方法があります。
  - 付随するテキスト文字列に平均を表示します ("平均: 3.5" など)。

- 2 つの評価コントロールを一緒に使い、1 つにはユーザー評価を表示します。もう 1 つには平均評価を表示し、ユーザーの入力は許可しません。



- 必要でない限り、既定の星の数 (最高評価) を変更しません。既定では、評価コントロールには 5 つの星があります。1 が最低、最悪の評価で、5 が最高、最良の評価です。 アプリがこの慣例に従っていると、ユーザーは評価の意味を簡単に理解できます。
- ユーザーが自分の評価を削除できないようにする必要がない限り、"自分の評価のクリア" 機能を無効にしません。

スクロール バーのガイドライン

パンとスクロールを行うと、画面の境界外のコンテンツを拡張表示することができます。

スクロール ビューアー コントロールは、ビューポート内に適合する量のコンテンツと、一方または両方のスクロール バーからなります。パンとズームのためにタッチ ジェスチャを使うことができ (操作中にのみスクロール バーはフェードインします)、またスクロールのためにポインターを使うことができます。フリック ジェスチャでは、慣性を伴ってパンします。

注 Windows: 検出された入力デバイスに基づいて、次の 2 種類のパン表示モードが使われます。パン インジケーター (タッチを使う場合) とスクロール バー (マウス、タッチパッド、キーボード、スタイラスなど、その他の入力デバイスを使う場合) です。

重要な API

[ScrollViewer クラス (XAML)]

[ScrollBar クラス (XAML)]

例

推奨と非推奨

- コンテンツ領域が 1 つのビューポート境界 (垂直方向または水平方向) を超えている場合は、単一軸のパンを使います。コンテンツ領域が両方のビューポート境界 (垂直方向と水平方向) を超えている場合は、2 軸のパンを使います。
- リスト ボックス、ドロップダウン リスト、テキスト入力ボックス、グリッド ビュー、リスト ビュー、ハブ コントロールの組み込みのスクロール機能を使います。こうしたコントロールでは、同時に表示する項目が多すぎる場合に、ユーザーが項目のリストを水平方向、垂直方向のいずれかにスクロールできます。
- ユーザーがより大きな領域の周囲で両方向にパンすること、そしておそらくズームできるようにする場合、たとえばユーザーが (画面に適合するサイズに設定された

イメージではなく) フル サイズのイメージをパンおよびズームできるようにする場合には、スクロール ビューアー内にイメージを配置します。

- ユーザーが長いテキスト パスをスクロールする場合、垂直方向にのみスクロールするようにスクロール ビューアーを構成します。
- 1 つのオブジェクトのみを含める場合にスクロール ビューアーを使います。1 つのオブジェクトをレイアウト パネルとし、その任意の数のオブジェクトを含めることができる点に注意してください。

検索のガイドライン

検索は、ユーザーがアプリでコンテンツを見つけることができる 2 つの方法のうちの 1 つです。この記事のガイダンスでは、検索エクスペリエンスの構成要素、検索スコープ、実装、コンテキストでの検索の例について説明します。

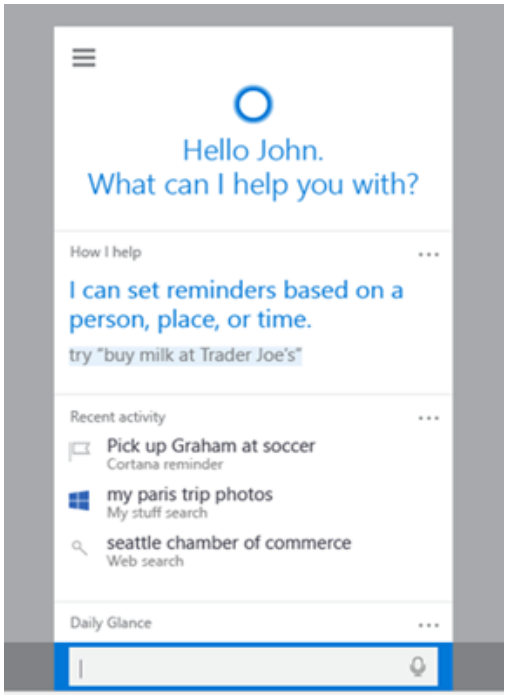
重要な API

AutoSuggestBox クラス (XAML)　　AutoSuggestBox オブジェクト (WinJS)

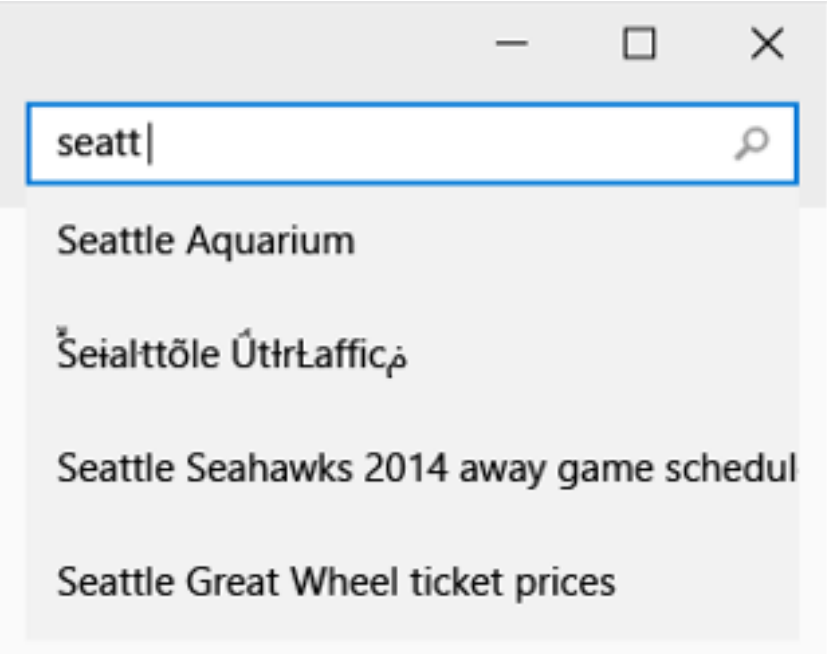
検索エクスペリエンスの構成要素

入力。 テキストは検索入力の最も一般的なモードであり、このガイドで重点的に説明します。その他の一般的な入力モードには音声やカメラがありますが、通常、それにはデバイスハードウェアを操作する機能が必要であり、追加のコントロールやカスタム UI がアプリ内で必要になる場合があります。

ゼロ入力。 ユーザーが入力フィールドをアクティブにしてからテキストを入力するまでに、"ゼロ入力キャンバス" と呼ばれるものを表示できます。通常、ゼロ入力キャンバスは、アプリのキャンバスに表示され、ユーザーがクエリの入力を開始したときに、このコンテンツがオート サジェスト ボックスで置き換えられます。最近の検索履歴、トレンド検索、状況依存の検索候補、ヒントが、すべてゼロ入力状態の候補となります。



クエリの生成/自動提案。 クエリの生成により、ユーザーが入力を開始するとすぐにゼロ入力のコンテンツが置き換えられます。ユーザーがクエリ文字列を入力すると、入力プロセスを高速化し、有効なクエリを生成できるように、継続的に更新される一連のクエリ候補、または不明瞭解消オプションが表示されます。このクエリ候補の表示動作は、オート サジェスト コントロールに組み込まれ、検索内部にアイコンを表示する方法にもなります (マイク アイコンやコミット アイコンなど)。これ以外のすべての動作は、アプリに基づきます。



結果のセット。 検索結果は、通常は検索入力フィールドのすぐ下に表示されます。これは必須ではありませんが、入力と結果の並置によりコンテキストが維持され、クエリの編集や新しい検索の入力をすぐに開始できます。このつながりは、ヒント テキストを、結果セットを作成したクエリで置き換えることで、さらに強化できます。

クエリの編集と再クエリの両方を効率的に開始できるようにする 1 つの方法として、フィールドが再アクティブ化されたときに、前のクエリを強調表示します。これにより、任意のキー入力によって前の文字列が置き換えられますが、文字列が保持されるため、ユーザーはカーソルを移動して、前の文字列を編集または追加することができます。

結果のセットは、コンテンツを最適に伝える任意の形式で表示できます。リスト ビューは十分な柔軟性を備えており、ほとんどの検索に最適です。グリッド ビューはイメージまたはその他のメディアに適しており、地図を使って空間的な配布を伝えることができます。

検索スコープ

検索は共通の機能であり、シェルおよび多くのアプリ内で検索 UI がユーザーに表示されます。検索のエントリ ポイントは同じように視覚化される傾向がありますが、広い範囲 (Web またはデバイスの検索) から狭い範囲 (ユーザーの連絡先一覧) までの結果を提供できます。検索エントリ ポイントは、検索対象のコンテンツに対して並置する必要があります。

一般的な検索スコープは次のとおりです。

グローバルとコンテキスト/絞り込み。 クラウドとローカル コンテンツの複数のソースにわたって検索します。結果には、URL、ドキュメント、メディア、操作、アプリなどが含まれます。

Web。 Web インデックスを検索します。結果には、ページ、エンティティ、回答が含まれます。

自分のコンテンツ。 デバイス、クラウド、ソーシャル グラフなどを対象に検索を実行します。結果はさまざまですが、ユーザー アカウントへの接続によって制限されます。

ヒントのテキストを使って検索スコープを伝えます。次のようなシナリオが考えられます。

"Windows と Web を検索する"

"連絡先の一覧を検索する"

"メールボックスを検索する"

"検索の設定"

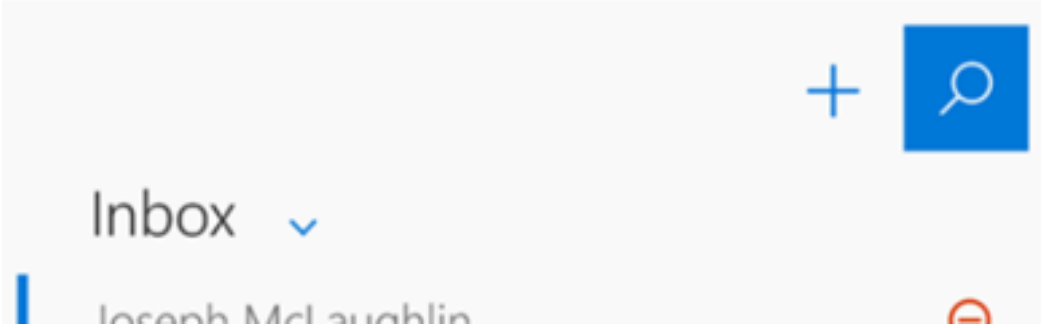
"場所を探す"

検索入力ポイントの範囲を効果的に伝えることにより、実行中の検索の機能がユーザーの期待事項を満たし、不満が発生する可能性を減らすことができます。

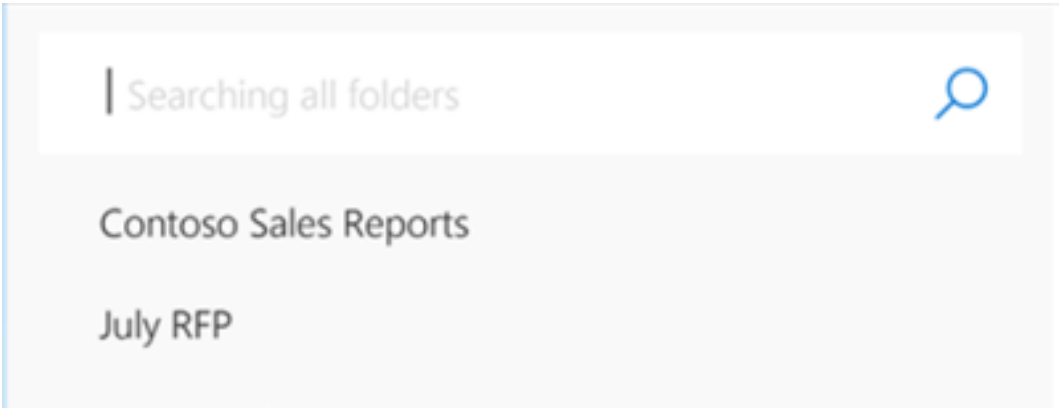
実装

ほとんどのアプリでは、検索のエントリ ポイントとしてテキスト入力フィールドを用意することをお勧めします。これにより、目立つビジュアルなフットプリントが提供されます。さらに、ヒントのテキストは検索機能を支援し、検索スコープを伝えることができます。検索がより副次的な操作であるか、またはスペースに制約がある場合、検索アイコンは、関連する入力フィールドのないエントリ ポイントとなります。アイコンとして視覚化するときは、次の例のように、必ずモーダルな検索ボックスの余地があることを確認します。

検索アイコンをクリックする前:



検索アイコンをクリックした後:



検索では、常にエントリ ポイントに右向きの虫眼鏡グリフを使います。使用するグリフは Segoe MDL2 Assets、16 進数の文字のコード 0xE094 で、通常は 15 epx のフォント サイズです。

検索のエントリ ポイントは、多数のさまざまな領域に配置でき、それによって検索スコープとコンテキストの両方が伝わります。さまざまなエクスペリエンスや外部からの結果をアプリに収集する検索は、通常、グローバル コマンド バーやナビゲーションなど、最上位にあるアプリのクロム内に配置されます。

検索スコープが狭くなるかにコンテキストに依存するにつれて、通常、配置は検索するコンテンツとより直接的に関連付けられます (キャンバス上、リスト ヘッダーとして、状況依存のコマンド バー内など)。いずれの場合も、検索入力と結果のつながり、または絞り込まれたコンテンツが視覚的に明確になります。

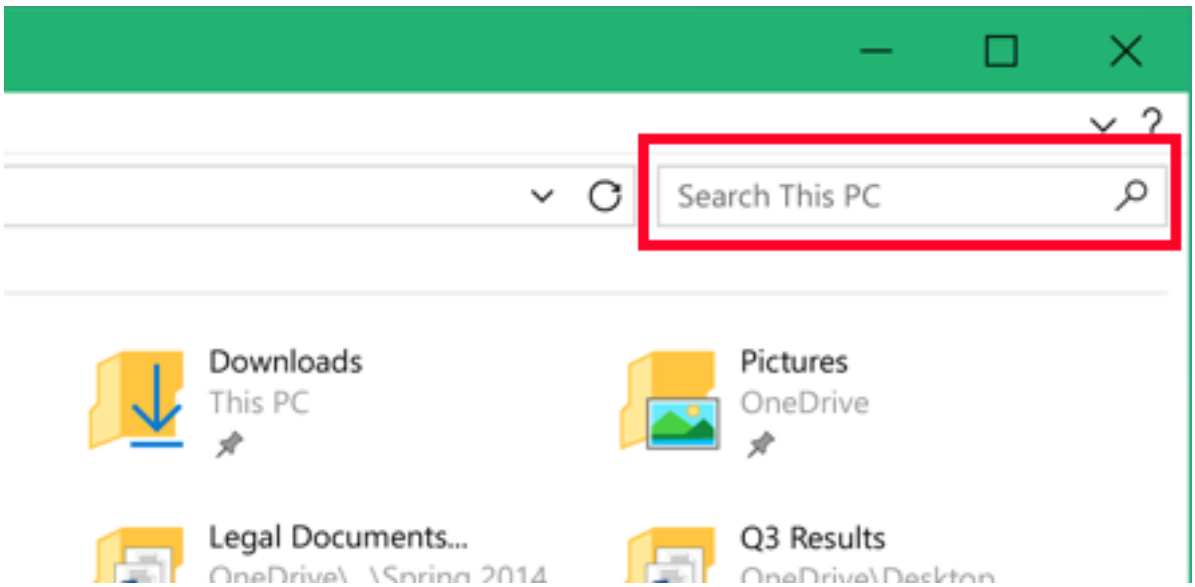
スクロール可能なリストの場合、常に検索入力を表示すると便利です。検索入力は固定し、コンテンツが背後をスクロールするようにすることをお勧めします。

ゼロ入力とクエリの生成機能は、コンテキスト/絞り込み検索ではオプションであり、リストはユーザーの入力によってリアルタイムで絞り込まれます。例外には、受信トレイのフィルター オプション (宛先:<入力文字列>、差出人: <入力文字列>、件名: <入力文字列>) など、クエリの書式設定の候補が表示される場合などがあります。

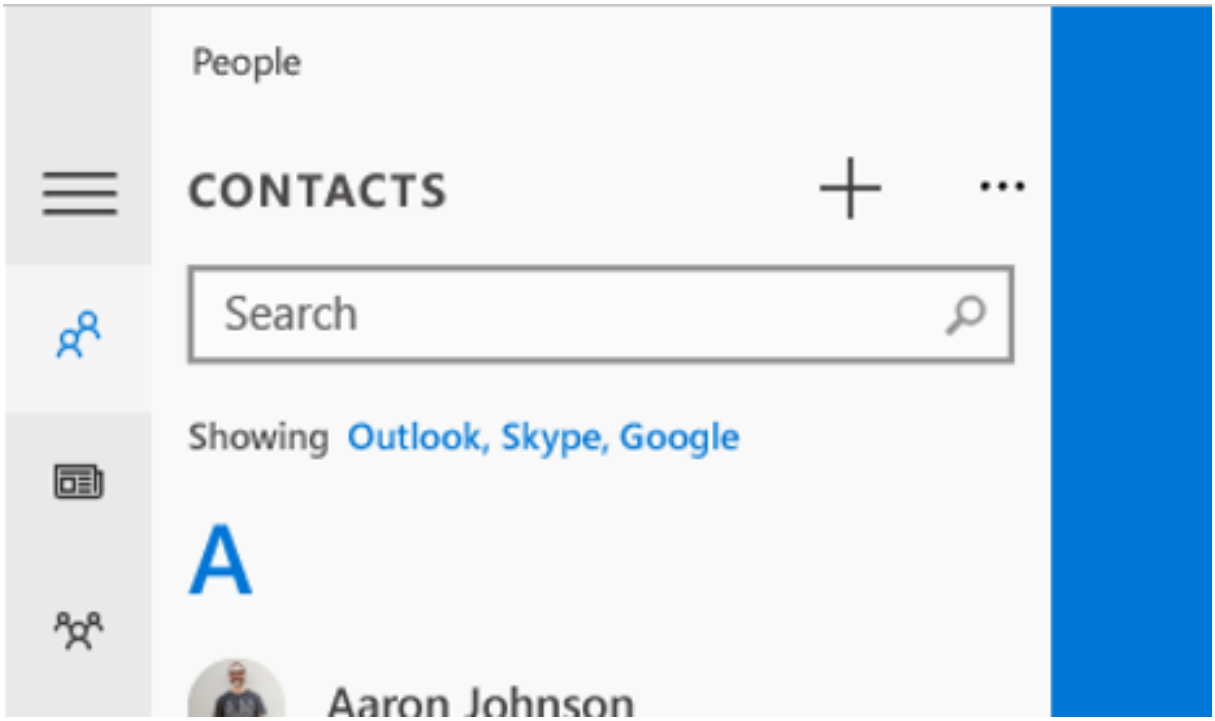
例

このセクションの例では、コンテキストに検索を配置します。

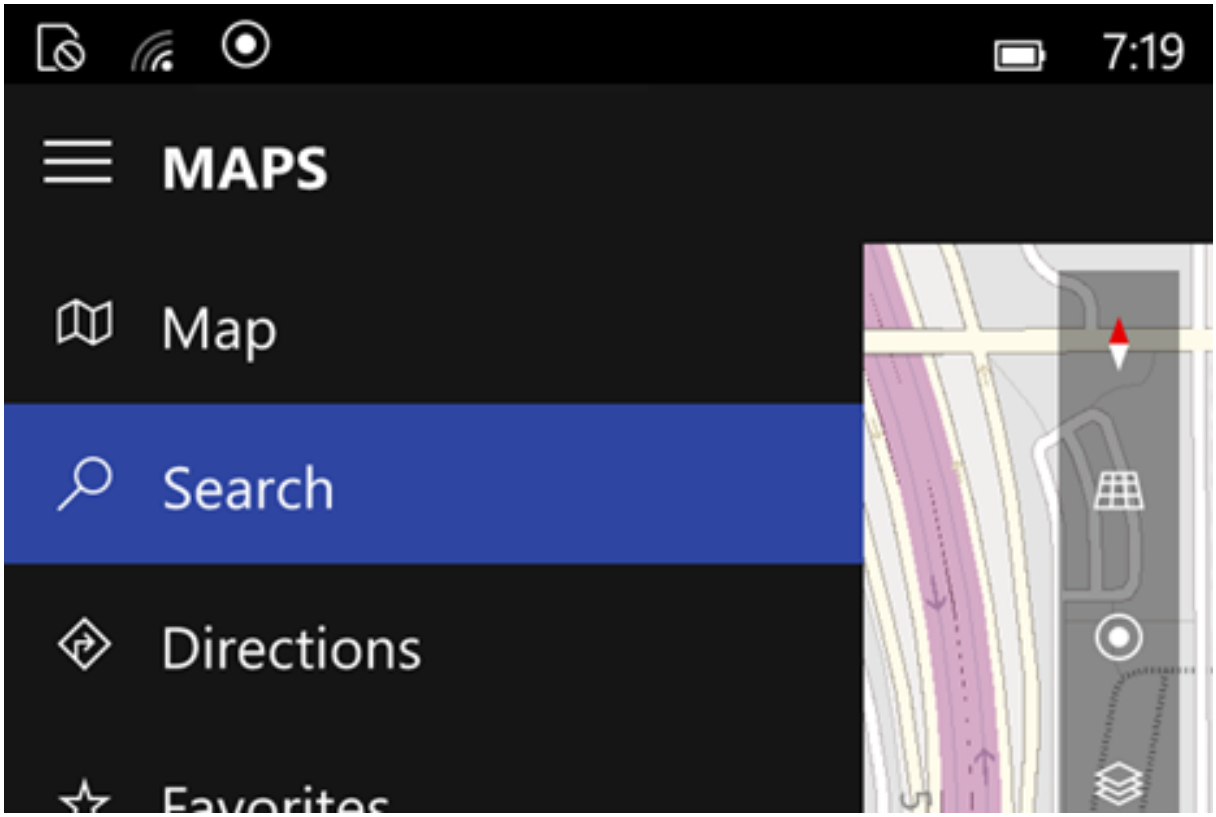
Windows ツール バーの操作としての検索:



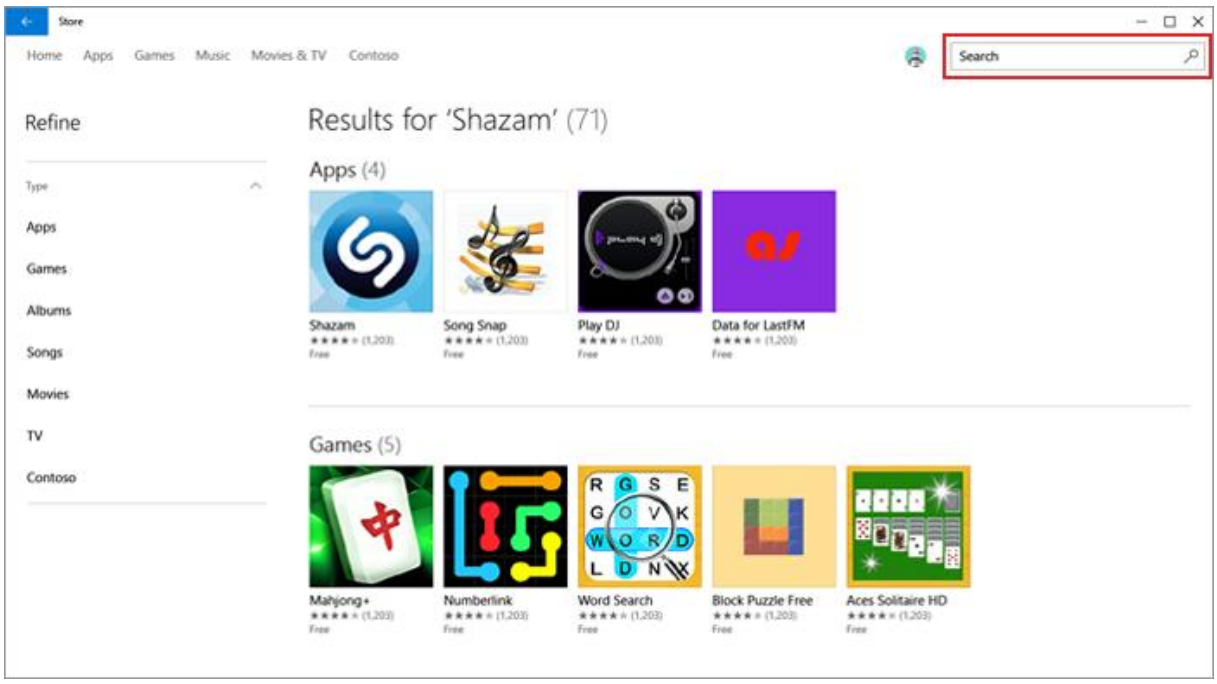
アプリ キャンバスでの入力としての検索:



ナビゲーション ウィンドウでの検索:



検索が頻繁にアクセスされないか、コンテキスト依存が高い場合に予約されるのが最適なインライン検索:



セマンティック ズームのガイドライン

セマンティック ズーム コントロールを使うと、ユーザーは同じデータ セットの 2 つの異なる表示間を切り替えることができるようになります。セマンティック ズームでは、グラフィック表示のパラメーターとレイアウトを変更して、データの構造や選択状況の表示を修正します。

重要な API

SemanticZoom クラス (XAML)

SemanticZoom オブジェクト (WinJS)

Input 名前空間 (XAML)

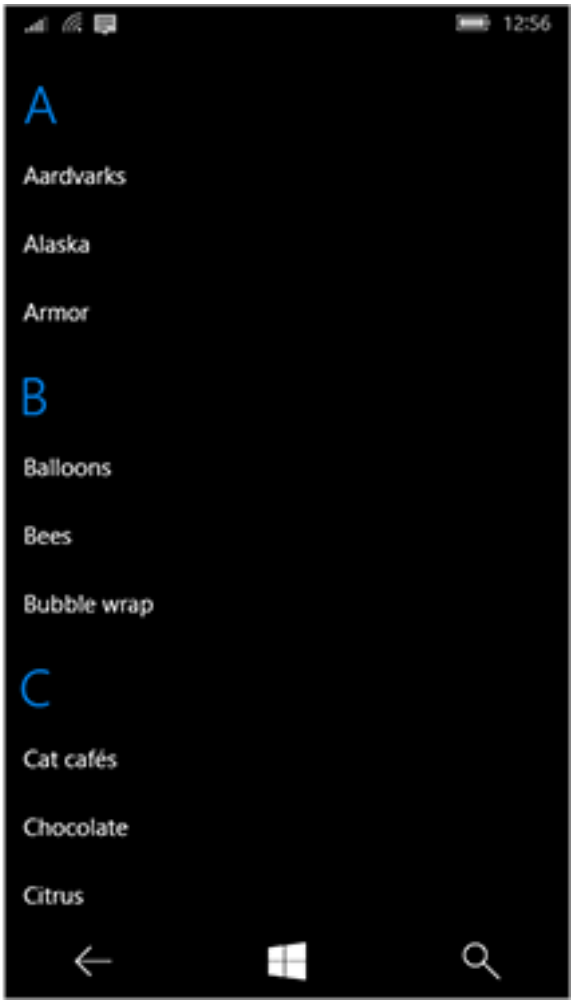
機能

- アプリに含めることができるセマンティック ズーム コントロールは 1 つだけです。

- 縮小表示ビューのサイズは、セマンティック ズーム コントロールの境界によって制限されます。
- グループ ヘッダーをタップするとビューが切り替わります。ピンチしてビューを切り替える方法を有効にできます。
- アクティブなヘッダーによりビューが切り替わります。

例

セマンティック ズーム コントロールを使って簡単にナビゲートできるデータ セットの例として、アドレス帳があります。この例では、セマンティック ズームはジャンプ リストを使用します。1 つのビューでは、アドレス帳に登録されているすべての人の概要がアルファベット順に表示されますが (左側のイメージ)、拡大表示ビューでは、データが順番に表示され、より詳細な情報が表示されます (右側のイメージ)。

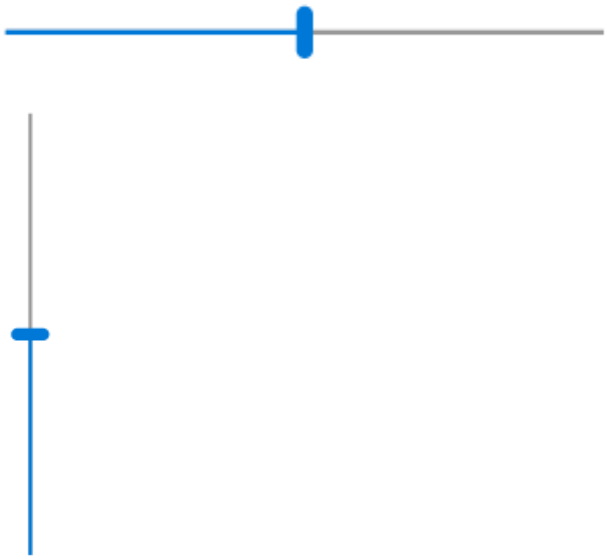


推奨事項

- アプリでセマンティック ズームを使うときは、ズーム レベルごとに項目のレイアウトとパン方向が変わらないようにする必要があります。レイアウトとパン操作は、ズーム レベルに関係なく一貫して予測できるものにしてください。
- セマンティック ズームを使ってすばやくコンテンツにジャンプできるようにするため、縮小モードでのページや画面の数は 3 つまでに制限します。パンが多すぎると、セマンティック ズームの実用性が損なわれます。
- セマンティック ズームを使ってコンテンツの範囲を変更しないでください。たとえば、フォト アルバムをファイル エクスプローラーのフォルダー表示に切り替えないでください。
- 各ビューに不可欠な構造とセマンティクスを使います。
- グループ化したコレクションの項目にはグループ名を使います。
- グループ化せずに並べ替えたコレクションには並べ替え順序を使います (日付の場合は時系列順、名前の一覧の場合はアルファベット順など)。

スライダーのガイドライン

スライダー コントロール (範囲コントロールとも呼ばれます) では、ユーザーがトラック上をタップするか前後にスクラブして、有効な範囲から値を設定できます。



重要な API

Slider クラス (XAML)　　Slider オブジェクト (HTML)

適切なコントロールの選択

定義された連続的な値 (音量や明るさなど) または個別の値の範囲 (画面解像度の設定など) をユーザーが設定できるようにする場合に、スライダーを使います。

スライダーは、ユーザーが値を数値でなく相対的な量であると考えている場合に適しています。たとえば、ユーザーはオーディオの音量を数値の 2 や 5 ではなく、低や中に設定しようと考えます。

二者択一の設定には、スライダーは使いません。代わりに、トグル スイッチ を使います。

スライダーを使うかどうかを決める際には、他にも次のような点を考慮します。

- **設定が相対的な量のように見えるか?**　見えない場合は、ラジオ ボタンまたは 選択コントロールを使います。

- **設定は正確な既知の数値か?** そのような数値である場合は、数値テキスト ボックスを使います。
- **設定の変更による効果をすぐに確認できると、ユーザーにとって便利か?** 便利である場合は、スライダーを使います。たとえば、色合い、鮮やかさ、明度の値を変更した場合の効果をすぐに確認できると、ユーザーは色をより簡単に選べるようになります。
- **設定に 4 つ以上の値の範囲があるか?** ない場合は、ラジオ ボタンを使います。
- **ユーザーが値を変えられるか?** スライダーは、ユーザーが操作するためのものです。ユーザーが値を変えられない場合は、代わりに読み取り専用のテキストを使います。

スライダーと数値テキスト ボックスのどちらを使うかを決める際に、次の場合には数値テキスト ボックスを使います。

- 画面領域が狭い。
- ユーザーがキーボードを使おうとする可能性が高い。

次の場合にはスライダーを使います。

- ユーザーにとって、すぐに結果がわかると便利。

推奨事項

- コントロールのサイズは、ユーザーが値を簡単に設定できる大きさにします。 個別の値を設定する場合は、ユーザーがマウスを使って値を簡単に選べるようにします。スライダーのエンドポイントが、常にビューの境界内にあることを確認します。
- ユーザーが選んでいるときまたは選んだ後で、すぐに結果が確認できるようにします (それが実際的な場合)。たとえば、Windows のボリューム コントロールは、選ばれたオーディオ音量を示すためにビープ音を鳴らします。
- 値の範囲を示すためにラベルを使います。 例外: スライダーが垂直方向で、上部のラベルが最大、高、多などの場合、下部の意味は明らかであるため、ラベルを省略できます。

- スライダーを無効にする場合は、関連するすべてのラベルまたはフィードバックのビジュアル効果も無効にします。
- スライダーのフロー方向や向きを設定するときには、テキストの方向を考慮してください。言語によって、左から右に書く場合と、右から左に書く場合があります。
- スライダーは、進行状況インジケーターとしては使いません。
- スライダーのつまみは、既定のサイズのままにします。
- 値の範囲が広く、ユーザーが選ぶのは範囲内のいくつかの代表的な値であることがほとんどの場合は、連続的なスライダーを作成しません。 代わりに、それらの値だけを、許可される間隔として使います。たとえば、時間の最大値は 1 か月であっても、ユーザーが 1 分、1 時間、1 日、1 か月のいずれかを選ぶ場合は、4 つの間隔位置だけのスライダーを作成します。

その他の使い方のガイダンス

適切なレイアウトの選択: 水平または垂直

スライダーは、水平方向または垂直方向にレイアウトできます。次のガイドラインを使って、使用するレイアウトを決めます。

- 自然な方向を使います。たとえば、スライダーが現実世界の値を表していて、通常は垂直方向に表示される場合 (気温など) は、垂直方向にします。
- 動画アプリのように、メディア内をシークするためにコントロールが使われる場合は、水平方向にします。
- 1 つの方向 (水平または垂直) にパンするページでスライダーを使う場合は、パンの方向とは異なる方向をスライダーに使います。 そうしないと、ユーザーはページをパンしようとしてスライダーをスワイプし、誤って値を変えてしまう場合があります。
- 使用する方向がまだ決まらない場合は、ページ レイアウトに適した方を使います。

範囲方向

範囲方向とは、現在の値から最大値へスライダーを動かす方向のことです。

- 垂直方向スライダーでは、読みの方向に関係なく、最大値をスライダーの上部に配置します。たとえば、音量スライダーでは、最大の音量設定を常にスライダーの上部に配置します。他の種類の値 (曜日など) では、ページの読みの方向に従います。
- 水平方向のスタイルでは、ページ レイアウトが左から右への場合は、低い方の値をスライダーの左側に配置します。ページ レイアウトが右から左への場合は、右側に配置します。
- 前のガイドラインの 1 つの例外は、メディア シーク バーです。このバーでは、低い方の値を常にスライダーの左側に配置します。

間隔と目盛り

- スライダーで最小値から最大値までの任意の値を許可するのではない場合は、間隔位置を使います。たとえば、購入する映画のチケットの数を指定するためにスライダーを使う場合、浮動小数点値は許可しません。間隔の値を 1 にします。
- 間隔(スナップ位置とも言います) を指定する場合、最後のステップがスライダーの最大値に揃うようにします。
- 主な、または重要な値の位置をユーザーに示す場合は、目盛りを使います。たとえば、ズームを制御するスライダーでは、50%、100%、200% の目盛りを設定します。
- 設定のおおよその値をユーザーが知る必要がある場合に、目盛りを表示します。
- 選択した設定の正確な値を、コントロールを操作しなくてもユーザーが確認できるようにするには、目盛りと値ラベルを表示します。または、値のツールチップを使って、正確な値が見られるようにします。
- 間隔位置が明白でない場合は、常に目盛りを表示します。たとえば、スライダーの幅が 200 ピクセルで、200 のスナップ位置がある場合は、ユーザーはスナップの動作に気付かないので、目盛りを非表示にできます。しかし、スナップ位置が 10 個しかない場合は、目盛りを表示します。

ラベル

- **スライダー ラベル**

  スライダー ラベルは、スライダーの使用目的を示します。
  - ラベルを使うときは末尾に句点を付けません (これはすべてのコントロール ラベルでの規則です)。

- スライダーのあるフォームで、ほとんどのラベルがコントロールの上にある場合は、ラベルをスライダーの上に配置します。
- スライダーのあるフォームで、ほとんどのラベルがコントロールの横にある場合は、ラベルをスライダーの横に配置します。
- ユーザーがスライダーにタッチするときに、指でラベルが見えなくなる場合があるので、ラベルをスライダーの下には配置しません。

- **範囲ラベル**

  範囲 (容量) ラベルは、スライダーの最小値と最大値を示します。
  - 垂直方向であることによって明白である場合以外は、スライダーの範囲の両端をラベルに表示します。
  - 各ラベルは、できれば 1 ワードだけにします。
  - 末尾に句点を付けません。
  - これらのラベルは、説明的で対比的なものにします。例: 最大/最小、多/少、低/高、小/大
- **値ラベル**

  値ラベルは、スライダーの現在の値を表示します。
  - 値ラベルが必要な場合は、スライダーの下に表示します。
  - テキストをコントロールに対して中央に配置し、単位 (ピクセルなど) を付記します。
  - スライダーのつまみはスクラブ中に隠れるため、ラベルや他のビジュアル効果で現在の値を表示することをお勧めします。スライダー設定のテキスト サイズは、スライダー以外の適切なサイズのサンプル テキストに連動させることができます。

外観と操作

スライダーはトラックとつまみで構成されます。トラックは入力できる値の範囲を表すバーです (オプションでさまざまなスタイルの目盛りを表示できます)。つまみは、ユーザーがトラックをタップするか、トラックを前後にスクラブして位置を調整できるセレクターです。

スライダーには大きなタッチ ターゲットが設定されています。タッチのアクセシビリティを維持するには、スライダーを表示の端から十分に離して配置する必要があります。

カスタム スライダーを設計する際は、余分な要素をできるだけなくし、ユーザーに必要なすべての情報を示す方法を検討してください。ユーザーが設定を理解できるように単位を表示する必要がある場合は、値ラベルを使います。これらの値を視覚的に示す方法を工夫してください。たとえば、音量を調整するスライダーでは、スライダーの最小の端に音波のないスピーカーのグラフィック、最大の端に音波のあるスピーカーのグラフィックを表示できます。

SplitView コントロールのガイドライン

SplitView コントロールには、展開/折りたたみ可能なウィンドウとコンテンツ領域があります。コンテンツ領域は常に表示されます。ウィンドウは展開/折りたたみを行うことも、開いた状態のままにすることもでき、アプリ ウィンドウの右側または左側から表示できます。このウィンドウには 3 つのモードがあります。

重要な API

SplitView クラス (XAML)　　SplitView オブジェクト (WinJS)

- **オーバーレイ**
  ウィンドウは開くまで表示されません。開くと、ウィンドウはコンテンツ領域をオーバーレイします。
- **インライン**
  ウィンドウは常に表示され、コンテンツ領域をオーバーレイしません。画面領域はウィンドウとコンテンツ領域に分割されます。
- **コンパクト**
  このモードでは、ウィンドウは常にアイコンを表示できるだけの大きさ (通常は幅 48 epx) で表示されます。画面領域はウィンドウとコンテンツ領域に分割されます。標準的なコンパクト モードはコンテンツ領域をオーバーレイしませんが、より多くのコンテンツを表示するより大きなウィンドウに変換でき、コンテンツ領域をオーバーレイします。

適切なコントロールの選択

SplitView コントロールは、ナビゲーション ウィンドウ パターンの作成に使うことができます。このパターンを構築するには、展開/折りたたみボタン ("ハンバーガー" ボタン) とリスト ビューを SplitView コントロールに追加する必要があります。

例

既定の形式の SplitView コントロールは、基本的なコンテナーです。ボタンとリスト ビューを追加すると、SplitView コントロールをナビゲーション メニューとして使うことができる

ようになります。ナビゲーション メニューとしての分割ビューの例を次に示します (展開モードとコンパクト モード)。

推奨事項

- ナビゲーション メニューの SplitView を使う場合、アプリの別の領域にアクセスできるようにするためのナビゲーション コントロールをウィンドウ内に配置することをお勧めします。ナビゲーションにウィンドウを使うことで、一貫したユーザー エクスペリエンスを提供できます。このメニューを実装すると、ユーザーはアプリのすべての部分の使い方を習得することができ、アプリのホーム ページにすばやくアクセスできるようになります。また、アプリのさまざまな領域を探索するようにユーザーを促すこともできます。

タブとピボット のガイドライン

タブとピボットは、アクセス頻度の高い個別のコンテンツ カテゴリ間を移動するために使います。タブ/ピボット パターンは、対応するカテゴリ ヘッダーがある 2 つ以上のコンテンツ ウィンドウで構成されます。ヘッダーは常に画面上に表示され、選択状態が明確に示されるので、ユーザーは現在使っているカテゴリを常に意識するようになります。

重要な API

Pivot クラス (XAML)　　Pivot オブジェクト (WinJS)

タブとピボットは実質的には同じパターンであり、どちらもピボット コントロールを使って構築されます。ピボット コントロールの基本機能については、この記事で後述します。

機能

タブ/ピボット パターンを使ってアプリを構築する場合は、パターンの構成可能な機能セットに基づいて考慮すべきいくつかの重要なデザインの変数があります。

ヘッダーの配置。 ヘッダーは、画面の一番上または一番下に配置できます。

ヘッダー ラベル。 ヘッダーは、テキスト付きのアイコン、テキストのみ、またはアイコンのみで示すことができます。

ヘッダーの整列。 ヘッダーは、左揃えまたは中央揃えにすることができます。

トップレベルまたはサブレベルのナビゲーション。 タブ/ピボットは、どちらかのレベルのナビゲーションに使うことができ、トップレベル/サブレベル パターンで並べて表示できます。タブ/ピボットのレベルが 2 つある場合は、トップレベルとサブレベルのヘッダーを視覚的に差別化して、ユーザーがその 2 つを明確に区別できるようにする必要があります。

タッチ ジェスチャのサポート。 タッチ ジェスチャをサポートするデバイスでは、次に示す 2 つの操作セットのどちらかを使ってコンテンツ カテゴリ間を移動できます。

1. タブ/ピボット ヘッダーをタップして、そのカテゴリに移動するか、またはコンテンツ領域上でスワイプして隣接するカテゴリに移動します。
2. タブ/ピボット ヘッダーをタップして、そのカテゴリに移動します (スワイプ操作なし)。

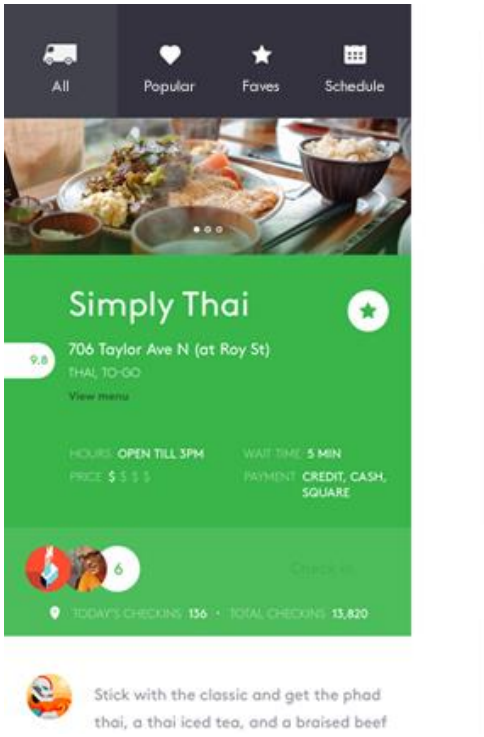
パターン構成

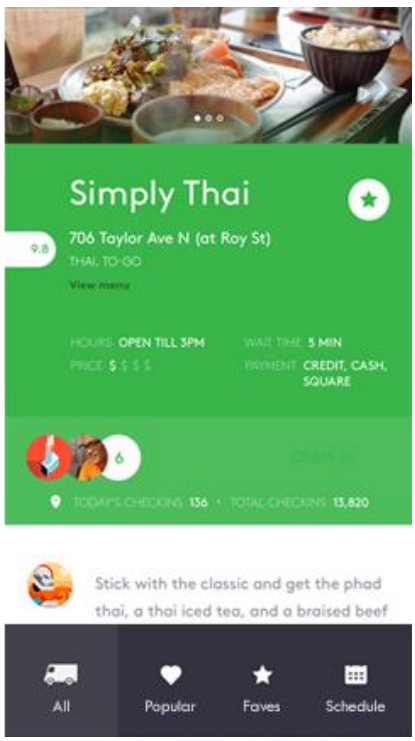
タブ/ピボット パターンの最適な配置は、操作のシナリオとアプリが表示されるデバイスによって異なります。次の表に、いくつかの使用頻度の高いシナリオとパターン構成の概要を示します。

操作のシナリオ	推奨される構成
携帯電話またはファブレットで 2 ～ 5 個のトップレベルの一覧またはグリッド ビューのコンテンツ カテゴリ間を横方向に移動する	タブ/ピボット: 画面の一番上に配置 (中央揃え) ヘッダー ラベル: アイコンとテキスト コンテンツ領域でのスワイプ: 有効
携帯電話またはファブレットでコンテンツ カテゴリの範囲内を移動する (ナビゲーション用としては実用的でないコンテンツ領域上でスワイプする)	タブ/ピボット: 画面の一番下に配置 (中央揃え) ヘッダー ラベル: アイコンとテキスト コンテンツ領域でのスワイプ: 無効
マウスとキーボードを使ったトップレベルのナビゲーション または タッチ デバイスでのページ レベルのナビゲーション	タブ/ピボット: 画面の一番上に配置 (左揃え) ヘッダー ラベル: テキストのみ コンテンツ領域でのスワイプ: 無効

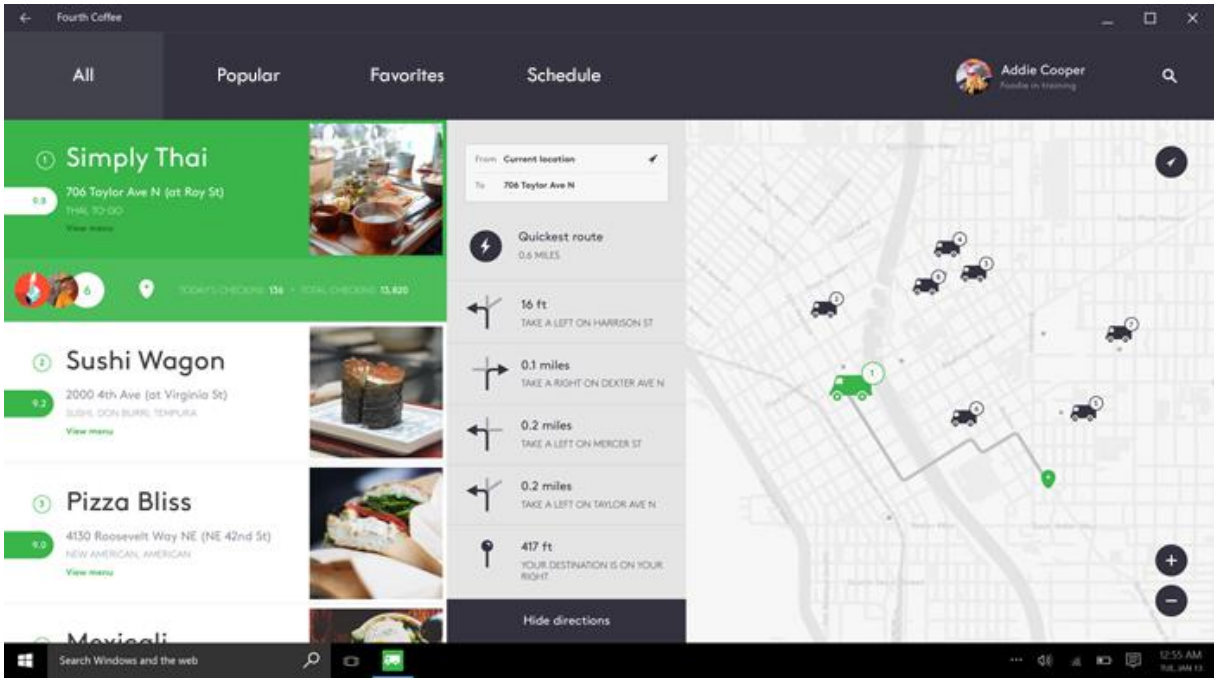
例

このフード トラック アプリのデザインは、タブ/ピボット ヘッダーを画面の一番上または一番下に配置するとどのようになるかを示しています。モバイル デバイスでは、一番下に配置すると手に届きやすくなります。





ノート PC/デスクトップ PC のフード トラック アプリのデザインではテキストのみのヘッダーを使います。テキスト付きのアイコンをヘッダーに使うと、タッチ補正には役立ちますが、マウスとキーボードを使う場合はテキストのみのヘッダーが適しています。



ピボット コントロール

タブ/ピボット ナビゲーション パターンはピボット コントロールを使って構築します。このコントロールには、このセクションで説明する基本的な機能が付属しています。

ピボット コントロールを使うと、次のタッチ ジェスチャ操作が可能になります。

- ヘッダーをタップすると、そのヘッダーのセクション コンテンツに移動します。
- ヘッダー上で左または右にスワイプすると、隣接するヘッダー/セクションに移動します。
- セクション コンテンツ上で左または右にスワイプすると、隣接するヘッダー/セクションに移動します。

コントロールには次の 2 つのモードがあります。

固定

- 許可されている領域内にすべてのピボット ヘッダーが収まる場合、ピボットは固定されます。
- ピボット ラベルをタップすると、ピボット自体は移動しませんが、対応するページに移動します。アクティブなピボットは強調表示されます。

カルーセル

- 許可されている領域内にすべてのピボット ヘッダーが収まらない場合、ピボットがカルーセル表示されます。
- ピボット ラベルをタップすると対応するページに移動し、アクティブなピボット ラベルは最初の位置までカルーセル表示されます。

このコントロールには、ヘッダーの数とラベルの文字列の長さに基づく組み込みのブレークポイント機能があります。

推奨事項

- タブ/ピボット ヘッダーの配置は画面サイズに基づいて行うことをお勧めします。画面幅が 720 epx 未満の場合、通常は中央揃えが適しています。720 epx 以上の画面幅では、ほとんどの場合、左揃えをお勧めします。
- ウィンドウの拡大/縮小時に、タブ/ピボット ヘッダーの数が利用可能な領域を超えると、ヘッダーをオーバーフロー領域に表示する処理を開始します。
- タブ/ピボットは横向きまたは縦向きの画面で使うことができますが、どちらの向きでもヘッダー (表示と非表示) の総数が同じになるようにしてください。
- カルーセル (ラウンド トリップ) モードを使う場合は、6 つ以上のヘッダーを使わないようにしてください。6 つ以上のヘッダーを使うと、ユーザーが画面の向きを認識できなくなる可能性があります。
- モバイル デバイスでは、UI の別の部分でスワイプを使う場合や、UI が上部に集中しすぎないようにするために、タブ/ピボットを一番下に配置すると手に届きやすくなります。
- スクリーン キーボードが表示されている場合は、ヘッダーを画面外に移動して領域を確保できます。

トグル スイッチのガイドライン

トグル スイッチは、ユーザーが項目をオンまたはオフに切り替えることができる物理的なスイッチを模したものです。 このコントロールには、オン (checked が **true**) とオフ (checked が **false**) の 2 つの状態があります。

重要な API

ToggleSwitch クラス (XAML)　　ToggleSwitch オブジェクト (WinJS)

適切なコントロールの選択

トグル スイッチは、ユーザーが変更した後すぐに有効になるバイナリ操作に対して使います。たとえば、トグル スイッチを使って、サービスまたはハードウェア コンポーネントをオンまたはオフにできます。

トグル スイッチを使うかどうかを判断するための良い方法は、そのような操作を行うために物理的なスイッチを使うかどうかを考えてみることです。

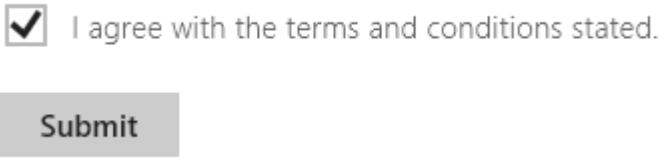
ユーザーがスイッチをオンまたはオフに切り替えた後は、対応する操作をすぐに実行します。

トグル スイッチとチェック ボックスの選択

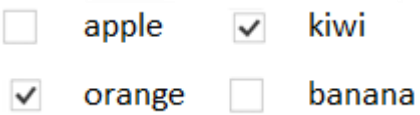
状況によっては、トグル スイッチとチェック ボックスのどちらでも使える場合があります。次のガイドラインに従って、どちらかを選択します。

- ユーザーが変更した後すぐに変更が有効になるようなバイナリ設定に対しては、トグル スイッチを使います。

  トグル スイッチの場合は、ワイヤレスがオンになっていることが明らかです。一方、チェック ボックスの場合は、ワイヤレスが現在オンになっているのか、またはオンにするためにチェック ボックスをオンにする必要があるのか、ユーザーが考える必要があります。
- 変更を有効にするためにユーザーが追加の手順を実行する必要があるときは、チェック ボックスを使います。たとえば、ユーザーが [送信] や [次へ] などのボタンをクリックして変更を適用する必要がある場合は、チェック ボックスを使います。



- ユーザーが複数の項目を選択できるときは、チェック ボックスまたはリスト ビューを使います。



推奨事項

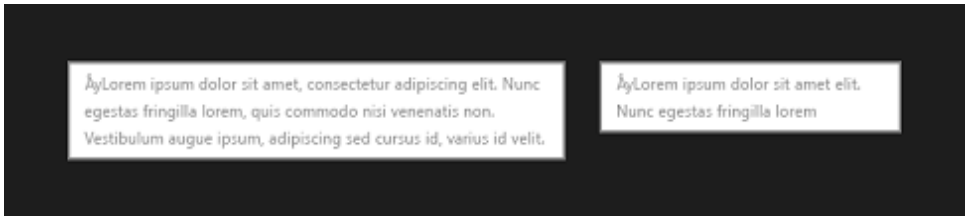
- 設定に対して具体的なラベルがある場合は、それらを "オン" や "オフ" の代わりに使います。特定の設定に対してより適切な、対になる項目を表す短い (3 ～ 4 文字

の) ラベルがある場合は、それらを使います。たとえば、設定が "画像の表示" である場合は、"表示/非表示" などを使います。より具体的なラベルを使うと、UI のローカライズ時に役立ちます。

- 必要な場合以外は、"オン" または "オフ" のラベルを変更しない。独自のラベルが必要な場合以外は既定のラベルを使います。
- ラベルは 4 文字以内である必要があります。

ツールチップのガイドライン

ツールチップは、他のコントロールまたはオブジェクトにリンクされた短い説明です。ツールチップを使うと、UI では直接説明されていない、なじみのないオブジェクトをユーザーが理解しやすくなります。ツールチップは、ユーザーがコントロール上で長押しするか、マウス ポインターをコントロール上にホバーすると、自動的に表示されます。ツールチップは、ユーザーが指、マウス ポインター、またはペン ポインターを移動したときに消えます。

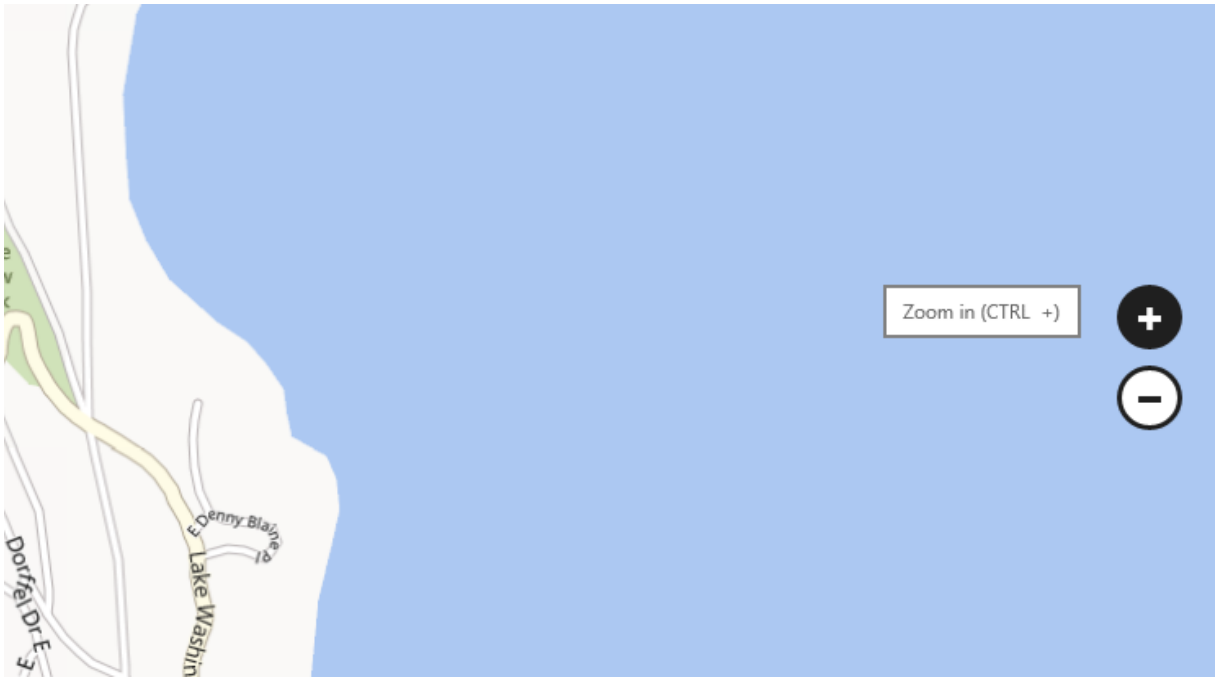


重要な API

ToolTip クラス (XAML)

Tooltip オブジェクト (WinJS)

例



適切なコントロールの選択

ツールチップは、他のコントロールまたはオブジェクトにリンクされた短い説明です。ツールチップを使うと、UI では直接説明されていない、なじみのないオブジェクトをユーザーが理解しやすくなります。ツールチップは、ユーザーがコントロール上で長押しするか、マウス ポインターをコントロール上にホバーすると、自動的に表示されます。ツールチップは、ユーザーが指、マウス ポインター、またはペン ポインターを移動したときに消えます。

ユーザーに操作の実行を指示する前に、ツールチップを使ってコントロールに関する詳しい情報を表示します。また、ツールチップを使うと、タッチ ダウン中に指の下に項目を表示できるため、ユーザーは自分がどこをタッチしているのかを知ることができます。ただし、最初は他の方法で場所を明確にするよう試みてください。たとえば、より大きなコントロールやより大きな領域を使ったり、コントロールのアクティブ状態やホバー状態にスタイルを適用したりします。

ツールチップはどのような場合に使えばよいでしょうか。それを判断するには、以下の質問を考えます。

- **情報はポインターをホバーすることで表示されますか?**
  そうでない場合は、別のコントロールを使います。ツールチップを、ユーザーの操作の結果としてのみ表示します。自動的には表示しません。
- **コントロールにはテキスト ラベルがありますか?**
  ない場合は、ツールチップを使ってラベルを表示します。ほとんどのコントロールにはラベルを付けることをお勧めします。それらのコントロールには、ツールチップは必要ありません。グラフィックのラベルを持つツール バー コントロールやコマンド ボタンには、ツールチップが必要です。
- **より詳しい説明や追加情報がオブジェクトに対して役立ちますか?**
  そうであれば、ツールチップを使います。ただし、このテキストは、主要なタスクに必須なものではなく、補助的なものである必要があります。必須なものであれば、直接 UI に配置して、ユーザーが探さなくても済むようにします。
- **表示する補助的な情報は、エラー、警告、または状態ですか?**
  その場合は、フライアウトなど、他の UI 要素を使います。
- **ユーザーがツールチップを操作する必要がありますか?**
  その場合は、別のコントロールを使います。ツールチップはマウスを動かすと消えるため、ユーザーはツールチップを操作できません。
- **ユーザーが補助的な情報を印刷する必要がありますか?**
  その場合は、別のコントロールを使います。
- **ユーザーがツールチップを煩わしいと感じますか?**
  その場合は、別の手段を使うことを検討します。何もしない、という選択肢もあります。煩わしいと感じる可能性があってもツールチップを使う場合は、ユーザーがツールチップをオフにできるようにします。

ツールチップの適切な使い方の例を次に示します。

- ユーザーがカレンダーの日付をタッチしたときに曜日を表示する。
- ユーザーがハイパーリンクをタッチしたときにリンク先の Web サイトのプレビューを表示する。

推奨事項

- ツールチップは慎重に使う (または使わない)。ツールチップは作業の中断になります。ツールチップはフライアウトと同じように煩わしい場合があるため、大きな付加価値がない限り使わないでください。
- ツールチップのテキストは簡潔なものにする。ツールチップは短い文やフレーズに適しています。大きなテキストのまとまりは読みにくく、圧迫感を与えます。
- 役に立つ補足的なツールチップ テキストを作成する。ツールチップのテキストは、情報として役に立つ必要があります。表示しなくても明らかな情報や、既に画面上に表示されている内容の繰り返しなどは避けます。ツールチップのテキストは常に表示されているわけではないため、ユーザーが必ずしも読まなくても問題がないような、補足的な情報である必要があります。重要な情報は、名前から判別できるコントロール ラベルを使うか、補足的なテキストを適切な場所に配置することで伝えるようにします。
- 状況に応じて画像を使う。ツールチップ内に画像を使うとよい場合もあります。たとえば、ユーザーがハイパーリンクをタッチしたときに、ツールチップを使ってリンク先ページのプレビューを表示できます。
- 既に UI に表示されているテキストは、ツールチップとして表示しない。たとえば、ボタンをタッチするとテキストが隠れる場合を除き、同じテキストが表示されているボタンにはツールチップを表示しません。
- ツールチップ内に対話的なコントロールを配置しない。
- 対話的に見えるような画像をツールチップ内に配置しない。

その他の使い方のガイダンス

ツールチップは慎重に使い、タスクを完了しようとしているユーザーにとって明らかに重要である場合にのみ追加します。1 つの目安は、情報が同じエクスペリエンスのどこかで入手できる場合、ツールチップは必要ありません。価値あるツールチップによって、不明瞭な操作を明確にします。

ユーザーに操作の実行を指示する前に、ツールチップを使ってコントロールに関する詳しい情報を表示します。また、ツールチップを使うと、タッチ ダウン中に指の下に項目を表示できるため、ユーザーは自分がどこをタッチしているのかを知ることができます。

Web ビューのガイドライン

Web ビュー コントロールは、Microsoft Edge のように動作するビューをアプリに組み込みます。また Web ビュー コントロールでは、ハイパーリンクの表示と動作が可能です。

Web ビュー コントロールを使うことで、Web サーバーより取得した HTML コンテンツや動的に生成されたコードとコンテンツ ファイルをブラウザのように表示する機能をアプリに提供します。HTML コンテンツは、スクリプト コードを使ってアプリのコードと連携することができます。

重要な API

WebView クラス (XAML)	x-ms-webview 要素 (HTML)

推奨事項

- 読み込まれた Web サイトがデバイスに対して正しく書式設定されており、アプリの他の部分に対して一貫性のある色、タイポグラフィ、ナビゲーションが使用されていることを確認します。
- 入力フィールドは適切なサイズに設定する必要があります。テキストを入力する際にズームインできることにユーザーが気付かない場合があります。
- Web ビューがアプリの他の部分とは異なって見える場合は、関連タスクを実行するための代替のコントロールまたは手段を検討します。Web ビューがアプリの他の部分に一致すれば、ユーザーにはすべてが 1 つのシームレスなエクスペリエンスとして認識されます。

インタラクションのガイドライン

このセクションのガイドラインでは、ユーザーがアプリを操作できるさまざまな方法と、使いやすいアプリを設計するための推奨事項について説明します。

このセクションの内容

トピック	説明
Cortana	音声コマンドを使い、アプリに用意されている機能を利用して **Cortana** を拡張します。
キーボード	キーボードはテキスト用の主要な入力デバイスであり、多くの場合、特定の障碍のあるユーザーや、キーボードを使った方がアプリをすばやく効率よく操作できると考えるユーザーにとって欠かせません。
マウス	ユニバーサル Windows プラットフォーム (UWP) アプリの設計はタッチ入力用に最適化し、既定の基本的なマウスのサポートを利用します。
ペン	ユニバーサル Windows プラットフォーム (UWP) アプリの設計はタッチ入力用に最適化し、既定の基本的なペンのサポートを利用します。
音声認識	音声認識や音声合成 (TTS: text-to-speech) をアプリのユーザー エクスペリエンスに直接統合します。
タッチ操作	タッチ用に最適化される一方で、さまざまな入力デバイスで一貫した機能を提供する、直観的で独特なユーザー操作エクスペリエンスを備えた UWP アプリを作成します。
タッチパッド	ユニバーサル Windows プラットフォーム (UWP) アプリの設計はタッチ入力用に最適化し、既定のタッチパッドのサポートを利用します。
複数の入力方法	人がお互いにコミュニケーションをとる際に音声とジェスチャを組み合わせて使うように、アプリの操作では、複数の種類とモードの入力を使用すると便利な場合があります。
アクセシビリティ	こうしたアクセシビリティ対応の設計の原則に従うことで、可能な限り広範な対象ユーザーにアプリを利用してもらい、Windows ストアでアプリに対してより多くのユーザーの興味を喚起するうえで役立ちます。

クロスライド	クロススライドは、スワイプ ジェスチャによる選択や、スライド ジェスチャによるドラッグ (移動) 操作をサポートするために使います。
光学式ズームとサイズ変更	このトピックでは、Windows のズームと要素のサイズ変更について説明し、アプリでこのような新しい操作のメカニズムを使うときのユーザー エクスペリエンスのガイドラインを示します。
パン	パンとスクロールにより、ユーザーは単一ビュー内で移動し、ビューポートに収まらないビューのコンテンツを表示できます。ビューの例として、コンピューターのフォルダー構造、ドキュメントのライブラリ、フォト アルバムなどがあります。
回転	このトピックでは、新しい Windows UI の回転について説明し、Windows アプリでこの新しい操作のメカニズムを使うときに考慮する必要があるユーザー エクスペリエンスのガイドラインを示します。
テキストと画像の選択	このトピックでは、テキスト、画像、コントロールを選んだり操作したりするための新しい Windows UI について説明します。また、Windows アプリでこの新しい選択と操作のメカニズムを使うときに考慮する必要のあるユーザー エクスペリエンスに関するガイドラインを示します。
ターゲット設定	このトピックでは、タッチ補正のための接触形状の使用について説明し、Windows ランタイム アプリでのターゲット設定のベスト プラクティスを紹介します。
ビジュアルなフィードバック	ビジュアルなフィードバックは、Windows アプリの対話操作が検出、解釈、処理されていることをユーザーに示すために使います。

Cortana の設計ガイドライン

音声コマンドを使い、アプリに用意されている機能を利用して **Cortana** を拡張します。アプリを起動するか、アプリを起動してコマンドを実行するか、またはここで説明するように、機能をアプリから **Cortana** UI に直接組み込みます。

音声コマンドは、具体的な目的を持って 1 つの言葉を声に出すことであり、音声コマンド定義 (VCD) ファイルで定義されています。**Cortana** を通じてインストール済みアプリに指示が伝えられます。 VCD ファイルは、1 つ以上の音声コマンドを定義する XML ファイルです。各音声コマンドは固有の目的を持っています。

VCD 内の各音声コマンド定義にはさまざまなものがあり、定義が複雑になる場合があります。また、音声コマンド定義では、制限された 1 つの言葉の発声から、より柔軟性の高い自然言語の発声のコレクションまで、あらゆる発声をサポートできます。ただし、これらの発声はすべて、同じ固有の目的を示している必要があります。

ターゲット アプリは、操作の複雑さに応じて、フォアグラウンドで起動したり (アプリがフォーカスを取得します)、バックグラウンドでアクティブ化されたりします (**Cortana** がフォーカスを維持しますが、アプリからの結果を表示します)。たとえば、追加のコンテキストやユーザー入力 (特定の連絡先へのメッセージの送信など) が必要な音声コマンドはフォアグラウンド アプリで処理するのが最適ですが、基本的なコマンドはバックグラウンド アプリを介して **Cortana** で処理できます。

アプリの基本的な機能を統合して、ユーザーが直接アプリを開かずにほとんどのタスクを実行できる中心的エントリ ポイントを提供することで、**Cortana** はアプリとユーザーの仲介役となります。多くの場合、これによってユーザーの時間と労力を大幅に減らすことができます。

音声機能が適切に設計され、実装されていると、ユーザーがアプリを楽しく確実に操作できる手段になります。音声機能によって、キーボード、マウス、タッチ、ジェスチャを補完することも、場合によってはこれらの代替として使うこともできます。

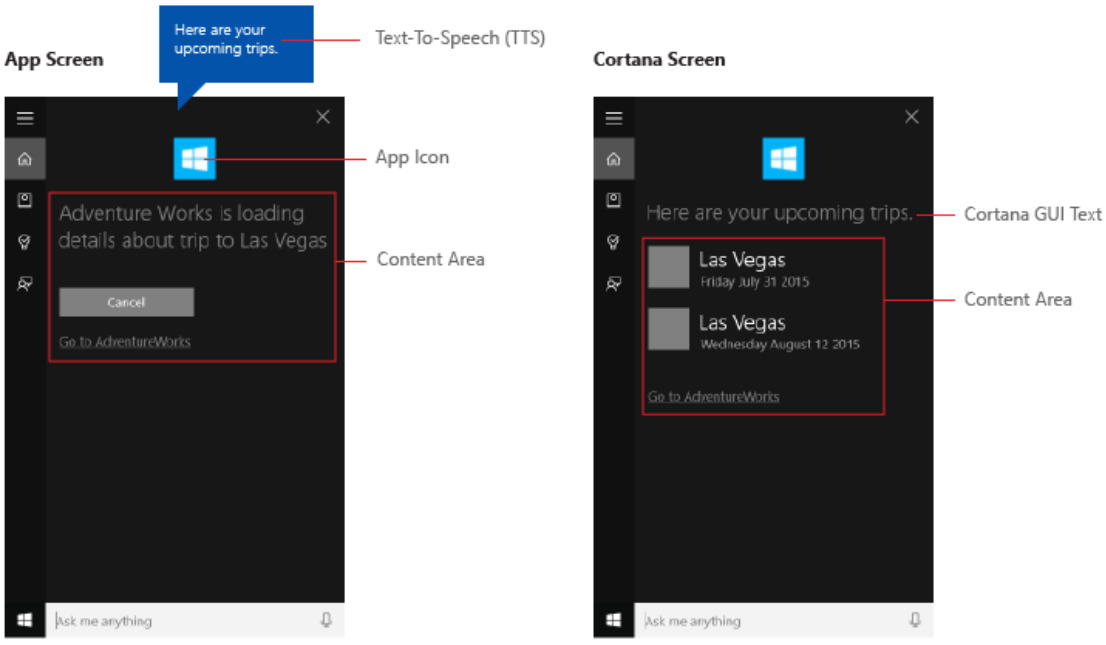
Cortana とのインタラクション設計

このガイドラインおよび推奨事項では、アプリがユーザーとやり取りしてタスクを実行し、どのように行われているかがすべて明らかになるように **Cortana** を有効に活用する方法について説明します。

Cortana では、アプリケーションをバックグラウンドで実行してユーザーに確認や不明瞭解消を求め、その返答として音声コマンドの状態に関するフィードバックを生成できます。このプロセスは軽量で高速のため、ユーザーが **Cortana** のエクスペリエンスから離れたり、アプリケーションにコンテキストを切り替えたりする必要がありません。

ユーザーは、できる限り軽量で簡単なプロセスになっているのは **Cortana** のおかげだと感じますが、アプリのおかげでタスクを実行できていることが **Cortana** ではっきりわかるようにすることもできます。

ここでは、**Cortana** UI に統合されている Adventure Works という旅行の計画および管理アプリを使って、さまざまな概念や機能について説明します。



メッセージの作成

Cortana をうまく操作するには、音声合成 (TTS) と GUI 文字列を作るときの基本的な原則に従う必要があります。

原則	悪い例	良い例
効率的である 使う単語はできるだけ少なくし、最も重要な情報が目立つようにします。	間違いなく可能です。今日はどのムービーをお探しですか。大きなコレクションがあります。	もちろんです。どのムービーをお探しですか。
関連性がある タスク、コンテンツ、コンテキストに関連する情報のみ提供します。	プレイリストに追加しました。念のためお知らせしますが、バッテリが少なくなっています。	プレイリストに追加しました。
明確である あいまいさを回避します。専門的な用語ではなく、日常的な言葉を使います。	クエリ「"ラスベガス旅行"」の結果はありません。	ラスベガス旅行が見つかりませんでした。
信頼できる できる限り正確な情報にします。バックグラウンドで何が行われているのか明白にします。タスクがまだ終わっていない場合、終わったと述べないでください。プライバシーを尊重して、個人情報を大きな音で読み上げないようにしてください。	そのムービーは見つかりませんでした。まだリリースされていません。	そのムービーはカタログで見つかりませんでした。

話し言葉を使います。自然な口調のためには、文法的な正確さを重視しません。たとえば、TTS の読み上げでは、「"見れる"」や「"食べれる"」のような、よく耳にする話し言葉であってもかまいません。

可能であり、自然であれば、暗示的な一人称を使います。たとえば、「"Adventure Works の次の旅行を探しています"」は、だれかが探していること示しますが、それを示すために「"私"」のような語句は使用しません。

アプリをより自然にするため、いくつかのバリエーションを使用します。実質的に同じことを言うために、異なる TTS と GUI 文字列を提供します。たとえば、「"どんな映画を見ます

か"」のバリエーションとして、「"どんな映画を鑑賞しますか"」があります。人は、同じことをいつもまったく同じ方法では言わないものです。ただし、TTS バージョンと GUI バージョンは必ず同期します。

「"OK"」や「"わかりました"」などの語句は、応答で慎重に使います。同意や前進の意味を伝えることができますが、変化を持たせずに使いすぎるとくどくなる可能性もあります。

注 同意の語句は TTS でのみ使用します。 **Cortana** のキャンバスのスペースは限られているため、対応する GUI 文字列で同じ語句を繰り返さないでください。

より自然な操作にし、**Cortana** のキャンバスのスペースを節約するため、応答では短縮形を使います。たとえば、「"そのムービーを見つけることができませんでした"」の代わりに「"そのムービーは見つかりません"」を使います。目ではなく耳に合わせて作ります。

システムが理解できる言葉を使います。ユーザーは、表示された用語を繰り返す傾向があります。 表示内容を把握してください。

代替応答のコレクションを回転したり、ランダムに選んで、応答に変化を持たせてください。たとえば、「"どんなムービーを見ますか"」や「"どんな映画を鑑賞しますか"」などです。これにより、アプリがより自然でユニークになります。

ローカライズ

音声コマンドを使ってアクションを開始するには、ユーザーがデバイスで選んだ言語で音声コマンドをアプリに登録する必要があります (設定 > 時刻と言語 > 音声認識)。

アプリが応答する音声コマンドと、すべての TTS および GUI 文字列をローカライズする必要があります。

長い GUI 文字列は避けてください。**Cortana** のキャンバスで応答に使うことができるのは 3 行のため、それよりも長い文字列は切り捨てられます。

「ファイル、データ、グローバリゼーションのガイドライン」をご覧ください。

画像リソースとスケーリング

ユニバーサル Windows プラットフォーム (UWP) アプリでは、特定の設定とデバイス機能 (ハイ コントラスト、有効ピクセル、ロケールなど) に基づいて最適なアプリ ロゴ画像を自動的に選択できます。必要な作業は、画像を提供し、リソースのバージョンごとに、アプリ プロジェクト内で適切な名前付け規則とフォルダー構造を使用していることを確認することだけです。推奨されるリソースのバージョンが提供されない場合、ユーザーの基本設定、身体能力、デバイスの種類、場所によって、アクセシビリティ、ローカライズ、画像の品質が影響を受ける可能性があります。

ハイ コントラストとスケール ファクター用の画像リソースについて詳しくは、「タイルとアイコン アセットのガイドライン」をご覧ください。

修飾子を使ってリソースに名前を付けます。リソース修飾子は、リソースの特定のバージョンが使われるコンテキストを識別するフォルダーとファイル名の修飾子です。

標準的な命名規則は、"foldername/qualifiername-value[_qualifiername-value]/filename.qualifiername-value[_qualifiername-value].ext" です。たとえば、images/en-US/logo.scale-100_contrast-white.png は、コード内ではルート フォルダーとファイル名を使用して単に images/logo.png と参照されます。詳しくは、「ファイル、データ、グローバリゼーションのガイドライン」と「修飾子を使ってリソースに名前を付ける方法」をご覧ください。

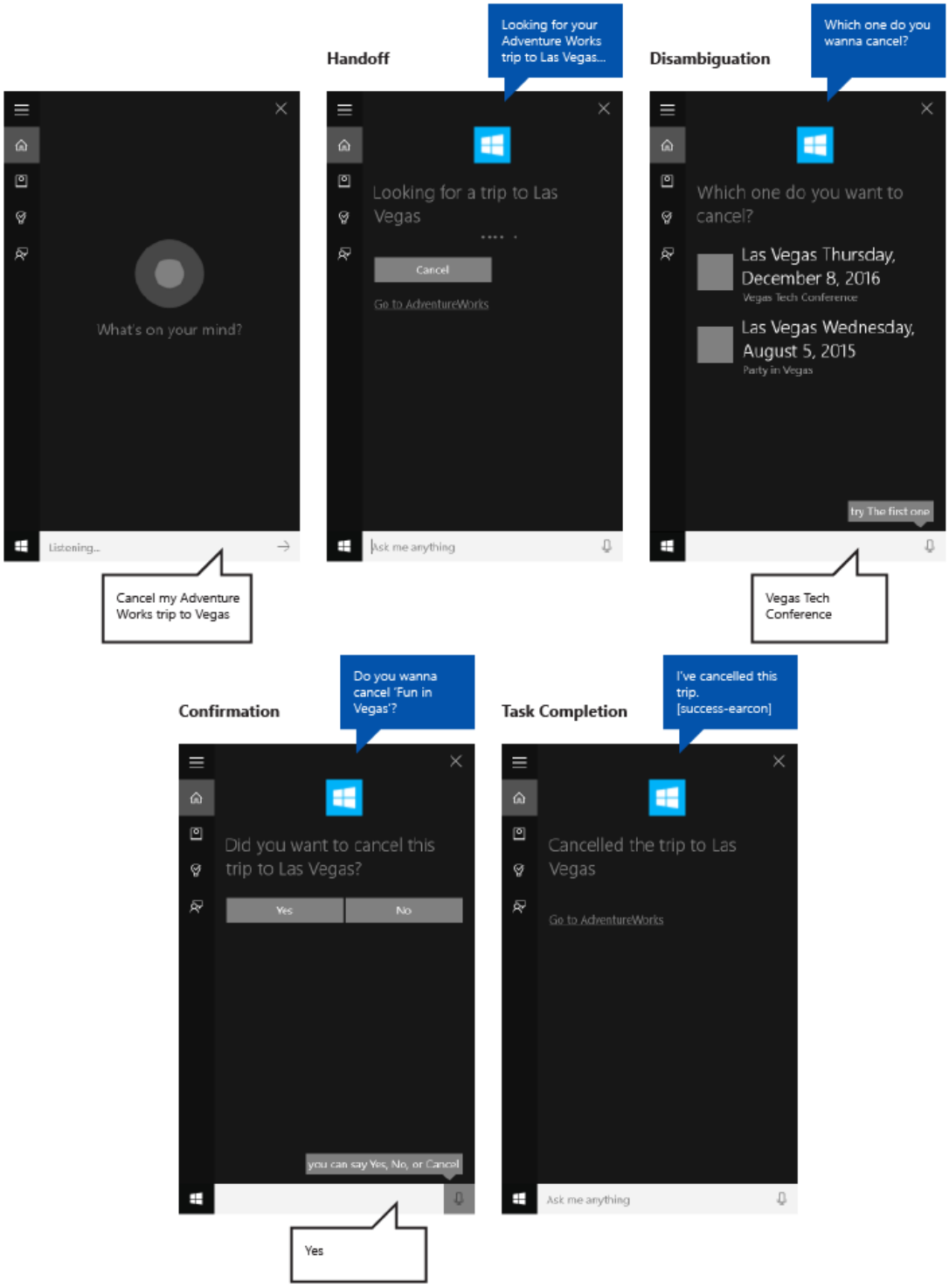
ローカライズされたリソースや複数の解像度のリソースの提供を現在計画していない場合でも、文字列リソース ファイルに既定の言語をマークし ("en-US\resources.resw" など)、画像に既定のスケール ファクターをマークする ("logo.scale-100.png" など) ことをお勧めします。ただし、100、200、400 のスケール ファクターのアセットを提供する必要があります。

重要 Cortana コンテンツ タイルの有効なアイコンのサイズは次のとおりです。

- 幅 68 x 高さ 68
- 幅 68 x 高さ 92
- 幅 280 x 高さ 140

例

この例では、**Cortana** のバックグラウンド アプリのエンド ツー エンドのタスク フローを示します。**Adventure Works** アプリは、ラスベガスへの旅行をキャンセルするために使います。

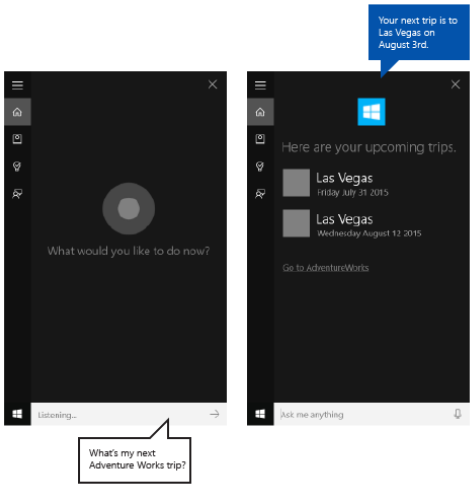


手順の概要を次の画像に示します。

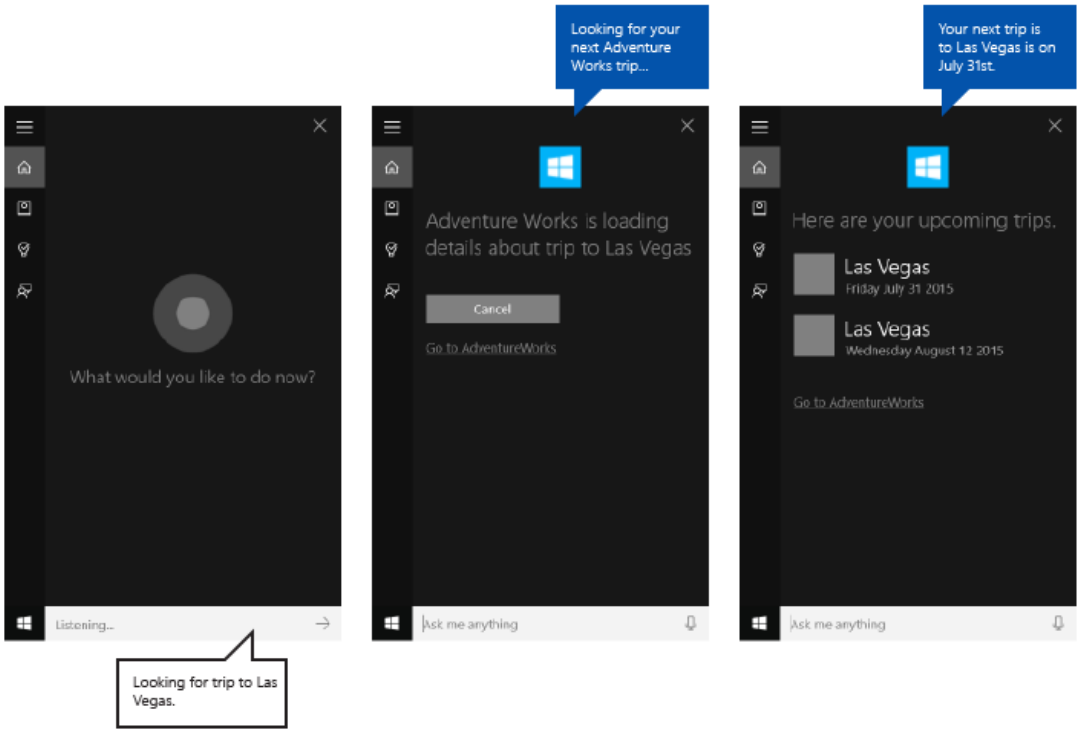
1. ユーザーは **Cortana** を開始するために、マイクをタップします。
2. ユーザーは、「"ベガスへの Adventure Works の旅行をキャンセル"」と話して、バックグラウンドで **Adventure Works** アプリを起動します。アプリは **Cortana** の音声とキャンバスの両方を使ってユーザーと対話します。
3. **Cortana** はユーザーへの確認のフィードバック ("Adventure Works で処理します")、ステータス バー、[キャンセル] ボタンが含まれたハンドオフ画面に切り替わります。
4. この場合、ユーザーには、クエリに一致する旅行が複数あることがわかったため、アプリは一致するすべての結果を示す不明瞭解消画面表示して「どの旅行をキャンセルしますか」とたずねます。
5. ユーザーは「"ベガス テック カンファレンス"」と指定します。
6. キャンセルを元に戻すことができないため、アプリは、ユーザーの意図を確認する確認画面を表示します。
7. ユーザーが「"はい"」と答えます。
8. 次に、アプリは、操作の結果を示す完了画面を表示します。

以下では、これらの手順をさらに詳しく説明します。

ハンドオフ



旅行のキャンセル (ハンドオフ画面なし)



旅行のキャンセル (ハンドオフ画面あり)

アプリが応答するまでの時間が 500 ミリ秒未満で、ユーザーからの追加の情報が必要ないタスクは、完了画面の表示を除いて、**Cortana** からのそれ以上の参加を必要とすることなく完了できます。

アプリの応答に 500 ミリ秒以上が必要な場合、**Cortana** にハンドオフ画面が表示されます。アプリ アイコンと名前が表示されます。音声コマンドが正しく理解されたことを示すために、GUI と TTS の両方のハンドオフ文字列を提供する必要があります。ハンドオフ画面は最大 5 秒間表示されます。この時間内にアプリが応答しない場合は、**Cortana** に汎用エラー画面が表示されます。

ハンドオフ画面用の GUI および TTS のガイドライン

タスクが進行中であることを明確に示します。

現在形を使用します。

開始中のタスクを確認し、特定のエンティティを参照する動作動詞を使用します。

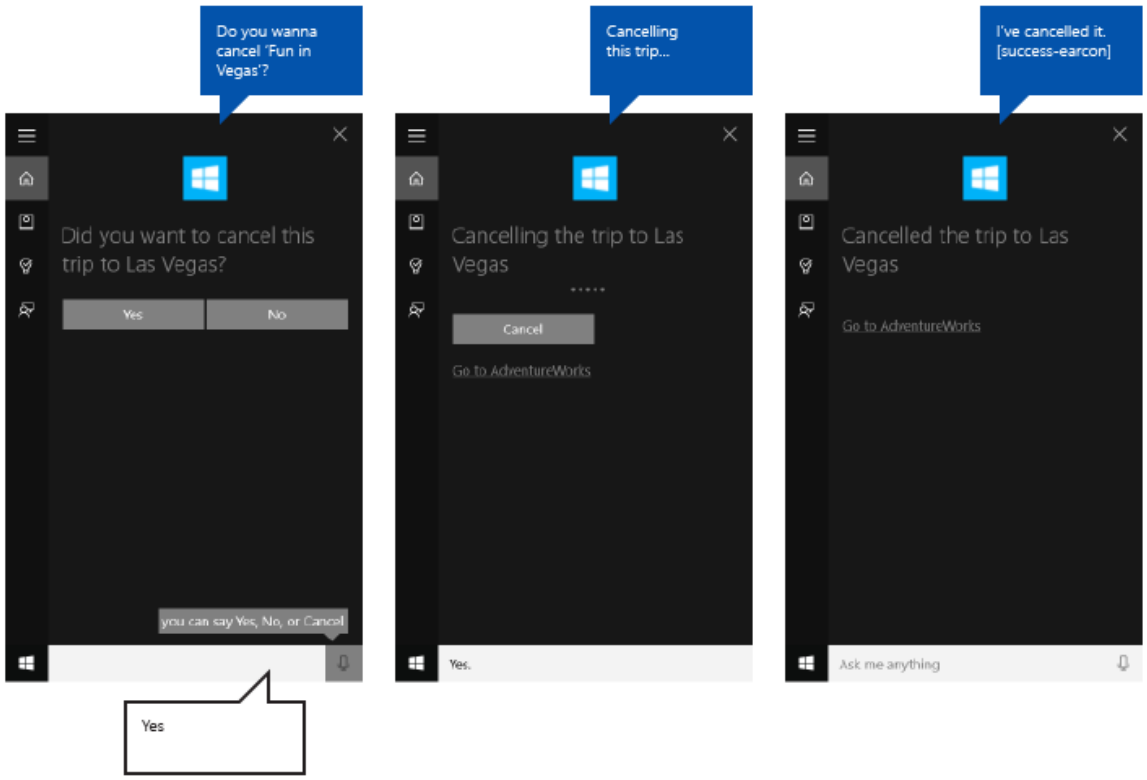
要求されたが完了していない操作をコミットしない汎用的な動詞を使います。たとえば、「"旅行をキャンセルしています"」ではなく「"旅行を探しています"」を使います。こうすれば、結果が返されない場合に、「"ラスベガスへの旅行をキャンセルしています。ラスベガスへの旅行が見つかりませんでした"」のような応答がユーザーに返されることはありません。

要求されたエンティティをまだアプリが解決する必要がある場合は、タスクが既に実行されていないことを明確にします。たとえば、「旅行をキャンセルしています」ではなく「旅行を探しています」と言っていますが、これは 0 個以上の旅行が一致し、まだ結果がわからないためです。

GUI 文字列と TTS の文字列は同じにすることができますが、その必要はありません。他のビジュアル資産が切り詰められたり、重複したりするのを避けるために、GUI 文字列は短くしてください。

TTS	GUI
Adventure Works の次の旅行を探しています。	次の旅行を探しています...
Falls City への Adventure Works の旅行を検索しています。	Falls City への旅行を検索しています...

進行状況



旅行のキャンセル (進行状況画面あり)

タスクのステップで時間がかかっている場合、アプリは何が起こっているのかを進行状況画面でユーザーに伝える必要があります。アプリ アイコンが表示されます。タスクが進行中であることを示すため、GUI と TTS の両方の進行状況文字列を提供する必要があります。

適切な状態でアプリを起動する起動パラメーターと共に、アプリへのリンクを提供する必要があります。これにより、ユーザーは自身でタスクを表示または完了できます。**Cortana** に、リンク テキスト (["Adventure Works に移動"] など) が表示されます。

進行状況画面は、それぞれについて 5 秒間表示されます。その後で、別の画面を表示する必要があり、そうしないとタスクはタイムアウトします。

進行状況画面に続いて、次の画面を表示できます。

- 進行状況

- 確認 (後で説明するように、明示的な確認)
- 不明瞭解消
- 完了

進行状況画面用の GUI および TTS のガイドライン

現在形を使用します。

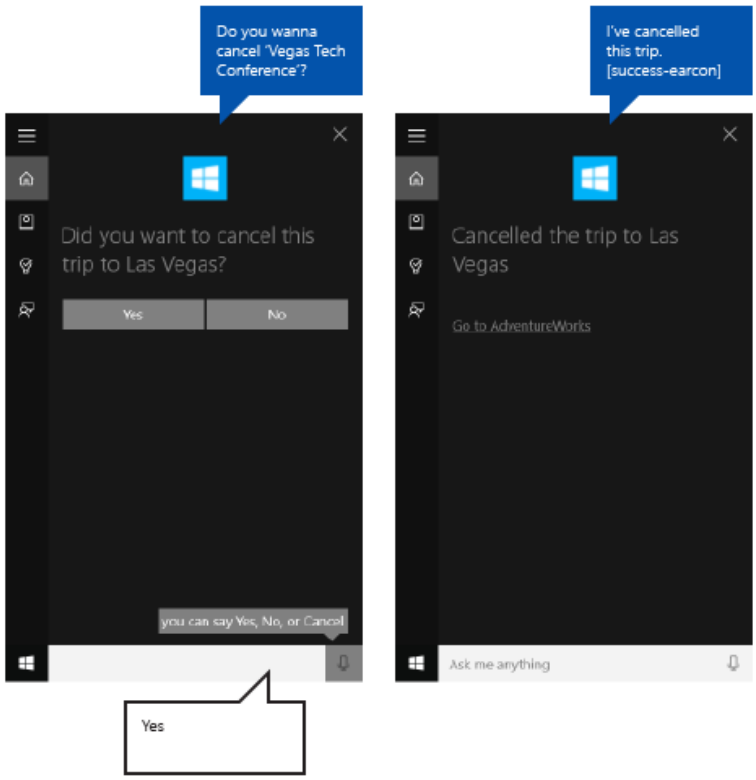
タスクが進行中であることを確認する動作動詞を使用します。

GUI: エンティティが表示される場合は、その参照 ("旅行をキャンセルしています...") を表示します。エンティティが表示されない場合は、エンティティを明示的に示します ("「ベガステック カンファレンス」をキャンセルしています")。

TTS: TTS 文字列は、最初の進行状況画面にのみ含めます。さらに進行状況画面が必要な場合は、TTS 文字列として空の文字列 {} を送信し、GUI 文字列のみを提供します。

条件	TTS	GUI
前の回でエンティティを読み取り/ エンティティを表示	この旅行をキャンセルしています...	この旅行をキャンセルしています...
前の回でエンティティを未読み取り/ エンティティを表示	ベガスへの旅行をキャンセルしています...	この旅行をキャンセルしています...
前の回でエンティティを未読み取り/ エンティティを非表示	ベガスへの旅行をキャンセルしています...	ベガスへの旅行をキャンセルしています...

確認



旅行のキャンセル (確認画面あり)

一部のタスクは、ユーザーのコマンドの性質によって暗黙的に確認できます。その他のタスクはより重要である可能性があり、明示的な確認が必要です。明示的な確認と暗黙的な確認を使う場合のいくつかのガイドラインを次に示します。

確認画面の GUI と TTS の両方の文字列はアプリおよびアプリ アイコン (提供された場合) によって指定され、**Cortana** アバターの代わりに表示されます。

ユーザーが確認に応答した後は、アプリケーションは進行状況画面に移動するのを避けるため、500 ミリ秒以内に次の画面を提供する必要があります。

次の場合に、明示的な確認を使用します。

- ユーザーからコンテンツが送信される (テキスト メッセージ、メール、ソーシャル投稿など)
- 操作を元に戻すことができない (購入の実行、何かの削除など)
- 厄介な結果が生じる可能性がある (間違い電話をかけるなど)
- より複雑な認識が必要である (オープンエンドのトランザクションなど)

次の場合に、暗示的な確認を使用します。

- コンテンツがユーザーに対してのみ保存される (自分自身へのメモなど)
- 簡単に取り消す方法がある (アラームのオンとオフなど)
- タスクをすばやくする必要がある (忘れる前にアイデアをすばやくキャプチャするなど)
- 精度が高い (シンプルなメニューなど)

確認画面用の GUI および TTS のガイドライン

現在形を使用します。

「"はい"」または「"いいえ"」で答えることができる明白な質問をユーザーにします。質問では、ユーザーが何を行おうとしているかを明示的に確認する必要があり、その他の明白なオプションが存在してはなりません。

音声コマンドを最初に認識できない場合に、再確認用に質問のバリエーションを提供します。

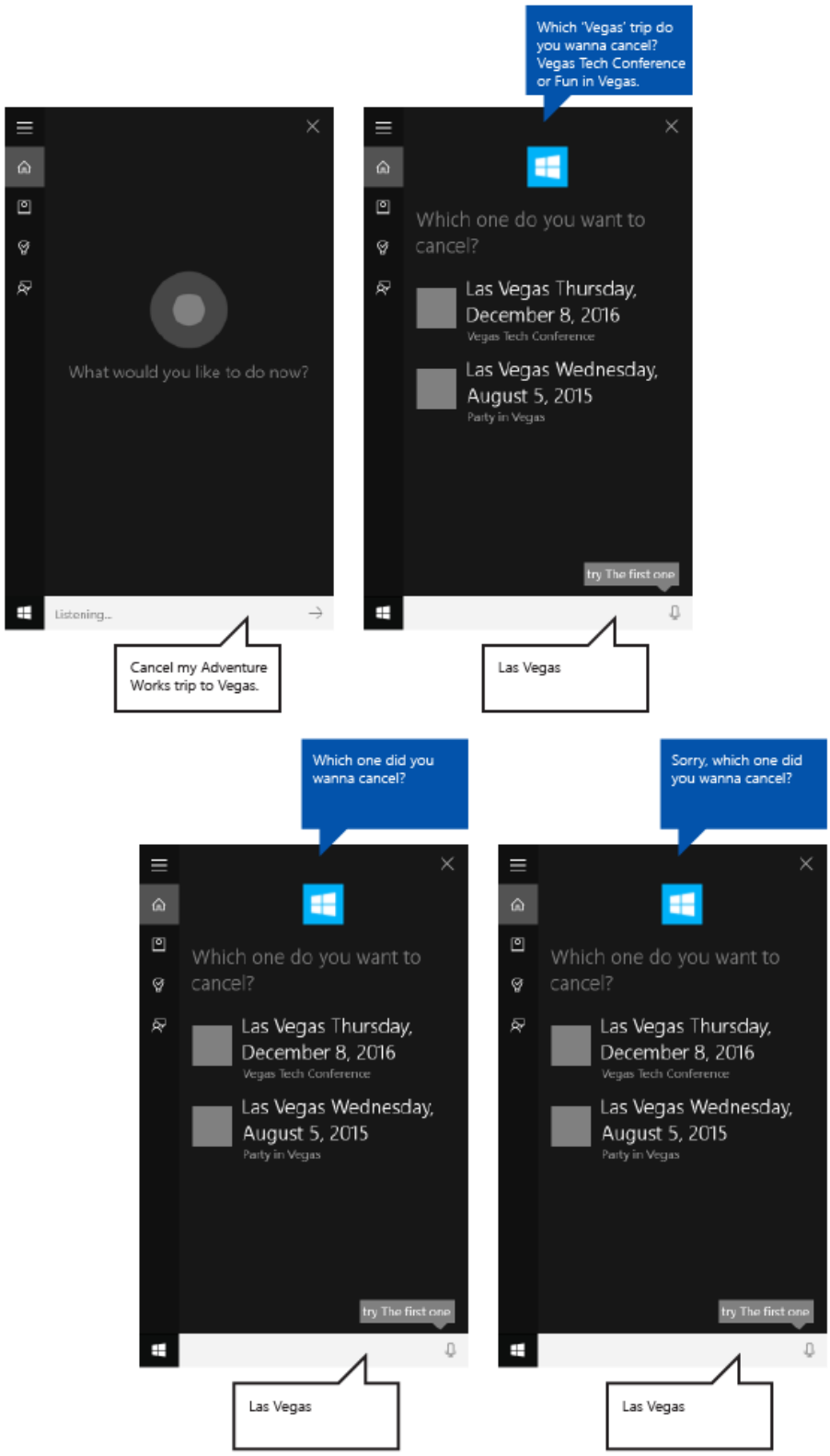
GUI: エンティティが表示される場合は、その参照を使います。エンティティが表示されない場合は、エンティティを明示的に示します。

TTS: わかりやすくするため、前の回にシステムによって読み上げられていない限り、常に特定の項目またはエンティティを参照します。

条件	TTS	GUI
前の回でエンティティを未読み取り/ エンティティを表示	ベガス テック カンファランスをキャンセルしますか。	この旅行をキャンセルしますか。

前の回でエンティティを未読み取り/ エンティティを非表示	ベガス テック カンファランスをキャンセルしますか。	ベガス テック カンファレンスをキャンセルしますか。
前の回でエンティティを未読み取り/ エンティティを非表示	この旅行をキャンセルしますか。	この旅行をキャンセルしますか。
エンティティを表示してメッセージを再表示	この旅行のキャンセルを希望しましたか。	この旅行をキャンセルすることを希望しましたか。
エンティティを表示せずにメッセージを再表示	この旅行のキャンセルを希望しましたか。	ベガス テック カンファランスをキャンセルすることを希望しましたか。

不明瞭解消



旅行のキャンセル (不明瞭解消画面あり)

一部のタスクでは、タスクを完了するには、ユーザーがエンティティの一覧からエンティティを選択する必要があります。

不明瞭解消画面の GUI と TTS の両方の文字列はアプリおよびアプリ アイコン (提供された場合) によって指定され、**Cortana** アバターの代わりに表示されます。

ユーザーが不明瞭解消の質問に応答した後は、アプリケーションは進行状況画面に移動するのを避けるため、500 ミリ秒以内に次の画面を提供する必要があります。

不明瞭解消画面用の GUI および TTS のガイドライン

現在形を使用します。

表示されるエンティティのタイトルまたはテキスト行で回答できる、明確な質問をユーザーにします。

最大 10 個のエンティティを表示できます。

各エンティティには固有のタイトルが必要です。

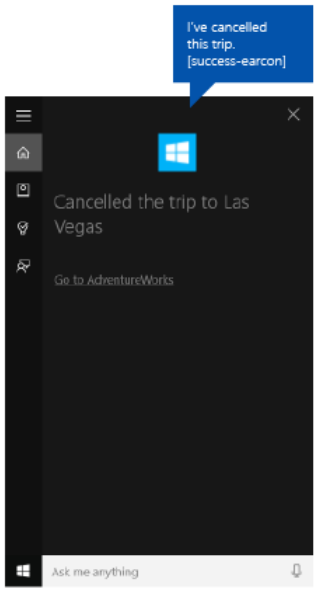
音声コマンドを最初に認識できない場合に、再確認用に質問のバリエーションを提供します。

GUI: わかりやすくするため、前の回に読み上げられていない限り、常に特定の項目またはエンティティを参照します。

TTS: 3 つ以下で短い場合を除いて、エンティティの一覧を読み上げないでください。

条件	TTS	GUI
プロンプト - 項目が 3 つ以下	どのベガス旅行をキャンセルしますか。ベガス テック カンファレンス、またはベガスのパーティのどちらですか。	どれをキャンセルしますか。
確認 - 3 つを超える項目	どのベガス旅行をキャンセルしますか。	どれをキャンセルしますか。
再確認	どのベガス旅行のキャンセルを希望しましたか。	どれをキャンセルしますか。

完了



旅行のキャンセル (完了画面あり)

タスクが正常に完了したら、アプリは要求されたタスクが正常に完了したことをユーザーに通知する必要があります。

完了画面の GUI と TTS の両方の文字列はアプリおよびアプリ アイコン (提供された場合) によって指定され、**Cortana** アバターの代わりに表示されます。

適切な状態でアプリを起動する起動パラメーターと共に、アプリへのリンクを提供する必要があります。これにより、ユーザーは自身でタスクを表示または完了できます。**Cortana** に、リンク テキスト (["Adventure Works に移動"] など) が表示されます。

完了画面用の GUI および TTS のガイドライン

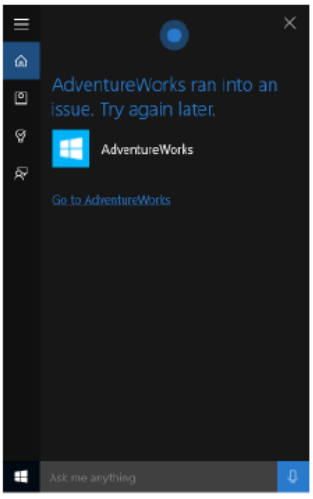
過去形を使用します。

タスクが完了したことを明示的に示すため、動作動詞を使います。

エンティティが表示される場合、または前の回に参照された場合は、参照するだけにします。

条件	TTS	GUI
前の回でエンティティを表示/読み取り	この旅行をキャンセルしました。	この旅行をキャンセルしました。
エンティティを非表示 /前の回でエンティティを未読み取り	ベガス テック カンファレンス旅行をキャンセルしました。	"ベガス テック カンファレンス"をキャンセルしました。

エラー



旅行のキャンセル (エラー画面あり)

次のいずれかのエラーが発生した場合、**Cortana** には同じ汎用エラー メッセージが表示されます。

- アプリ サービスが予期せず終了する。
- **Cortana** がアプリ サービスとの通信に失敗する。
- **Cortana** でハンドオフ画面または状況進行画面が 5 秒間表示された後で、アプリが画面を表示できない。

キーボードのガイドライン

キーボードはテキスト用の主要な入力デバイスであり、多くの場合、特定の障碍のあるユーザーや、キーボードを使った方がアプリをすばやく効率よく操作できると考えるユーザーにとって欠かせません。

ユーザーはハードウェア キーボードと 2 つのソフトウェア キーボード (スクリーン キーボード (OSK) およびタッチ キーボード) を通じて、ユニバーサル アプリを操作できます。

スクリーン キーボード (OSK)

スクリーン キーボードは、物理的なキーボードの代わりに使うことができるビジュアルなソフトウェア キーボードです。タッチ、マウス、ペン/スタイラス、またはその他のポインティング デバイス (タッチ スクリーンは必須ではありません) を通じてデータを入力します。スクリーン キーボードは、物理的なキーボードが存在しないシステムや、運動障碍により一般的な物理入力デバイスを使うことができないユーザーのために用意されています。スクリーン キーボードは、ハードウェア キーボードの機能のすべて、または少なくともほとんどをエミュレートします。

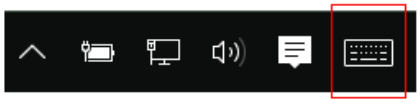
スクリーン キーボードは、[設定] – [簡単操作] - [キーボード] から有効にすることができます。

注：スクリーン キーボードは、タッチ キーボードより優先され、スクリーン キーボードが表示されている場合は表示されません。

タッチ キーボード

タッチ キーボードは、タッチ入力でのテキスト入力に使われる、ビジュアルなソフトウェア キーボードです。タッチ キーボードはテキスト入力専用であり (ハードウェア キーボードをエミュレートしません)、スクリーン キーボードの代わりになるものではありません。

デバイスに応じて、タッチ キーボーは、テキスト フィールドやその他の編集可能なテキスト コントロールがフォーカスを取得したとき、またはユーザーが**通知センター**で手動でタッチ キーボードを有効にしたときに表示されます。



注　ユーザーは [設定] – [システム] – [タブレット モード] 画面に移動して、「デバイスをタブレットとして使用すると、Windows のタッチ機能がより使いやすくなります」をオンにして、タッチ キーボードを自動的に有効にします。

テキスト入力コントロールへのフォーカスをプログラムで設定するアプリの場合、タッチ キーボードは呼び出されません。これにより、ユーザーの直接的な操作による予期しない動作を回避できます。ただし、プログラムによってキーボードがテキスト入力コントロール以外に移動されると、キーボードが自動的に非表示になります。

通常、ユーザーがフォームでコントロール間を移動している間は、タッチ キーボードは表示されたままです。 この動作は、フォーム内の他のコントロールの種類に基づいて異なります。

タッチ キーボードを使用するテキスト入力セッション中に、キーボードを閉じずにフォーカスを受け取ることができる非編集コントロールの一覧を次に示します。ユーザーがこれらのコントロールとタッチ キーボードによるテキスト入力との間で何度も行き来することが考えられるため、UI の表示を不必要に切り替えてユーザーを混乱させることのないよう、タッチ キーボードは表示されたままになります。

- チェック ボックス

- コンボボックス
- ラジオ ボタン
- スクロール バー
- ツリー
- ツリー項目
- メニュー
- メニュー バー
- メニュー項目
- ツール バー
- リスト
- リスト項目

次に、タッチ キーボードのさまざまなモードの例を示します。最初の画像は既定のレイアウトであり、2 つ目の画像は親指レイアウトです (一部の言語では利用できません)。

既定のレイアウト モードのタッチ キーボード

拡張レイアウト モードのタッチ キーボード

既定の親指レイアウト モードのタッチ キーボード

数字親指レイアウト モードのタッチ キーボード

原則

キーボード操作に成功すると、ユーザーはキーボードのみを使って基本のアプリ シナリオを実行できます。つまり、ユーザーはすべての対話型要素にアクセスし、既定の機能をアクティブにすることができます。成功の度合いには、キーボード ナビゲーション、アクセシビリティ対応のアクセス キー、上級ユーザー用のアクセラレータ (ショートカット) キーなど、さまざまな要因が影響します。

注　タッチ キーボードでは、トグルや、ほとんどのシステム コマンドがサポートされません (「パターン」をご覧ください)。

ナビゲーション

キーボードでコントロール (ナビゲーション要素を含む) を操作するには、そのコントロールがフォーカスを取得する必要があります。コントロールにキーボード フォーカスを受け取る方法の 1 つは、そのコントロールにタブ ナビゲーションでアクセスできるようにすることです。 適切にデザインされたキーボード ナビゲーション モデルでは、ユーザーが迅速かつ効率的にアプリを移動して使用するための予測可能な論理タブ オーダーを提供します。

すべての対話的なコントロールには (グループ内のコントロールでなければ) タブ ストップがあり、ラベルなど非対話型のコントロールにはありません。

一連の関連するコントロールをコントロール グループとしてまとめ、1 つのタブ ストップを割り当てることもできます。コントロール グループは、ラジオ ボタンなどのように、1 つのコントロールとして動作するコントロールのセットに使用されます。コントロール数が多すぎて、Tab キーだけでは効率的に移動できない場合にも使用できます。[方向]、

[Home]、[End]、[Page Up]、[Page Down]キー を使うと、グループ内のコントロール間で入力フォーカスを移動できます (これらのキーを使用してコントロール グループの外に移動することはできません)。

最初のキーボード フォーカスは、アプリの起動時にユーザーが最初に直感的に操作する (または、その可能性が最も高い) 要素に設定する必要があります。多くの場合、これはアプリのメイン コンテンツ ビューです。ユーザーはすぐに方向キーを使ってアプリ コンテンツのスクロールを開始できます。

最初のキーボード フォーカスは、ネガティブ (または破滅的) な結果を招く可能性のある要素に設定しないでください。これにより、データの消失またはシステム アクセスの喪失を回避します。

コマンド、コントロール、コンテンツを順序付けし、最も重要なものをタブ オーダーと表示順序 (または視覚的な階層) の両方で最初に提示するようにしてください。ただし実際の表示位置は、親レイアウト コンテナーと、レイアウトに影響する子要素の特定のプロパティに左右されることがあります。特に、グリッド形式または表形式を使用しているレイアウトでは、読み取り順序がタブ オーダーとまったく異なる場合があります。このことは必ずしも問題ではありませんが、タッチ可能な UI とキーボードからアクセス可能な UI の両方についてアプリの機能をテストする必要があります。

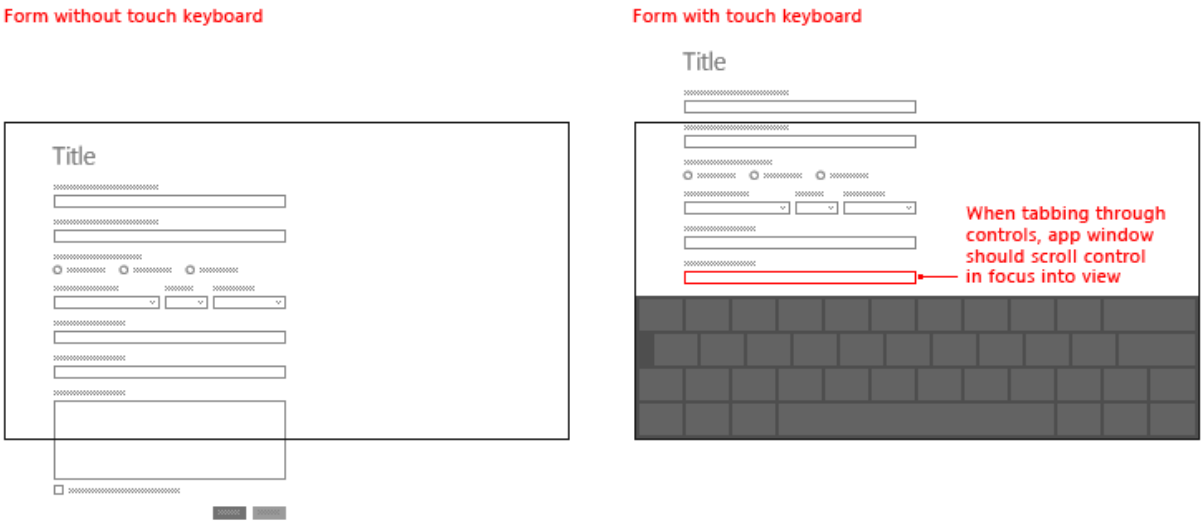
タブ オーダーは、可能な限り読み取り順序に従う必要があります。これにより混乱を軽減できる可能性があります (ロケールと言語に依存します)。

キーボード ボタンを、アプリの適切な UI ("戻る" ボタンと "進む" ボタン) に関連付けてください。

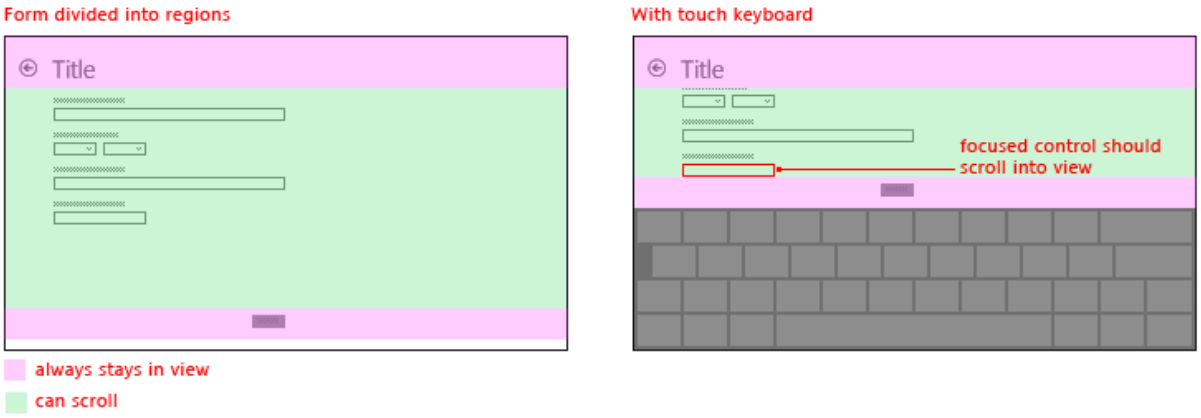
アプリのスタート画面に戻る場合と重要なコンテンツ間を移動する場合に、できるだけ単純かつ容易に移動できるようにします。

コンポジット要素の子要素間で正しい内部ナビゲーションを実行するために、キーボード ショートカットとして [方向] キーを使います。ツリー ビュー ノードに、展開折りたたみとノードのアクティブ化を処理するための別の子要素がある場合は、左右の [方向] キーを使って、キーボードの展開折りたたみ機能を提供します。この動作は、プラットフォーム コントロールと一致します。

タッチ キーボードによって画面の大部分が見えなくなるため、ユニバーサル Windows プラットフォーム (UWP) では、ユーザーがフォームのコントロール間を移動するときに、フォーカスのある入力フィールドが見えるようにスクロールします。これには、現在ビューに表示されていないコントロールも含まれます。カスタム コントロールでも、この動作をエミュレートする必要があります。



場合によって、画面にずっと表示されたままであることが必要な UI 要素もあります。フォーム コントロールがパン領域に含まれ、重要な UI 要素が静的であるように UI を設計します。次に例を示します。



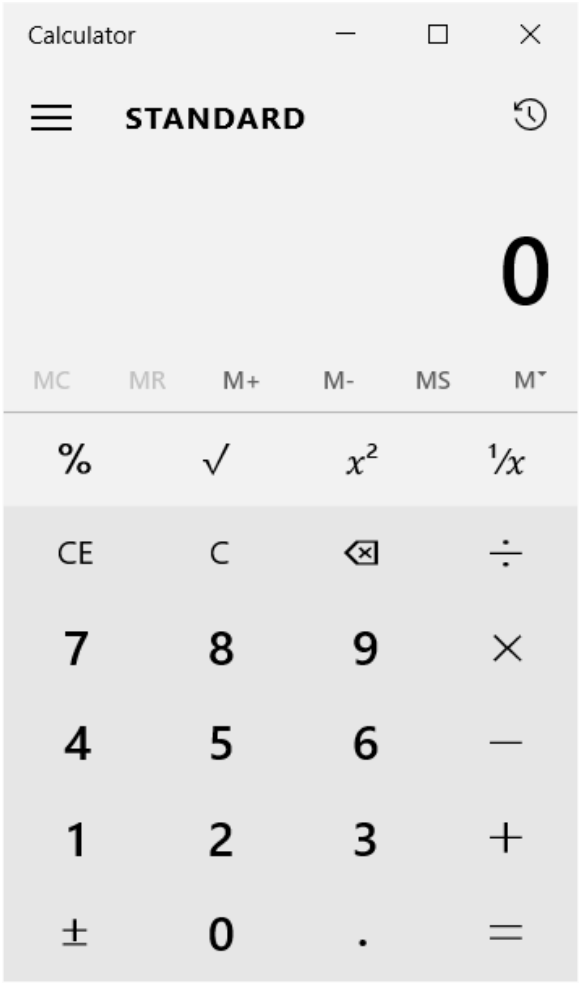
アクティベーション

コントロールは、現在フォーカスがあるかどうかにかかわらず、さまざまな方法でアクティブ化できます。

[Sapce]、[Enter]、[Esc] キー

[Space] キーは、入力フォーカスのあるコントロールをアクティブ化します。[Enter] キーは、既定のコントロールや入力フォーカスのあるコントロールをアクティブ化します。既定のコントロールとは、初期フォーカスのあるコントロールまたは [Enter] キーにのみ応答するコントロール (通常は、入力フォーカスと共に変化します) を指します。[Esc] キーは、メニューやダイアログなどの一時的な UI を終了します。

次に示す電卓アプリでは、[Space] キーを使用してフォーカスのあるボタンをアクティブ化し、[Enter] キーを "=" ボタンにロックして、[Esc] キーを "C" ボタンにロックします。



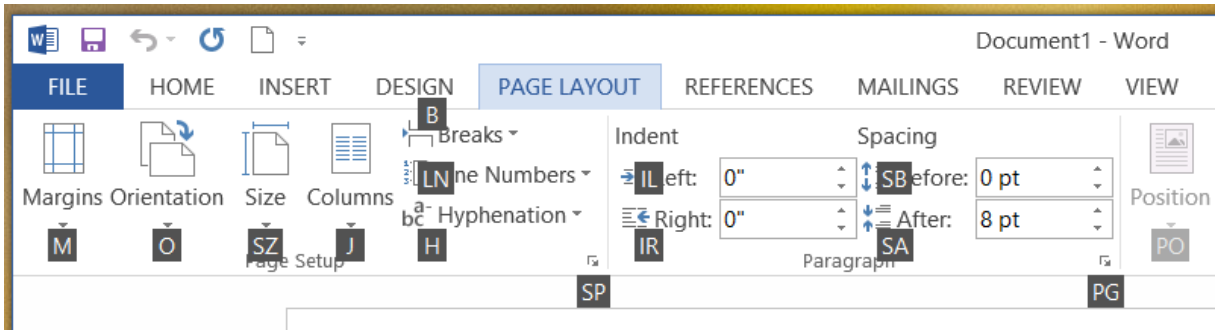
アクセス キーとショートカット キー

アプリの主な機能にキーボード ショートカットを実装します (ショートカットは、ユーザーが効率的にアプリの機能にアクセスできるようにして、生産性を向上させるためのキーの組み合わせです)。

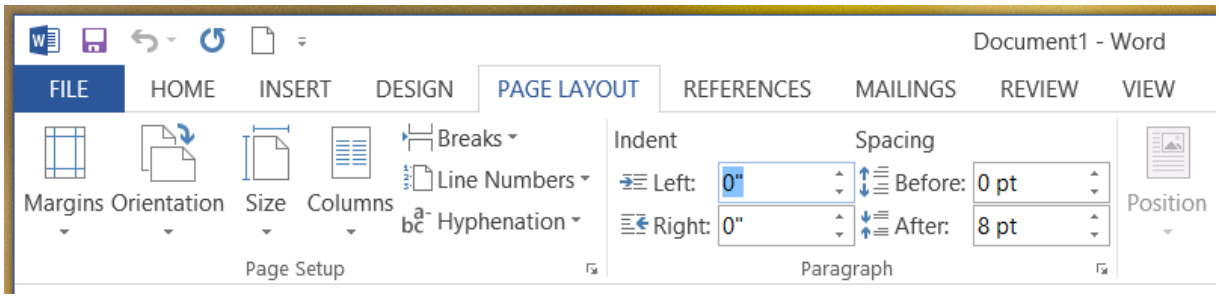
[Tab] キーによるナビゲーションの代わりに、コンロールの直接操作をサポートするには、アクセス キーとショートカット キー (キーボードのパターンに関するセクションで説明します) を通じてキーボード ショートカットを提供します。

注　コントロールには、固有ラベルがあるもの (コマンド ボタン、チェック ボックス、ラジオ ボタンなど) と、外部ラベルのあるもの (リスト ビューなど) があります。外部ラベルのあるコントロールの場合、アクセス キーはラベルに割り当てられ、呼び出されると、対応コントロール内の要素または値にフォーカスを設定します。

次の例では、**Word** の [**ページ レイアウト**] タブのアクセス キーを示しています。



ここで、[インデント] の [Left] ボックスに対応するラベルに示されたアクセス キーを入力すると、このテキスト ボックスの値が強調表示されます。



ユーザービリティとアクセシビリティ

適切に設計されたキーボードの操作エクスペリエンスは、ソフトウェア アクセシビリティの重要な要素であり、視覚に障碍のあるユーザーや特定の運動障碍のあるユーザーによるアプリ内の移動や、その機能の操作を実現します。このようなユーザーにはマウス操作が困難な場合があり、キーボード強化ツールやスクリーン キーボードなどさまざまな支援技術をスクリーン拡大機能、スクリーン リーダー、音声入力ユーティリティなどと組み合わせて使用することが必要になります。このようなユーザーにとっては、一貫性より包括性の方が重要です。

多くの経験豊富なユーザーには、キーボードの使用の方がはるかに好まれます。キーボードベースのコマンドであれば、すばやく入力することができ、キーボードから手を離す必要がないためです。このようなユーザーにとっては、効率性と一貫性が重要です。包括性が重要になるのは、特に頻繁に使用するコマンドに対してのみです。

ユーザビリティとアクセシビリティの設計には、わずかな相違があります。キーボード アクセスにおいて 2 つの異なるメカニズムがサポートされているのはこのためです。

アクセス キーには、次の特徴の特徴があります。

- アクセス キーは、アプリ内の UI 要素へのショートカットです。
- [Alt] キーを押しながら [英数字] キーを押します。
- 主な目的はアクセシビリティです。
- すべてのメニューと大部分のダイアログ ボックス コントロールに割り当てます。
- アクセス キーは覚えて使うことを意図していないため、UI に直接示されています (コントロールのラベルに含まれる対応する文字に下線)。
- アクセス キーは、対応するメニュー項目やコントロールへの移動に使用します。効力は現在のウィンドウ内に限られます。
- 常に一貫して割り当てることはできないため、割り当てに一貫性はありません。ただし、使用頻度の高いコマンド (特にコミット ボタン) については、アクセス キーの割り当てに一貫性が必要です。
- アクセス キーはローカライズされます。

アクセス キーは覚えて使うことを意図していないため、キーワードがラベルの後半にある場合も、見つけやすいようにラベルの最初の方に出現する文字に割り当てます。

これに対し、ショートカット キーには次の特徴があります。

- ショートカット キーは、アプリ コマンドへのショートカットです。
- 主に [Ctrl] キーとファンクション キーのシーケンスを使用します (Windows のシステム ショートカット キーには、[Alt] キーと英数字以外のキーの組み合わせと [Windows] ロゴ キーが使用されています)。
- ショートカット キーの主な目的は、上級ユーザーによる使用の効率です。
- 特に使用頻度の高いコマンドにのみ割り当てます。
- 覚えて使うことを意図しており、メニュー、ツールヒント、ヘルプ内にのみ示されています。
- 効力はプログラム全体に及びますが、該当しない場合には効力がありません。
- 覚えて使うことを意図しており、直接示されないため、割り当てに一貫性が必要です。
- ローカライズされません。

ショートカット キーは覚えて使うことを意図しているため、特に使用頻度の高いショートカット キーには、コマンド キーワードに含まれる先頭の文字または最も記憶しやすい文字を使用するのが理想的です。たとえば、コピー (Copy) に [Ctrl] + [C] を、要求 (Request) に [Ctrl] + [Q] を割り当てます。

ユーザーが、アプリでサポートされているすべてのタスクをハードウェア キーボードまたはスクリーン キーボードだけで実行できるようにしてください。

スクリーン リーダーやその他の支援技術を使うユーザーがアプリのショートカット キーを簡単に見つけられるようにする必要があります。ヒント、アクセシビリティ対応の名前、アクセシビリティ対応の説明、またはその他の画面上の伝達形式を使ってショートカット キーが確認できるようにします。少なくとも、アプリのヘルプ コンテンツにはアクセス キーとショートカット キーについて十分な説明を用意しておく必要があります。

よく知られているショートカット キーや標準のショートカット キーを他の機能に割り当てないでください。たとえば、[Ctrl] + [F] は通常、検索に使用されます。

密度の高い UI に含まれるすべての対話型コントロールにアクセス キーを割り当てる必要はありません。ただし、特に重要なコントロールと特に使用頻度の高いコントロールにアクセ

ス キーを必ず割り当てるか、コントロール グループを作成して、そのコントロール グループのラベルにアクセス キーを割り当ててください。

キーボードの修飾キーを使うコマンドは変更しないでください。変更すると、検出できなくなり、混乱を招きます。

入力フォーカスがあるコントロールを無効にしないでください。キーボード入力の妨げになる可能性があります。

キーボード操作のエクスペリエンスを確実なものにするには、キーボードのみを使ってアプリを徹底的にテストすることが重要です。

テキスト入力

キーボード入力に依存している場合は、常にデバイスの機能を確認するようにします。一部のデバイス (電話など) では、ハードウェア キーボードにあるようなアクセラレータやコマンド キー ([Alt] キー、[Function] キー、[Windows] ロゴ キーなど) の多くがタッチ キーボードに備わっていないため、タッチ キーボードの用途がテキスト入力に限定されます。

ユーザーがタッチ キーボードを使ってアプリ内を移動しないようにしてください。フォーカスを取得するコントロールによっては、タッチ キーボードが閉じられることがあります。

フォームに対する操作が行われている間は終始キーボードを表示しておくようにしてください。これにより、フォームまたはテキスト入力フローの途中で UI 表示が切り替わることでユーザーを混乱させるような状況を回避します。

入力しているフィールドをユーザーが常に見られるようにします。タッチ キーボードによって画面の半分が見えなくなるため、ユーザーがフォーム内を移動するときに、フォーカスのある入力フィールドが見えるようにスクロールします。

パターン

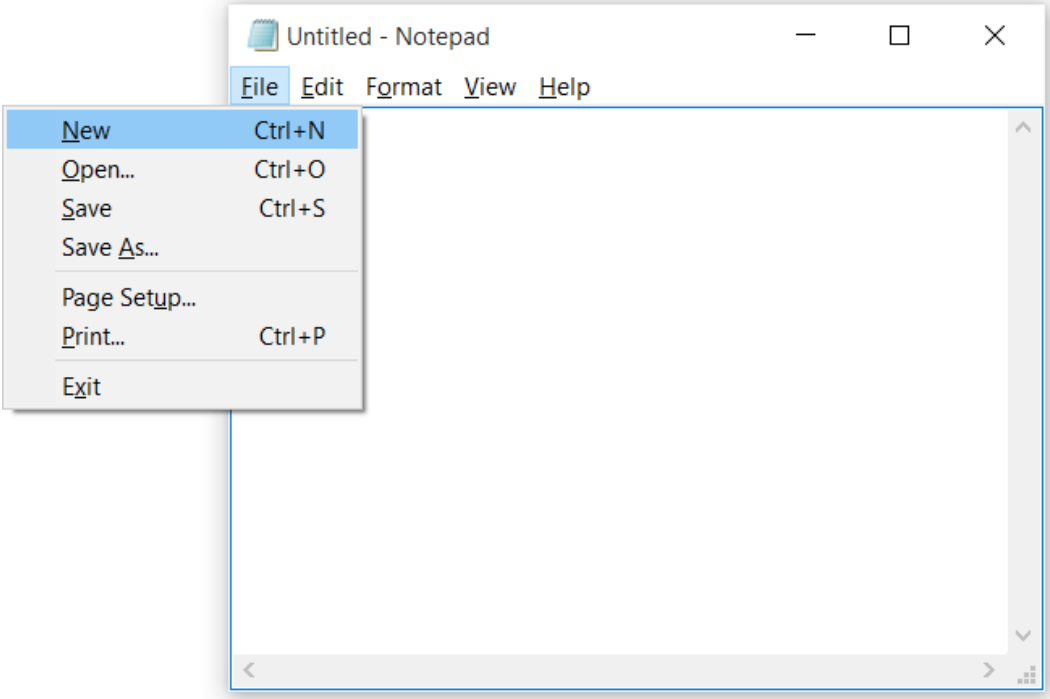
標準的なハードウェア キーボード (OSK) は、それぞれ異なる機能をサポートする 7 種類のキーで構成されています。各サポート固有の機能で構成されます。

- 文字キー：表示どおりの文字を入力フォーカスのあるウィンドウに送ります。
- 修飾キー：[Ctrl] キー、 [Alt] キー、[Shift] キー、 [Windows] ロゴ キーなど。プライマリ キーと組み合わせて使用し、そのキーの意味を変えます。
- ナビゲーション キー：[Tab] キー、 [Home] キー、[End] キー、[Page Up] キー、[Page Down] キー、 [方向] キーなど。入力フォーカス (テキスト入力の場所) を移動します。
- 編集キー：[Shift] キー、[Tab] キー、[Enter] キー、[Ins] キー、[BackSpace] キー、[Del] キーなど。テキストを操作します。
- ファンクション キー：[F1] から [F12] までのキー。特殊な機能の実行に使います。
- 切り替えキー：[CapsLock] キー、[ScrollLock] キー、[NumLock] キーなど。システムを特定のモードに切り替えます。
- コマンド キー：[Space] キー、[Enter] キー、[Esc] キー、[Pause/Break] キー、[Print Screen] キーなど。システム タスクまたはコマンドのアクティブ化を実行します。

これらのカテゴリに加え、セカンダリ クラスのキー (およびキーの組み合わせ) が存在します。これらは、アプリ機能のショートカットとして使用できます。

- アクセス キー：[Alt] キーと文字キーを押すことでコントロールまたはメニュー項目を操作します。アクセス キーに使用する文字キーは、メニュー内では下線のある文字として示されます。オーバーレイに表示することもできます。
- ショートカット キー：[Function] キーを押すか、[Ctrl] キーと文字キーを押すことでアプリ コマンドを操作します。アプリにはコマンドに対応する UI がある場合とない場合があります。

次の図では、メモ帳アプリで展開された [ファイル] メニューに、アクセス キーとショートカット キーの両方が含まれています。



キーボード コマンド

キーボード入力をサポートするさまざまなデバイスで提供されるキーボード操作の一覧を次に示します。一部のデバイスやプラットフォームではキー入力や操作が異なる場合がありますが、これらも記載されています。

カスタムのコントロールや操作を設計する場合は、このキーボード言語を統一して使用してください。これにより、アプリの信頼性と学びやすさが向上し、なじみのある使用感が実現します。

既定のキーボード ショートカット定義を変更しないでください。

次の表に、よく使われるキーボード コマンドを示します。キーボード コマンドの一覧については、Windows のキーボード ショートカット キーに関するページをご覧ください。

ナビゲーション コマンド

操作	キー コマンド
戻る	Alt + ←、または特殊キーボードの戻るボタン
進む	Alt + →
上へ	Alt + ↑
キャンセルまたは現在のモードの終了	Esc
一覧の項目間の移動	方向キー (←、→、↑、↓)
次の項目一覧へジャンプ	Ctrl + ←
セマンティック ズーム	Ctrl + 正符号 (+) または Ctrl + マイナス記号 (-)
コレクション内の名前付き項目にジャンプ	項目の名前を入力します
次のページ	[PageUp]、[PageDown]、または [Space]
次のタブ	[Ctrl] + [Tab]
前のタブ	[Ctrl] + [Shift] + [Tab]
アプリ バーを開く	[Windows] + [Z]
アクティブ化、または項目を開く	[Enter]
選択	[Space]
連続して選択	[Shift] + [方向キー]
すべてを選択	[Ctrl] + [A]

一般的な」コマンド

操作	キーコマンド
項目をピン留めする	[Ctrl] + [Shift] + 数字 1
保存	[Ctrl] + [S]
検索	[Ctrl] + [F]
印刷	[Ctrl] + [P]
コピー	[Ctrl] + [C]
切り取り	[Ctrl] + [X]
新規項目	[Ctrl] + [N]
貼り付け	[Ctrl] + [V]
開く	[Ctrl] + [O]
アドレスを開く (たとえば、Internet Explorer で URL を開く)	[Ctrl]+[L] または [Alt]+[D]

メディア ナビゲーション コマンド

操作	キーコマンド
再生/一時停止	[Ctrl] + [P]
次の項目	[Ctrl] + [F]
前の項目	[Ctrl] + [B]

注　メディア ナビゲーションの "再生または一時停止" のキー コマンドは "印刷" のキー コマンドと同じであり、"次の項目" のキー コマンドは "検索" のキー コマンドと同じです。メディア ナビゲーションのコマンドよりも、一般的なコマンドの方が優先されます。たとえば、再生メディアと印刷の両方をサポートするアプリの場合、[Ctrl] + [P] キーを押すと印刷が実行されます。

ビジュアルなフィードバック

キーボード操作でのみフォーカス用の四角形を使います。ユーザーがタッチ操作を始めたら、キーボードの UI を徐々にフェード アウトします。 これにより、UI の簡潔さが保たれます。

静的テキストなど、要素で対話操作がサポートされていない場合は、ビジュアルなフィードバックを返さないでください。これにより、UI の簡潔さが保たれます。

同じ入力対象を表すすべての要素に対してビジュアルなフィードバックを同時に表示します。

パン、回転、ズームなど、タッチ ベースの操作をエミュレートするためのツールチップとして画面ボタン (+、- など) を提供するようにしてください。

ビジュアル なフィードバックのガイダンスについては、「ビジュアルなフィードバックのガイドライン」を参照してください。

マウスのガイドライン

ユニバーサル Windows プラットフォーム (UWP) アプリの設計はタッチ入力用に最適化し、既定の基本的なマウスのサポートを利用します。

ユーザーがマウスで操作できる UWP ストア アプリを設計し、構築します。

マウス入力は、ポイントとクリックの正確さが求められるユーザー操作に最適です。この固有の正確さは、Windows の UI でも当然サポートされていますが、タッチの本来の不正確さに合わせて最適化されています。

マウス入力とタッチ入力が異なるのは、タッチでは画面上の UI 要素に対する物理的なジェスチャ —(スワイプ、スライド、ドラッグ、回転など) を通じて、それらのオブジェクトへの直接の操作をエミュレートする機能があることです。マウスによる操作では、オブジェクトのサイズ変更や回転を実行するためにハンドルを使用するなど、他の UI アフォーダンスが必要になることが普通です。

このトピックでは、マウス操作の設計時の考慮事項について説明します。

UWP アプリのマウスの言語

システム内で一貫して、マウス操作の簡単なセットが使われます。

用語	説明
ホバーによる説明の表示	要素にホバーすると、詳しい情報や説明を伝える視覚効果 (ツールチップなど) が表示されます。操作はコミットされません。
左クリックによるプライマリ操作	要素の左クリックにより、プライマリ操作 (アプリの起動、コマンドの実行など) が呼び出されます。

スクロールによるビューの変更	スクロール バーを表示し、コンテンツ領域内で上下左右に移動します。スクロール バーのクリック、またはマウス ホイールの回転により、スクロールできます。スクロール バーは、コンテンツ領域内の現在のビューの位置を示します (タッチによるパンでも同様の UI が表示されます)。
右クリックによる選択とコマンド	右クリックして、ナビゲーション バー (使用できる場合) と、グローバル コマンドを含むアプリ バーを表示します。要素を右クリックして選択し、その要素に対応する状況依存のコマンドを備えたアプリ バーを表示します。 **注** 選択またはアプリ バーのコマンドが適切な UI 動作ではない場合は、右クリックによりコンテキスト メニューが表示されます。ただし、すべてのコマンド動作にアプリ バーを使うことを強くお勧めします。
ズームの UI コマンド	アプリ バーに UI コマンドを表示するか (+、- など)、[Ctrl] キーを押しながらマウス ホイールを回転させて、ズームのためのピンチ ジェスチャーとストレッチ ジェスチャーをエミュレートします。
回転の UI コマンド	アプリ バーに UI コマンドを表示するか、[Ctrl] キーと [Shift] キーを押しながらマウス ホイールを回転させて、回転のための回転ジェスチャーをエミュレートします。画面全体を回転させるには、デバイスを回転させます。
左クリックとドラッグによる移動	要素を左クリックしてドラッグし、移動します。
左クリックとドラッグによるテキストの選択	選択可能なテキスト内を左クリックしてドラッグし、選択します。単語を選択するには、ダブルクリックします。

ビジュアルなフィードバックのガイドライン

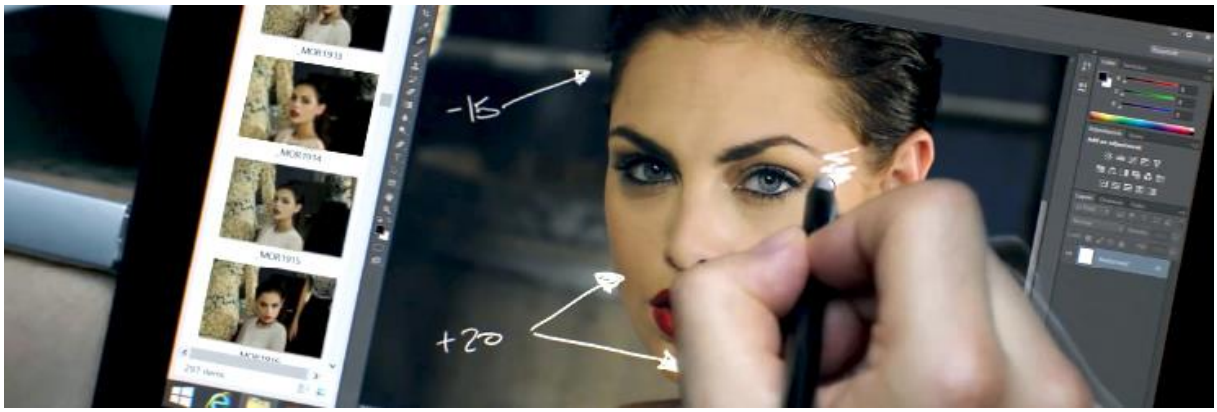
- 移動イベントまたはホバー イベントを通じてマウスが検出されたら、マウス固有の UI を表示して、要素によって公開されている機能を示します。マウスが一定の期間動かされなかった場合や、ユーザーがタッチ操作を始めた場合は、マウス UI を徐々に非表示にします。 これにより、UI の簡潔さが保たれます。

- ホバーのフィードバックにカーソルを使わないでください。要素によるフィードバックで十分です (以下の「カーソル」をご覧ください)。
- 静的テキストなど、要素で対話操作がサポートされていない場合は、ビジュアルなフィードバックを返さないでください。
- マウス操作ではフォーカス用の四角形を使わないでください。これはキーボード操作専用です。
- 同じ入力対象を表すすべての要素に対してビジュアルなフィードバックを同時に表示します。
- パン、回転、ズームなど、タッチ ベースの操作をエミュレートするためのボタンを提供します (+、- など)。

ペンのガイドライン

ユニバーサル Windows プラットフォーム (UWP) ストア アプリの設計はタッチ入力用に最適化し、既定の基本的なペンのサポートを利用します。

UWP アプリのインク プラットフォームでペン デバイスを使うと、自然な形で手書きノート、描画、コメントを作ることができます。このプラットフォームは、デジタイザー入力からのインク データのキャプチャ、インク データの生成、出力デバイスへのひと筆としてのデータのレンダリング、インク データの管理、手書き認識の実行をサポートします。



このトピックでは、ペン操作の設計時の考慮事項について説明します。ペン操作の実装について詳しくは、「ペン操作とスタイラス操作への反応」をご覧ください。

ペン操作

ペンを使うことで、ユーザーが手書きノート、描画、コメントの書き込み操作を行うことができる Windows ストア アプリを設計できます。

ペンは、ポインティング デバイスとして使うことができます。より興味深い用途は、ペンをデジタル インクに関連付けられた描画デバイスとして使うことです。 Windows のインク プラットフォームでペン デバイスを使うと、自然な形で手書きノート、描画、コメントを作れます。

ユーザーが書いたり描画したりするときのペンの空間移動のキャプチャに加えて、アプリで筆圧、形状、色、不透明度などの情報を収集して、紙の上でペン、鉛筆、ブラシを使っているときに近いユーザー エクスペリエンスを実現することもできます。

ペン入力が持つ固有の正確さは、UI でも当然サポートされていますが、タッチの本来の不正確さに合わせて最適化されています。

ビジュアル なフィードバックのガイダンスについては、「ビジュアルなフィードバックのガイドライン」を参照してください。

音声認識のガイドライン

音声認識や音声合成 (TTS: text-to-speech) をアプリのユーザー エクスペリエンスに直接統合します。

音声認識: ユーザーが発声した単語を、フォーム入力やテキストのディクテーション用にテキストに変換し、アクションやコマンドを指定したり、タスクを実行したります。この機能は、フリーテキストのディクテーションと Web 検索向けの定義済みの文法、および Speech Recognition Grammar Specification (SRGS) Version 1.0 を使って作成されたカスタム文法をサポートしています。

音声合成 (TTS): 音声合成エンジン (声) を使って、テキスト文字列を音声に変換します。入力文字列は、基本的でシンプルなテキスト、またはより複雑な Speech Synthesis Markup Language (SSML) のいずれかになります。 SSML は、発音、音量、ピッチ、速度、強調など、音声出力の特性を制御する標準的な方法です。

注 **Cortana** とカスタマイズした音声コマンドを使うと、アプリをフォアグラウンドで起動したり ([スタート] メニューから起動した場合と同様にアプリがフォーカスを取得します)、バック グラウンド サービスとしてアクティブ化したりすることができます (**Cortana** がフォーカスを維持しますが、アプリからの結果を表示します)。追加のコンテキストやユーザー入力 (特定の連絡先へのメッセージの送信など) が必要なコマンドはフォアグラウンド アプリで処理するのが最適ですが、基本的なコマンドはバックグラウンド アプリを介して **Cortana** で処理できます。

Cortana UI の音声コマンドを使って、機能をバック グラウンド サービスとして公開する場合は、「**Cortana** の設計ガイドライン」をご覧ください。

音声機能が適切に設計され、実装されていると、ユーザーがアプリを楽しく確実に操作できる手段になります。音声機能によって、キーボード、マウス、タッチ、ジェスチャを補完することも、場合によってはこれらの代替として使うこともできます。

音声操作の設計

次のガイドラインと推奨事項では、最適な形で音声認識と TTS をアプリの操作エクスペリエンスに統合する方法について説明します。

アプリで音声操作のサポートを検討している場合:

- 音声認識では、どのような操作を実行できるか。ユーザーは、ページ間の移動やコマンドの呼び出しを実行できるか。また、データをテキスト フィールド、簡単なメモ、長いメッセージとして入力できるか。
- 音声入力はタスクの実行に適した機能であるか。
- 音声入力が利用可能になっていることをユーザーはどのように認識するか。
- アプリが常に聞き取りを行っているか、またはユーザーが操作を実行してアプリを聞き取りモードに切り替える必要があるか。
- どのような語句が操作や動作を開始するか。語句と操作を画面に列挙する必要があるか。
- プロンプト画面、確認画面、不明瞭解消画面、TTS が必要か。
- アプリとユーザー間の対話ダイアログは何か。
- アプリのコンテキストに対してカスタムのボキャブラリや制約のあるボキャブラリが必要か (医学、科学、ロケールなど)。
- ネットワーク接続が必要か。

テキスト入力

テキスト入力用の音声は、短い形式 (1 つの単語または語句) から長い形式 (継続的なディクテーション) までさまざまなものがあります。短い形式の入力は 10 秒未満の長さにする必要があり、長い形式の入力セッションは最大で 2 分の長さにすることができます (長い形式の入力はユーザーが操作しなくても再開でき、継続的なディクテーションのような印象を与えることができます)。

音声認識がサポートされていて、利用可能になっていること、およびユーザーが音声認識を有効にする必要があるかどうかを示すために、視覚的な合図を使うことをお勧めします。たとえば、マイクのグリフが表示されるコマンド バー ボタン (「コマンド バーのガイドライン」をご覧ください) を使って、音声認識が利用可能になっていることやその状態を示すことができます。

実行中の音声認識に対するフィードバックを提供することで、音声認識を実行しているときに、不十分な応答性を最小限に抑えることができます。

キーボード入力、不明瞭解消のプロンプト、修正候補、追加の音声認識を使って、ユーザーが認識テキストを変更できるようにします。

入力が音声認識以外のデバイス (タッチ入力やキーボードなど) から検出された場合は、認識を停止します。このような入力は、ユーザーが他のタスク (認識テキストの修正や他のフォーム フィールドの操作など) に移行したことを示している可能性があります。

音声入力がないときに認識を終了までの時間を指定します。この時間が経過した後で音声認識を自動的に再開しないでください。これは、ユーザーがアプリの操作を停止したことを示している場合が多いためです。

ネットワーク接続が利用できない場合は、すべての継続的な認識 UI を無効にし、認識セッションを停止します。継続的な認識では、ネットワーク接続が必要です。

コマンドの実行

音声入力によって、操作の開始、コマンドの呼び出し、タスクの実行を行うことができます。

画面に余裕がある場合は、現在のアプリのコンテキストでサポートされている応答を、有効な入力の例と一緒に表示することを検討してください。これにより、場合によっては、アプリで処理する必要がある応答を減らすことできます。また、ユーザーが入力に迷うことが少なくなる場合もあります。

できる限り具体的な応答を引き出せるように、質問を構成します。たとえば、""今日は何をしたいですか"" は何とおりもの応答ができる質問であり、さまざまな応答の可能性があるため膨大な量の文法定義が必要になります。これに対して、""ゲームをプレイするか、音楽を聴くか、どちらにしますか"" という質問では、2 つの有効な回答のいずれか 1 つに応答が制限されます。このため、文法定義の量も少なくなります。文法定義の量が少ないと作成が容易になり、認識結果の精度が向上します。

音声認識の信頼性が低い場合には、ユーザーに確認を求めます。ユーザーの意図が明らかでない場合は、ユーザーが意図していない操作を開始するよりも、ユーザーの意図を明確にする方が、良い結果につながります。

音声認識がサポートされていて、利用可能になっていること、およびユーザーが音声認識を有効にする必要があるかどうかを示すために、視覚的な合図を使うことをお勧めします。たとえば、マイクのグリフが表示されるコマンド バー ボタン (「コマンド バーのガイドライン」をご覧ください) を使って、音声認識が利用可能になっていることやその状態を示すことができます。

通常、音声認識のスイッチが画面から消える場合は、アプリのコンテンツ領域に状態インジケーターを表示することを検討してください。

音声認識がユーザーによって開始される場合は、一貫性を維持するために組み込みの認識エクスペリエンスを使うことを検討してください。組み込みのエクスペリエンスには、プロンプト、例、不明瞭解消、確認、エラーが表示されるカスタマイズ可能な画面が含まれています。

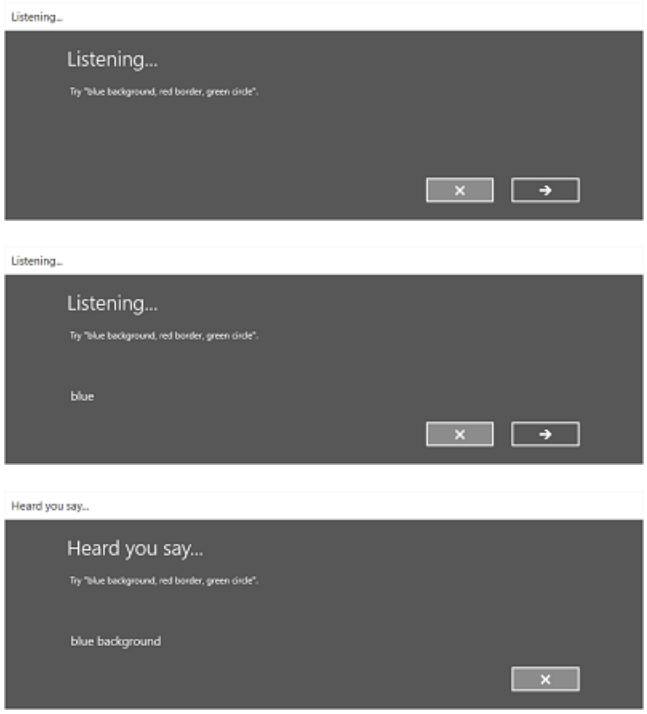
画面は、指定した制約によって異なります。

- 定義済みの文法の場合 (ディクテーションまたは Web 検索)
    - [**Listening**] 画面。
    - [**Thinking**] 画面。
    - [**Heard you say**] 画面またはエラー画面。
- 単語や語句の一覧または SGRS 文法ファイルの場合
    - [**Listening**] 画面。
    - [**Did you say**] 画面 (ユーザーの発言に複数の解釈が可能な場合)。
    - [**Heard you say**] 画面またはエラー画面。

[**Listening**] 画面では、次の操作を実行できます。

- 見出しテキストのカスタマイズ。
- ユーザーが声に出すことができる内容のサンプル テキストの指定。
- [**Heard you say**] 画面が表示されるかどうかの指定。
- 認識された文字列を [**Heard you say**] 画面でユーザーに対して読み返す。

SRGS で定義された制約を使う音声認識エンジンにおける組み込みの認識フローの例を、次に示します。この例では、音声認識に成功しています。



常に聞き取る

ユーザーが操作しなくても、アプリを起動してすぐに、音声入力の聞き取りと認識を実行できます。

アプリのコンテキストに基づいて、文法の制約をカスタマイズしてください。これにより、音声認識エクスペリエンスがターゲットとして維持され、現在のタスクとの関連が継続されます。また、エラーを最小限に抑えることができます。

"音声操作の項目"

音声入力が有効になっているとき、どのような単語と語句が正確に理解されるか、またどのような操作を実行できるかを、ユーザーが検出できるようにすることが重要です。

音声認識がユーザーよって有効化される場合は、コマンド バーやメニュー コマンドを使って、現在のコンテキストでサポートされているすべての単語と語句を表示することを検討してください。

音声認識が常に有効になっている場合は、すべてのページに ""音声操作の項目"" という語句を追加することを検討してください。ユーザーがこの語句を発声すると、現在のコンテキ

ストでサポートされているすべての単語と語句が表示されます。このフレーズを使うと、ユーザーは一貫した方法でシステムに実装されている音声機能を検出することができます。

認識の失敗

音声認識は失敗する場合があります。オーディオ品質が低い場合、語句の一部しか認識できない場合、または入力がまったく検出されない場合は、音声認識が失敗する可能性があります。

失敗を適切に処理することで、ユーザーは認識が失敗した理由を理解し、回復することが可能になります。

音声入力が理解されなかった点と再試行する必要がある点を、アプリでユーザーに伝えてください。

サポートされている 1 つ以上の語句を例として提示することを検討します。ユーザーは提示された語句を再度入力する可能性があります。これにより、認識が成功する確率が高くなります。

一致する可能性がある語句の一覧を表示し、ユーザーが選ぶことができるようにすることをお勧めします。こうした対処方法は、認識プロセスを再度やり直すよりも効率的です。

別の種類の入力方法を常にサポートしてください。認識の失敗が繰り返し発生した場合、その失敗を処理する際に特に役立ちます。たとえば、キーボード、タッチ入力、またはマウスを使って、一致する可能性がある語句の一覧から選ぶように提案します。

組み込みの音声認識エクスペリエンスを使います。この音声認識エクスペリエンスには、音声認識に失敗したことをユーザーに通知し、ユーザーが音声認識をもう一度試行できる画面が含まれています。

オーディオ入力の問題を検出し、修正を試みます。音声認識エンジンでは、音声認識の正確さにマイナスの影響を与える可能性があるオーディオ品質の問題を検出できます。音声認識エンジンから提供される情報を使うと、問題をユーザーに通知し、可能であれば対処するように指示することができます。たとえば、マイクの音量設定が低すぎる場合は、もっと大きな声で話すことやボリュームを上げることをユーザーに求めることができます。

制約

制約 (文法) は、発声される単語や語句を定義します。これらの単語や語句は、音声認識エンジンによって照合することができる単語や語句です。定義済みの Web サービスの文法のいずれかを使ったり、アプリと共にインストールされるカスタムの文法を作成したりすることができます。

定義済みの文法

定義済みのディクテーション文法と Web 検索文法を使うと、文法を作らずにアプリに音声認識を実装できます。これらの文法を使った場合、音声認識がリモート Web サービスで実行され、結果がデバイスに返されます。

- 既定のフリーテキストのディクテーション文法では、ユーザーが特定の言語で話すほとんどの単語と語句を認識できます。これは短い語句の認識に最適化されています。フリーテキストのディクテーションは、ユーザーが話す内容を限定しない場合に便利です。一般的な用途としては、メモの作成やメッセージ内容の口述などがあります。
- Web 検索文法は、ユーザーが話す可能性のある多数の単語と語句を含んでいる点でディクテーション文法と似ていますただし、ユーザーが Web 検索で一般的に使う用語の認識に最適化されています。

注 定義済みのディクテーション文法と Web 検索文法は容量が大きく、(デバイス上ではなく) オンライン上に存在するため、カスタム文法をデバイスにインストールした場合に比べるとパフォーマンスが劣る可能性があります。

このような定義済みの文法は、10 秒までの長さの音声入力を認識でき、開発者による作成作業は必要ありません。ただし、ネットワークへの接続が必要になります。

カスタム文法

カスタム文法はお客様が設計および作成を行い、アプリと共にインストールされます。カスタム制約を使う音声認識は、デバイスで実行されます。

- プログラムによる一覧の制約は、単語や語句の一覧を使って単純な文法を作成する手法で、軽量です。個別の短い語句を認識するには、一覧の制約が適しています。文法にすべての単語を明示的に指定すると、音声認識エンジンは音声と単語の一致を確認する際に音声だけを処理すればよいので、認識の精度が向上します。また、一覧はプログラムで更新することもできます。
- SRGS 文法は静的ドキュメントで、プログラムによる一覧の制約とは異なり、SRGS Version 1.0 で定義された XML 形式を使います。SRGS 文法では、1 回の認識で複数の意味をキャプチャすることができるため、音声認識エクスペリエンスを最大限に制御することができます。

  SRGS 文法を作成するためのヒントを次にいくつか紹介します。
  - 各文法の規模を小さくします。文法に含める語句を少なくする方が、規模の大きな文法に多数の語句が含まれている場合よりも、認識精度が高くなる傾向があります。アプリ全体に対して 1 つの文法を設定するよりも、特定のシナリオごとに別々の小規模な文法を設定することをお勧めします。
  - ユーザーには、各アプリのコンテキストに基づいて何と話しかければよいかを知らせ、必要に応じて文法を無効にします。
  - 文法は、ユーザーがさまざまな形でコマンドを音声入力できるように設計します。たとえば、GARBAGE 規則を使って、文法で定義されていない音声入力を照合することができます。これにより、ユーザーはアプリにとって意味を持たない語句を含めて話すことができます。たとえば、""お願い""、""それと""、""ええと""、""多分"" などの語句を含めることができます。
  - 音声入力の認識率を高めるには、sapi:subset 要素を使います。この要素は、部分的な語句の照合をサポートするための、SRGS 仕様に対する Microsoft の拡張機能です。
  - 音節が 1 つしかない語句は、文法に定義しないようにしてください。音節が 2 つ以上ある語句の方が、正確に認識されやすくなります。
  - 同じように聞こえる語句を使わないようにしてください。たとえば、"hello"、"bellow"、"fellow" などの語句を使うと音声認識エンジンが混乱し、認識精度が低くなる可能性があります。

注 どの種類の制約を使うかは、作成する認識エクスペリエンスの複雑さによって決まります。どの種類の制約も特定の認識タスクに最適な選択肢となる可能性があり、アプリですべての種類の制約を使う場合もあります。

カスタムの発音

一般的ではない単語や架空の単語を含む特殊なボキャブラリや、普通とは異なる発音の単語がアプリに含まれる場合は、カスタムの発音を定義することで、認識性能が高まる可能性があります。

単語や語句の一覧が小規模な場合や、あまり使われない単語や語句の一覧の場合、カスタムの発音を SRGS 文法で作成できます。詳しくは、「token 要素」をご覧ください。

単語や語句の一覧が大規模な場合や、頻繁に使われる単語や語句については、発音辞書のドキュメントを別途作成することもできます。詳しくは、辞書と音標文字に関するページをご覧ください。

テスト

音声認識の精度のテストとサポートされている UI のテストは、アプリの対象ユーザーに対して行います。このようなテスト方法は、アプリの音声操作エクスペリエンスの有効性を判断するために最適な方法です。たとえば、アプリが一般的な語句を聞き取らないために、ユーザーが良好な認識結果を得られない可能性があります。

このような場合は、該当する語句をサポートするように文法を変更したり、サポートされている語句の一覧をユーザーに提供したりします。サポートされている語句の一覧が既に提供されている場合は、その一覧を簡単に見つけることができるかどうかを確認してください。

音声合成 (TTS)

TTS では、プレーンテキストまたは SSML から音声出力が生成されます。

ていねいな言葉使いで音声入力を促すようなプロンプトを設計します。

長いテキスト文字列を読み取る必要があるかどうかを検討します。1 つのテキスト メッセージを聞き取ることと、記憶するのが困難な長い検索結果の一覧を聞き取ることは別のことです。

ユーザーが TTS を一時停止または停止できるように、メディア コントロールを用意します。

すべての TTS 文字列を聞き取り、それらの文字列が明瞭かつ自然な話し方になっていることを確認します。

- 単語が不自然な順番で連続している場合や、文字列に含まれる数値や句読点を発声する場合に、語句が不明瞭になる可能性があります。
- 韻律や抑揚がネイティブ スピーカーによる発声と異なると、音声が不自然に聞こえる場合があります。

どちらの問題も、スピーチ シンセサイザーへの入力にプレーンテキストではなく SSML を使うことで対処できます。SSML について詳しくは、SSML による合成音声の制御に関するページと、Speech Synthesis Markup Language (SSML) のリファレンスをご覧ください。

タッチ操作のガイドライン

タッチ用に最適化される一方で、さまざまな入力デバイスで一貫した機能を提供する、直観的で独特なユーザー操作エクスペリエンスを備えたユニバーサル Windows プラットフォーム (UWP) アプリを作成します。

推奨と非推奨

- 期待される主な入力方法としてタッチ操作を使うアプリを設計します。
- あらゆる種類 (タッチ、ペン、スタイラス、マウスなど) の操作に対するビジュアルなフィードバックを提供します。
- タッチ ターゲットのサイズ、接触形状、スクラブ、揺らす操作を調整してターゲット設定を最適化します。
- スナップ位置と方向 "レール" を使って精度を最適化します。
- 密集した UI 項目のタッチの精度を高めるためにヒントとハンドルを用意します。
- できるだけ時間制限のある操作を使わないようにします (適切な使用例: 長押し)。
- できる限り、操作の区別に使われた数の指は使わないようにします。

その他の使い方のガイダンス

まず、タッチがユーザーの主な入力方法になるという想定でアプリを設計します。プラットフォーム コントロールを使う場合は、タッチパッド、マウス、ペン/スタイラスをサポートするために追加のプログラミングを行う必要はありません。Windows 10 では、それらが標準で提供されます。

ただし、タッチ用に最適化された UI が従来の UI よりも常に優れているとは限らないことに留意してください。どちらの UI にも、テクノロジとアプリに固有の長所と短所があります。タッチ操作主体の UI に移行する際に、タッチ (タッチパッドを含む)、ペン/スタイラス、マウス、キーボードの各入力の主な違いを理解することが重要です。Windows 10 のタッチは単に機能をエミュレートするだけではないので、使い慣れた入力デバイスのプロパティと動作に変化がないとは考えないでください。

このガイドラインに目を通すと、タッチ入力では異なる UI 設計方法が必要になるとわかります。

タッチ操作の要件の比較

次の表に、タッチ操作に最適な Windows アプリの設計時に考慮する必要がある、入力デバイス間の違いをいくつか示します。

要因	タッチ操作	マウス、キーボード、ペン/スタイラス操作	タッチパッド
正確性	指先が接触する領域は単一の XY 座標よりも広いので、意図していないコマンドがアクティブ化される可能性が高くなります。	マウスとペン/スタイラスを使うと正確な XY 座標を指定できます。	マウスと同じです。
	接触する領域の形は移動を通じて変化します。	マウスの移動とペン/スタイラスのストロークによって正確な XY 座標を指定できます。キーボード フォーカスは明示的です。	マウスと同じです。
	ターゲット設定に役立つマウス カーソルはありません。	マウス カーソル、ペン/スタイラス カーソル、キーボード フォーカスはすべてターゲット設定に役立ちます。	マウスと同じです。
人体構造	1 本または複数の指で直線移動を行うのは困難なので、指先の動きは正確さに欠けます。これは、手関節が曲がることや動きに関係する関節の数が原因です。	マウスまたはペン/スタイラスを制御する手の物理的な移動距離は、画面上のカーソルの移動距離よりも短いので、マウスまたはペン/スタイラスで直線移動を行う方が簡単です。	マウスと同じです。
	ディスプレイ デバイスのタッチ画面には、指の位置とユーザーのデバイスの持ち方が原因で届きにくくなる領域もあります。	マウスとペン/スタイラスは画面のどの部分にも届き、キーボードではタブ オーダーによってどのコントロールにもアクセスできます。	指の位置と持ち方が問題になることがあります。

	オブジェクトは、1 本以上の指先またはユーザーの手で隠れる場合があります。これをオクルージョンと呼びます。	間接的な入力デバイスでは、オクルージョンは発生しません。	マウスと同じです。
オブジェクトの状態	タッチでは、ディスプレイ デバイスのタッチ画面がタッチされているか (オン) タッチされていないか (オフ) の 2 状態モデルが使われます。追加のビジュアルなフィードバックをトリガーできるホバー状態はありません。	マウス、ペン/スタイラス、キーボードはすべて、離した状態 (オフ)、押した状態 (オン)、ホバー状態 (フォーカス) の 3 状態モデルを公開します。ホバーすると、UI 要素に関連付けられたツールチップを調べて参考にすることができます。ホバーまたはフォーカスを合わせたときの効果によって操作可能なオブジェクトがわかり、ターゲット設定にも役立ちます。	マウスと同じです。
豊富な操作	タッチ画面において複数の入力ポイント (指先) で操作できるマルチタッチをサポートします。	単一の入力ポイントをサポートします。	タッチと同じです。
	タップ、ドラッグ、スライド、ピンチ、回転などのジェスチャーによるオブジェクトの直接操作をサポートします。	マウス、ペン/スタイラス、キーボードは間接的な入力デバイスなので、直接操作はサポートされません。	マウスと同じです。

注　間接的な入力には、25 年以上の改良を経ているという利点があります。ホバーすると表示されるツールチップなどの機能は、タッチパッド、マウス、ペン/スタイラス、キーボード入力での UI の操作を解決するために特別に設計されています。このような UI 機能は、他のデバイスのユーザー エクスペリエンスを損なうことなく、タッチ入力で充実したエクスペリエンスを提供するために再設計されました。

タッチのフィードバックの使用

アプリの対話的操作中にビジュアルなフィードバックが適切に表示されると、その対話的操作がアプリと Windows 10 の両方でどのように解釈されるかに関する認識、学習、適応に役立ちます。ビジュアルなフィードバックの用途は、対話的操作の成功の表示、システム状態の中継、コントロール感の向上、エラーの低減、システムと入力デバイスに関するユーザーの理解の支援、対話的操作の促進などです。

位置に基づく正確性が求められる操作をタッチ入力で行う場合は、ビジュアルなフィードバックが重要です。タッチ入力が検出された場所に必ずフィードバックを表示して、アプリとそのコントロールで定義されたカスタム ターゲット設定規則をユーザーが把握できるようにします。

イマーシブな操作性の実現

次の方法を使って、Windows アプリのイマーシブな操作性を高めます。

ターゲット設定

ターゲット設定は、次の要素によって最適化します。

- タッチ ターゲットのサイズ
  明確なサイズのガイドラインによって、ターゲット設定しやすいオブジェクトとコントロールが含まれる快適な UI を備えたアプリになるようにします。
- 接触形状
  指が接触する領域全体によって、意図された可能性が最も高いターゲット オブジェクトを特定します。
- スクラブ
  グループ内の項目 (ラジオ ボタンなど) 間で指をドラッグすると、それらの項目を簡単にターゲット設定し直すことができます。指を離すと現在の項目がアクティブ化されます。
- 揺らす
  密集した複数の項目 (ハイパーリンクなど) を指で押してスライドせずに前後に揺らすと、それらの項目を簡単にターゲット設定し直すことができます。 オクルージ

ョンが原因で、現在の項目はツールチップまたはステータス バーで特定され、指を離すとアクティブ化されます。

正確性

以下を使って、対話的操作が雑な場合に備えて設計します。

- コンテンツの操作時に目的の位置で簡単に停止できるようにするスナップ位置。
- 手をわずかに曲げて動かした場合でも垂直方向または水平方向のパンを実行できる方向 "レール"。詳しくは、「パンのガイドライン」をご覧ください。

オクルージョン

指と手のオクルージョンは、次の要素によって回避します。

- UI のサイズと配置
  UI 要素を十分に大きくして、指先が接触する領域で完全にふさぐことができないようにします。
  メニューとフライアウトは、できる限り接触する領域の上に配置します。
- ツールチップ
  指がオブジェクトに接触し続けているときは、ツールチップを表示します。オブジェクトの機能について説明する場合に便利です。ツールチップが呼び出されないようにするには、ユーザーは指先をオブジェクトの外にドラッグします。
  小さいオブジェクトの場合は、ツールチップをずらして、指先が接触する領域でふさがれないようにします。これはターゲット設定に役立ちます。
- 正確さを確保するためのハンドル
  正確さが要求される場合 (テキスト選択など)、正確さを向上させるためにオフセットされる選択ハンドルを用意します。詳しくは、「テキストと画像の選択のガイドライン」をご覧ください。

タイミング

直接操作を行うために、時間制限のあるモードの変更を避けます。直接操作は、オブジェクトに対するリアルタイムで物理的な直接処理をシミュレートします。オブジェクトは指の動きに合わせて反応します。

一方、時間制限のある操作は、タッチ操作の後に発生します。通常、時間制限のある操作では、時間、距離、速度などの見えないしきい値に基づいて実行するコマンドが決定されます。システムで操作が実行されるまで、時間制限のある操作ではビジュアルなフィードバックは返されません。

直接操作には、時間制限のある対話操作と比べて、いくつかの利点があります。

- 対話操作中にビジュアルなフィードバックがすばやく返されるので、ユーザーはより集中して、すべてを制御しているという安心感を持って操作できます。
- 直接操作は元に戻すことができるので、安心してシステムを使うことができます。ユーザーは、論理的かつ直観的な方法で操作を簡単にさかのぼることができます。
- オブジェクトに直接影響して実際の対話操作を模倣する操作は、より直観的で見つけやすく覚えやすい対話操作です。わかりにくい抽象的な対話操作には依存しません。
- 時間制限のある対話操作は、任意の見えないしきい値に達する必要があるので、実行が難しい場合があります。

また、次の点を考慮することを強くお勧めします。

- 操作は、使う指の数で区別しないでください。
- 複合操作をサポートしてください。たとえば、ピンチによるズームを行いながら指をドラッグしてパンできるようにします。
- 対話操作を時間で区別しないでください。実行にかかる時間に関係なく、同じ対話操作を行うと同じ結果が得られるようにします。時間ベースのアクティブ化では、ユーザーは遅延を強いられるので、直接操作のイマーシブの特性が損なわれ、システムの応答性が低く感じられるようになります。

  注 ただし、特定の時間制限のある対話操作を使って学習や調査に役立てる場合は例外です (長押しなど)。

- 適切な説明とビジュアルな合図を使うと、高度な対話操作を非常に効果的に使用できます。

タッチパッドのガイドライン

ユニバーサル Windows プラットフォーム (UWP) アプリの設計はタッチ入力用に最適化し、既定のタッチパッドのサポートを利用します。

ユーザーがタッチパッドで操作できるアプリを設計します。 タッチパッドは、間接的なマルチタッチ入力と、マウスのようなポインティング デバイスの精密入力を組み合わせたものです。この組み合わせにより、タッチパッドはタッチに最適化された UI にも、デスクトップ環境での生産性アプリのより小さいターゲットにも適しています。

UWP アプリのタッチパッド言語

システム内では一貫して、タッチパッド操作の簡単なセットが使われます。

用語	説明
2 本指でのタップによる右クリック	2 本の指で同時にタップすると、グローバル コマンドが含まれているアプリ バーが表示されます。要素上でタップすると、その要素が選択され、状況依存のコマンドが含まれているアプリ バーが表示されます。 **注** 選択またはアプリ バーのコマンドが適切な UI 動作ではない場合は、2 本指でのタップによりコンテキスト メニューが表示されます。ただし、すべてのコマンド動作にアプリ バーを使うことを強くお勧めします。
2 本指でのスライドによるパン	スライドは主にパン操作に使われますが、移動、描画、筆記などの操作に使うこともできます。
ピンチとストレッチによるズーム	タッチパッドでのピンチとストレッチは、サイズ変更とセマンティック ズームに使われます。
左と右のクリック ゾーン	マウス デバイスの左ボタンと右ボタンの機能をエミュレートします。
ホバーによる説明の表示	要素にホバーすると、詳しい情報や説明を伝えるビジュアル効果 (ツールチップなど) が表示されます。操作はコミットされません。
1 本指でのタップによるプライマリ操作	1 本指を使って要素をタップすると、プライマリ操作 (アプリの起動、コマンドの実行など) が呼び出されます。
1 本指でのプレスとスライドによる移動	要素をドラッグします。
1 本指でのプレスとスライドによるテキストの選択	選択可能なテキスト内を押してスライドし、選択します。単語を選択するには、ダブルタップします。
システム コマンドのためのエッジの操作 (システムに依存する)	タッチパッドの右端 (右から左のレイアウトでは左端) からスワイプすると、システム コマンドが含まれているチャームが表示されます。 タッチパッドの左端 (右から左のレイアウトでは右端) からスワイプすると、実行中のアプリの一覧が表示されます。

ビジュアルなフィードバックのガイドライン

- 移動イベントまたはホバー イベントを通じてタッチパッド カーソルが検出されたら、マウス固有の UI を表示して、要素によって公開されている機能を示します。タッチパッド カーソルが一定の期間動かされなかった場合や、ユーザーがタッチ操作を始めた場合は、タッチパッド UI を徐々に非表示にします。 これにより、UI の簡潔さが保たれます。
- ホバーのフィードバックにカーソルを使わないでください。要素によるフィードバックで十分です。
- 静的テキストなど、要素で対話操作がサポートされていない場合は、ビジュアルなフィードバックを返さないでください。
- タッチパッド操作ではフォーカス用の四角形を使わないでください。これはキーボード操作専用です。
- 同じ入力対象を表すすべての要素に対してビジュアルなフィードバックを同時に表示します。

ビジュアル なフィードバックのガイダンスについては、「ビジュアルなフィードバックのガイドライン」を参照してください。

複数の入力方法のガイドライン

人がお互いにコミュニケーションをとる際に音声とジェスチャを組み合わせて使うように、アプリの操作では、複数の種類とモードの入力を使用すると便利な場合があります。

説明

できるだけ多くのユーザーやデバイスに対応するため、可能な限り多くの入力の種類 (ジェスチャ、音声、タッチ、タッチパッド、マウス、キーボード) で作業できるようにアプリを設計することをお勧めします。これにより、柔軟性、操作性、アクセシビリティが最大限に高まります。

最初に、アプリで入力を処理するさまざまなシナリオを検討します。アプリ全体で一貫性を保つようにし、プラットフォーム コントロールでは、複数の入力の種類に対応する組み込みサポートを用意します。

- ユーザーは、複数の入力デバイスを使ってアプリケーションを操作できますか?
- すべての入力方法が常にサポートされていますか?特定のコントロールでサポートされていますか?特定の時間や環境でサポートされていますか?
- 1 つの入力方法が優先されますか?

単一 (排他的) モードの操作

単一モードの操作では、複数の入力の種類がサポートされますが、1 つのアクションで使用できるのは、1 つのみです。たとえば、コマンドに音声認識、ナビゲーションにジェスチャなどです。または近接度に応じて、タッチかジェスチャを使用してテキストを入力します。

マルチ モーダル操作

マルチモーダル操作では、1 つのアクションを完了するために複数の入力方法が順番に使われます。

音声認識 + ジェスチャ

ユーザーは製品をポイントし、「カートに追加」と言います。

音声認識 + タッチ

ユーザーは長押しを使用して写真を選択し、「写真の送信」と言います。

アクセシビリティのガイドライン

アプリを設計する際は、ユーザーによってできる操作、できない操作、好ましい操作が大きく異なることを常に心に留めておいてください。こうしたアクセシビリティ対応の設計の原則に従うことで、可能な限り広範な対象ユーザーにアプリを利用してもらい、Windows ストアでアプリに対してより多くのユーザーの興味を喚起するうえで役立ちます。

アクセシビリティのための計画が必要な理由

アクセシビリティを考慮して設計したアプリは、次のシナリオに適しています。

- **画面の読み上げ:** 視覚に障碍があるユーザーは、アプリの UI の画像を頭の中に描いて保つために、スクリーン リーダーを使います。UI 要素の名前など、UI に関する情報を耳で聞くことで、UI コンテンツを把握し、アプリを操作できるようになります。

  画面の読み上げをサポートするには、名前、役割、説明、状態、値など、UI 要素に関する正確な情報を十分にアプリで提供する必要があります。詳しくは、「基本的なアクセシビリティ情報の公開」をご覧ください。

  また、JavaScript と HTML を使った Windows ストア アプリにおけるライブ領域など、動的コンテンツを含む UI 要素に関する追加のアクセシビリティ情報を提供する必要もあります。スクリーン リーダーで追加のアクセシビリティ情報を使うことで、コンテンツに加えられた変更をユーザーに知らせることができます。HTML でライブ領域に関するアクセシビリティ情報を提供するには、動的コンテンツを含む要素の **aria-live** 属性を設定します。 詳しくは、「ライブ領域をアクセシビリティ対応にする」をご覧ください。XAML でライブ コンテンツの ARIA メタファーを使ってライブ領域のアクセシビリティ情報を提供するには、AutomationProperties.LiveSetting 添付プロパティを使います。
- **キーボードのアクセシビリティ:** キーボードはスクリーン リーダーを使ううえで不可欠であり、キーボードを使った方がアプリを効率よく操作できるユーザーにとっても重要です。キーボードでアクセスできるアプリでは、ユーザーはすべての対話型 UI 要素にキーボードだけでアクセスでき、次の操作を実行できます。
  - [Tab] キーと方向キーを使って、アプリ内を移動する。
  - [Space] キーと [Enter] キーを使って、UI 要素をアクティブ化する。

- キーボード ショートカットを使って、コマンドとコントロールにアクセスする。

物理的なキーボードが存在しないシステムや、運動障碍により一般的な物理入力デバイスを使えないユーザーのために、スクリーン キーボードが用意されています。

詳しくは、キーボードのアクセシビリティの実装についてのページをご覧ください。

- **アクセシビリティに対応したビジュアルな効果:** 視覚に障碍があるユーザーには、テキストを高いコントラスト比で表示する必要があります。また、UI がハイ コントラスト モードで適切に表示され、[コンピューターの簡単操作] コントロール パネルで **[画面上のすべてのものを大きくする]** を選んだときに適切に拡大されることも必要です。色を使って情報を伝える場合、色覚に障碍があるユーザーに対しては、色の代わりにテキスト、図形、アイコンなどを使う必要があります。詳しくは、「ハイ コントラスト テーマのサポート」についてのページをご覧ください。詳しくは、「アクセシビリティに対応したテキストの要件の適合」についてのページをご覧ください。

推奨と非推奨

- 名前、役割、説明、状態、値を含めて、アプリの UI 要素について情報を提供することにより、画面の読み上げをサポートします。
- ユーザーが [Tab] キーと方向キーを使ってアプリをナビゲートできるようにします。
- [Space] キーと [Enter] キーを使って、UI 要素をアクティブ化する。
- キーボード ショートカットを使って、コマンドとコントロールにアクセスする。
- ハイ コントラスト テーマをサポートするようにテキストと UI を設計します。
- **[簡単操作]** 設定が変更されたときに適切なスケーリングの調整を行うためのテキストおよび UI を準備します。
- 色を情報を伝える唯一の手段として使わないようにします。 色覚に障碍があるユーザーは、色によるステータス インジケーターのような色を通じてのみ伝えられる情報は受け取ることができません。他の視覚的な合図 (テキストが望ましい) を含めるようにして、情報にアクセスできるようにします。

- 1 秒間に 4 回以上光る UI 要素は使わないようにします。 明滅する要素は一部の人にとって発作を起こす原因になります。明滅する UI 要素の使用は避けた方が良いでしょう。
- ユーザー コンテキストを変えたり自動的に機能をアクティブ化しないようにします。 コンテキストやアクティブ化の変更は、フォーカスのある UI 要素上でユーザーが直接操作したときにだけ行うようにします。 ユーザー コンテキストの変更には、フォーカスの変更、新しいコンテンツの表示、別のページへの移動が含まれます。ユーザーと無関係に行われるコンテキストの変更は、障碍のあるユーザーを混乱させる可能性があります。 この要件の例外としては、サブメニューの表示、フォームの検証、別のコントロールでのヘルプ テキストの表示、非同期イベントへの応答によるコンテキストの変更などがあります。
- Windows ランタイムに含まれる既定のコントロール、または Microsoft UI オートメーション サポートを既に実装しているコントロールを使うことができる場合には、カスタム UI 要素を作成しないようにします。Windows ランタイムの標準コントロールは、既定でアクセシビリティに対応しており、追加する必要があるのは、通常、アプリ固有のわずかなアクセシビリティ属性のみです。それに対し、純粋なカスタム コントロールに **AutomationPeer** サポートを実装するのは、これより複雑です (「カスタム オートメーション ピア」をご覧ください)。
- 静的テキストなどの非対話型の要素をタブ オーダーに含めないようにします (たとえば、対話的に操作できない要素に **TabIndex** プロパティを設定することは避けます)。非対話型の要素をタブ オーダーに含めると、キーボード ナビゲーションの効率が下がるため、キーボードのアクセシビリティ ガイドラインで推奨されていません。多くの支援技術では、支援技術のユーザーにアプリのインターフェイスを表示する方法に関するロジックの一部として、要素をフォーカスする機能とタブ オーダーが使われています。タブ オーダーにテキスト専用要素が含まれると、タブ オーダーに対話型の要素 (ボタン、チェック ボックス、テキスト入力フィールド、コンボ ボックス、リストなど) しか含まれていないと想定しているユーザーを混乱させる可能性があります。
- UI 要素の絶対配置 (**Canvas** 要素内など) は、しばしば表示の順序が (事実上の論理的な順序である) 子要素の宣言の順序と異なるため、使用を避けます。 スクリーン リーダーが UI 要素を正しい順序で読み取ることができるように、これらの要素をできるだけドキュメントの順序または論理的な順序に並べます。UI 要素の視覚的

な順序がドキュメントの順序または論理的な順序とは異なる可能性がある場合は、正しい読み取り順序を定義するために明示的なタブ インデックス値 (**TabIndex** に設定) を使います。

- アプリの機能に本当に必要でない限り、アプリのキャンバス全体を自動的に更新しないようにします。 ページの内容を自動的に更新する必要がある場合には、ページの一部の領域だけを更新するようにします。支援技術では通常、更新されたアプリのキャンバスは、実質的な変更がわずかな場合でも、完全に新しい構造であると見なす必要があります。このため、更新されたアプリのドキュメント ビューや説明を再作成し、支援技術を使うユーザーに再提示する必要があります。

  注 領域内のコンテンツを更新する場合は、要素上の **AccessibilityProperties.LiveSetting** アクセシビリティ プロパティを既定以外の設定である **Polite** または **Assertive** に設定することをお勧めします。支援技術の中には、ライブ領域の Accessible Rich Internet Applications (ARIA) 概念にこの設定をマップして、コンテンツの領域が変更されたことをユーザーに通知できるものもあります。

  注 ユーザーが開始する意図的なページのナビゲーションによってアプリの構造が更新されることは、問題ありません。ただし、ナビゲーションを開始する UI 項目に適切な ID または名前を設定し、呼び出すとコンテキストが変更されてページが再読み込みされることがわかるようにしてください。

その他の使い方のガイダンス

HTML カスタム コントロールのアクセシビリティ対応

HTML カスタム コントロールを使う場合、アクセシビリティに対応する名前、役割、状態、値など、コントロールに関する基本的なアクセシビリティ情報をすべて用意しておく必要があります。また、コントロールがキーボードで完全にアクセスでき、UI が視覚に関するアクセシビリティの要件を満たしていることを確認する必要もあります。

たとえば、カスタムの対話要素を表す (つまり、**onclick** イベントを処理する) **div** 要素を含めるとします。 次の操作を行う必要があります。

- **div** 要素のアクセシビリティ対応の名前を設定します。
- role 属性を、Accessible Rich Internet Applications (ARIA) の対応する対話型の役割 ("button" など) に設定します。
- tabindex 属性を、タブの順序で要素を含むように設定します。
- キーボードによるアクティブ化をサポートする keyboard イベント ハンドラーを追加します。これは **onclick** イベント ハンドラーのキーボード版です。

アクセシビリティのためのカスタム HTML UI 要素の公開について詳しくは、「Accessible Rich Internet Applications (ARIA)」をご覧ください。

注 HTML5 の **canvas** 要素では、アクセシビリティがサポートされません。**canvas** にはコンテンツのアクセシビリティ情報を公開する方法がないため、どうしても必要な場合以外は、使わないでください。**canvas** を使う場合は、カスタム UI 要素として扱います。

XAML カスタム コントロールのアクセシビリティ対応

XAML カスタム コントロールを使う場合、アクセシビリティに対応する名前、役割、状態、値など、コントロールに関する基本的なアクセシビリティ情報の一部を調整することが必要になる可能性があります。また、コントロールがキーボードで完全にアクセスでき、UI が視覚に関するアクセシビリティの要件を満たしていることを確認する必要もあります。XAML 用のカスタム コントロールを作成する場合、カスタム コントロールの基底クラスとして使用する、あらゆるコントロールで使用できる UI オートメーション サポートを継承します。これで十分な場合もあります。コントロールをカスタマイズする度合いに応じて、既定の UI オートメーション サポートを変更または強化する、カスタム UI オートメーション ピア クラスを作成することもできます。これは、Windows.UI.Xaml.Automation 名前空間と Windows.UI.Xaml.Automation.Peers 名前空間の API によって実現されます。詳しくは、「カスタム オートメーション ピア」をご覧ください。

開発プラットフォームでのアクセシビリティのサポート

Windows Runtime の開発プラットフォームでは、開発サイクルのすべての段階でアクセシビリティをサポートしています。

- **作成:** Microsoft Visual Studio のアプリ テンプレートから生成されたコードには、アクセシビリティ情報が含まれます。
- **コード作成:** 開発プラットフォームでは、コード作成段階で次のアクセシビリティをサポートしています。
  - Visual Studio の Microsoft IntelliSense には、アクセシビリティ情報が含まれています。
  - Windows 10 プラットフォームに含まれているコントロールには、アクセシビリティのサポートが組み込まれています。標準の HTML とプラットフォームのコントロールを使うと、ほとんどのアクセシビリティを既定のプラットフォームの動作としてサポートできます。たとえば、評価コントロールは完全にアクセシビリティに対応しており、追加の作業は必要ありません。また、**ListView** コントロールは、メインのリスト要素のアクセス可能な名前だけが必要です。それ以外のアクセシビリティのサポートはすべて組み込まれています。プラットフォーム コントロールの一覧については、「コントロールの一覧 (HTML)」または「コントロールの一覧」をご覧ください。
  - Windows デベロッパー センターのドキュメントには、アクセシビリティのガイドラインとサンプル アプリが含まれています。
- **テスト:** Windows ソフトウェア開発キット (Windows SDK) には、アクセシビリティのテスト ツールが用意されています。詳しくは、アプリのアクセシビリティのテストについてのページをご覧ください。
- **販売:** Windows ストアではアプリを公開する際にアクセシビリティ対応として登録できます。そうすると、ユーザーが Windows ストアでアクセシビリティ フィルターを使ってアプリを見つけられるようになります。 詳しくは、Windows ストアでアプリをアクセシビリティ対応として宣言する方法についてのページをご覧ください。

クロススライドのガイドライン

クロススライドは、スワイプ ジェスチャーによる選択や、スライド ジェスチャーによるドラッグ (移動) 操作をサポートするために使います。

重要な API

CrossSliding イベント

CrossSlideThresholds プロパティ

Windows.UI.Input 名前空間

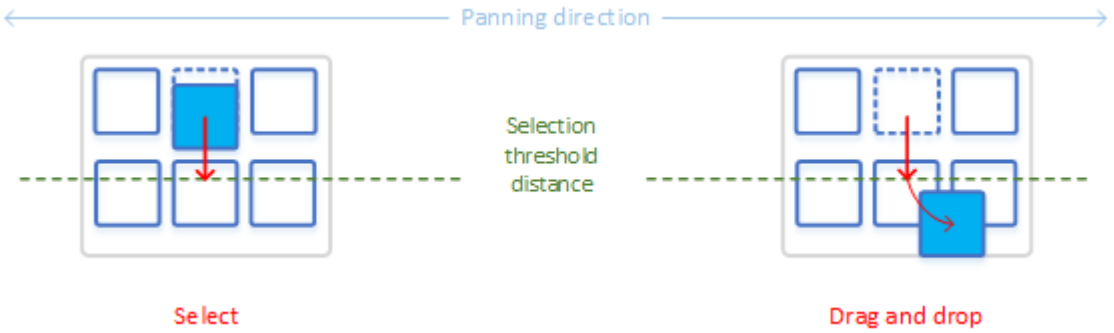
推奨と非推奨

- クロススライドは、単一の方向にスクロールするリストやコレクションだけに使います。
- クロススライドは、タップ操作が別の目的で使われる場合に、項目を選ぶために使います。
- キューに項目を追加するためにクロススライドを使わないでください。

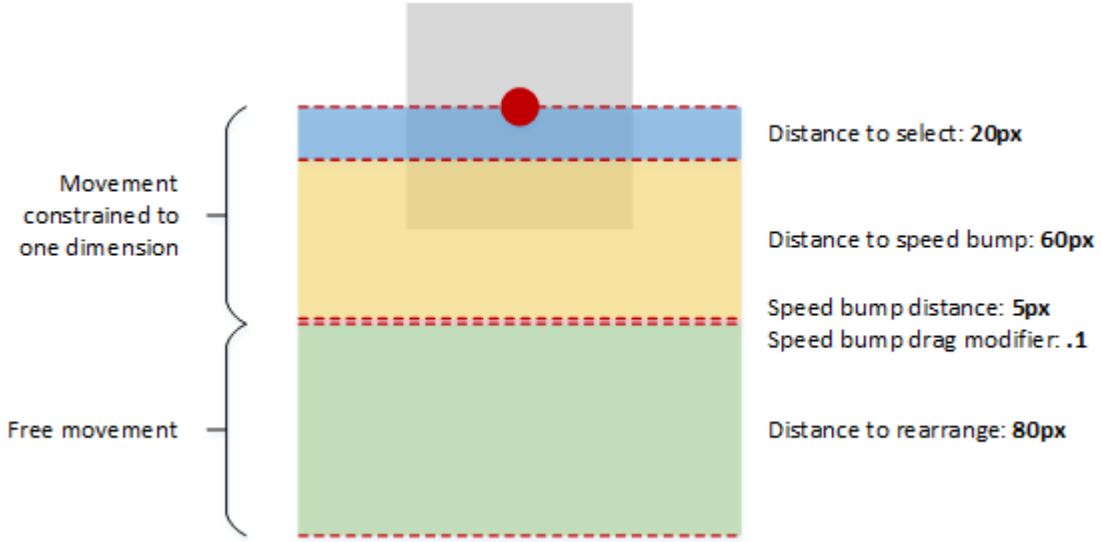
その他の使い方のガイダンス

選択とドラッグは、1 方向 (垂直または水平) にパンできるコンテンツ領域内でだけ行うことができます。これらの操作が機能するには、1 つのパン方向がロックされていて、ジェスチャーがパン方向に対して垂直な方向に行われる必要があります。

ここでは、クロススライドを使ってオブジェクトを選び、ドラッグする方法を示します。左の図は、スワイプ ジェスチャーで距離のしきい値を超える前に指を離してオブジェクトを解放することで、項目が選択された状態を示しています。右の図は、距離のしきい値を超えてスライド ジェスチャーを行うことで、オブジェクトがドラッグされた状態を示しています。

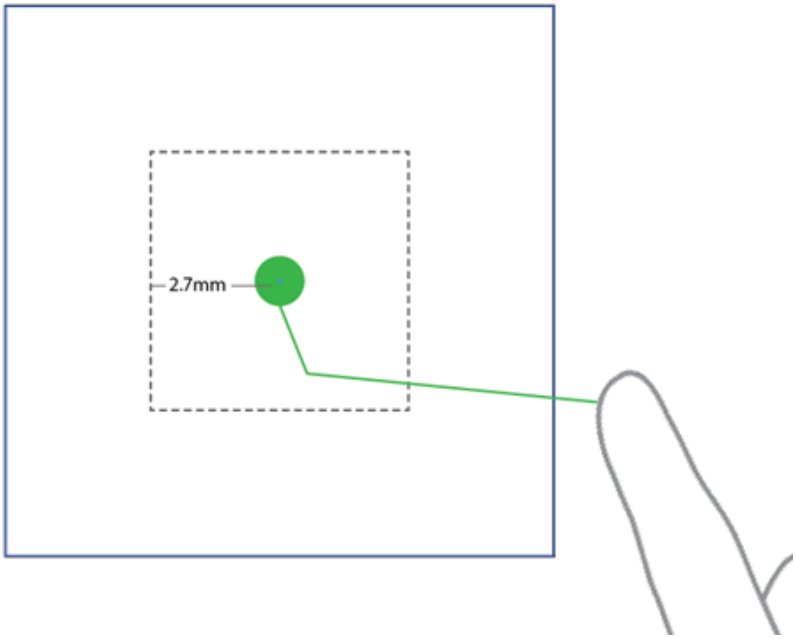


クロススライド操作で使われるしきい値の距離を次の図に示します。



パンの機能を維持するために、選択操作とドラッグ操作は、2.7 mm (ターゲット解像度で約 10 ピクセル) という小さいしきい値を超えないと有効にならないしくみになっています。この小さいしきい値は、クロススライドとパンの区別に使われるほか、タップ ジェスチャーをクロススライドやパンと区別する目的でも使われます。

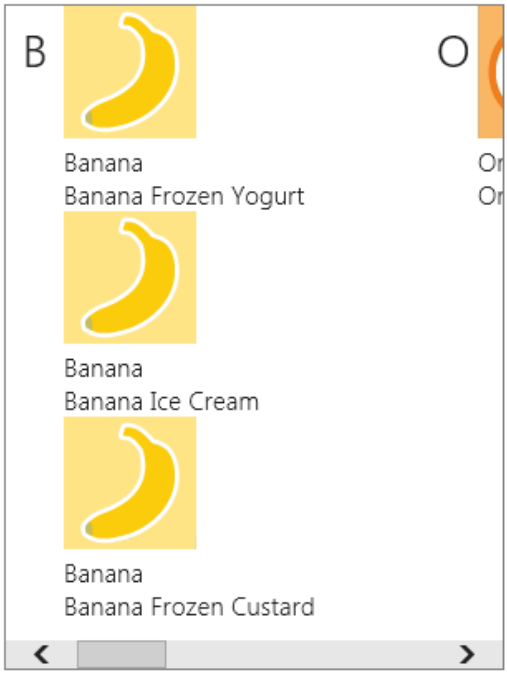
この図は、ユーザーが UI の要素にタッチしたときに、指の位置がわずかに下に動いて しまった状態を示しています。 しきい値がなければ、最初に垂直方向に 移動しているため、この操作はクロススライドと解釈されてしまいます。このしきい値があるおかげで、水平方向のパンと正しく解釈されます。

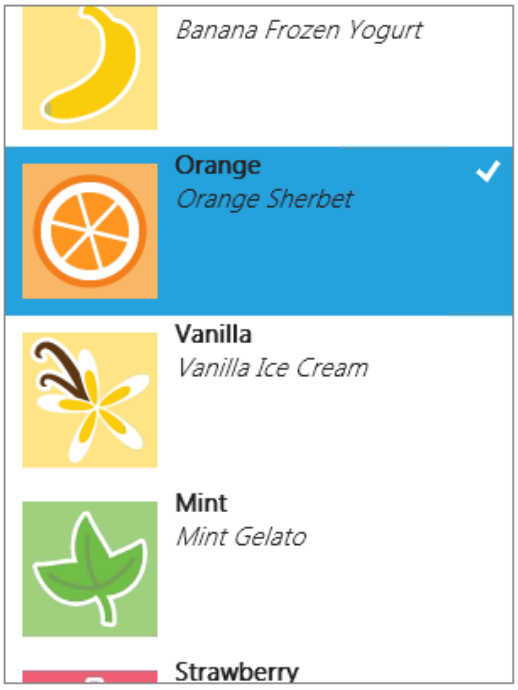


次に、クロススライド機能をアプリに含める際に考慮する必要がある、いくつかのガイドラインを示します。

クロススライドは、単一の方向にスクロールするリストやコレクションだけに使います。

注 Web ブラウザーや電子ブック リーダーのように、コンテンツ領域を 2 方向にパンできる場合は、時間制限のある長押し操作を使って、画像やハイパーリンクなどのオブジェクトのコンテキスト メニューを呼び出すようにしてください。





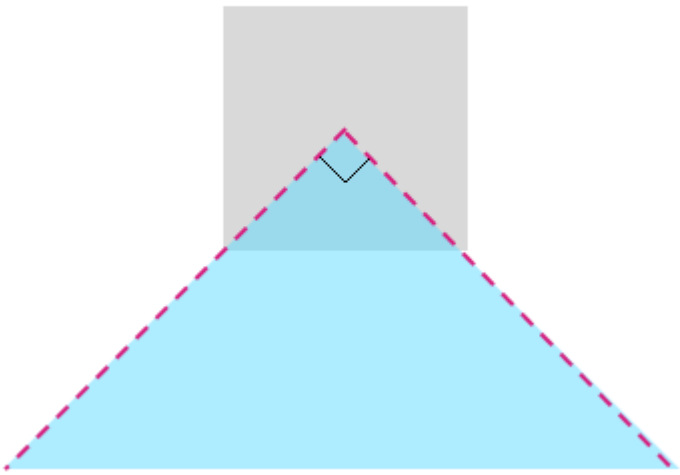
水平方向にパンする 2 次元のリスト。 項目を選択または移動するには垂直方向にドラッグします。	垂直方向にパンする 1 次元のリスト。項目を選択または移動するには水平方向にドラッグします。

選択

選択は、1 つ以上のオブジェクトを起動またはアクティブ化せずにマークする操作です。これは、マウスを 1 回クリックする操作、または Shift キーを押しながらクリックする操作 (オブジェクトが複数の場合) に相当します。

クロススライド選択を行うには、要素をタッチし、少しドラッグして放します。この選択方法を使えば、他のタッチ インターフェイスで必要になるような、専用の選択モードや時間制限のある長押し操作は必要ありません。また、アクティブ化のためのタップ操作と競合することもありません。

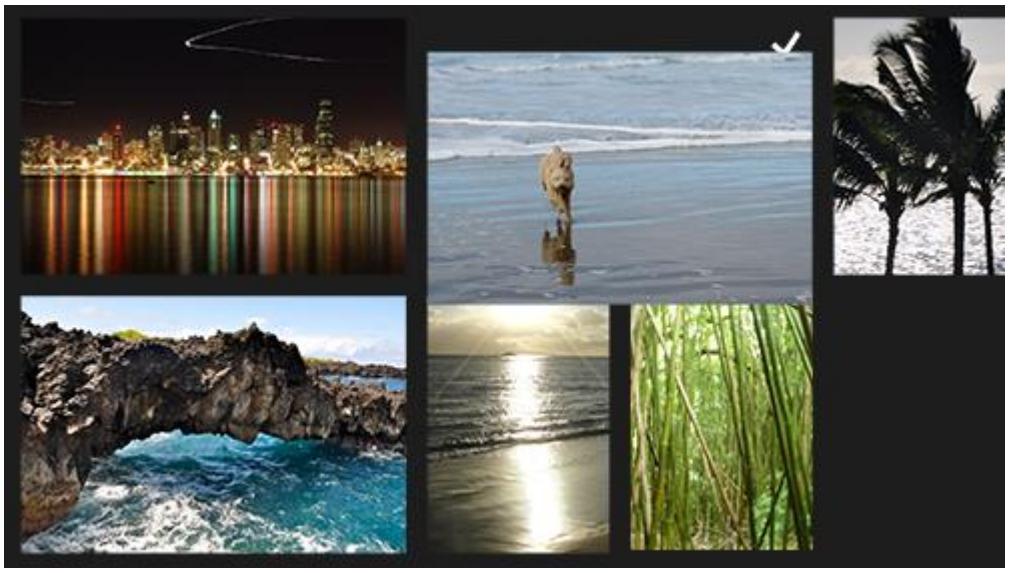
クロススライド選択には、距離のしきい値のほかにも領域のしきい値があり、次の図に示すように範囲が 90° に制限されます。この領域の外にドラッグすると、オブジェクトは選択されません。



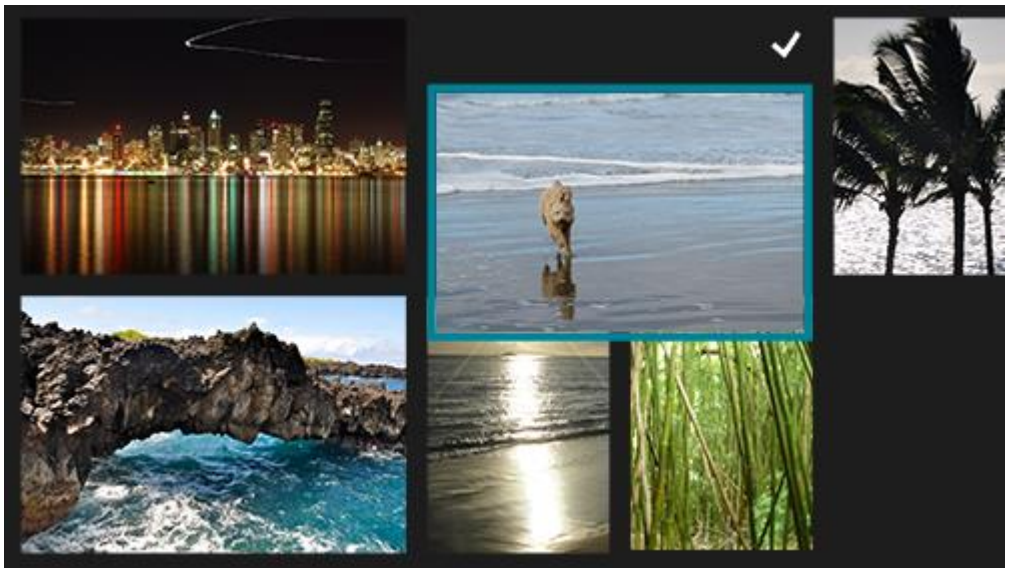
クロススライド操作を補完する操作に、"自己表明" 操作とも呼ばれる時間制限のある長押し操作があります。この補助操作でアクティブ化されるアニメーションによって、オブジェクトに対して実行できる操作が示されます。

次のスクリーン ショットは、自己表明操作のアニメーションの動作を示しています。

1. 長押しして、自己表明操作のアニメーションを開始します。項目が選ばれているかどうかによって、アニメーションで説明される内容が変わります。選ばれていない場合はチェック マークが付き、選ばれている場合はチェック マークが付きません。



2. スワイプ ジェスチャ —(上または下) を使って項目を選びます。



3. この時点で、項目が選ばれています。スライド ジェスチャーを使って選択動作を上書きし、項目を移動します。

主な操作が選択だけであるアプリケーションでは、選択にシングル タップを使います。この場合、アクティブ化やナビゲーションのための標準のタップ操作と区別するために、クロススライドの自己表明のアニメーションが表示されます。

選択バスケット

選択バスケットは、アプリの主要なリストやコレクションから選択された項目を視覚的に区別して動的に表す機能です。これは選択された項目の追跡に役立つ機能で、次のようなアプリで使うと便利です。

- 項目を複数の場所から選択できる。
- 複数の項目を選択できる。
- 選択リストによって操作やコマンドが異なる。

選択バスケットの内容は、操作やコマンドの実行後も保持されます。たとえば、ギャラリーから一連の写真を選択して各写真に色補正を適用し、それらの写真を何らかの方法で共有した場合、それらの項目は選択されたままになります。

アプリで選択バスケットを使わない場合は、操作やコマンドの実行後に現在の選択がクリアされます。たとえば、再生リストから曲を選択して評価した後、その選択はクリアされるなどです。

また、選択バスケットを使わない場合は、リストやコレクションで別の項目がアクティブ化されたときにも現在の選択がクリアされます。たとえば、受信トレイのメッセージを選択すると、プレビュー ウィンドウが更新されます。その後、受信トレイで別のメッセージを選択すると、前のメッセージの選択が取り消され、プレビュー ウィンドウが更新されます。

キュー

キューと選択バスケットのリストは異なるものであるため、混同しないように注意してください。主な違いは次のとおりです。

- 選択バスケットの項目のリストは、視覚的に表すことだけを目的としたものです。キューの項目は、特定の操作を想定してまとめられたものです。
- 選択バスケットでは同じ項目は 1 回しか表示できませんが、キューでは複数回表示できます。
- 選択バスケットの項目の順序は、選択の順序を表します。 キューの項目の順序は、機能に直接関連します。

これらの理由から、キューに項目を追加する目的でクロススライド選択操作を使わないでください。 キューに項目を追加するときは、代わりにドラッグを使います。

ドラッグ

1 つまたは複数のオブジェクトを別の場所に移動するには、ドラッグを使います。

複数のオブジェクトを移動する必要がある場合は、ユーザーが複数の項目を選択してから、すべてを同時にドラッグできるようにします。

光学式ズームとサイズ変更のガイドライン

このトピックでは、新しい Windows のズームと要素のサイズ変更について説明し、アプリでこのような新しい操作のメカニズムを使うときのユーザー エクスペリエンスのガイドラインを示します。

重要な API

Windows.UI.Input 名前空間

ズームとパン (HTML)

Input 名前空間 (XAML)

説明

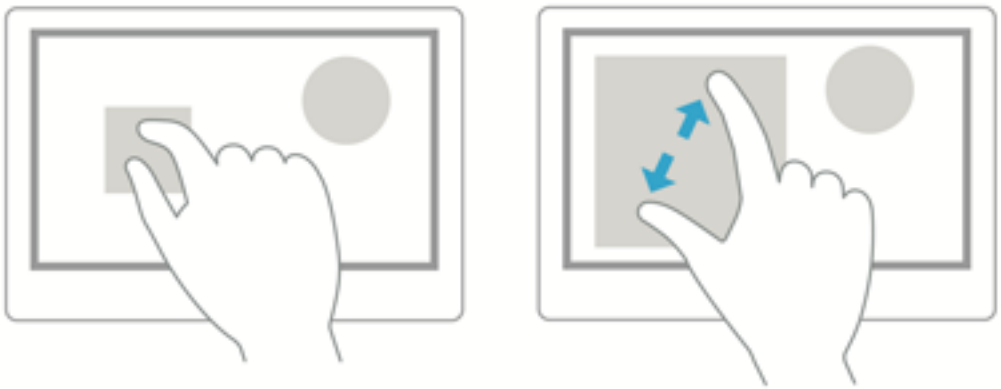
光学式ズームを使うと、ユーザーはコンテンツの表示を拡大できます (コンテンツ領域自体に対して実行されます)。一方、サイズ変更を使うと、コンテンツ領域に対する表示は変更せずに、1 つまたは複数のオブジェクトの相対的なサイズをユーザーが変更できます (コンテンツ領域内のオブジェクトに対して実行されます)。

光学式ズーム操作とセマンティック ズーム操作は両方とも、ピンチ ジェスチャーとストレッチ ジェスチャー (指を広げて拡大、互いに近づけて縮小)、[Ctrl] キーを押しながらマウスのスクロール ホイールをスクロール、または [Ctrl] キーを (テンキーがない場合は [Shift] キーも同時に) 押しながらプラス (+) キーまたはマイナス (-) キーを押して実行します。

次の図にサイズ変更と光学式ズームの違いを示します。

光学式ズーム: ユーザーは領域を選び、領域全体を拡大します。

サイズ変更: ユーザーは領域内のオブジェクトを選び、そのオブジェクトのサイズを変更します。



注　光学式ズームとセマンティック ズーム操作を混同しないように気を付けてください。これらは同じジェスチャーを共有しますが、セマンティック ズームは、構造化されたデータまたはコンテンツを単一のビュー内で表示したりナビゲーションしたりする場合に使われます (コンピューターのフォルダー構造、ドキュメント ライブラリ、フォト アルバムなど)。

推奨と非推奨

サイズ変更または光学式ズームをサポートするアプリでは、次のガイドラインに従ってください。

- 最大サイズと最小サイズの制限または範囲が定義されている場合には、ビジュアルなフィードバックを使って、ユーザーがこの制限に達したことや超過したことを示します。
- スナップ位置を使うと、論理的な操作停止位置を指定してズームとサイズ変更の動作を変更し、コンテンツの特定の部分がビューポートに表示されるようにできます。一般的なズーム レベルまたは論理ビューに対してスナップ位置を設定して、ユーザーがこれらのレベルを簡単に選べるようにします。たとえば、写真のアプリ

では 100% の位置にサイズ変更用のスナップ位置を設定します。また、地図のアプリでスナップ位置を設定すると、市、県、国を表示する場合に便利です。
スナップ位置があると、ユーザーの操作が正確でなくても意図された操作を実行できます。XAML を使う場合は、ScrollViewer のスナップ位置のプロパティをご覧ください。JavaScript と HTML の場合は、-ms-content-zoom-snap-points を使います。
スナップ位置には次の 2 種類があります。
  - 近接: 指を離した後、スナップ位置の距離のしきい値の範囲内で慣性に従った動きが止まると、スナップ位置が選ばれます。近接スナップ位置の場合は、ズームとサイズ変更をスナップ位置とスナップ位置の間で止めることができます。
  - 強制: 指を離す前に通過した最後のスナップ位置の直前または直後のスナップ位置が選ばれます (ジェスチャの方向と速度によって異なります)。操作が必ず強制スナップ位置で止まるようにする必要があります。
- 慣性の法則を使う必要があります。これには次のものがあります。
  - 減速: ユーザーが 2 本の指を互いに近づけたり、遠ざけたりしたときに発生します。これは滑りやすい表面で滑っている状態から止まるまでの動きに似ています。
  - バウンド: サイズの制限または範囲を超えると、わずかな跳ね返りの効果が発生します。
- 「ターゲットの設定のガイドライン」に従った領域制御。
- 制限付きのサイズ変更のためにスケーリング ハンドルを提供します。ハンドルが指定されない場合は、等角投影、つまり比が一定のサイズ変更が既定値です。
- UI の操作またはアプリ内の追加コントロールの公開用にはズームを使わず、パン領域を使います。パンについて詳しくは、「パンのガイドライン」をご覧ください。
- サイズ変更できるコンテンツ領域内にサイズ変更できるオブジェクトを置かないようにします。ただし、次のような例外があります。
  - サイズ変更できるアイテムがサイズ変更できるキャンバスまたはアート ボードに表示される描画アプリケーション。
  - 地図などの埋め込みオブジェクトがある Web ページ。

注　どのような場合でも、すべてのタッチ ポイントがサイズ変更できるオブジェクト内にある場合以外は、コンテンツ領域のサイズが変更されます。

パンのガイドライン

パンとスクロールにより、ユーザーは単一ビュー内で移動し、ビューポートに収まらないビューのコンテンツを表示できます。ビューの例として、コンピューターのフォルダー構造、ドキュメントのライブラリ、フォト アルバムなどがあります。

重要な API

Windows.UI.Input 名前空間

ズームとパン (HTML)

Input 名前空間 (XAML)

推奨と非推奨

パン インジケーターとスクロール バー

- アプリにコンテンツを読み込む前に、パン/スクロールが可能であることを確認します。
- パン インジケーターとスクロール バーを表示して、位置とサイズがわかるようにします。 これらのコントロールは、カスタム ナビゲーション機能がある場合には非表示にします。

  注 標準のスクロール バーとは異なり、パン インジケーターは情報提供のみを目的としています。 入力デバイスには公開されず、一切操作できません。

単一軸パン (1 次元のオーバーフロー)

- コンテンツ領域が 1 つのビューポート境界 (垂直方向または水平方向) を超えている場合は、単一軸のパンを使います。
  - 1 次元の項目の一覧の場合は、垂直方向のパンを使います。
  - 項目のグリッドの場合は、水平方向のパンを使います。
- ユーザーのパン操作をスナップ位置以外の位置で停止できるようにする必要がある場合は、単一軸パンで強制スナップ位置を使わないでください。 強制スナップ位置を使うと、スナップ位置で必ず停止します。代わりに、近接スナップ位置を使ってください。

フリーフォーム パン (2 次元のオーバーフロー)

- コンテンツ領域が両方のビューポート境界 (垂直方向と水平方向) を超えている場合は、2 軸のパンを使います。
  - 複数の方向へ動かされる可能性がある、構造化されていないコンテンツの場合は、既定のレール動作を上書きしてフリーフォーム パンを使います。
- フリーフォーム パンは通常、画像や地図内の移動に適しています。

ページ ビュー

- コンテンツが個別の要素で構成されている場合、または 1 つの要素全体を表示する必要がある場合は、強制スナップ位置を使います。書籍や雑誌のページ、項目の列、個々の画像がその例です。
  - スナップ位置はそれぞれの論理的な境界に置く必要があります。
  - 各要素のサイズや倍率を、ビューに収まるように調整する必要があります。

論理的な位置と主要位置

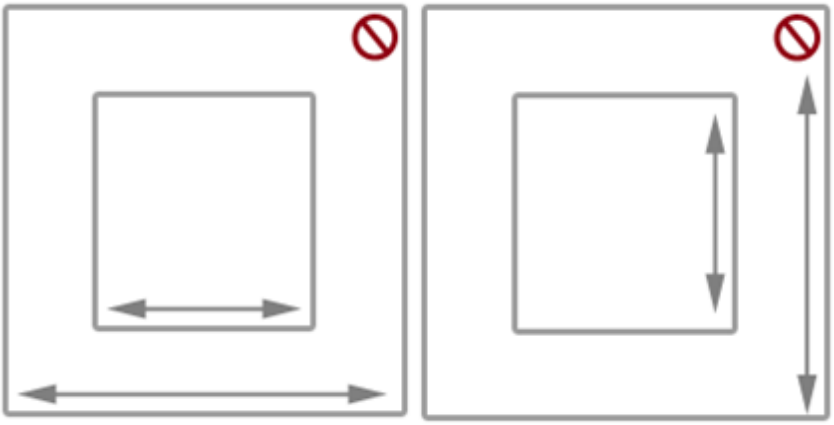
- コンテンツ内にユーザーが停止する可能性が高い主要位置または論理的な位置がある場合は、近接スナップ位置を使います。たとえば、セクション ヘッダーなどです。
- 最大サイズと最小サイズの制限または範囲が定義されている場合には、ビジュアルなフィードバックを使って、ユーザーがこの制限に達したことや超過したことを示します。

埋め込まれたコンテンツまたは入れ子になったコンテンツの連結

- テキストとグリッド ベースのコンテンツに対して単一軸パン (通常は水平方向) と列レイアウトを使います。このような場合は、コンテンツは通常列から列へと自然に折り返し、遷移するので、Windows アプリ全体で一貫性があり見つけやすいユーザー エクスペリエンスを維持できます。
- テキストまたは項目の一覧を表示する目的で、埋め込まれたパン対応領域を使わないでください。領域内で入力の接触が検出されたときしかパン インジケーターと

スクロール バーが表示されず、直感的で見つけやすいユーザー エクスペリエンスが得られません。

- 下の図に示すように、2 つのパン対応領域がどちらも同じ方向にパンする場合は、パン対応領域を別のパン対応領域内に連結 (配置) しないでください。この場合に連結すると、子領域の境界に到達したときに親領域が意図せずパンされる可能性があります。パンの軸は相互に対して垂直になるようにしてください。



その他の使い方のガイダンス

タッチでのパン (1 本または複数の指でのスワイプまたはスライド ジェスチャー) は、マウスでのスクロールと似ています。パンはスクロール バーのクリックよりも、マウス ホイールの回転やスクロール ボックスのスライドに最も近い操作です。API で区別されているか、一部のデバイス固有の Windows UI によって区別が必要とされていない限り、両方の操作を単にパンと呼びます。

入力デバイスに応じて、ユーザーは次のいずれかを使って、パン対応領域内でパンを実行します。

- マウス、タッチパッド、またはアクティブなペン/スタイラスを使って、スクロール矢印をクリックするか、スクロール ボックスをドラッグするか、スクロール バー内をクリックする。
- マウスのホイール ボタンを使って、スクロール ボックスのドラッグと同じ動作を実現する。
- マウスでサポートされている場合は、拡張ボタン (XBUTTON1 と XBUTTON2)。

- キーボードの方向キーを使ってスクロール ボックスのドラッグと同じ動作を実現するか、ページ キーを使ってスクロール バー内のクリックと同じ動作を実現する。
- タッチ、タッチパッド、またはパッシブなペン/スタイラスを使って、任意の方向に指をスライドまたはスワイプする。

スライドでは、指をパン方向にゆっくり移動します。これにより、コンテンツが指と同じ速度で同じ距離だけパンする 1 対 1 の関係ができます。スワイプ (指をすばやくスライドして離す) では、パンのアニメーションに次の物理的効果が適用されます。

- 減速 (慣性): 指を離すとパンが減速し始めます。これは滑りやすい表面で滑っている状態から止まるまでの動きに似ています。
- 吸収: 減速時に、パン操作の勢いがスナップ位置またはコンテンツ領域の境界まで保たれた場合、反対方向に少し押し戻される効果があります。

パンの種類

Windows 8 以降 では 3 種類のパンがサポートされます。

- 単一軸: 一方向 (水平または垂直) へのパンのみがサポートされます。
- レール: 全方向へのパンがサポートされます。ただし、特定の方向への距離のしきい値を超えると、パンはその軸に制限されます。
- フリーフォーム: 全方向へのパンがサポートされます。

パンの UI

パンの操作エクスペリエンスは、機能的には類似していても、入力デバイスごとに異なります。

パン対応領域

パン対応領域の動作は、JavaScript を使った Windows ストア アプリの開発者に対して、設計時にカスケード スタイル シート (CSS) を通じて公開されます。

検出された入力デバイスに基づいて、次の 2 種類のパン表示モードが使われます。

- パン インジケーター (タッチを使う場合)。
- スクロール バー (マウス、タッチパッド、キーボード、スタイラスなど、その他の入力デバイスを使う場合)。

注 パン インジケーターは、タッチによる接触がパン対応領域内であるときにだけ表示されます。同様に、スクロール バーは、スクロール対応領域内にマウス カーソル、ペン/スタイラス カーソル、またはキーボード フォーカスがあるときにのみ表示されます。

パン インジケーター

パン インジケーターは、スクロール バーのスクロール ボックスに似ています。パン対応領域全体に対する表示されているコンテンツの比率と、パン対応領域内の表示されているコンテンツの相対的な位置を示します。

次の図は、長さが異なる 2 つのパン対応領域とそれらのパン インジケーターを示しています。

パン の動作

スナップ位置

パンとスワイプ ジェスチャーを使うと、タッチによる接触が離れたときの操作に慣性の動作が生じます。慣性によって、コンテンツのパンは、ユーザーによる直接入力がなければ距離のしきい値に到達するまで継続されます。この慣性の動作を変更するには、スナップ位置を使います。

スナップ位置は、アプリのコンテンツの論理的な停止を指定します。スナップ位置は、認識に基づくユーザー用のページング メカニズムとして機能し、ユーザーが大きなパン対応領域でスライドまたはスワイプしすぎて疲れるのを防ぎます。これ

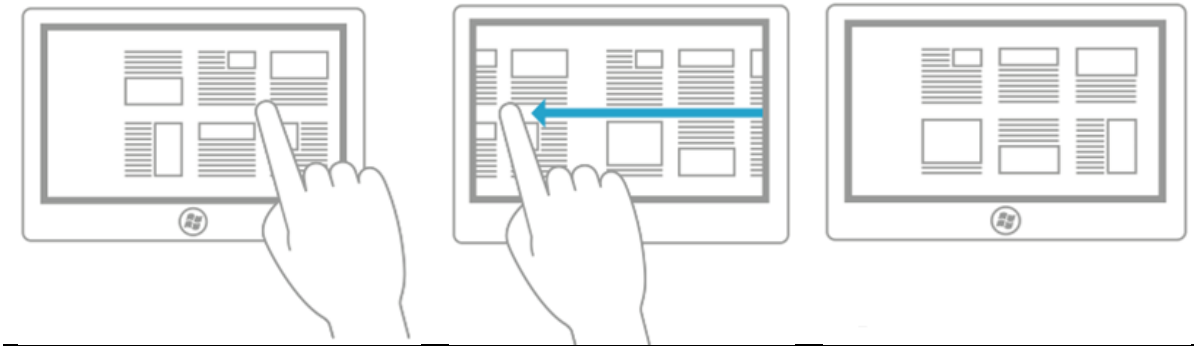
らを使用すると、不正確なユーザー入力を処理し、コンテンツの特定の部分や主要な情報がビューポートに確実に表示されるようにすることができます。

スナップ位置には次の 2 種類があります。

- 近接: 指を離した後、スナップ位置の距離のしきい値の範囲内で慣性に従った動きが止まると、スナップ位置が選ばれます。パンは近接スナップ位置の中間で停止することもできます。
- 強制: 指を離す前に通過した最後のスナップ位置の直前または直後のスナップ位置が選ばれます (ジェスチャーの方向と速度によって異なります)。パンは強制スナップ位置で停止する必要があります。

パンのスナップ位置は、ページ付けされたコンテンツと同じ動作を実現したり、項目を論理的にグループ化して、ビューポートまたはディスプレイに収まるように動的に再グループ化できるようにしたりする、Web ブラウザーやフォト アルバムのようなアプリで便利です。

次の図は、特定の位置にパンして離すことでコンテンツを論理的な位置に自動的にパンする方法を示しています。

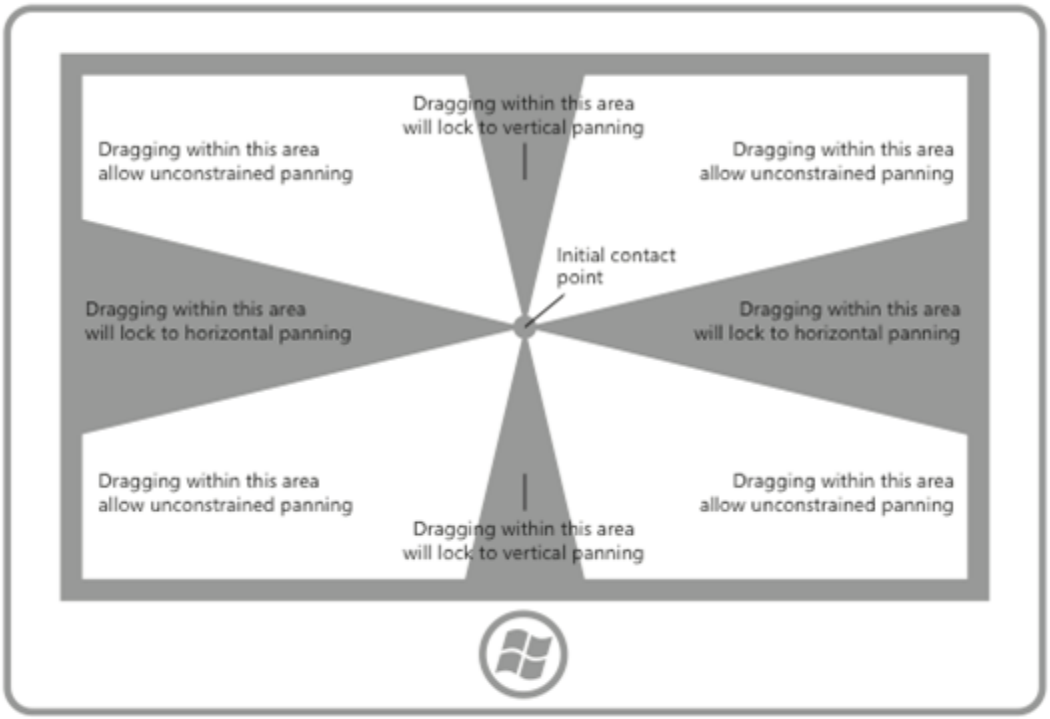


スワイプしてパンします。	タッチによる接触を離します。	パン対応領域は、タッチによる接触が離れた場所ではなく、スナップ位置で停止します。

レール

コンテンツは、ディスプレイ デバイスのサイズと解像度より広かったり高かったりする場合があります。このため、2 次元のパン (水平方向と垂直方向) が必要になることがよくあります。レールは、このような場合に動作の主軸 (垂直方向または水平方向) に沿ってパンを強調表示することで、ユーザー エクスペリエンスを向上させます。

次の図は、レールの概念を示しています。

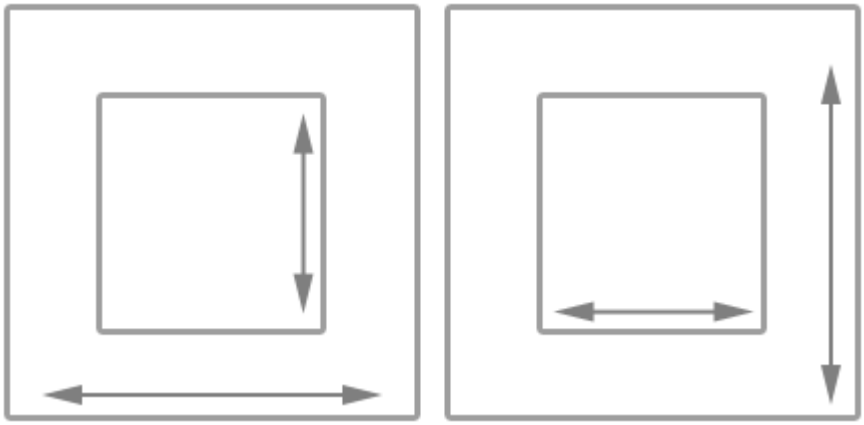


埋め込まれたコンテンツまたは入れ子になったコンテンツの連結

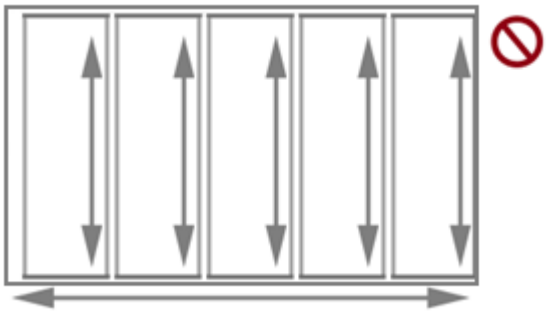
他のズーム可能またはスクロール可能な要素の入れ子になっている要素のズームまたはスクロールが限界に達した後で、親要素が子要素のズーム操作またはスクロール操作を継続して開始するかどうかを指定します。これはズームまたはスクロールのチェーンと呼ばれます。

1 つ以上の単一軸パン領域またはフリーフォーム パン領域が含まれる単一軸のコンテンツ領域内で (これらの子領域のいずれかでタッチによる接触があったときに) パンを行う場合は、連結を使います。子領域の特定の方向のパン境界に到達すると、親領域の同じ方向にパンがアクティブ化されます。

パン対応領域を別のパン対応領域内に入れ子にするときは、コンテナーと埋め込まれたコンテンツ間に十分な領域を指定することが重要です。次の図では、パン対応領域が別のパン対応領域内に置かれており、それぞれが相互に対して垂直方向に移動します。各領域にユーザーがパンできる十分な領域があります。



十分な領域がないと、次の図に示すように、埋め込まれたパン対応領域によってコンテナーでのパンが妨げられ、1 つ以上のパン対応領域で意図しないパンが発生する可能性があります。



このガイダンスは、たとえば、フォト アルバムや地図のようなアプリでも役に立ちます。各画像または地図内の制約のないパンをサポートしながら、アルバム内の前の画像または次の画像や詳細な領域への単一軸パンもサポートできます。フリーフォーム パンの画像や地図に対応する詳細領域またはオプション領域を提供するアプリでは、ページ レイアウトを詳細領域やオプション領域で始めることをお勧めします。画像や地図の制約のないパン領域が、詳細領域へのパンを妨げる可能性があるためです。

回転のガイドライン

このトピックでは、新しい Windows UI の回転について説明し、Windows アプリでこの新しい操作のメカニズムを使うときに考慮する必要があるユーザー エクスペリエンスのガイドラインを示します。

重要な API

Windows.UI.Input 名前空間　　Input 名前空間 (XAML)

推奨と非推奨

- ユーザーが直接 UI 要素を回転できるように回転を使います。

その他の使い方のガイダンス

回転の概要

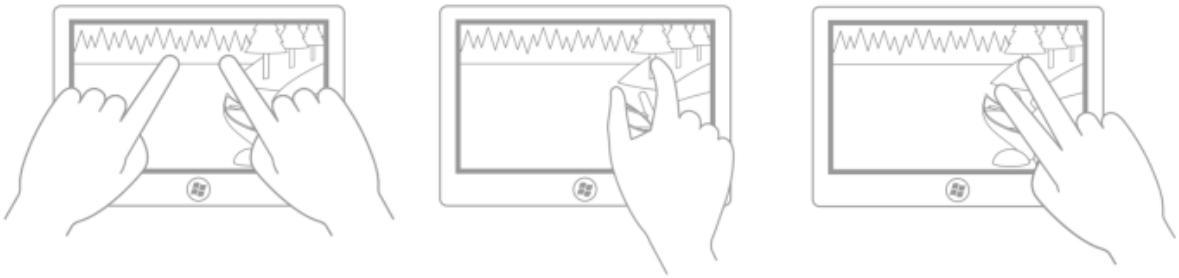
回転は Windows 8 以降の Windows アプリで使われるタッチ操作に最適な手法であり、ユーザーがオブジェクトを回転 (時計回りまたは反時計回り) できるようにします。

入力デバイスに応じて回転操作は次のように実行されます。

- マウスまたはアクティブなペン/スタイラスを使って、選んだオブジェクトの回転グリッパーを移動する。
- タッチまたはパッシブなペン/スタイラスを使って、回転ジェスチャーによって任意の方向にオブジェクトを回転させる。

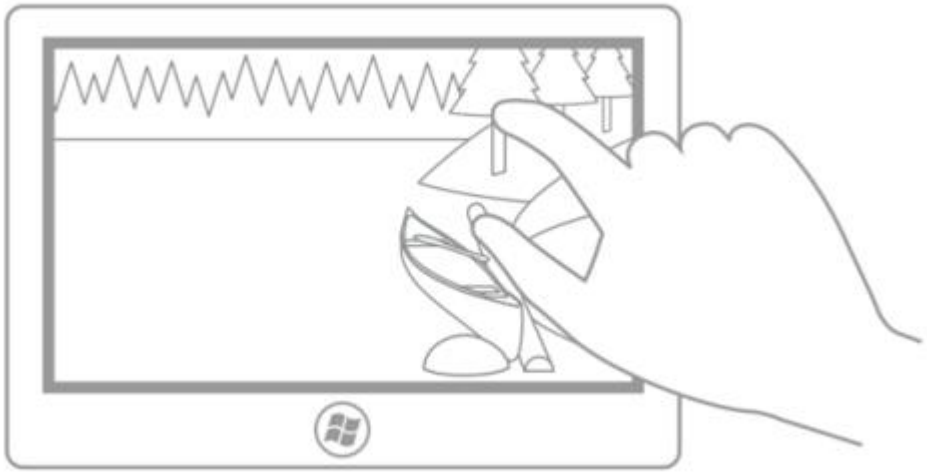
回転を使う状況

ユーザーが直接 UI 要素を回転できるように回転を使います。次の図は、サポートされる回転操作の指の配置をいくつか示しています。

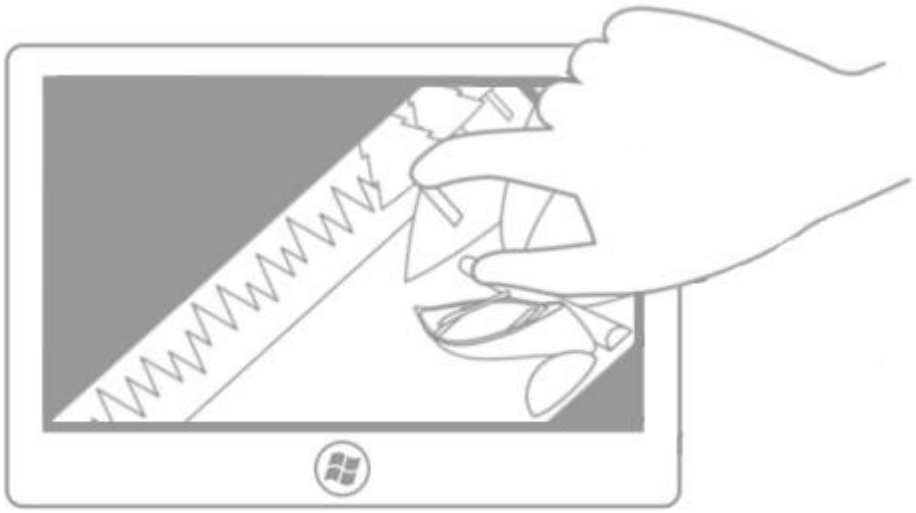


注 ユーザーが接触点とは無関係に回転の中心点を指定できる場合を除いて (例: 描画アプリやレイアウト アプリ)、直観に従い多くの場合、回転の中心点は 2 つのタッチした点のどちらかになります。以下の図では、回転の中心点がこのような制約を受けない場合に、どのようにユーザー エクスペリエンスが低下するかについて説明します。

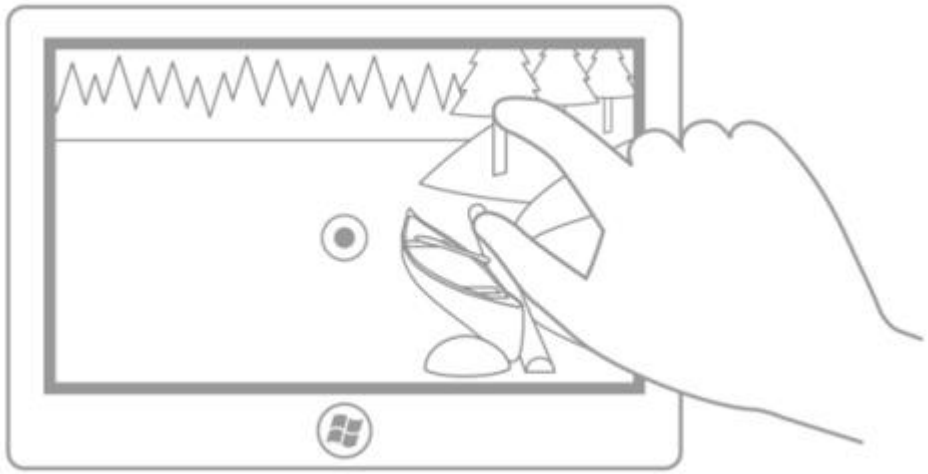
1 番目の図は、最初のタッチ ポイント (親指) と 2 番目のタッチ ポイント (人差し指) を示します。人差し指は木に、親指は丸太にタッチしています。



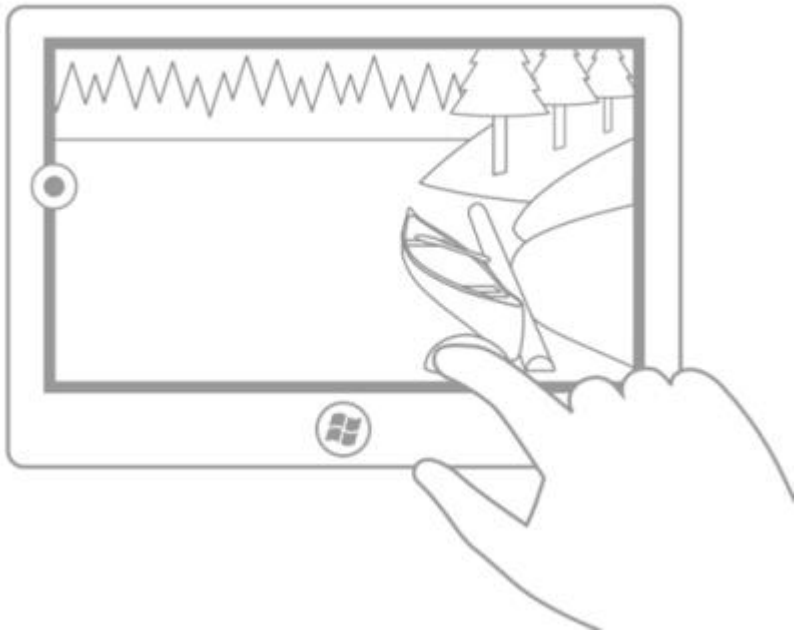
2 番目の図では、最初のタッチ ポイント (親指) の周りで回転が行われています。回転の後で、人差し指は相変わらず木の幹にタッチし、親指は相変わらず丸太 (回転の中心点) にタッチしています。



3 番目の図では、回転の中心がアプリによって絵の中心点に定義されています (またはユーザーによって設定されています)。回転の後で、絵が指の 1 つの周りで回転しなかったために、直接操作の画像が失われます (ユーザーがこの設定を選んだ場合を除きます)。



最後の図では、回転の中心がアプリによって絵の左端の中央の点に定義されています (またはユーザーによって設定されています)。この場合も、ユーザーがこの設定を選んだ場合を除いて、直接操作の画像が失われます。



Windows 8 以降 では、自由、制約付き、複合の 3 種類の回転をサポートします。

種類	説明
自由回転	自由回転では、ユーザーはコンテンツを 360°の任意の位置に自由に回転できます。ユーザーがオブジェクトを離すと、オブジェクトは選んだ位置にとどまります。自由回転は、Microsoft PowerPoint、Word、Visio、ペイントと Adobe Photoshop、Illustrator、Flash などの描画アプリやレイアウト アプリで便利です。
制約付き回転	制約付き回転は、操作中は自由回転をサポートしますが、離したときに 90°単位のスナップ位置が強制されます (0、90、180、270)。ユーザーがオブジェクトを離すと、オブジェクトは自動的に最も近いスナップ位置まで回転します。 制約付き回転は回転の最も一般的な方法で、コンテンツのスクロールと同じように機能します。スナップ位置があることで、ユーザーは操作が正確でなくても目標の位置に到達できます。制約付きの回転は Web ブラウザーやフォト アルバムのようなアプリで便利です。
複合回転	複合回転は自由回転をサポートしますが、(パンにおけるレールのように) 90°単位のスナップ位置のゾーンでは制約付き回転によって強制されます。ユーザーが各 90°のゾーンの外でオブジェクトを離した場合にはオブジェクトはその位置にとどまりますが、それ以外の場合にはオブジェクトは自動的にスナップ位置まで回転します。 **注** ユーザー インターフェイスのレールは、ターゲットの周辺の領域において、特定の値または位置に向けて動きが制約され選択に影響を与える機能です。

テキストと画像の選択のガイドライン

このトピックでは、テキスト、画像、コントロールを選んだり操作したりするための新しい Windows UI について説明します。また、Windows アプリでこの新しい選択と操作のメカニズムを使うときに考慮する必要のあるユーザー エクスペリエンスに関するガイドラインを示します。

重要な API

Windows.UI.Input 名前空間　　Input 名前空間 (XAML)

推奨と非推奨

- 独自のグリッパー UI を実装する場合は、フォント グリフを使います。グリッパーは、システム全体で利用できる 2 つの Segoe UI フォントを組み合わせたものです。フォント リソースを使うと、さまざまな dpi におけるレンダリングの問題が軽減され、さまざまな UI 表示スケール プラトーに対応できます。独自のグリッパーを実装する場合は、どのグリッパーにも次の UI の特性を持たせてください。
  - 円の図形
  - 背景に隠れず表示される
  - 一貫したサイズ
- グリッパー UI が収まるように、選択可能なコンテンツの周囲に余白を設けます。パンとスクロールに対応していない領域でテキストを選択できる場合は、テキスト領域の左右に 1/2 のグリッパー余白を設け、テキスト領域の上下に 1 つのグリッパーの高さを設けます (次の図を参照)。こうすることで、グリッパー UI 全体がユーザーに表示されるため、エッジ (端)に基づく他の UI が誤って操作されるのを防ぐことができます。

- 対話操作中はグリッパー UI を非表示にします。対話操作中にグリッパーによって領域がふさがれないようにします。 これは、グリッパーが指で完全に隠れない場合や、テキスト選択グリッパーが複数存在する場合に有効です。子ウィンドウを表示しているときは、ビジュアルなアーティファクトを取り除きます。
- コントロール、ラベル、画像、独自のコンテンツなどの UI 要素は選択できないようにします。通常、Windows アプリケーションでは、特定のコントロール内でのみ選択できます。ボタン、ラベル、ロゴなどのコントロールは選択できません。JavaScript を使った Windows アプリでは、選択を無効にする必要があります。選択がアプリにとって問題になるかどうかを評価し、問題になる場合は、選択を禁止する UI 領域を特定します。

その他の使い方のガイダンス

テキストの選択と操作は、タッチ操作で導入されたユーザー エクスペリエンスの問題の影響を特に受けやすくなっています。マウス、ペン/スタイラス、キーボード入力は非常に細かく制御されます。1 回のマウスのクリックまたはペン/スタイラスの接触は 1 ピクセルにマッピングされ、キーは押されるか、放されます。 タッチ入力は細かく制御されません。指先の表面全体を、画面上の特定の x-y の位置にマッピングしてテキスト キャレットを正確に配置することは困難です。

考慮実行と推奨事項

Windows 10 の言語フレームワークによって公開されるビルトイン コントロールを利用して、選択や操作の動作など、完全なプラットフォームのユーザー操作エクスペリエンスを実現するアプリを作成してください。ビルトイン コントロールの対話操作の機能は、大部分の Windows アプリにとって十分なものです。

標準の Windows 8 以降 テキスト コントロールを使う場合、このトピックで説明した選択の動作とビジュアル効果はカスタマイズできません。

テキスト選択

アプリにテキストの選択をサポートするカスタム UI を実装する必要がある場合は、ここで説明する Windows 10 の選択動作に従うことをお勧めします。

編集可能なコンテンツと編集不可のコンテンツ

タッチ操作の精度の低いターゲット設定動作を受け入れるために、選択と操作のユーザーエクスペリエンスのビジュアル効果 (選択が調整可能でかつ操作ターゲットを特定する "グリッパー" など) が Windows 10 (正確には Windows 8 以降) 用に完全に再設計されています。

タッチでは、選択操作は主に挿入カーソルの設定や単語の選択を行うタップ、選択範囲の変更を行うスライドなどのジェスチャーを通じて実行されます。他の Windows 10 タッチ操作と同様に、時間制限のある対話操作は情報 UI を表示するための長押しジェスチャーに制限されます。詳しくは、「ビジュアルなフィードバックのガイドライン」をご覧ください。

Windows 8 以降でマウス、ペン/スタイラス、キーボードを使う場合は、選択の動作に変わりはありません。

Windows 8 以降では、選択操作のために "編集可能" と "編集不可" の 2 つの状態が認識され、その状態に合わせて選択 UI、フィードバック、機能が調整されます。

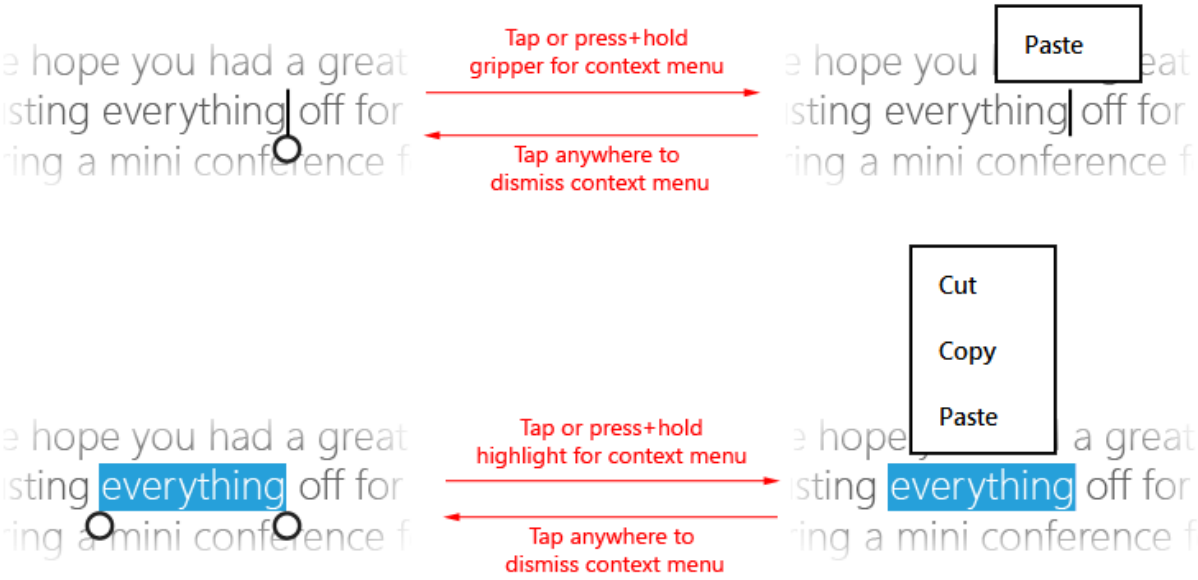
編集可能なコンテンツ

単語内の左半分をタップすると、単語のすぐ左にカーソルが配置されます。単語内の右半分をタップすると、単語のすぐ右にカーソルが配置されます。

次の図は、単語の先頭または終わりの近くでタップして、グリッパーを持つ最初の挿入カーソルを配置する方法を示しています。

次の図は、グリッパーをドラッグして選択範囲を調整する方法を示しています。

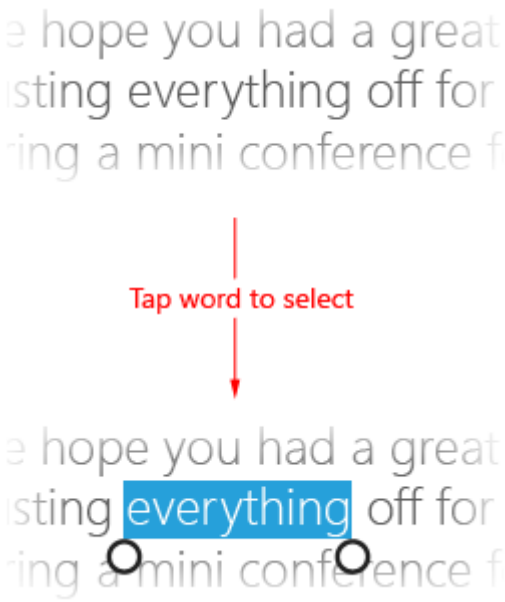
次の図は、選択範囲内またはグリッパー上でタップしてコンテキスト メニューを呼び出す方法を示しています (長押しを使うこともできます)。



注 これらの対話的操作は、綴りに間違いのある単語の場合は若干異なります。綴りに誤りがあるとしてマークされている単語をタップすると、単語全体が強調表示されて、スペル候補のコンテキスト メニューが呼び出されます。

編集不可のコンテンツ

次の図は、単語内でタップして単語を選ぶ方法を示しています (最初の選択にスペースは含まれていません)。



編集可能なテキストと同じ手順に従って、選択範囲を調整し、コンテキスト メニューを表示します。

オブジェクトの操作

Windows アプリでカスタム オブジェクト操作を実装する場合は、できる限り、テキストの選択と同じ (類似する) グリッパー リソースを使います。そうすれば、プラットフォーム間で操作エクスペリエンスの一貫性が保たれます。

たとえば、次の図に示すように、グリッパーは、サイズ変更とトリミングをサポートする画像処理アプリや、調節可能なプログレス バーを備えたメディア プレーヤー アプリでも利用できます。

調節可能なプログレス バーを備えたメディア プレーヤーです。

トリミング グリッパーが表示された画像エディターです。

ターゲット設定のガイドライン

Windows のタッチ補正では、タッチ デジタイザーで検出されるそれぞれの指が接触する領域全体を使います。デジタイザーから伝えられる、より広く複雑なこの入力データのセットを使うと、ユーザーが意図した (または意図した可能性が高い) ターゲットをより正確に特定できます。

このトピックでは、タッチ補正のための接触形状の使用について説明し、Windows ランタイム アプリでのターゲット設定のベスト プラクティスを紹介します。

重要な API

Windows.UI.Core 名前空間
Input 名前空間 (XAML)
Windows.UI.Input 名前空間

サイズと表示スケール

画面サイズとピクセル密度が変わっても一貫性が維持されるように、ターゲット サイズはすべて物理単位 (ミリメートル) で表されます。物理単位は、次の式でピクセルに変換できます。

ピクセル数 = ピクセル密度 × サイズ

次の例ではこの式を使って、135 ppi (pixel per inch) のディスプレイと 1x の表示スケール プラトーでの 9 mm ターゲットのピクセル サイズを計算します。

ピクセル数 = 135 ppi × 9 mm

ピクセル数 = 135 ppi × (0.03937 インチ/mm × 9 mm)

ピクセル数 = 135 ppi × 0.35433 インチ

ピクセル数 = 48 ピクセル

この結果は、システムで定義されている各表示スケール プラトーに従って調整する必要があります。

しきい値

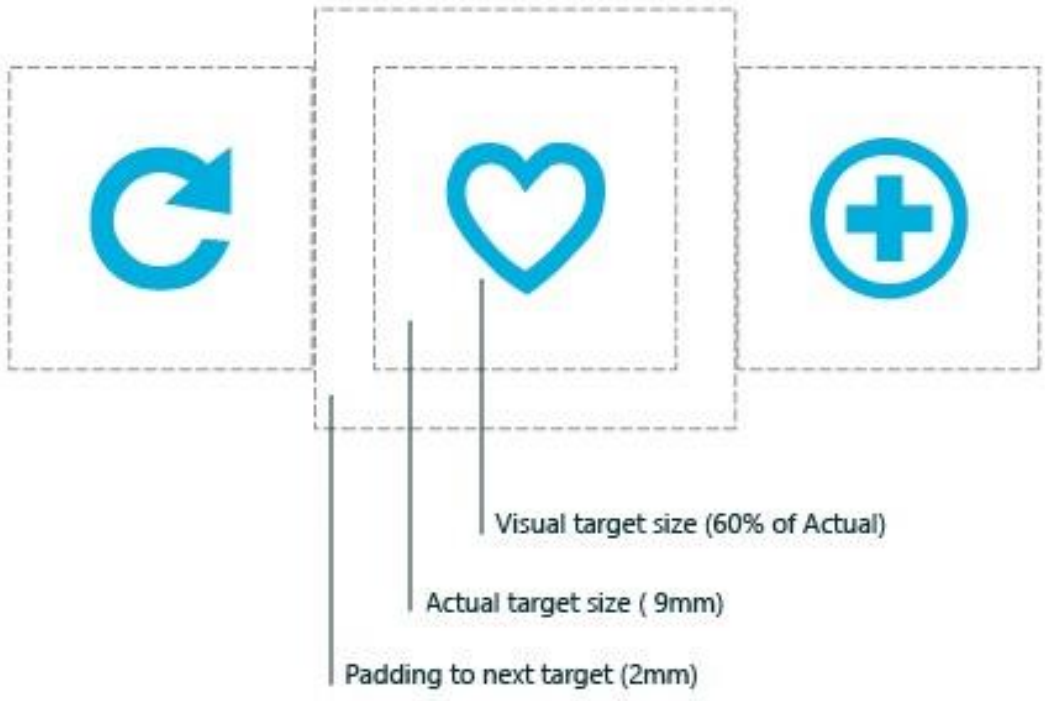
操作の結果を判断するために距離と時間のしきい値を使うことがあります。

たとえば、タッチ ダウンが検出されたとき、オブジェクトがタッチ ダウン ポイントから 2.7 mm 未満の範囲でドラッグされて、タッチ ダウンから 0.1 秒以内に指が上げられた場合は、タップが登録されます。この 2.7 mm のしきい値を超えて指を動かすと、オブジェクトはドラッグされ、選択または移動されます (詳しくは、「クロススライドのガイドライン」をご覧ください)。アプリによっては、0.1 秒より長く押し続けると自己説明操作が実行されることもあります (詳しくは、「ビジュアルなフィードバックのガイドライン」をご覧ください)。

ターゲット サイズ

一般に、タッチ ターゲットのサイズを一辺 9 mm 以上の正方形に設定します (1.0x のスケール プラトーで 135 ppi のディスプレイで 48x48 ピクセル)。一辺 7 mm 未満の正方形であるタッチ ターゲットを使わないようにしてください。

次の図は、ターゲット サイズが一般には外観上のターゲット、実際のターゲット サイズ、実際のターゲットと他の指定可能なターゲットの間の余白の組み合わせで決まることを示しています。



タッチ ターゲットのすべてのコンポーネントの最小サイズと推奨サイズを次の表に示します。

ターゲット コンポーネント	最小サイズ	推奨サイズ
余白	2 mm	該当なし
外観上のターゲット サイズ	実際のサイズの 60% 未満	実際のサイズの 90 から 100% ほとんどのユーザーは一辺 4.2 mm の正方形 (7 mm の推奨の最小ターゲットのサイズの 60%) よりも小さい場合、外観上のターゲットがタッチ可能であると実感しません。
実際のターゲット サイズ	7 x 7 mm の正方形	一辺 9 mm 以上の正方形 (48 x 48 ピクセル @ 1x)
全体的のターゲット サイズ	11 x 11 mm (約 60 ピクセル: 3 個の 20 ピクセル グリッド @ 1x)	13.5 x 13.5 mm (72 x 72 ピクセル @ 1x) これは、実際のターゲットと余白を組み合わせたサイズをそれぞれの最小サイズより大きくする必要があることを意味します。

表に示したターゲット サイズの推奨サイズは、個々のシナリオの必要に応じて調整できます。この推奨サイズの設定では、次の点が考慮されています。

- タッチの頻度: 繰り返しタッチされたり、頻繁にタッチされたりするターゲットは、最小サイズより大きくするようにしてください。
- エラー防止: 誤ってタッチすると重大な結果をもたらすターゲットは、大きな余白を取り、コンテンツ領域の端から離して配置する必要があります。特に当てはまるのは頻繁にタッチされるターゲットです。
- コンテンツ領域での位置
- フォーム ファクターと画面サイズ
- 指の位置
- タッチのビジュアル エフェクト

- ハードウェアとタッチ デジタイザー

ターゲット設定支援

Windows では、ここで示した最小サイズや推奨する余白サイズを適用できない状況に対応するためのターゲット設定支援機能を提供しています。対象となるのは、Web ページ上のハイパーリンク、カレンダー コントロール、ドロップダウン リストとコンボ ボックス、テキスト選択などです。

このようにターゲット設定プラットフォームを強化し、ユーザー インターフェイスの動作をビジュアルなフィードバック (不明瞭解消 UI) と連携させることで、ユーザーがより正確に、また安心して操作できるようになります。 詳しくは、「ビジュアルなフィードバックのガイドライン」をご覧ください。

タッチ可能な要素を推奨の最小ターゲット サイズより小さくする必要がある場合は、次のテクニックを使ってターゲット設定で発生する問題を最小化できます。

テザー

テザーとは、入力接点がオブジェクトに直接触れていなくても、ユーザーがそのオブジェクトにつながっているか操作していることをユーザーに示すために使われる視覚的な合図 (接点からオブジェクトの境界の四角形までを結ぶコネクタ) です。テザーは次の場合に使われます。

- タッチによる接触がオブジェクトまでの近接しきい値の範囲内で最初に検出され、そのオブジェクトがそのタッチのターゲットとして最も可能性が高いと特定された場合。
- タッチによる接触がオブジェクトから離れたが、その接触が近接しきい値内にとどまっている場合。

この機能は、JavaScript を使った Windows アプリの開発者には公開されていません。

スクラブ

スクラブとは、ターゲットのフィールド内をタッチし、そのまま指を持ち上げずに目的のターゲットまでスライドさせてそのターゲットを選ぶことを意味します。 この操作は "指を離すことによるアクティブ化" とも呼ばれます。この場合、アクティブ化されるオブジェクトは、指が画面から離れたときに最後にタッチされたオブジェクトです。

スクラブ操作を設計するときは、次のガイドラインに従ってください。

- スクラブは、不明瞭解消 UI と併用します。詳しくは、「ビジュアルなフィードバックのガイドライン」をご覧ください。
- スクラブ対象のタッチ ターゲットの推奨最小サイズは、20 ピクセル (3.75 mm @ 1x サイズ) です。
- スクラブは、Web ページなどのパン対応サーフェイスで実行されたときに優先されます。
- スクラブ対象ターゲットは互いに近づける必要があります。
- ユーザーがドラッグによってスクラブ対象ターゲットから指を離すと、操作は取り消されます。
- ターゲットによって実行される操作が破棄的でなければ (カレンダーでの日付の切り替えなど)、スクラブ対象ターゲットに対してテザーが指定されます。
- テザーは、水平、垂直のいずれかの方向で指定されます。

ビジュアルなフィードバックのガイドライン

ビジュアルなフィードバックは、対話操作が検出、解釈、処理されていることをユーザーに示すために使います。 ビジュアル的なフィードバックは、対話操作を促進することによってユーザーを支援します。対話操作の成功を示すことによって、ユーザーのコントロール感を向上させます。また、システム状態の中継やエラーの削減も可能になります。

重要な API

Windows.Devices.Input 名前空間

Windows.UI.Core 名前空間

Windows.UI.Input 名前空間

推奨と非推奨

- たとえ接触時間が短くても、ビジュアルなフィードバックを返す必要があります。理由は次のとおりです。
  - タッチ スクリーンが機能していることを示すため。
  - ターゲットがタッチに対応しているか、応答できるかを示すため。
  - タッチしたところが目的のターゲットから外れているかどうかを示すため。
- すべての操作イベントに対してフィードバックをすぐに表示します。
- ユーザーの注意をそらさない、繊細かつ直感的なフィードバックを提供します。
- すべての操作において、タッチ ターゲットが指先から離れないようにします。
- パンの方向が 1 方向に制限されている場合は、スワイプ ジェスチャーを使った項目の選択を有効にします。
- タッチのビジュアル エフェクトがアプリの使用を妨げる可能性がある場合は、使わないでください。
- どうしても必要な場合以外は、フィードバックを表示しないでください。その場所でしか意味がない場合を除き、ビジュアルなフィードバックを表示せず UI をすっきりさせてください。既に表示されているテキストをさらに表示することになる場合は、ツールチップを表示しないでください。ツールチップは、項目を選択するときにテキストが途中までしか表示されていない (テキストに省略記号が付いている) 場合や、アプリの理解や使用に追加情報が必要な場合など、特定の場面でのみ使います。
- 情報提供型 UI 以外には長押しジェスチャーを使わないでください。

重要 水平方向と垂直方向のパンが有効である場合は、長押しを選択に使うことができます。

- Windows 8 以降の組み込みジェスチャーのビジュアルなフィードバックの動作をカスタマイズしないでください。この動作をカスタマイズすると、ユーザー エクスペリエンスに一貫性がなくなり、混乱する可能性があります。
- パンやドラッグの操作中はビジュアルなフィードバックを返さないでください。画面上のオブジェクトの動きだけで十分です。ただし、コンテンツ領域がパンまたはスクロールしない場合は、ビジュアル エフェクトで境界条件を示します。詳しくは、「パンのガイドライン」をご覧ください。
- ターゲットとして識別されないコントロールにはフィードバックを表示しないでください。位置に基づく正確性が求められる操作をタッチ入力で行う場合は、ビジュアルなフィードバックが重要です。タッチ入力が検出されるたびにフィードバックが表示されるようにすると、ユーザーは、アプリとそのコントロールで定義されたカスタム ターゲット設定ヒューリスティックを把握できます。
- 特定の種類の入力に対するフィードバック動作を、別の種類の入力に使わないでください。たとえば、キーボード フォーカスの四角形はキーボード入力のみに使い、タッチ操作には使わないでください。

その他の使い方のガイダンス

正確性が求められるタッチ操作では、接触のビジュアル エフェクトが特に重要です。たとえば、アプリでタップの位置を正確に示し、対象から外れていたかどうか、どの程度外れていたか、合わせるにはどうすればよいかをユーザーが把握できるようにしなければなりません。

Windows アプリの言語フレームワーク (JavaScript を使った Windows アプリ、C++、C#、Visual Basic を使った Windows アプリ) を通じて公開されるプラットフォーム コントロールを利用すると、Windows 8 以降のビジュアル エフェクトを標準機能として実装できます。カスタマイズされたフィードバックが必要なカスタム操作をアプリに実装する場合は、適切なフィードバックを実装し、入力デバイス間の連絡を取り、ユーザーがタスクに集中できるようにする必要があります。このことは、ビジュアルなフィードバックが重要な UI と競合したり、重要な UI を隠したりする可能性があるゲームや描画アプリでは特に重要です。

重要　組み込みジェスチャーの操作の動作を変更することはお勧めしません。

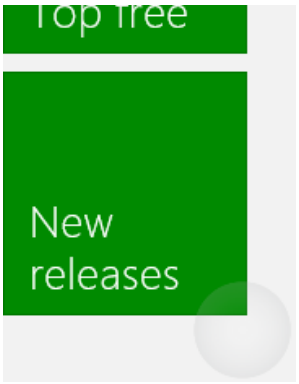
フィードバック UI

フィードバック UI は、一般に入力デバイス (タッチ、タッチパッド、マウス、ペン/スタイラス、キーボードなど) に依存します。たとえば、マウスの組み込みフィードバックには、通常はカーソルの移動と変化が伴います。一方、タッチとペンの場合は接触のビジュアル エフェクトが必要です。キーボードによる入力とナビゲーションの場合は、フォーカス用の四角形と強調表示を使います。

プラットフォーム ジェスチャーのフィードバック動作を設定するには、ShowGestureFeedback を使います。

フィードバック UI をカスタマイズする場合は、すべての入力モードをサポートした適切なフィードバックを提供してください。

Windows 8 以降には、次のような接触のビジュアル エフェクトが組み込まれています。



タッチのビジュアル エフェクト

マウス/タッチパッドのビジュアル エフェクト

ペンのビジュアル エフェクト

キーボードのビジュアル エフェクト

情報提供型 UI(ポップアップ)

ビジュアルなフィードバックの主な形態の 1 つに、情報提供型 UI (不明瞭解消型 UI) があります。 情報提供型 UI は、オブジェクトに関する情報を識別および表示し、その機能の説明と使用方法を示します。さらに、必要に応じてガイダンスも提供します。

次に、Windows アプリでサポートされているさまざまな情報提供型 UI の種類を示します。

- ツールチップ
- リッチ ツールチップ
- メニュー
- メッセージ ダイアログ
- フライアウト

情報提供型 UI は、指先によるオクルージョン (干渉) を克服し、アプリのタッチ操作を向上させるうえで特に有用です。長押しというこの UI 専用の組み込みジェスチャーも用意されています。

長押しは時間制限のある操作であり、一般に Windows 8 以降では推奨されていません。学習や調査に役立つため、この場合は時間制限のある操作を使ってもかまいません。推奨される時間は情報提供型 UI の種類によって変わります。推奨される時間のしきい値を次に示します。

情報提供型 UI の種類	タイミング	アクティブ化	用途
オクルージョン ツールチップ (スクラブ操作と小さなターゲット用)	0 ミリ秒	あり	操作についてすばやく説明するために使います。 通常はコマンドに使います。
オクルージョン ツールチップ(操作用)	200 ミリ秒	あり	
リッチ ツールチップ	2000 ミリ秒以下	なし	比較的低速で意図的な調査や学習に使います。 通常はコレクション項目に使います。
インストラクショナル	2000 ミリ秒以下	なし	
コンテキスト メニュー	2000 ミリ秒以下	なし	選択したオブジェクトに関連する限られたコマンドのセットを表示します。
フライアウト	2000 ミリ秒以下	なし	選択したオブジェクトに関連する限られたコマンドのセットを表示します。

ツールチップ

ツールチップを使ってコントロールに関する詳しい情報を表示してから、ユーザーに操作を実行するように求めます。

ツールチップ (Tooltip) は、ユーザーがコントロールまたはオブジェクト上で長押しジェスチャーを行った (またはホバー イベントが検出された) ときに自動的に表示されます。接触またはカーソルがコントロールやオブジェクトの外に移動すると消えます。ツールチップにはテキストや画像を含めることができますが、対話的な操作はできません。

小さなターゲット用のオクルージョン ツールチップ

オクルージョン ツールチップは、隠れたターゲットについて説明するために使います。Web ページ上のハイパーリンクなど、標準のタッチ ターゲット サイズよりも小さな項目をターゲット設定してアクティブ化するときに、これらのツールチップが役立ちます。

これらのツールチップは、一定の時間しきい値の経過後に情報ポップアップに変えることができます。 たとえば、オクルージョン ツールチップを使って、隠れたハイパーリンク テキストを示した後で、このツールチップを URL を含むポップアップに変えることができます。

操作とコマンド用のオクルージョン ツールチップ

これらのツールチップは、ユーザーが要素から指を離したときに発生するアクションまたはコマンドについて説明するために使います。ボタンやそれに似たコントロールをターゲット設定してアクティブ化するときに、これらのツールチップが役立ちます。

小さなターゲットのツールチップは、一定の時間しきい値の経過後にアクションのツールチップに変えることができます。 この場合、小さなターゲットのツールチップに情報を追加してアクションのツールチップにします。

リッチ ツールチップ

これらのツールチップでは、要素に関する補助的な情報を提供します。たとえば、画像の説明、途中で切れているタイトルの完全なテキストなどの、ターゲットに関する情報です。

一般的に、リッチ ツールチップには、すぐには必要でない情報や表示のタイミングが早すぎると煩わしいような情報を含めます。 時間のしきい値を長くすると、情報を得ようとするユーザーの意図が強まります。

リッチ ツールチップが表示された後にユーザーが指を離すと、オブジェクトはアクティブでなくなります。 ツールチップから情報を得たユーザーが、その項目をアクティブ化しようと思わない場合もあるためです。

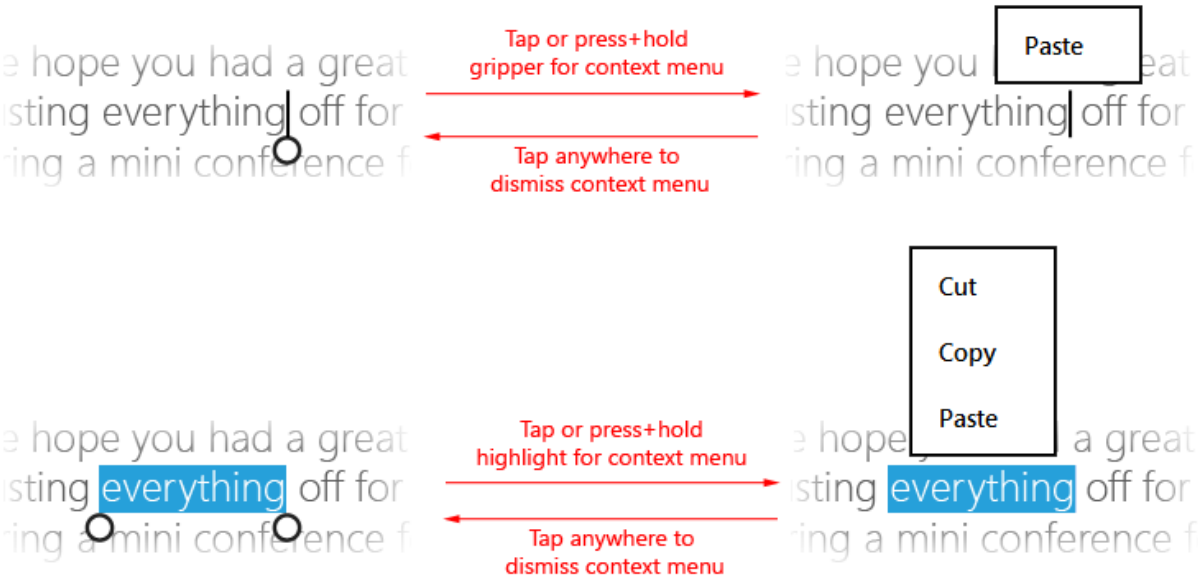
リッチ ツールチップのビジュアル デザインと情報は、標準のツールチップと区別しやすく、充実した内容にすることをお勧めします。

コンテキスト メニュー

コンテキスト メニュー (PopupMenu) は、ユーザーが Windows ストア アプリ内のテキストまたは UI オブジェクトに対する操作 (クリップボード コマンドなど) をすばやく実行できるようにするための簡易メニューです。

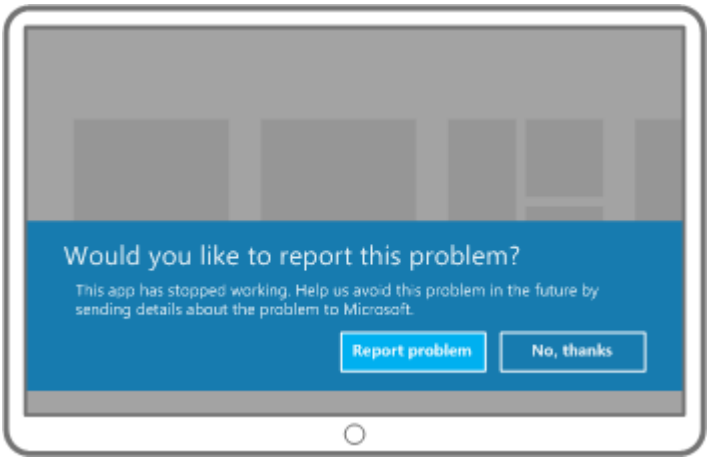
タッチ操作のメリットを活かしたコンテキスト メニューは、2 つの部分で構成されます。長押しによってビジュアルな合図 (ツールチップ) が表示されます。その後、ツールチップが消えてユーザーが指を離すと、コンテキスト メニュー自体が表示されます。

次の図は、選択範囲内またはグリッパー上でタップしてテキストの既定のコンテキスト メニューを呼び出す方法を示しています (長押しを使うこともできます)。



メッセージ ダイアログ

処理を続行する前に、ユーザーの操作やアプリの状態に基づいてユーザーに応答を求めるには、メッセージ ダイアログ (MessageDialog) を使います。明示的なユーザー操作が必須であり、ユーザーが応答するまで、アプリへの入力はブロックされます。



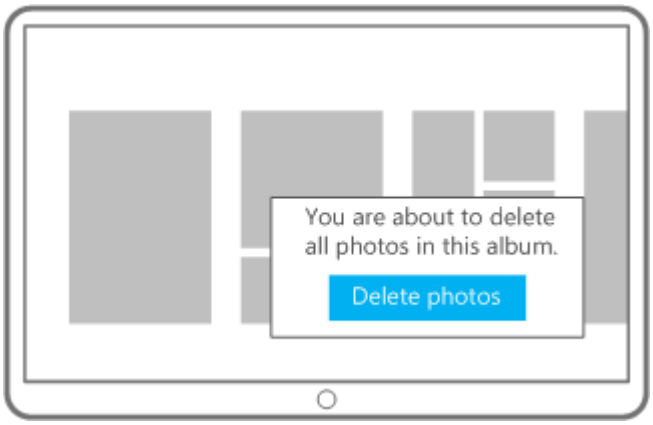
次に、メッセージ ダイアログを表示する一般的な理由をいくつか示します。

- 緊急の情報を提示する
- 実行を継続する前に質問する
- エラー メッセージを表示する

フライアウト

フライアウト (Flyout) は、タップ、クリック、またはその他のアクティブ化によって表示される軽量の UI パネルです。現在のアクティビティに関連する情報、質問、またはオプションのメニューをユーザーに表示するために使われます。簡易非表示にすることができます (ユーザーがフライアウト パネルの外部をタッチまたはクリックするか、Esc キーを押すと消えます)。つまり、フライアウトを無視できます。

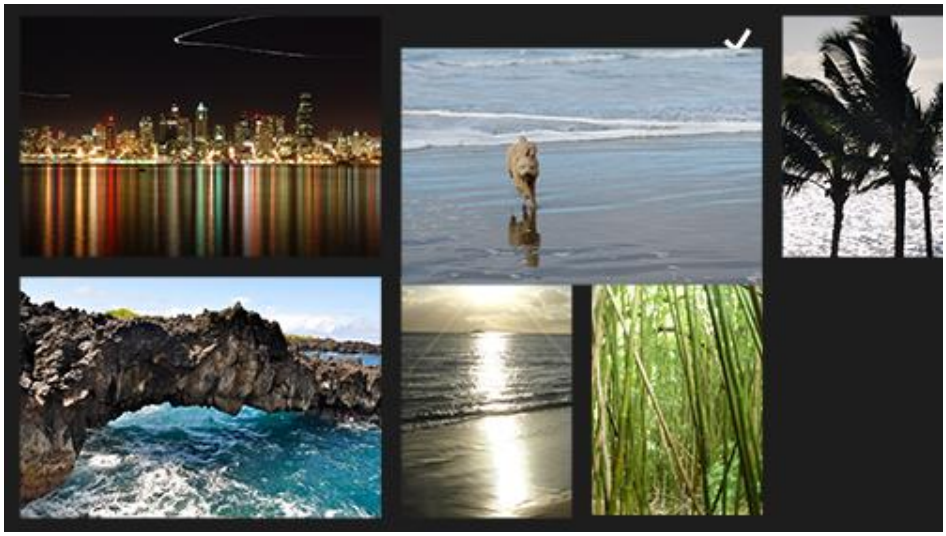
ツールチップとは異なり、フライアウトは入力を受け取ることができます。メッセージ ダイアログとは異なり、アプリはアクティブなままで、入力を受け取ります。



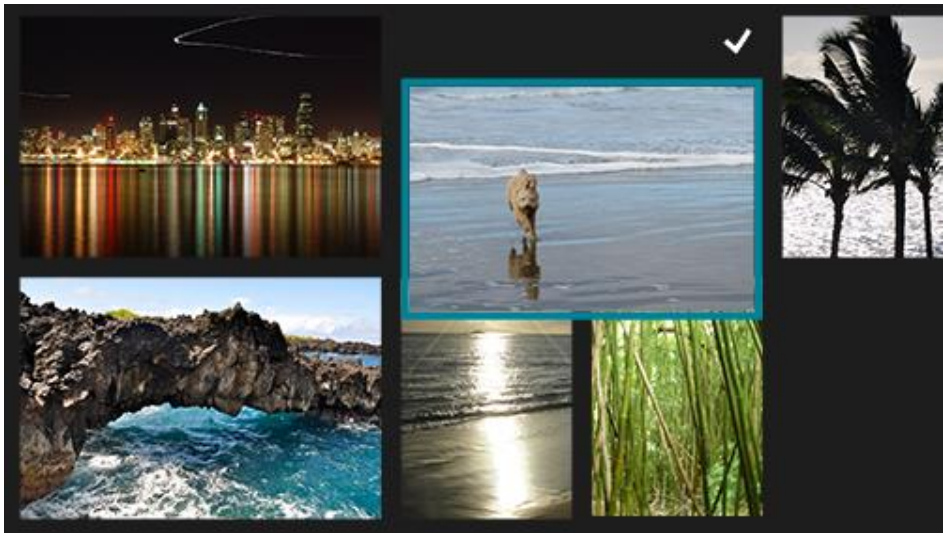
自己表明 UI

自己表明操作とは、ターゲット オブジェクトの操作方法を説明し、操作結果のプレビューを提供する、情報量の多いビジュアルな合図またはアニメーションです。

次のいくつかの画像は、スタート画面でのクロススライド選択の自己表明操作を示しています。ユーザーがアプリのタイルに (ドラッグしないで) タッチすると、タイルが (ドラッグされたかのように) 下方向へスライドし、アプリを実際に選んだ場合に表示される選択チェック マークが現れます。



指で項目を押すと、選択内容に対する自己表明が始まります。自己表明により、項目に対して実行される操作が示されます。



指を離さずに、スワイプして実際に項目を選択します。

ユーザーがそのまま指をスライドすると、自己表明のビジュアル エフェクトが変わり、オブジェクトを移動できるようになったことが示されます。

自己表明が表示された後にユーザーが指を離すと、オブジェクトはアクティブでなくなります。

ファイル、データ、接続のガイドライン

このセクションでは、アプリからファイルやデータにアクセスする方法と、ユーザーを Web へ接続する方法について説明します。

このセクションの内容

トピック	説明
カスタム データ形式	ユーザーは、さまざまな情報をオンラインで共有します。成功するアプリを作成するには、ユーザーが共有する可能性が高い情報の種類を分析し、その情報を受信側のアプリが正しく処理できるようにパッケージ化します。ユーザーが共有する情報は、多くの場合、Windows でサポートされている 6 つの標準形式のいずれかに当てはまります。しかし、より的を絞ったデータ型を使うことによってユーザー エクスペリエンスを向上させることができる場合もあります。そのような場合は、アプリでカスタム データ形式をサポートすることができます。
ファイルピッカー	アプリは、ファイル ピッカーを使って、ファイルとフォルダーにアクセスし、ファイルを保存できます。
ファイルピッカー コントラクト	他のアプリにアプリのコンテンツ、保存場所、ファイルの更新へのアクセスを提供するために、ファイル オープン ピッカー コントラクト、ファイル保存ピッカー コントラクト、キャッシュ ファイル アップデーター コントラクトに参加しているアプリのファイル ピッカーをカスタマイズするには、次のガイドラインに従ってください。
ファイルの種類と URI	ユニバーサル Windows プラットフォーム (UWP) アプリと、アプリがサポートするファイルの種類やプロトコルの間の関係を理解すると、より一貫性があり、洗練されたエクスペリエンスをユーザーに提供できます。
印刷	アプリでコンテンツの印刷を可能にする場合は、以下のガイドラインに従ってください。
近接通信	このトピックでは、近接通信を使ってアプリを接続し、コンテンツを共有する際のベスト プラクティスについて説明します。
サムネイル	このガイドラインでは、サムネイル イメージを使って、UWP アプリでファイルをプレビューする方法について説明します。

カスタム データ形式の作成のガイドライン

ユーザーは、さまざまな情報をオンラインで共有します。成功するアプリを作成するには、ユーザーが共有する可能性が高い情報の種類を分析し、その情報を受信側のアプリが正しく処理できるようにパッケージ化します。ユーザーが共有する情報は、多くの場合、Windows でサポートされている 6 つの標準形式のいずれかに当てはまります。しかし、より的を絞ったデータ型を使うことによってユーザー エクスペリエンスを向上させることができる場合もあります。そのような場合は、アプリでカスタム データ形式をサポートすることができます。

重要な API

DataPackege クラス	Windows.ApplicationModel.DataTransfer 名前空間

例

ここでは、カスタム形式の作成についての考え方をよりわかりやすく示すために、次の例について検討します。

Fabrikam という架空の会社の開発者が、オンラインに保存されているファイルを共有するためのアプリを作成します。1 つのオプションとしてストリーム方式の StorageItems を使う方法が考えられますが、ターゲット アプリでファイルを読むためにはファイルのダウンロードが必要になるため、時間がかかり、非効率です。そのため、それらのファイルの種類で使うためのカスタム形式を作成することにしました。

まず、新しい形式の定義について検討します。この場合、新しい形式は、オンラインに保存されているファイルの種類 (ドキュメント、画像など) のコレクションです。これらのファイルはローカル コンピューターではなく Web にあるため、形式の名前は WebFileItems にすることにしました。

次に、その形式の詳細について決定する必要があります。次のように決定されました。

- この形式は、URI を表す InspectableArray を含む **IPropertyValue** で構成されます。

- この形式には、少なくとも 1 つの項目が含まれている必要がありますが、含めることのできる項目の数に制限はありません。
- 任意の有効な URI を使うことができます。
- ソース アプリケーションの境界の外からはアクセスできない URI (認証される URI など) は推奨されません。

これで、カスタム形式を作成して使用するための十分な情報が集まりました。

アプリでカスタム形式を使うかどうか

Windows 10 の共有機能では、次の 6 つの標準データ形式がサポートされています。

- Text
- HTML
- Bitmap
- StorageItems
- URI
- RTF

これらの形式は汎用性が高いため、共有コンテンツの共有と受信をアプリですばやく簡単にサポートすることができます。その一方で、受信側のアプリに対してデータの豊富な説明が必ずしも提供されないという欠点もあります。たとえば、住所を表す次のような文字列があったとします。

1234 Main Street, New York, NY 98208

アプリでこの文字列を共有するには、**DataPackage.setText** を使います。しかし、受信側のアプリでは、この文字列が何を表すのか正確にはわからないため、このデータを使ってできることは限られます。カスタム データ形式を使うと、共有されるデータを、http://schema.org/Place 形式を使った "場所" としてソース アプリで定義することができます。これにより、受信側のアプリに追加情報が提供されて、ユーザーの期待どおりに情報を処理できるようになります。既にあるスキーマ形式を使うと、定義されている形式のより大きなデータベースにアプリをフックできます。

カスタム形式は、データをより効率的に共有できるようにするために使うこともできます。たとえば、画像のコレクションを Microsoft OneDrive に保存しているユーザーが、その一部をソーシャル ネットワークで共有することにしたとします。このシナリオは、次のような理由から、標準形式を使って実現するには問題があります。

- Uniform Resource Identifier (URI) 形式は、一度に 1 つの項目しか共有できません。
- OneDrive では、コレクションはストリーム方式の StorageItems として共有されます。このシナリオで StorageItems を使うには、アプリで各画像をダウンロードしてから共有する必要があります。
- Text と HTML ではリンクの一覧を提供できますが、リンクの意味は失われます。したがって、受信側のアプリには、それらのリンクが、ユーザーが共有しようとしている画像を表すことはわかりません。

既に存在するスキーマ形式またはカスタム形式を使ってこれらの画像を共有することには、次の 2 つの大きな利点があります。

- すべての画像をローカルにダウンロードする代わりに URI のコレクションを作ることができるため、アプリでよりすばやく画像を共有できます。
- 受信側のアプリで、それらの URI が画像を表すことを認識し、それに応じた処理を行うことができます。

推奨と非推奨

カスタム形式の定義

アプリでカスタム形式を定義するメリットがあることがわかったら、次の点について検討する必要があります。

- 標準データ形式について理解していることを確認します。理解していないと、無駄にカスタム形式を作成することになります。自分のシナリオに合う形式が既にあるかどうかを http://www.schema.org で確認する必要があります。自分のカスタム形式をカバーする既に存在する形式が別のアプリで使われている可能性があります。

その場合は、より多くの対象ユーザーが、自分の考えるエンド ツー エンドのシナリオを実現できることになります。

- 実現するエクスペリエンスについて検討します。ユーザーが求めている操作と、その操作をサポートするのに最適なデータ形式について考えることが重要です。
- カスタム形式の定義を他のアプリ開発者が使用できるようにします。
- 形式には、内容に合った名前を付けるようにします。たとえば、UriCollection は、あらゆる種類の URI が含まれていることを表し、WebImageCollection は、オンライン イメージを指す URI のみが含まれていることを表します。
- その形式の意味について慎重に検討します。その形式が何を表し、どのように使用されるのかを明らかにします。
- その形式の構造について吟味します。複数の項目やシリアル化をサポートするかどうかや、どのような制限があるのかについてよく検討します。
- いったん公開したカスタム形式は変更しないでください。API のようなものと考えて、要素が追加されたり廃止されたりすることがあっても、下位互換性と長期的なサポートが確保されるようにします。
- 形式の組み合わせに依存しないようにします。たとえば、一方の形式が見つかったらもう一方の形式を探すなどの動作をアプリに期待しないようにしてください。形式はそれぞれ自己完結している必要があります。

アプリへのカスタム形式の追加

カスタム形式を定義した後にその形式をアプリに追加するときは以下の推奨事項に従ってください。

- その形式を他のアプリでテストします。ソース アプリからターゲット アプリにデータが正しく処理されることを確認します。
- その形式の当初の目的を見失わないようにします。他の用途に使用しないようにしてください。
- ソース アプリを作成する場合、標準の形式も少なくとも 1 つは提供するようにします。そうすると、そのカスタム形式をサポートしていないアプリでもデータを共有できるようになります。理想的なエクスペリエンスは得られないかもしれませんが、特定のアプリでしかデータを共有できなくなるよりはましです。さらに、他の

アプリでこの形式を利用できるように、形式に関するドキュメントをオンラインで提供します。

- ターゲット アプリを作成する場合、標準の形式を少なくとも 1 つサポートすることを検討します。そうすれば、ソース アプリで目的のカスタム形式が使われていなくてもデータを受信できます。
- 他のアプリの特定のカスタム形式 (特に、他社の形式) を利用する場合は、その形式に関するドキュメントがオンラインで一般に公開されているかどうかをもう一度確認する必要があります。ドキュメント化されていない形式は予告なく変更される可能性が高く、不整合が生じたり、アプリが失敗したりする原因となります。

その他の使い方のガイダンス

データの選択

カスタム形式を定義するときに行う最も重要な決定の 1 つは、ソース アプリケーションとターゲット アプリケーションの間の転送に使う WinRT データ型です。**DataPackage** クラスでは、カスタム形式用に以下のデータ型がサポートされています。

- **IPropertyValue** の任意のスカラー型 (integer、string、**DateTime** など)
- **IRandomAccessStream**
- **IRandomAccessStreamReference**
- **IUri**
- **IStorageItem**
- 上記のいずれかの項目の異種コレクション

カスタム形式を定義するときには、そのデータに適した型を選択します。その形式のすべてのコンシューマーと受信側が (他の選択肢があったとしても) 同じデータ型を使うことがとても重要です。そうしないと、データ型の不一致によるエラーがターゲット アプリケーションで予期せず発生する可能性があります。

カスタム形式のデータ型に string を選ぶと、ターゲット側で **GetTextAsync** 関数の **GetTextAsync(formatId)** 形式を使ってデータを取得できるようになります。この関数では、データ型の確認が行われます。その他のデータ型を使う場合は、**GetDataAsync** を使う必要があります。この場合は、データ型の不一致に対する保護が必要になります。 たとえば、ソース アプリケーションから 1 つの URI が提供された場合に、ターゲットが URI のコ

レクションとして取得しようとすると、不一致が生じます。こうした競合を防ぐには、次のようなコードを追加します。

DataPackage の設定

```
var uris = new Array();
uris[0] = new Windows.Foundation.Uri("http://www.msn.com");
uris[1] = new Windows.Foundation.Uri("http://www.microsoft.com");
var dp = new Windows.ApplicationModel.DataTransfer.DataPackage();
dp.setData("UriCollection", uris);
```

```
System.Uri[] uris = new System.Uri[2];
uris[0] = new System.Uri("http://www.msn.com");
uris[1] = new System.Uri("http://www.microsoft.com");
DataPackage dp = new DataPackage();
dp.SetData("UriCollection", uris);
```

データの取得

```
if (dpView.contains("UriCollection")) {
    dpView.getDataAsync("UriCollection").done(function(uris) {
        // Array.isArray doesn't work – uris is projected from InspectableArray
        if (uris.toString() === "[object ObjectArray]") {
            var validUriArray = true;
            for (var i = 0; (true === validUriArray) && (i < uris.length); i++) {
                validUriArray = (uris[i] instanceof Windows.Foundation.Uri);
            }
            if (validUriArray) {
                // Type validated data
            }
        }
    }
}
```

```
if (dpView.Contains("UriCollection"))
{
    System.Uri[] retUris = await dpView.GetDataAsync("UriCollection") as System.Uri[];
    if (retUris != null)
    {
      // Retrieved Uri collection from DataPackageView
    }
}
```

ファイル ピッカーのガイドライン

アプリでファイル ピッカーを使うと、ファイルやフォルダーへのアクセス、ファイルとフォルダーの添付、ファイルを開く操作や保存する操作が可能になります。

重要な API

FileOpenPicker クラス

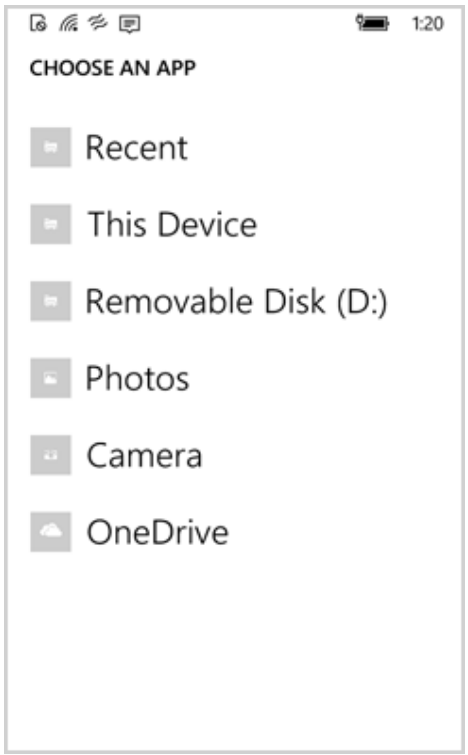
FileSavePicker クラス

FolderPicker クラス

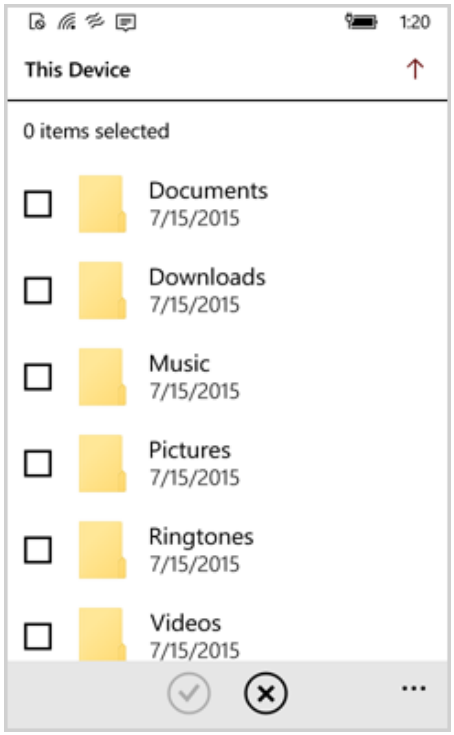
使い方のガイダンス

アプリでユーザーがファイルまたはフォルダーを選ぶことができるようにするには、ファイル ピッカーを呼び出すコントロールをアプリに追加します。

ファイル ピッカーの一般的なフローは、次の例で示されているように、ファイルまたはフォルダーをメールに添付する処理で表すことができます。ユーザーが "添付" を選択した場合は、一般的な場所の一覧を表示する必要があります。場所として、ローカル ストレージ、クラウド記憶域、他のアプリなどの場所を表示します。

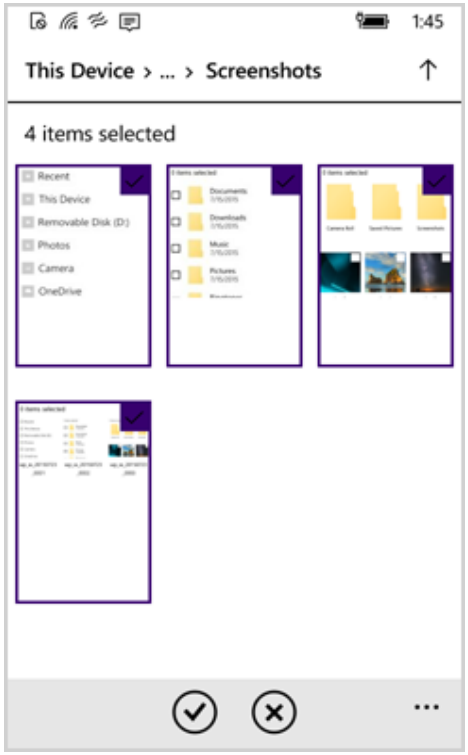


ユーザーが場所を選択したときに、一覧を表示する必要があります。フォルダー構造を示すには、リスト ビューまたはグリッド ビューを使うと便利です。この例では、リスト ビューが使用されており、フォルダーを選択できます。



ユーザーが次のレベルにドリルダウンすると、構造がグリッド ビューで表示される必要があります。

複数選択を利用すると、複数のファイルを添付できます。ファイルまたはフォルダーを選択して [OK] をタップすると、そのファイルまたはフォルダーが添付されます。



推奨事項

- ファイル ピッカーを呼び出すコントロールをアプリに追加して、ユーザーがアプリでファイルを選ぶことができるようにします。
- ファイル ピッカーを呼び出すコントロールをアプリの UI に追加して、(他のアプリと同様に) ユーザーが保存するファイルの名前、種類、保存場所を指定できるようにします。
- アプリに専用のページと UI を作って、ユーザーがファイルの内容を参照、使用、管理できるようにすることをお勧めします。これにより、ユーザーは現在の作業に集中できるようになり、ファイルを選ぶときに不要な機能に惑わされることがなくなります。
- ファイルまたはフォルダーを選ぶ場合も保存する場合も、アプリがサポートし、ユーザーの現在の作業に関連するファイルの種類だけを表示するように、ファイル ピッカーをカスタマイズします。たとえば、ユーザーが動画を選択または保存する場合、アプリで処理できる形式を使った動画ファイルだけを選択または保存できるようにファイルの種類を設定します。

- ユーザーがファイル名、ファイルの種類、または保存場所を指定する必要がない場合は、(ファイル ピッカーを起動せずに) アプリで自動的にファイルをバックグラウンド保存することをお勧めします。このようにすると不必要なユーザー操作がなくなるため、ファイルの保存プロセスが迅速になり、煩わしさも減ります。
- ユーザーが画像または動画を選ぶ場合は、表示モードを **Thumbnail** に設定します。 ユーザーがその他の種類のファイルまたはフォルダーを選ぶ場合は、表示モードを **List** に設定します。
- これは、ユーザーがファイル ピッカーで表示されるファイルを使っているときにフォルダーを選ぶ際にも適用されるため、選ぶフォルダーを決めるのに役立ちます。 適切なファイルの種類を表示するようフィルター処理を行うことで、ユーザーは正しいフォルダーをより速く特定できます。
- 場合によっては、ユーザーが画像 / 動画やその他の種類のファイルを選ぶこともあります (たとえば、ユーザーが電子メールに添付したり IM で送信したりするファイルを選ぶ場合)。 この場合は、アプリに 2 つの UI コントロールを追加して、両方の表示モードをサポートする必要があります。1 つのコントロールは **Thumbnail** 表示モードを使ってファイル ピッカーを呼び出して、ユーザーが画像や動画を選べるようにし、もう 1 つのコントロールは **List** 表示モードを使ってファイル ピッカーを呼び出して、ユーザーが他の種類のファイルを選べるようにします。たとえば、メール アプリには、**[画像または動画の添付]** と **[ドキュメントの添付]** という 2 つのボタンを用意します。
- ファイルまたはフォルダーを選ぶ場合も保存する場合も、コミット ボタンにユーザーの現在の作業に適したテキストを設定してファイル ピッカーをカスタマイズします。たとえば、ユーザーがアプリにアップロードする一連のファイルを選ぶ場合は、コミット ボタンのテキストを "アップロード" に設定します。
- ファイルまたはフォルダーを選ぶ場合も保存する場合も、ユーザーの現在の作業と **PickerLocationId** 列挙体で提供される有効な開始場所のリストに基づいて、できるだけ関連性のある開始場所を提示するようにファイル ピッカーをカスタマイズします。たとえば、ユーザーが画像を選ぶ場合は、提示する開始場所をユーザーのピクチャに設定します。
- ユーザーがプロフィール画像を選ぶ場合は、1 つのファイルを選ぶファイル ピッカーを呼び出します。ユーザーが友人に送る写真を選ぶ場合は、複数のファイルを選ぶファイル ピッカーを呼び出します。

- ユーザーが用意されている既定のファイル名をそのまま使うと、別の名前を入力する手間が省けるため、"名前を付けて保存" 作業を速やかに完了できます。既定のファイル名を設定するには、**FileSavePicker.SuggestedFileName** プロパティを使います。
- アプリで処理できるファイルの種類だけをユーザーが選択または保存できるようにファイルの種類を設定する。
- ファイルまたはフォルダーにアクセスする場合は、ユーザーが選ぶ項目の種類に基づいて表示モードを設定する
- コミット ボタンにユーザーの現在の作業に対応するテキストを設定する
- 提示する開始場所は、ユーザーの現在の作業に基づいてできるだけ関連性のある場所に設定する
- ファイルにアクセスする場合は、ユーザーが現在の作業に応じて 1 つのファイルまたは複数のファイルを選べるようにする
- ファイルを保存する場合は、保存するファイルの既定のファイル名を設定する
- ファイルの内容の参照、使用、管理にファイル ピッカーを使わない
- ユーザーが独自の名前や場所を指定する必要がない場合は、ファイルの保存にファイル ピッカーを使わない

ファイル ピッカー コントラクトのガイドライン

他のアプリにアプリのコンテンツ、保存場所、ファイルの更新へのアクセスを提供するために、ファイル オープン ピッカー コントラクト、ファイル保存ピッカー コントラクト、キャッシュ ファイル アップデーター コントラクトに参加しているアプリのファイル ピッカーをカスタマイズするには、次のガイドラインに従ってください。

重要な API

Pickers 名前空間

AccessCache 名前空間

Pcikers.Provider 名前空間

推奨事項

- **ファイルを提供する**　ファイル オープン ピッカー コントラクトと統合すると、ファイル ピッカーを通じてアプリのコンテンツにユーザーと他のアプリがアクセスできるようになります。
- **保存場所を提供する**　ファイル保存ピッカー コントラクトと統合すると、ファイル ピッカーを通じてユーザーと他のアプリに保存場所を提供できます。
    - アプリで保存場所を提供する場合は、ファイル オープン ピッカー コントラクトと統合して、アプリのコンテンツへのアクセスも提供する必要があります。
- **ファイルのリアルタイム更新を提供する**　キャッシュ ファイル アップデーター コントラクトと統合すると、アプリのリポジトリにあるファイルの更新を実行したり、リポジトリにあるローカル バージョンのファイルに更新を提供したりできます。これによりユーザーは、アプリのリポジトリにあるリモート ファイルをまるでローカルにあるかのように操作することができます。たとえば、ユーザーはテキスト エディター アプリを使ってファイルを編集し、Microsoft OneDrive のリポジトリにあるそのファイルのバージョンを更新することができます。

    - アプリでファイルを更新する場合は、ファイル保存ピッカー コントラクトとファイル オープン ピッカー コントラクトと統合して、保存場所とファイルへのアクセスも提供する必要があります。

使い方のガイダンス

アプリでファイル ピッカーを使って、ファイル、保存場所、ファイルの更新を提供する場合、ユーザーにファイル (または他の UI) を表示するためのページを設計することが必要になります。このページは、ファイル ピッカーの中央の領域に表示されます。このページとファイル ピッカーについて詳しくは、「ファイル ピッカー コントラクトとの統合」をご覧ください。

- すべてのサイズのウィンドウに対応するようにファイル ピッカーのページを設計します。
- アプリでファイルを表示するために既に使っているページを基に、ファイル ピッカーで表示するページ (ファイル ピッカーのページ) を設計します。
    - ユーザーがファイル ピッカーを使って選ぶファイルをアプリで提供する場合、ユーザーがファイルを表示するためのページがアプリに既に用意されている必要があります。ファイル ピッカーのページを設計するときは、お使いのこのファイル ビュー ページとの一貫性が維持されたページにすることをお勧めします。
    - ファイル ピッカーのページでの快適な操作をさらに確実にするには、ファイル ピッカーのページで、アプリでお使いのファイル ビュー ページと同じ (または類似した) ナビゲーション UI とエラー報告を使います。特にナビゲーションについては、ファイル ピッカーのページと、お使いのファイル ビュー ページでほぼ同じコマンドと場所を利用できることをユーザーは求めています。
- ユーザーの現在の作業に関係するファイル ピッカーのページを設計します。
    - ファイル ピッカーのページでは、直接関係のない UI を削除して、ファイルの選択、保存、更新など、ユーザーの現在の作業に焦点を合わせた UI を用意します。
    - たとえば、アプリが提供するファイルにアクセスするためにファイル ピッカーを使う場合は、選ぶことができない複雑な (詳しい) ナビゲーション、検索、情報をサポートする UI を削除します。
    - ユーザーがファイルの使用、変更、管理などの他の作業を行うことができるようにする場合は、その作業用のコントロールまたは他の UI をメイン ア

プリに追加します。コントロールを追加する方法については、「コントロールとコンテンツの追加」をご覧ください。

- ファイル ピッカーのタイトルをユーザーの現在の場所の名前に設定する。これにより、ユーザーがファイル ピッカーからアプリを使うときに、現在の場所を確認しやすくなります。
- アプリからアクセスできるすべてのファイルの場所にファイル ピッカーのページからアクセスできるようにします。アプリで通常アクセスできる場所にあるファイルには、ファイル ピッカーのページからもアクセスできるようにする必要があります。また、アプリにファイル ピッカーのページが複数ある場合は、アクセスできる場所がすべてのページで一貫している必要があります。そうすると、ユーザーがファイルや場所に予想どおりにアクセスできるようになります。
- Microsoft Visual Studio に用意されている組み込みのテンプレートを使います。これらは、UWP アプリでファイル ピッカーのビューを作成するときに使うことができます。
- ユーザーがファイル ピッカーからアプリを起動したときに、サインインとセットアップの対話操作をシンプルに保ちます。サインインやセットアップのタスクがシンプルな (1 回の操作で実行できる) 場合は、コンテキストを変更する必要がないように、ユーザーがファイル ピッカーからタスクを完了できるようにしてください。

追加の UX ガイドライン: ファイル オープン ピッカー コントラクト

- ファイル ピッカーのページには、Windows や他のアプリからアクセスできないファイルを表示します。他のアプリや Windows からはアクセスできない場所 (アプリの保存フォルダー、リモート サーバーなど) にあるファイルにアクセスできるようにして、Windows や他のアプリとの差別化を実現します。
- ファイル ピッカーのページでは、呼び出し元アプリの選択モードに応じた UI を設計します。

- アプリでファイル ピッカーを呼び出してファイルにアクセスする場合、その呼び出し元のアプリによって、ユーザーが単一の項目を選べるか複数の項目を選べるかが指定されます。選択対象ファイルを選択モードごとに異なる方法で適切に示すようにアプリのページを設計することをお勧めします。たとえば、ユーザーがアプリで用意されているファイルからプロフィール画像を選ぶ場合 (単一項目の選択)、どの写真を選ぶか決めるまでに複数の写真をタップまたはクリックする可能性があります。この場合、アプリの UI で一度に 1 項目しか選べないようにします。一方、ユーザーが友人と共有する複数のファイルを選ぶ場合は (複数項目の選択)、アプリの UI で一度に複数の項目を選べるようにします。

- Web カメラ アプリ、写真アプリ、カメラ アプリの場合、ファイル ピッカーのページでは写真撮影に関係する UI を設計します。
  - ファイル ピッカーのページではアプリの UI をシンプルにして、ユーザーが使っていた元のアプリ (呼び出し元アプリまたは呼び出し元) に確実に戻れるようにします。ファイル ピッカーのページに用意するコントロールは、ユーザーが写真を撮ったり、少数の前処理効果 (フラッシュの切り替えやズームなど) を適用したりするためのコントロールに限定します。
  - ファイル ピッカーからコマンド バーにアクセスすることはできないため、使用可能なすべてのコントロールをファイル ピッカーのページに表示する必要があります。ファイル ピッカーのページに表示するコントロールは、コマンド バーのコントロールと同様の構成にし、コマンド バーでコントロールが表示される場所のできるだけ近くに (ページの上部または下部)、ファイル ピッカーのページのコントロールを配置することをお勧めします。

追加の UX ガイドライン: ファイル保存ピッカー コントラクト

- Windows や他のアプリからユーザーがアクセスできない保存場所を提供します。Windows や他のアプリから簡単にアクセスできない場所 (アプリの保存フォルダー、リモートの保存場所など) にユーザーがファイルを保存できるようにします。
- 選ばれたファイルの種類に基づいて、ファイル ピッカーのページに表示されるファイルを変更します。ユーザーがファイル ピッカーのファイルの種類のドロップダウン リストでファイルの種類を変更した場合、選ばれたファイルの種類に一致するファイルだけを表示するようにビューを更新する必要があります。表示されるファイルを種類で絞り込むと、ユーザーが目的の種類のファイルを一貫した方法で簡単に見つけられるようになります。
- アプリのファイル ピッカーのページでファイルを選択することで、ユーザーがそのファイルを簡単に置き換えることができるようにします。ユーザーがファイル ピッカーのページでファイルを選択する場合は、既存のファイルを簡単に置き換えることができるように、ファイル ピッカーのファイル名ボックスでそのファイルの名前を自動的に置き換える必要があります。

追加の UX ガイドライン: キャッシュ ファイル アップデーター コントラクト

- ファイルの追跡と更新に対応したリポジトリを提供します。ユーザーがアプリを主な保存場所 (ユーザーがファイルを保存したりファイルにアクセスしたりするために通常使う場所) として使う場合は、アプリで一部のファイルを追跡してリアルタイム更新を提供することができます。
- 堅牢なリポジトリを提供するようにアプリとファイル ピッカーのページを設計します。ユーザーがアプリをファイルの主な保存場所として使う場合は、アプリと、関連するファイル ピッカーのビューを、ファイルの頻繁な更新やバージョンの競合などによるデータ消失を防ぐように設計します。
- 更新中に発生した問題をユーザーが解決できるようにします。更新を正常に完了できるようにするには、ファイルの更新中または保存中に発生した問題を効果的に解決するためにユーザーが操作する必要がある場合に、(**UIRequested** を使って) リア

ルタイムにユーザーに通知する必要があります。資格情報、ファイルのバージョンの競合、ディスク容量の問題の解決をサポートすることは特に重要です。

- 作成する UI は、軽量で、問題の解決に特化したものにする必要があります。複数の手順が必要な場合は (ログインなど)、すべての手順をアプリのファイル ピッカーのページで処理する必要があります。 完了したら、アプリでファイル ピッカーのコミット UI を有効にすることができます。アプリでは、ファイル ピッカーのタイトルを更新して、ユーザーに現在の場所についての情報を示す必要もあります。
- ユーザーが問題をリアルタイムで解決できない場合や、ユーザーに状況を通知するだけでよい場合 (ユーザーには解決できないエラーが発生した場合など) は、問題の発生時に **UIRequested** を使ってすぐに通知するのではなく、アプリの次回起動時に通知することをお勧めします。

- 更新操作と保存操作に関する追加情報をアプリの通常のページで提供する メイン アプリの UI で、ユーザーが進行中の操作や今後の操作の設定を管理したり、進行中の操作や以前の操作に関する情報を確認したり、発生したエラーに関する情報を入手したりできるようにする必要があります。

ファイルの種類と URI のガイドライン

Windows 10 では、アプリとアプリがサポートするファイルの種類の間の関係は以前のバージョンの Windows とは異なります。これらの違いを理解すると、より一貫性があり、洗練されたエクスペリエンスをユーザーに提供できます。

重要な API

FileActivatedEventArgs クラス (XAML)　　WebUIFileActivatedEventArgs (HTML)

推奨と非推奨

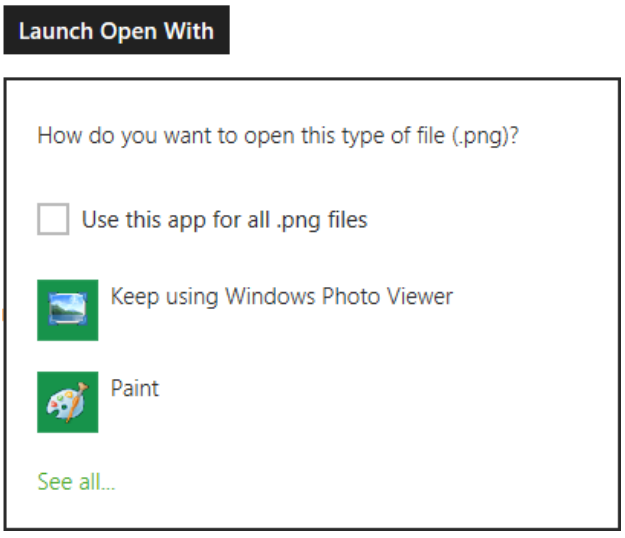
- フライアウトは、呼び出した位置の近くに配置します。

その他の使い方のガイダンス

ユニバーサル Windows プラットフォーム (UWP) アプリのガイドライン

ファイルまたは URI を開くときに、既定で使用するアプリを選ぶために、ユーザーが **[プログラムから開く]** の一覧を使うことが必要になる場合があります。Windows 10 ではこの一覧をフライアウトで実装しています。**[プログラムから開く]**フライアウトのコンテンツはカスタマイズできませんが、アプリ内の位置は制御できます。このガイドラインに従って、呼び出した位置のできるだけ近くにフライアウトを配置してください。

フライアウトの望ましい使用方法の例を次に示します。呼び出したボタンのすぐ横にフライアウトが配置されていることを確認してください。



ファイルや URI を適切と思われるもの (通常はサムネイルやハイパーリンクで表示される) に渡すことができます。これらのアイテムのプライマリ動作は **Open** になります。この動作はファイルまたは URI の既定のハンドラーを呼び出し、場合によっては **[プログラムから開く]** フライアウトが表示される結果になります (フライアウトが表示される場合があることを想定して適切に配置することをお勧めします)。

ファイルまたは URI の任意のセカンダリ動作 (Save As や Download など) をアプリに実装する場合は、**[プログラムから開く]** フライアウトでユーザーに別のアプリを選択させることを考えてください。

UWP アプリでは、ファイルの種類や URI に対する既定のアプリの設定、変更、照会はできないため、アプリにこの機能を追加しないでください。

印刷のガイドライン

アプリでコンテンツを印刷する場合は、以下のガイドラインに従ってください。印刷機能の詳細については、「アプリからの印刷」を参照してください。

推奨と非推奨

- **ShowPrintUIAsync** メソッドを呼び出す場合は、必ず try/catch でラップする必要があります。デバイスで印刷をサポートしていない場合、このメソッドではエラーがスローされます。印刷をサポートしていないプラットフォームの場合は、エラー メッセージが表示されます。
- 印刷ウィンドウに表示される設定の順序を変更しないでください。ユーザーに表示される設定の順序はカスタマイズ可能ですが、設定を既定の順序を保持することで、エクスペリエンスの一貫性が維持されます。たとえば、既定の印刷エクスペリエンスでは **[部数]** の設定は最初に表示されているため、ユーザーはこの表示順序をアプリの印刷エクスペリエンスでも想定します。

- 絶対に必要でない限り、印刷ウィンドウにプリンター設定を追加しないでください。代わりに、プリンターの製造元がプリンター固有の設定を追加できるようにします。製造元によってプリンター固有の設定が提供されている場合、ユーザーは印刷ウィンドウの **[その他の設定]** をクリックして、追加の設定を表示できます (この表示を有効にする Windows ストア デバイス アプリがインストールされている場合)。
- **PrintTaskRequested** イベント ハンドラーと **PrintTaskSourceRequestedHandler** ハンドラーでは、最小限の処理を実行します。これらのハンドラーには、タイムアウトが影響します。**PrintManager** によって印刷プレビュー UI に表示するための印刷ページのコレクションが要求されたら、印刷登録中はオブジェクトの初期化を最小限に抑え、負荷の大きいタスクは **Pagainate** イベント ハンドラーに任せることをお勧めします。

近接通信のガイドライン

このトピックでは、近接通信を使ってアプリを接続し、コンテンツを共有する際のベストプラクティスについて説明します。

重要な API

Proximity 名前空間　　PeerFinder クラス

近接通信は、2 台のデバイスで実行されているアプリの 2 つのインスタンス間で共有されるアプリのエクスペリエンスを実現するための優れた方法です。近接通信が有効なアプリでは、アプリのユーザーが 2 台のデバイスを同時にタップして接続を開始したり、ユーザーがワイヤレス範囲内でアプリを実行している別のデバイスを検索したりすることができます。PC では、ユーザーは Wi-Fi Direct を使って他の PC で実行されているアプリを検索でき

ます。Windows Phone では、ユーザーは Bluetooth を使って他の Windows Phone で実行されているアプリを検索できます。

近接通信を使って通信を行うには、いくつかの方法があります。

- **帯域外セッション:** アウトオブバンド トランスポート (Bluetooth、インフラストラクチャ ネットワーク、または Wi-Fi Direct) でデバイスどうしを接続する **PeerFinder** オブジェクトを使って、セッションを確立できます。 タップの場合の範囲は 3 ～ 4 センチに制限されますが、アウトオブバンド トランスポート オプションの場合の範囲はそれよりも広くなります。 リソースを共有するだけであれば、アプリに近接通信を含める必要はありません。Windows で共有シナリオがサポートされている場合は、共有コントラクトを有効にし、Windows の組み込み機能を使って、タップ ジェスチャでリソースを共有します。
  注 Wi-Fi Direct は Windows Phone ストア アプリではサポートされていません。
- **ピアの参照:** **PeerFinder.FindAllPeersAsync()** メソッドを使って、セッションを確立できます。このメソッドは、**PeerFinder.Start()** メソッドを呼び出してピア セッションで使用可能であることをアドバタイズしているすべてのリモート ピアを検出します。
  注 ピアの参照は、Windows Phone ストア アプリで Bluetooth を使って実行します。このため、Windows Phone で実行されているアプリは電話のピアのみを検出でき、コンピューターで実行されているアプリはコンピューターのピアのみを検出できます。
- **メッセージの発行と購読:** **ProximityDevice** オブジェクトを使って、タップ ジェスチャーを行っているときにメッセージを送受信できます。

アプリが **ConnectAsync** メソッドを呼び出してピアとの接続を作ると、アプリは接続のためのアドバタイズを行わなくなり、アプリが **StreamSocket.Close** メソッドを呼び出してソケット接続を閉じないと **FindAllPeersAsync()** メソッドで見つからなくなります。

ピアが見つかるのは、コンピューターがワイヤレス範囲内にあって、ピア アプリがフォアグラウンドで実行されている場合のみです。ピア アプリがバックグラウンドで実行されている場合、近接通信ではピア接続のためのアドバタイズを行いません。

ConnectAsync メソッドを呼び出してソケット接続を開く場合、コンピューターで開くことができるソケット接続は一度に 1 つのみです。自分のアプリ、または別のアプリが **ConnectAsync** メソッドを呼び出すと、既にあるソケット接続が閉じられます。

デバイス上の各アプリから別のデバイス上のピア アプリに対して開かれた接続がタップ ジェスチャーを使って確立された場合、各アプリでは開かれた接続を 1 つ使うことができます。各デバイスをタップすると、1 つのアプリから複数のデバイス上のピア アプリに対してソケット接続を開くことができます。タップ ジェスチャーを使って接続を作る場合、新しいタップ ジェスチャーで既にある接続は閉じられません。ソケット オブジェクトの **StreamSocket.Close** メソッドを呼び出し、タップ ジェスチャーを使って同じピア デバイス上の同じピア アプリに対して新しい接続を作る必要があります。

注 近接通信ではネットワーク接続の StreamSocket オブジェクトのみを作ります。アプリで **StreamSocket** オブジェクト以外の種類の接続オブジェクトが必要な場合は、接続のために近接通信を使うことはできません。

推奨事項

- アプリで同じアプリを実行している他のピアを参照する場合は、ピアを連続して参照しないようにします。代わりに、Wi-Fi 範囲内のピアを参照するためのオプションを用意して、ユーザーが操作を開始できるようにします。
- 接続された近接通信エクスペリエンスを開始し、アプリをマルチユーザー モードにする)ときは、必ずユーザーの同意を得る必要があります。たとえば、あるゲームをプレイする 2 人のプレイヤーがいる場合は、プレイヤーが同意を示したうえでゲームを一緒にプレイできるようにします。 アプリの起動時にタップが発生した場合は、スタート メニューまたはアプリのロビーで同意を示す機会をユーザーに提供する必要があります。
- ユーザーがアプリをマルチユーザー モードに移行したときは、UI に接続状態を表示する必要があります。接続状態は、次の 3 つのうちのいずれかになります。
  - タップの待機中
  - デバイスに接続中 (進捗状況を表示)
  - デバイスへの接続完了、または接続の失敗

- 接続が中断した場合または設定できない場合は、単一ユーザー モードに戻します。接続が失敗したことを示すメッセージを表示します。
- ユーザーが近接通信エクスペリエンスを簡単に終了できるようにします。
- 同じアプリを実行している他のデバイスを連続して参照しないようにします。代わりに、Wi-Fi 範囲内のピアを参照するためのオプションを用意します。
- 接続に関して定期的な更新 (帯域幅使用状況や速度に関する更新など) を必要とするアプリの場合は、近接通信を使わないでください。

サムネイルのガイドライン

このガイドラインでは、サムネイル イメージを使って、ユニバーサル Windows プラットフォーム (UWP) アプリでファイルを参照するユーザーに最適なプレビューを提供する方法について説明します。

重要な API

ThumbnailMode 列挙

アプリにサムネイルを含めるかどうか

アプリでユーザーがファイルを参照できるようにする場合は、サムネイル イメージを表示して、ファイルをすばやくプレビューできるようにします。

サムネイルは次の場合に使います。

- 多くの項目 (ファイルやフォルダーなど) に対するプレビューを表示する場合。たとえば、フォト ギャラリー アプリでは、ユーザーが写真ファイルを参照するときにサムネイルを使って各写真が小さく表示されるようにします。
- 個別の項目 (個々のファイルなど) に対するプレビューを表示する場合。たとえば、ユーザーがファイルを開くかどうかを決める前に、より見やすい大きなサムネイルと共に、ファイルの詳しい情報を表示できます。

推奨と非推奨

- サムネイルを取得するメソッドを呼び出すときに、サムネイル モード (**picturesView**、**videosView**、**documentsView**、**musicView**、**listView**、**singleItem**) を指定します。これにより、ユーザーが参照するファイルの種類を表示するのに最適なサムネイル イメージが用意されます。
- ファイルの種類に関係なく単一項目用のサムネイルを取得するには、**singleItem** モードを使います。その他のサムネイル モード (**picturesView**、**videosView**、**documentsView**、**musicView**、**listView**) の目的は、複数ファイルのプレビューを表示することです。
- サムネイルの読み込み中は、サムネイルの代わりに汎用のプレースホルダー イメージを表示します。このようにプレースホルダーを使うことで、ユーザーはプレビュー イメージを読み込む前に項目を操作できるため、アプリの見かけの応答速度を高めることができます。

  プレースホルダー イメージは次の条件を満たす必要があります。
  - 代わりとなる項目の種類に固有である。たとえば、フォルダー、画像、動画にはすべて、それぞれ異なるアイコン、テキスト、色を使った専用のプレースホルダーを用意する必要があります。
  - 代わりとなるサムネイル イメージとサイズおよび縦横比が同じである。
  - サムネイル イメージが読み込まれるまで表示される。サムネイルを取得できない場合は、代わりにプレースホルダー イメージを表示します。
- テキスト ラベル付きのプレースホルダー イメージを使って、フォルダーとファイル グループを表現します。こうすることで、フォルダーやファイル グループのようなシステム構造と個別のファイルを区別することができます。これらの項目の種類をそれぞれ視覚的に区別することで、アプリでユーザーが参照しやすくなります。フォルダーの名前、またはファイル グループを作るために使われた条件を、テキスト ラベルとしてプレースホルダーに含めます。
- 項目 (ファイル、フォルダー、ファイル グループなど) のサムネイルを取得できない場合はプレースホルダー イメージを表示します。
- ドキュメントと音楽ファイルのプレビューを表示するときは、サムネイル イメージに加えてファイル情報も表示します。これによってユーザーは、サムネイル イメージだけからはすぐに得られない可能性のある、ファイルに関する重要な情報を

識別できます。たとえば、アルバム アートを示すサムネイルと並べて音楽ファイルのアーティスト名を表示したりできます。

- 画像ファイルと動画ファイルのサムネイルと共に追加のファイル情報を表示しないでください。ほとんどの場合、ユーザーが画像と動画を参照するには、サムネイル イメージだけで十分です。

その他の使い方のガイダンス

推奨サムネイル モードとその特徴

プレビューの表示対象	サムネイル モード	取得するサムネイル イメージの特徴
画像 ビデオ	picturesView videosView	**サイズ:** 中、190 以上を推奨 (画像サイズが 190 × 130 の場合) **縦横比:** 均一な横長の縦横比 (約 0.7) (サイズが 190 の場合は 190 × 130) プレビューの場合はトリミング 縦横比が統一されているため、画像をグリッド内で揃えるときに便利です。
ドキュメント ミュージック	documentsView musicView listView	**サイズ:** 小、40 × 40 ピクセル以上を推奨 **縦横比:** 均一な正方形の縦横比 縦横比が正方形であるため、アルバム アートのプレビューに最適 ドキュメントは、ファイル ピッカーのウィンドウと同じように表示 (同じアイコンを使用)
任意の 1 つの項目	singleItem	**サイズ:** 大、長辺が 256 ピクセル以上 **縦横比:** 可変、ファイルの元の縦横比を使用

ヒント 今後、各モードのサムネイル イメージの特徴は、さらに具体的になる可能性があります。それに対応するために、プレビューを表示するファイルの種類に最も近いサムネイル モードを指定することをお勧めします。たとえば、ビデオ ファイルを表示する場合は、**videosView** サムネイル モードを使います (ただし単一のビデオ ファイルを表示する場合は、singleItem モードを使います)。

推奨サムネイル モードを使う理由

以下の例は、取得したサムネイル イメージが、ファイルの種類とサムネイル モードに応じてどのように異なるかを示します。

項目の種類	指定するサムネイル モード		
	取得時に使ったモード： • picturesView • videosView	取得時に使ったモード： • documentsView • musicView • listView	取得時に使ったモード： • singleItem
画像		サムネイルは縦横比が正方形になるようにトリミングされています。 	サムネイル イメージには、ファイルの元の縦横比が使われます。 
ビデオ	サムネイル イメージには、画像と区別するためのアイコンが追加されます。 	サムネイルは縦横比が正方形になるようにトリミングされています。 	サムネイル イメージには、ファイルの元の縦横比が使われます。 

ミュージック	サムネイルは、適切なサイズの背景に配置されたアイコンです。背景色は、ファイルに関連付けられたアプリによって決まります (関連付けられたアプリが Windows アプリの場合は、アプリのタイルの背景色が使われます。)。 	ファイルにアルバム アートが含まれる場合、サムネイルはアルバム アートになります。 それ以外の場合、サムネイルは、適切なサイズの背景に配置されたアイコンです。背景色は、ファイルに関連付けられたアプリによって決まります (関連付けられたアプリが Windows アプリの場合は、アプリのタイルの背景色が使われます。)。 	ファイルにアルバム アートが含まれる場合、サムネイルはアルバム アートになり、ファイルの元の縦横比が使われます。それ以外の場合、サムネイルはアイコンです。 
ドキュメント	サムネイルは、適切なサイズの背景に配置されたアイコンです。背景色は、ファイルに関連付けられたアプリによって決まります。 (関連付けられたアプリが Windows アプリの場合は、アプリのタイルの背景色が使われます。) 	サムネイルは、適切なサイズの背景に配置されたアイコンです。背景色は、ファイルに関連付けられたアプリによって決まります (関連付けられたアプリが Windows アプリの場合は、アプリのタイルの背景色が使われます。)。 	ドキュメントのサムネイルがある場合は、そのサムネイルが表示されます。  それ以外の場合、サムネイルはアイコンです。 

フォルダー	フォルダーに画像ファイルが含まれる場合は、画像のサムネイルが使われます。  それ以外の場合、サムネイル イメージは取得されません。	サムネイル イメージは取得されません。	サムネイルは、フォルダーを表すアイコンです。
ファイル グループ	グループ内のファイルに画像ファイルが含まれる場合は、画像のサムネイルが使われます。  それ以外の場合、サムネイル イメージは取得されません。	グループ内のファイルにアルバム アートを含むファイルがある場合、サムネイルはアルバム アートになります。  ファイル グループにアルバム アートが存在しない場合、サムネイル イメージは取得されません。	グループ内のファイルにアルバム アートを含むファイルがある場合、サムネイルはアルバム アートになり、ファイルの元の縦横比が使われます。  それ以外の場合、サムネイルはファイルのグループを表すアイコンです。 

グローバリゼーションとローカライズのガイドライン

世界市場に向けて容易に提供できるアプリの設計方法について検討します。アプリのリソース (文字列や画像など) をコードから分離します。これにより、ローカライズが容易になります。

このセクションの内容

トピック	説明
アプリ リソース	このトピックでは、ユニバーサル Windows プラットフォーム (UWP) アプリでアプリ リソースを使うためのベスト プラクティスについて説明します。
グローバリゼーションとローカリゼーション	広範なユーザー向けにアプリをグローバル化したり、特定の市場を対象にアプリをローカライズするときは、次のベスト プラクティスに従ってください。

アプリ リソースのガイドライン

文字列や画像などのアプリ リソースをコードから分離することは、アプリの保守やローカライズのプロセスを簡単にする方法の 1 つです。このトピックでは、ユニバーサル Windows プラットフォーム (UWP) アプリでアプリ リソースを使うためのベスト プラクティスについて説明します。

アプリ リソースの詳細と、リソース管理 API の使い方を説明したトピックについては、次をご覧ください。

- アプリ リソースの定義 (JavaScript と HTML を使った Windows ストア アプリ)
- アプリ リソースの定義 (C#/VB/C++ と XAML を使った Windows ストア アプリ)

重要な API

Resources 名前空間

Resources.Core 名前空間

推奨と非推奨

リソースの作成

- UI 文字列や画像などのリソースをコードに格納しないでください。代わりに、.resjson ファイルや .resw ファイルなどのリソース ファイルに格納してください。
- 修飾子を使って、さまざまな表示スケール、UI 言語、ハイ コントラスト設定に合わせたファイルと文字列リソースをサポートします。
- アプリ マニフェスト (package.appxmanifest) に既定の言語を設定します。
- 文字列リソースには、既定の言語であっても、その言語タグの名前を持つファイルまたはフォルダーが必要です。
- ローカライズ担当者に向けたコメントを文字列リソースに追加します。

リソースの名前付けについて詳しくは、「修飾子を使ってリソースに名前を付ける方法 (HTML)」または「修飾子を使ってリソースに名前を付ける方法 (XAML)」をご覧ください。

リソースの参照

- リソースを参照するための一意のリソース識別子をコードとマークアップに追加します。
- マークアップ、コード、またはマニフェスト ファイルで修飾子を除いて画像を参照します。
- システムが変更されて、異なるセットの修飾子の使用が始まるときに発生するイベントをリッスンします。正しいリソースが読み込まれるようにドキュメントを再処理します。

UI リソースの翻訳とローカライズの準備について詳しくは、「アプリのグローバル化 (HTML)」または「アプリのグローバル化 (XAML)」をご覧ください。世界中のユーザーに適合するアプリを作成するための推奨事項については、「グローバリゼーションのガイドライン」をご覧ください。

グローバリゼーションのガイドライン

広範なユーザーや市場向けにアプリをグローバル化したり、特定のユーザーや市場を対象にアプリをローカライズするときは、次のベスト プラクティスに従ってください。

重要な API

Globalization 名前空間

Globalization.NumberFormatting 名前空間

Globalization.DateTimeFormatting 名前空間

推奨と非推奨

グローバリゼーション

グローバル化に適した UI の用語と画像が選択され、**Globalization** API を使ってアプリ データがフォーマットされ、場所や言語に基づく前提のない、異なる市場に簡単に適応できるアプリを準備します。

推奨事項	説明
数値、日付、時刻、住所、電話番号には正しい形式を使う。	数値、日付、時刻などのデータに使われる形式は、カルチャ、地域、言語、市場により異なります。数値、日付、時刻などのデータを表示する場合は、**Globalization** API を使って特定のユーザーに適した形式を取得します。
国際的な用紙サイズをサポートする。	最も一般的な用紙サイズは国によって異なるため、用紙サイズによって変化する機能 (印刷など) を含める場合には、必ず一般的な国際サイズをサポートし、テストしてください。
国際的な計測単位と通貨をサポートする。	使われる単位と尺度は国によって異なりますが、最も使われているのはメートル法とヤード ポンド法です。長さ、温度、範囲などの計測を扱う場合は、**Globalization** 名前空間を使って正しいシステム計測を取得してください。アプリが通貨の表示をサポートする場合は、必ず正しい書式設定を使ってください。**CurrenciesInUse** プロパティを使って、ユーザーの地理的な地域の通貨を取得することもできます。
テキストとフォントを正しく表示する。	テキストに適したフォント、フォント サイズ、方向は、市場によって異なります。

文字エンコードに Unicode を使う。	既定では、最近のバージョンの Microsoft Visual Studio は、すべてのドキュメントに Unicode 文字エンコードを使います。別のエディターを使っている場合は、適切な Unicode 文字エンコードでソース ファイルが保存されるようにしてください。Windows ランタイム API はどれも、UTF-16 エンコードの文字列を返します。
入力の言語を記録する。	アプリがユーザーにテキスト入力を求めるときに、入力の言語が記録されるようにします。こうすると、その入力が後で表示されるときに適切な書式設定でユーザーに提示されます。現在の入力言語の取得には、**CurrentInputMethodLanguage** プロパティを使います。
言語からユーザーの位置を想定する、または位置からユーザーの言語を想定することはしない。	Windows では、ユーザーの言語と位置は別の概念です。特定の地理的な言語バリアント (en-gb (英国で話される英語) など) を話しても、住んでいる国または地域はまったく異なる場合があります。UI テキストなどのためにアプリがユーザーの言語について認識する必要があるか、ライセンス問題などのためにアプリがユーザーの位置について認識する必要があるかを検討してください。
俗語と比喩を使わない。	一部のカルチャや年齢などの集団にしか伝わらない言葉は、その集団の人しか使わないので、理解や翻訳が難しい場合があります。同様に、比喩も人によって伝わったり伝わらなかったりします。たとえば、"ブルーバード" はスキーをする人には伝わりますが、スキーをしない人には伝わりません。アプリをローカライズしてくだけた表現を使うことを計画している場合は、ローカライズ担当者に翻訳対象の意味と口調を十分に説明してください。

専門的な用語、省略形、略語を使わない。	専門用語は、専門知識のないユーザーや他のカルチャまたは地域の人々には意図が伝わりにくく、翻訳も困難です。このような言葉は日常会話では使われません。専門用語は、ハードウェアとソフトウェアの問題を特定するため、エラー メッセージ内でよく使われます。専門用語が必要な場合があるかもしれませんが、普通の言葉に置き換えることが望まれます。
不快感を与えかねない画像は避ける。	自分が所属するカルチャでは妥当な画像でも、別のカルチャでは不快感を与えたり、誤って解釈されたりすることがあります。宗教的なシンボル、動物、国旗や政治運動に関連付けられる色の組み合わせなどは避けてください。
地図や、地域についての言及では、政治的侵害を避ける。	地図には論争の的になっている地域や国境が含まれている可能性があり、それらはしばしば政治的な侵害のきっかけになります。国家の選択に使う UI は、必ず "国/地域" という名称にしてください。(住所フォームなどで) "国" という名称の一覧に領有権未決の領域を含めると、トラブルになりかねません。
言語タグを比較する目的で文字列比較を単独で使わない。	BCP-47 言語タグは複雑です。言語タグの比較では、スクリプト情報、前のタグ、複数の地域バリアントの対応付けに伴う問題など、多数の問題が発生します。Windows のリソース管理システムでは、対応付けが自動的に行われます。開発者はどの言語で作られたリソース セットでも指定でき、システムがユーザーとアプリのために適切なものを選びます。

並べ替えが常にアルファベット順で行われると想定しない。	ラテン文字を使わない言語の場合、並べ替えは発音、ペン ストロークの数などの要素に基づいて行われます。ラテン文字を使う言語でも、常にアルファベット順の並べ替えを行うわけではありません。たとえば、一部のカルチャでは電話帳はアルファベット順では並んでいない場合があります。システムによって並べ替えが自動的に行われますが、自分で独自の並べ替えアルゴリズムを作る場合は、必ず、アプリの対象市場で使われている並べ替え方法を考慮してください。

ローカライズ

推奨事項	説明
UI 文字列や画像などのリソースをコードから分離する。	アプリは、文字列や画像などのリソースがコードから分離されるように設計してください。こうすることで、さまざまなスケール ファクター、アクセシビリティ オプション、ユーザーとコンピューターに関する多くのコンテキストに対して、それらの保守、ローカライズ、カスタマイズを個別に行うことができます。 文字列リソースはアプリのコードから分離し、言語に依存しない単一のコードベースを作ってください。常にアプリ コードとマークアップから文字列を分離し、リソース ファイル (ResW や ResJSON ファイル) に入れてください。 Windows のリソース インフラストラクチャを使って、ユーザーの実行時環境に最適なリソースが選ばれるように処理してください。詳しくは、「アプリ リソースのガイドライン」をご覧ください

他のローカライズ可能なリソース ファイルを分離する。	ローカライズが必要な他のファイル (翻訳するテキストを含んだ画像やカルチャ上の配慮のために変更が必要な画像など) は、言語名でタグ化されたフォルダーに入れてください。
既定の言語を設定し、既定の言語で作られているものも含め、すべてのリソースをマークする。	アプリの既定の言語は常に適切に設定してください。アプリでサポートされる言語のどれもユーザーが話さない場合、使われる言語は既定の言語によって決定されます。既定の言語リソースをその言語でマークしてください (例: en-us/Logo.png)。こうすることで、システムはリソースで使われている言語と、個々の状況でそのリソースがどのように使われるかを判断できます。
ローカライズが必要なアプリ リソースを特定する。	他の市場向けにアプリをローカライズすることになった場合には、何を変更する必要があるでしょうか。テキスト文字列を他の言語に翻訳する必要があります。他のカルチャに合わせて画像を変更する必要がある場合もあります。アプリが使う他のリソース (オーディオやビデオなど) にローカライズがどのような影響を与えるかを考慮してください。
リソースを参照するには、コードとマークアップでリソース識別子を使ってください。	文字列リテラル、または画像の特定のファイル名をマークアップに含めるのではなく、リソースの参照を利用してください。必ず、リソースごとに一意の識別子を使ってください。詳しくは、「修飾子を使ってリソースに名前を付ける方法 (HTML)」をご覧ください。

テキスト サイズを拡大できるようにする。	翻訳されるとテキスト サイズが大きくなる可能性があるため、テキスト バッファーは動的に割り当ててください。静的なバッファーを使う必要がある場合は、(英語文字列の長さを倍にするなどの方法で) それらを特大サイズにし、文字列が翻訳された時点で起きる可能性がある拡大に対応してください。 ユーザー インターフェイスに利用可能な領域が制限されることもあります。ローカライズされた言語に対応するには、文字列の長さが、日本語に必要となりそうな長さよりも約 40% 長くします。単一の語句のように非常に短い文字列の場合、必要領域が約 300% も増える可能性があります。さらに、コントロール内で複数行のサポートとテキストの折り返しを有効にすると、各文字列を表示するための領域を増やすことができます。
左右反転をサポートする。	テキストの配置と読み取りは、英語のように左から右の順にも、アラビア語やヘブライ語のように右から左の順 (RTL) にも行うことができます。読み取り順が自国語とは異なる言語に製品をローカライズする場合は、UI 要素のレイアウトが左右反転をサポートする必要があります。戻るボタン、UI 切り替え効果、画像などのアイテムですら、左右反転が必要になることがあります。
文字列にコメントする。	文字列に適切にコメントを入れ、翻訳が必要な文字列だけをローカライズ担当者に提供してください。過剰なローカライズは、よく問題を引き起こします。
短い文字列を使う。	文字列を短くすると翻訳が簡単になり、翻訳データを再使用できます。翻訳データを再使用すると、同じ文字列はローカライズ担当者に再び送られることがないため、コストを節約できます。 8,192 文字を超える文字列は、一部のローカライズ ツールではサポートされない可能性があります。このため、文字列は 4,000 文字以内に抑えてください。

文全体が入った文字列を提供する。	単語の訳は文におけるその位置によって変化する可能性があるため、文を個々の単語に分割せず、文全体が入った文字列を提供してください。1 つのフレーズが複数のパラメーターから構成される場合、どの言語でもパラメーターの順序は変わらないと想定してはなりません。
画像ファイルとオーディオ ファイルをローカライズ用に最適化する。	画像内にテキストを入れることやオーディオ ファイルに音声を入れることを避けると、ローカライズ コストが抑えられます。読み取り順が自国語とは異なる言語にローカライズする場合は、左右対称の画像や効果を使うと左右反転をサポートしやすくなります。
文字列は異なるコンテキストで再使用しない。	文字列は異なるコンテキストで再使用してはなりません。"オン" や "オフ" などの簡単な語句でも、コンテキストに基づいて別の翻訳がなされる可能性があります。

ヘルプと説明のガイドライン

ユーザーにヘルプやトラブルシューティングのヒントを提供して、効果的にアプリの操作法を示す必要があります。このセクションでは、アプリを使うユーザー向けの操作説明に関するベスト プラクティスを提供します。

このセクションの内容

トピック	説明
アプリのヘルプ	このガイドラインは、アプリの効果的なヘルプ コンテンツの設計方法を説明しています。
インストラクショナル UI	Windows アプリを使う方法をユーザーに気付かせるユーザー インターフェイス (UI) をデザインします。

アプリのヘルプのガイドライン

このガイドラインは、アプリの効果的なヘルプ コンテンツの設計方法を説明しています。ヘルプ コンテンツは 1 ページに収めます。テキスト、リンク、画像を含めることができます。ダイナミックなヘルプ コンテンツを提供する場合は、サポート Web サイトにリンクするか、ヘルプ セクションにオンライン ページを埋め込みます。

アプリにヘルプ コンテンツを含めるかどうか

ヘルプを含めるかどうかは、必要に応じて選択できます。すべてのアプリに専用のヘルプ セクションが必要になるわけではありません。たとえば、ユーザーが混乱する可能性のある UI 要素がアプリに 1 つか 2 つしかない場合は、独立したヘルプ セクションを作成する代わりに、インストラクショナル UI を組み込むか、単純なアプリ内デモを作成することができます。UI 要素の設計を見直すこともできます。専用のヘルプ セクションを作成する場合は、できるだけ簡潔にすることをお勧めします。

推奨事項

- ヘルプ ページは短くしてユーザーが参照しやすくします。

- ヘルプ コンテンツが 1 ページに収まらない場合、または更新が必要な情報をダイナミック コンテンツに含める必要がある場合は、サポート Web サイトにリンクするか、ヘルプ ポップアップにオンライン ページを埋め込みます。Web ページにリンクすると、ユーザーがアプリから引き離されることを忘れないでください。可能な場合は、オンライン コンテンツを埋め込んで、より統一感のあるユーザー エクスペリエンスを実現します。
- 専門的な用語や業界用語を使わないでください。
- ヘルプを使って、アプリのすべての機能を説明しないでください。アプリについて詳しく説明する場合は、ヘルプ コンテンツの最後に、サポート Web ページへのリンクを記載することを検討してください。
- アプリの新しいバージョンが利用できることをユーザーに知らせるためにヘルプを使わないでください。
- 設定ページからヘルプにアクセスできるようにします。

インストラクショナル UI のガイドライン

ユニバーサル Windows プラットフォーム (UWP) アプリを使う方法をユーザーに気付かせるユーザー インターフェイス (UI) をデザインします。

推奨事項

- アプリでできることを新しいユーザーに紹介するためにインストラクショナル UI を使います。
- 新機能に関するヒントや更新後のアプリの変更点に関する詳細に使います。
- インストラクショナル UI を特定のタスクと統合します。
- アプリケーション UI の操作を妨げないようにします。

その他の使い方のガイダンス

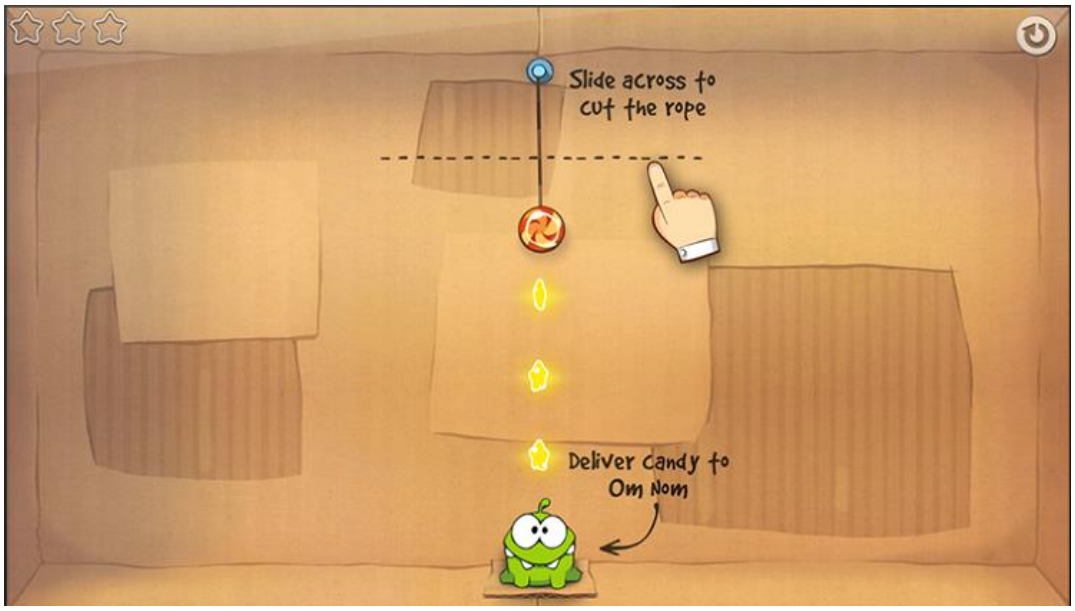
状況によっては、アプリの UI から操作を気づかせると効果的な場合があります。この種のガイダンスは、インストラクショナル UI という用語で表します。良い例の 1 つとして、

タスクを完了するためにタッチ操作を使う必要があるときに、ユーザーへの指示のためにインライン テキストやフライアウトなどの UI 要素を使うことが挙げられます。

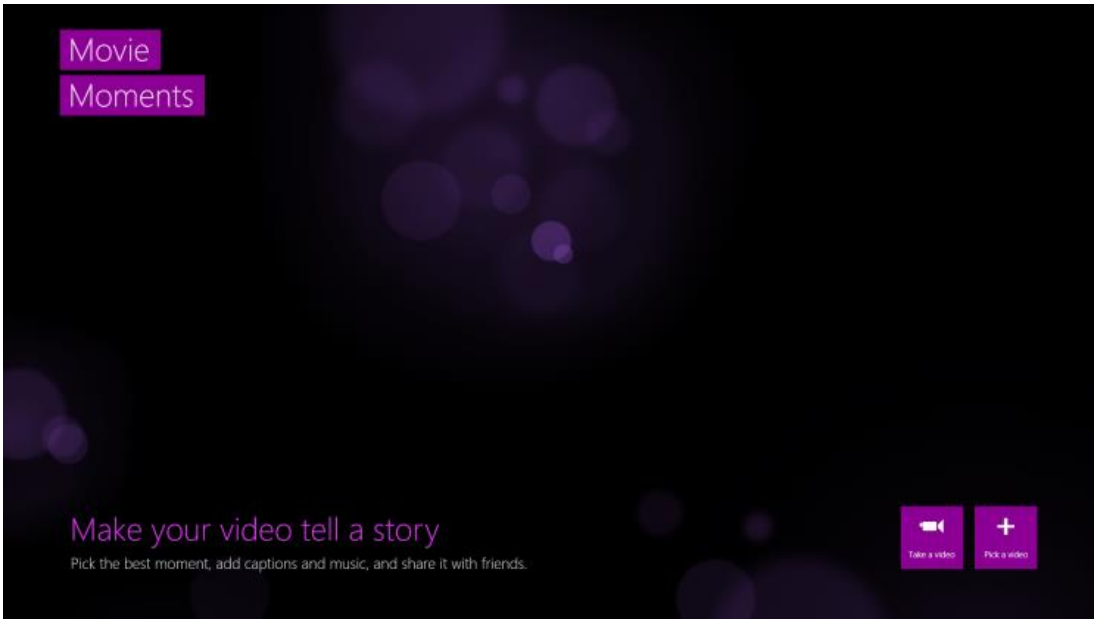
インストラクショナル UI はよく練られた設計に代わるものではないことに注意してください。使用頻度が多すぎる場合、またはコンテキストから逸脱する場合には、アプリのフローが中断され、実効性が低下するおそれがあります。インストラクショナル UI を追加する前に、ユーザーをアプリに導く他の方法を検討してください。

ここでは、インストラクショナル UI がユーザーが操作を習得するのに役立ついくつかの例を示します。

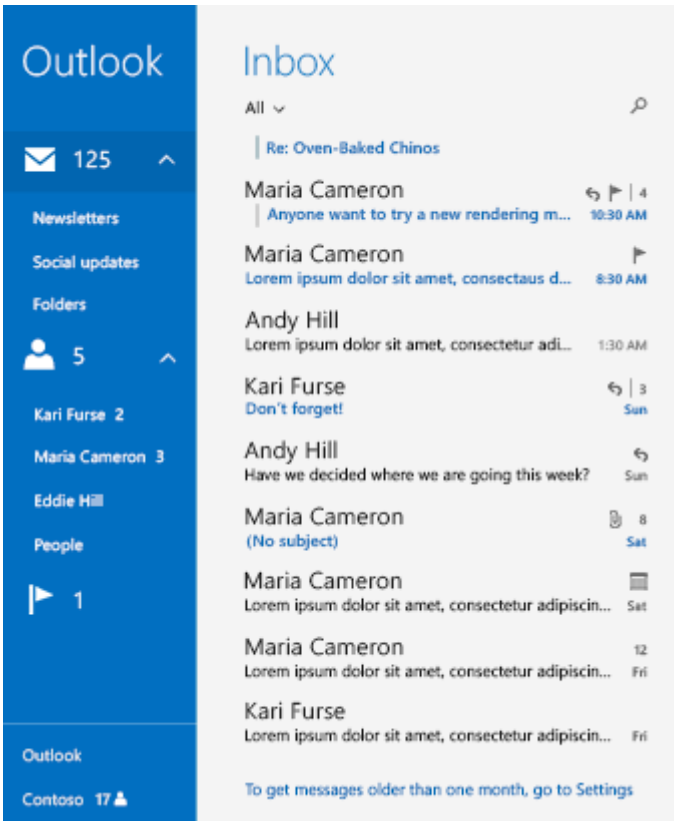
- **ユーザーがタッチ操作を見つけられるようにする。** 次のスクリーン ショットには、Cut the Rope というゲーム内でタッチ ジェスチャーを使う方法をプレイヤーに気付かせるインストラクショナル UI が示されています。



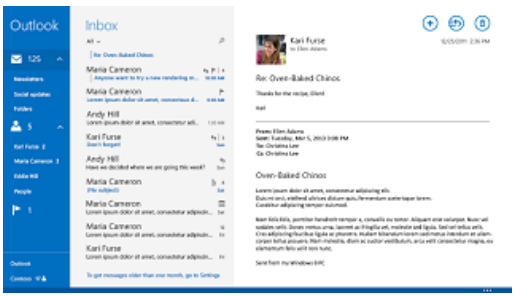
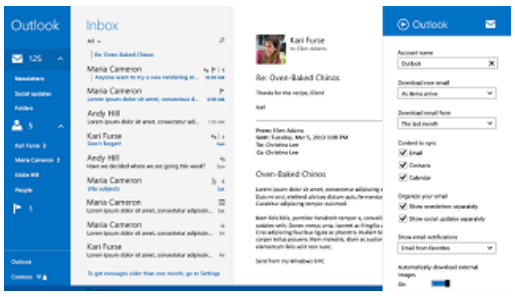
- **第一印象を良くする。** 新しいユーザーにアプリで可能な処理を紹介するためにインストラクショナル UI を使うことを考慮してください。たとえば、ムービー モーメントの初回起動時には、インストラクショナル UI を使ってユーザーは映画を作成するように求められます。



- **複雑なタスクの次の手順にユーザーを導く。**Windows メール アプリの受信トレイの下のヒントは、以前のメッセージにアクセスできるように、[設定] にユーザーを導きます。



ユーザーがメッセージをクリックすると、アプリの設定フライアウトが画面の右側に表示され、ユーザーがタスクを実行できるようになります。次のスクリーン ショットは、ユーザーがヒントのメッセージをクリックする前後のメール アプリを示しています。

クリック前	クリック後
	

- **UI の変更を示す。**最新バージョンのアプリの UI に大幅な変更が加えられた場合、ユーザーはアプリの新しい機能や変更の詳細についてのヒントを希望することが考えられます。

インストラクショナル UI の設計原則

- **シンプルな状態を保つ。**一度に 1 つの基本的な概念を紹介し、可能な限り図を使います。複雑な機能について説明する場合は、UWP アプリにヘルプ セクションを追加することを検討してください。
- **コンテキストの中で説明する。**インストラクショナル UI をタスクに統合します。そうすれば、ユーザーがタスクを完了しやすくなります。最も必要としているときに紹介された概念は、ユーザーの記憶に残りやすいものです。
- **操作を妨げない。**インストラクショナル UI が表示されている間も、アプリを操作し続けられるようにします。インストラクショナル UI は、ユーザーの邪魔になることなく、有用であることが必要です。
- **説明が終わったら消える。**説明が終わったらすぐにインストラクショナル UI が消えるようにするか、ユーザーが終了できるようにします。 また、ほとんどの場合、インストラクショナル UI は一度表示されれば十分です。繰り返し同じインストラクショナル UI を表示することは避けてください。
- **控え目に使う。**設計とヘルプ セクションが練られていれば、新しいアプリを利用するために把握する必要があることを、ユーザーに十分に伝えることができます。アプリにインストラクショナル UI を追加する前に、さまざまな設計オプションを検討してください。

アイデンティティとセキュリティのガイドライン

このセクションでは、ユーザーを管理する方法と、ユーザーがアカウントに接続する方法について説明します。

ユーザーに対し、アプリからフォルダー、ファイル、データへのアクセスを許可できます。ユーザーがアカウントから、または Microsoft OneDrive などのクラウド サービスで、データにアクセスできるように、ユーザーにサインイン エクスペリエンスを提供できます。

このセクションの内容

トピック	説明
ログイン	多くのユニバーサル Windows プラットフォーム (UWP) アプリは、ユーザーがログインすると、カスタマイズされたエクスペリエンスや初回用エクスペリエンスを提供します。アプリを活用するのに、登録したアカウントへのログインが必須である場合もあります。一方で、どのユーザーにもリッチな基本エクスペリエンスを提供し、ユーザーがログインしたら拡張機能を有効にするアプリもあります。
OneDrive	Microsoft OneDrive にあるユーザーのファイル、ドキュメント、画像、動画、フォルダー、アルバム、コメントを操作する UWP アプリを設計する場合は、次のガイドラインに従ってください。
個人データにアクセスするアプリ	位置情報、カメラ、マイク、連絡先などは、ユーザーの個人データへのアクセスまたはユーザーへの課金が発生する可能性のあるリソースです。そのため、これらは機密性の高いリソースであると見なされます。 Windows 10 では、プライバシー設定を使うことで、ユーザーは機密性の高いリソースへのアクセスを動的に制御することができます。
シングル サインインと接続されているアカウント	Microsoft アカウントを持っているユーザーに対して認証されたエクスペリエンスを提供する、UWP アプリのガイドラインを説明します。
ユーザー名とアカウントの画像	Windows 8 以降の場合、アプリで現在のユーザーの名前と画像 (アカウントの画像) を取得し、それらを使って、ユーザーを識別したうえで、ユーザーに合わせたエクスペリエンスを作成できます。

ログインのガイドライン

多くのユニバーサル Windows プラットフォーム (UWP) アプリは、ユーザーがログインすると、カスタマイズされたエクスペリエンスや初回用エクスペリエンスを提供します。アプリを活用するのに、登録したアカウントへのログインが必須である場合もあります。一方で、どのユーザーにもリッチな基本エクスペリエンスを提供し、ユーザーがログインしたら拡張機能を有効にするアプリもあります。

重要な API

SettignsFlyout クラス (XAML)　　SettingsFlyout オブジェクト (WinJS)

推奨と非推奨

ログイン設定

- アプリの中で、アプリにログインし、アカウントを作り、アカウント設定を管理する手段を提供する場合は、ユーザーがエッジ (端) からスワイプして、設定フライアウトでログイン設定を変更できるようにすることをお勧めします。このような設計にすることにより、ユーザーはアプリ内のワークフローのどこにいても、わかりやすく簡単な方法でアクセスできます。またこの設計では、アプリのキャンバスをログイン関連の UI で占拠しないため、場所を広く使うことができます。

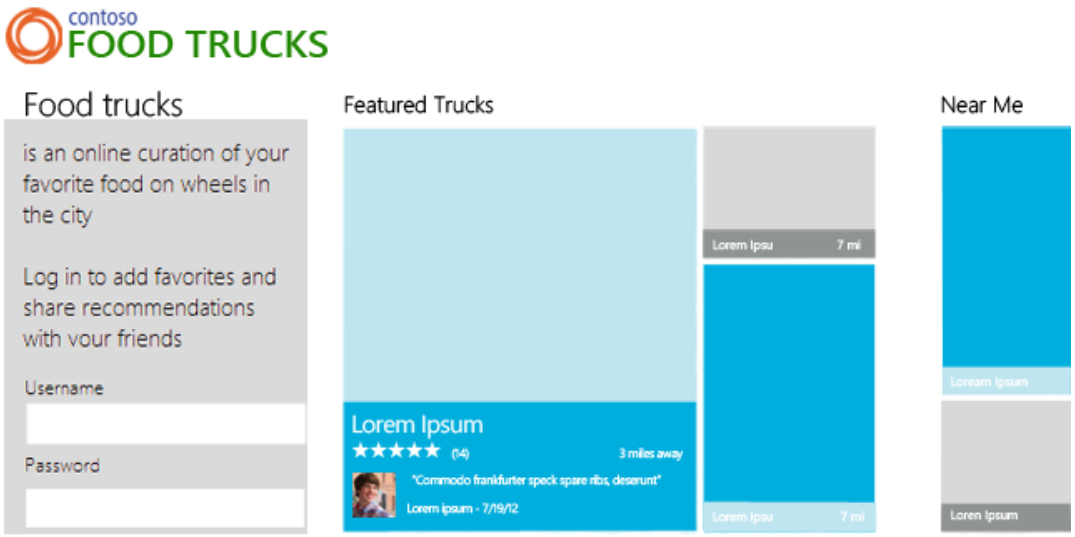
ログインが必須の場合

- アプリにおいて、ユーザーのログイン、またはアプリの初回実行時のアカウント作成を必須とする場合は、アプリの最初の画面でログイン UI を目立つように強調します。ユーザーがログインしたら、画面上にログイン UI を表示する必要はありません。
- ユーザーは設定チャームを使ってログアウトします。

ログインが推奨の場合

- ログイン UI をアプリのキャンバス上に配置する必要がある場合は、コントロールをコンテンツ内にインラインで提供します。このような設計にすると、ユーザーはアプリの初回起動時にランディング ページでログイン オプションを必ず目にすることになりますが、ログイン UI がエクスペリエンス全体を邪魔することはありません。

- ログイン UI を ListView コントロールの最初のセクションとしてアプリのランディング ページに配置します。ユーザーがアプリのコンテンツやビューを閲覧する際、ログオン UI はスクロールによって表示されなくなります。ただし、アプリには存在し続けます。
- 常にユーザーが設定フライアウトでログイン UI を見つけられるようにします。

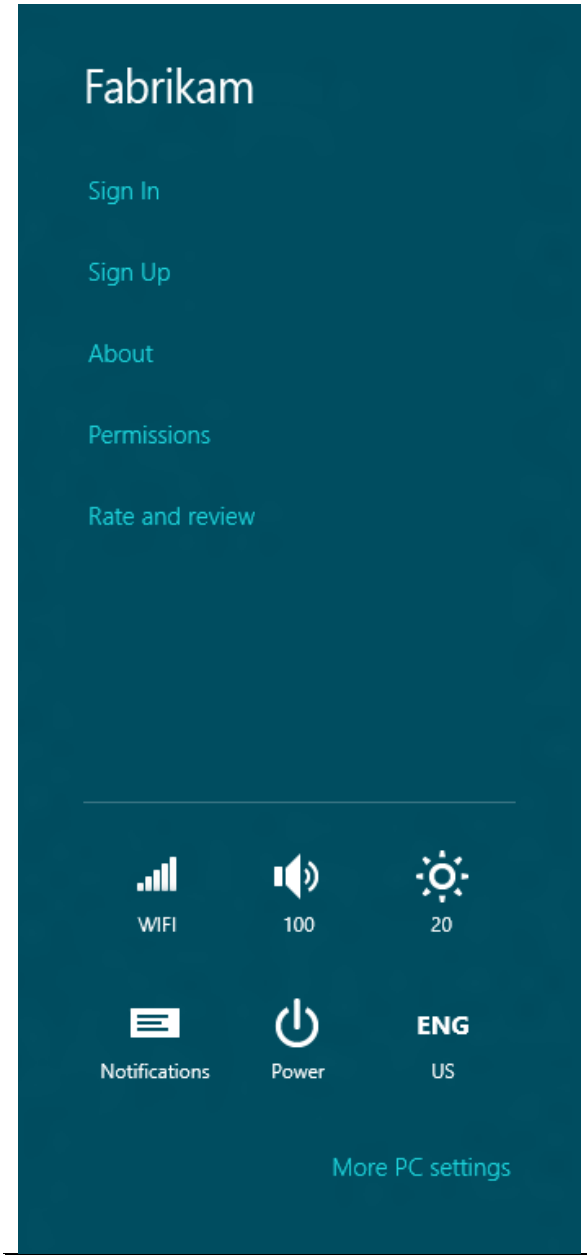


ログイン UI が ListView コントロールの最初のセクションとしてホストされているアプリ

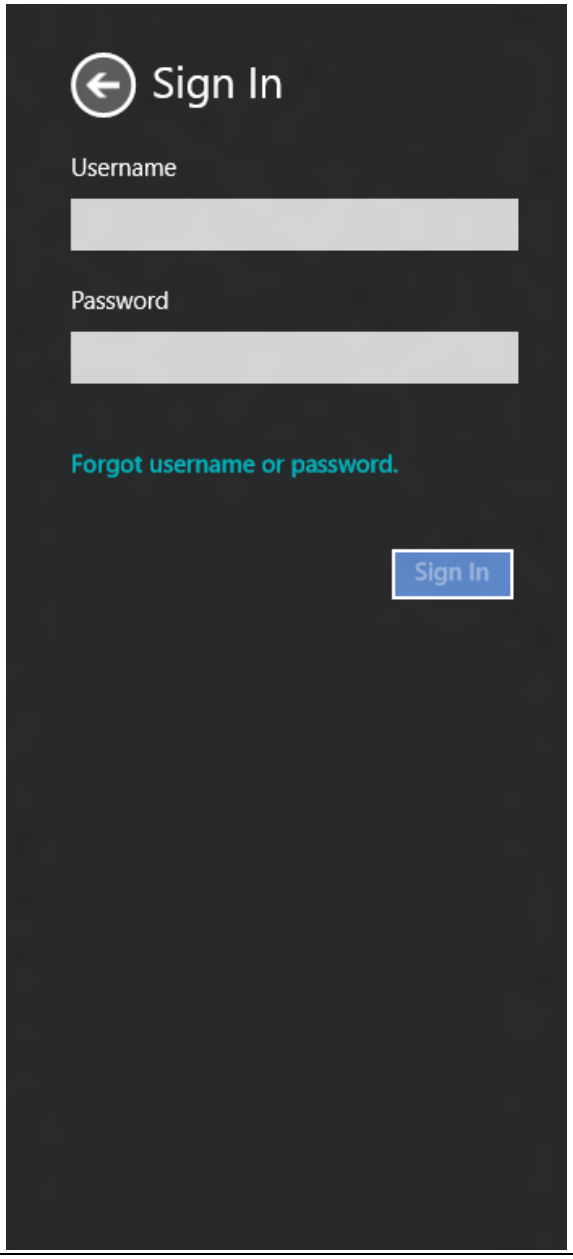
ログインが任意の場合

- アプリへのログインが任意である場合は、ログイン UI を設定フライアウトに配置します。こうすると、ログイン UI のためにアプリのコンテンツからユーザーの注意がそれることや、キャンバス上のスペースが狭くなることがありません。アプリの中には、ユーザーにログインを求めることなく優れた価値を提供するものもあります。たとえば、ニュース アプリでは多くの読者が関心を持つニュース記事の冒頭部分を提供することがあります。ユーザーは、ログインしなくてもアプリから大きな価値を得られます。
- アプリの一部のコンテンツだけに専用のログイン UI が必要な場合は、ユーザーにログインが必要なことを示すために、コンテキストに応じてログイン ボタンをページに表示します。このログイン ボタンで、ログイン UI を提供する設定フライア

ウトを起動します。たとえば、ニュース アプリでユーザーがニュース記事にコメントを投稿するにはログインが必要な場合などです。



設定チャーム フライアウト ログイン



設定フライアウト ログイン

ログアウト UI

- ログアウト UI は設定フライアウトに配置します。アプリに一度ログインしたユーザーは、アプリがそのユーザー向けに有意義なコンテンツを提供している限り、ほとんどログアウトすることはありません。必要に応じて、見慣れた場所でログアウトできるようにします。

ログイン後のアプリのカスタマイズ

- アプリへのログイン後のエクスペリエンスを、アプリをカスタマイズし、接続を持続させるコンテンツを使ってデザインします。
- ユーザーがログインしたら、汎用的なコンテンツを表示するのではなく、そのユーザーの設定に基づいてコンテンツを更新します。各アプリで個人向けのコンテンツを独自の方法で表示し、提供すれば、Windows アプリのユーザー エクスペリエンスを向上できます。
- アプリのキャンバス上に ID を表示する UI を常に配置しておくことは避けます。ユーザーのアプリへのログインと、Windows へのログインは一致しないことがあります。たとえば、自分のものではないノート PC やタブレットを使うユーザーが、ソーシャル ネットワークにログインする場合です。スタート画面とアプリで ID が異なる場合、ユーザーにとってメリットよりも混乱する可能性の方が大きくなります。

アプリから OneDrive へアクセスする場合のガイドライン

Microsoft OneDrive にあるユーザーのファイル、ドキュメント、画像、動画、フォルダー、アルバム、コメントを操作するユニバーサル Windows プラットフォーム (UWP) アプリを設計する場合は、次のガイドラインに従ってください。

重要な API

Live SDK Managed APIs
Live SDK JavaScript APIs
Windows.Storage.Pickers 名前空間

推奨と非推奨

OneDrive のユーザーは、Microsoft がデータのセキュリティとプライバシーを保護することを前提としています。また、重要なドキュメントの保管、写真の保存、友人とのエクスペリエンスの共有に OneDrive を利用します。アプリからデータにアクセスしやすければ、ユーザーにとっての OneDrive の価値が高まります。

ユーザーが OneDrive に対して持っている信頼感を維持するために、次の設計原則に従ってアプリを設計してください。

ユーザーがオプトインできるようにする

ユーザーは、アプリによるデータの処理方法を選べることや、アプリが自分のアカウントに接続する前に、アプリからアクセス許可を求められることを望みます。データが変更される場合は、事前に通知されることも望みます。こうした期待に応えるには、次のプラクティスに従ってください。

- 必ずユーザーによる明示的な要求または選択に基づいて OneDrive にファイルをアップロードする。
  OneDrive に接続するアプリには、ユーザーが自分の意思で OneDrive にファイルをアップロードできるボタンを含める必要があります。アプリが既定で OneDrive にファイルを同期する場合は、データが保存される前にユーザーにこのことを知らせ、オプトインする機会を提供してください。
- アカウントへのユーザーのサインインとサインアウトには、アカウント チャームを使う。

「Microsoft アカウントへのユーザーのサインインとサインアウト」で説明しているように、アプリでは、ユーザーが Microsoft アカウントにアクティブにサインインおよびサインアウトできる手段を提供する必要があります (ただし、ユーザーが既に Microsoft アカウントで Windows にサインインしている場合、アプリが明示的にユーザーをサインアウトさせることはできません)。
詳しくは、「Microsoft アカウントのサインインの要件」をご覧ください。

- サインインしたユーザーが所有しているファイルにのみアクセスする。
  アプリで OneDrive ユーザー間でのファイルの共有を想定していない場合は、サインインしているユーザーのファイルにのみアクセスします。ユーザー自身が共有を選んだ場合にのみ、ユーザーと共有されているファイルやフォルダーにアクセスする必要があります。逆に、ユーザーの許可なく共有フォルダーにファイルを保存することはできません。
- ユーザーが OneDrive にデータを保存する場面について、選択肢を用意する。
  アプリでは、**Windows.Storage.Pickers** 名前空間を通じて Windows ファイル ピッカーを利用し、ファイルをユーザーの OneDrive に保存したり、OneDrive にあるファイルを開いたりできます。アプリが複数のファイルを同期する場合は、ユーザーのフォルダーに一意の名前のサブフォルダーを作成することを検討してください。
  ユーザーが OneDrive からファイルを開くときにファイル ピッカーを使う方法について詳しくは、「フォルダーとファイル」をご覧ください。

ユーザーのデータとプライバシーを保護する

アプリは OneDrive に対するユーザーの信頼を揺るがしてはなりません。ユーザーのデータは個別に扱ってください。ユーザーは、自分が選んだユーザーとのみファイルが共有されるものと考えます。重要な情報は、必要なときに使用できるように保管する必要があります。

重要 OneDrive オブジェクトに対してプログラムで設定したアクセス許可を、後からアプリが変更することはできません。

- ファイルを OneDrive にアップロードするときに、そのユーザーしかアクセスできないように既定で設定する。
  ユーザーがファイルの共有を明示的に要求した場合にのみ、ファイルを他のユーザーと共有します。
- 他のユーザーとのファイルへのリンクの共有に関して警告する。

ユーザーがファイルへのリンクの共有を要求したら、アプリでその結果を知らせるようにしてください。特に、アプリでユーザーがファイルへの事前認証済みリンクを共有できるようにする場合は、そのリンクを受け取ったユーザーはだれでもファイルを確認できるようになることを知らせます。これらのリンクについてはファイルのアクセス許可は評価されず、リンクを開いたユーザーはだれでもコンテンツを表示できます。

詳しくは、「OneDrive の中心となる概念」をご覧ください。

- リンクの用途に基づいて OneDrive オブジェクトへのリンクを作成する。

  可能な限り、埋め込みリンク、読み取り専用リンク、読み取り/書き込みリンクを共有するようにします。これらのリンクは、そのファイルを表示するアクセス許可があるユーザーだけが利用できます。ユーザーが特定のユーザーとフォルダーやファイルを共有する場合にのみ、ファイルへの事前認証済みリンクを提供します。これらのリンクについてはファイルのアクセス許可は評価されず、リンクを開いたユーザーはだれでもコンテンツを表示できます。

  詳しくは、「OneDrive の中心となる概念」をご覧ください。
- 既にあるファイルを上書きする場合、ユーザーに警告します。

  ファイルを OneDrive にアップロードしたときの既定の動作として、同じ名前を持つ既にあるファイルが上書きされます。競合がある場合は、既にあるファイルが上書きされることをユーザーに知らせます。**Overwrite** ヘッダーを追加して "false" に設定すると、既にあるファイルは上書きされません。

意図したとおりに OneDrive と Windows を使う

OneDrive を通じて自由に使える記憶域を、あらゆるデータを保管するクラウド データ ソリューションとして利用することは魅力的です。OneDrive には Windows アプリ向けのさまざまなオプションがありますが、意図したとおりに使われる場合に、アプリに対するメリットが最も大きくなります。OneDrive は、任意のデバイスからドキュメント、写真、その他重要情報にアクセスできるように設計されています。

- ドキュメントの保存、表示、編集や、フォト アルバムの作成と共有に OneDrive を使う。

OneDrive は、スケーラブル データベースの格納、構成ファイルの共有、Web アプリケーションのホストなどを行わずに済む手段として利用されます。 ユーザーの個別のファイルを簡単に格納、共有する目的でのみ使われます。

- ファイルのアップロードの前に OneDrive にスペースがあることを確認する。
  OneDrive のユーザーごとに、使用可能な記憶域の量に制限があります。ユーザーのアカウントに対する割当量を超える場合は、アプリでファイルを保存しようとすると、呼び出しエラーが返されます。OneDrive にファイルを保存する前にユーザーの利用可能な記憶域をチェックすることをお勧めします。
  OneDrive の利用可能な領域をチェックする方法について詳しくは、「一般的なタスク」をご覧ください。
- 組み込みの Windows 機能を使う。
  可能な限り、OneDrive のホストまたは操作には Windows 機能と Windows UI を使います。たとえば、ファイルを開く処理や保存には **Windows.Storage.Pickers** 名前空間で提供されるファイル ピッカーを使います。また、ユーザーのさまざまなデバイスで少量のデータを保存する場合は、Windows アプリケーション データ API を使います。
  OneDrive からファイルを開くときにファイル ピッカーを使う方法について詳しくは、「フォルダーとファイル」をご覧ください。
  Windows アプリケーション データ API を使う方法について詳しくは、「アプリ データとアプリ設定データの保存と読み取り」をご覧ください。

その他の使い方のガイダンス

OneDrive は、ユーザーがファイルをクラウドに保存し、アクセスできる信頼性の高い場所です。ユーザーは、Microsoft アカウントを使ってサインインすると、好きな Windows デバイスから OneDrive にある自分のファイルにアクセスできます。OneDrive では 7 GB の記憶域が無料で提供され、写真、ドキュメント、動画、オーディオ ファイルの保存と共有に利用できます。

Windows アプリは、OneDrive にあるファイルやフォルダーへのアクセスをユーザーに提供することができます。OneDrive への接続を利用すると、ハード ディスクを散らかさずに、OneDrive にあるファイルを操作 (開く、読み取り、保存、ダウンロード) できるアプリを開

発できます。OneDrive API は、Windows アプリからの使用を想定して設計され、アプリの設計にスムーズに統合されます。

OneDrive を使うための設計

広い意味で、OneDrive は、個々のファイルとやり取りするすべてのアプリの役割を果たすことができます。アプリにファイルの操作機能 (読み取り、表示、保存、ダウンロード、開く) がある場合は、アプリの設計に OneDrive を追加できます。OneDrive は、追加のコードを大量に記述しなくても、Windows の組み込みの機能を利用して、Windows アプリのアーキテクチャとうまく統合できます。

重要 OneDrive API は Live Connect SDK に含まれています。OneDrive に接続する Windows アプリの開発を始める前に、Live Connect SDK をインストールし、プロジェクトに SDK への参照を追加する必要があります。

- Live Connect SDK をダウンロードするには、Live Connect SDK のダウンロード ページにアクセスしてください。
- OneDrive API のドキュメントを確認するには、「OneDrive for Developers (開発者向け OneDrive)」と「OneDrive API」をご覧ください。

Microsoft アカウントへのユーザーのサインインとサインアウト

もちろん、OneDrive とやり取りするどのアプリでも、OneDrive に関連付けられた Microsoft アカウントを使ってユーザーがサインインとサインアウトを実行できるようにする必要があります。アプリ自体の設計ではないものの、アカウントへのユーザーのサインインは、OneDrive と統合されるアプリを作成するうえで重要な手順です。

ユーザーのサインインに関しては、アプリの設定チャームにアカウントのページとプライバシーに関する声明のページを作成することをお勧めします。アカウント ページには、ユーザーがアカウントへのサインインとサインアウトを行うためのボタンを設けます。アプリの残りのサインイン プロセスは Windows UI で処理します。

詳しくは、次のリソースをご覧ください。

- Microsoft アカウントへのユーザーのサインインについて詳しくは、「ユーザーのサインイン」をご覧ください。

OneDrive への新しいファイルの保存と既にあるファイルの更新

一部のユーザーにとっては、OneDrive が "マイ ドキュメント" です。ファイルの保存に OneDrive を使うユーザーのために、アプリでは OneDrive にユーザーのデータを保存するオプションを提供できます。たとえば、ユーザーがアプリで新しいファイルを作成したときに、保存場所として OneDrive を提供できます。 ユーザーがアプリでファイルを編集した場合は、その編集内容も OneDrive に保存できます。

実際のところ、ユーザーが新しいファイルを作ることができるアプリでは、OneDrive へのユーザー アクセスを提供することには大きな意味があります。

- OneDrive と統合されるアプリを作成する方法のガイドラインについては、「推奨と非推奨」をご覧ください。
- ユーザーの OneDrive から画像、動画、オーディオ ファイルをアップロードする方法について詳しくは、「アルバム、写真、動画、オーディオ、タグ」をご覧ください。
- ユーザーの OneDrive でファイルを保存および更新する方法について詳しくは、「フォルダーとファイル」をご覧ください。

OneDrive からのファイルのダウンロード、オープン、表示

既に述べたように、一部のユーザーはクラウドに多くのデータを保存します。ユーザーはクラウドのデータを表示できることを望みます。アプリでは、OneDrive からファイルを開いて読むオプションを提供できます。アプリは、ユーザーが確認するファイルのコンテンツをダウンロードして開き、表示できます。

たとえば、動画を再生するアプリの場合、ユーザーが OneDrive のフォルダーから映画を開く機能を提供できます。ユーザーが特定の種類のファイルを開いて表示できるリーダー アプリもあります。

注 Windows アプリでは、OneDrive のファイルを表示するだけにとどまらない機能を提供することをお勧めします。Windows には最初から OneDrive アプリがインストールされています。アプリに独特の機能があれば、ユーザーにダウンロードしてインストールしてもらえる可能性が高まります。

詳しくは、次のリソースをご覧ください。

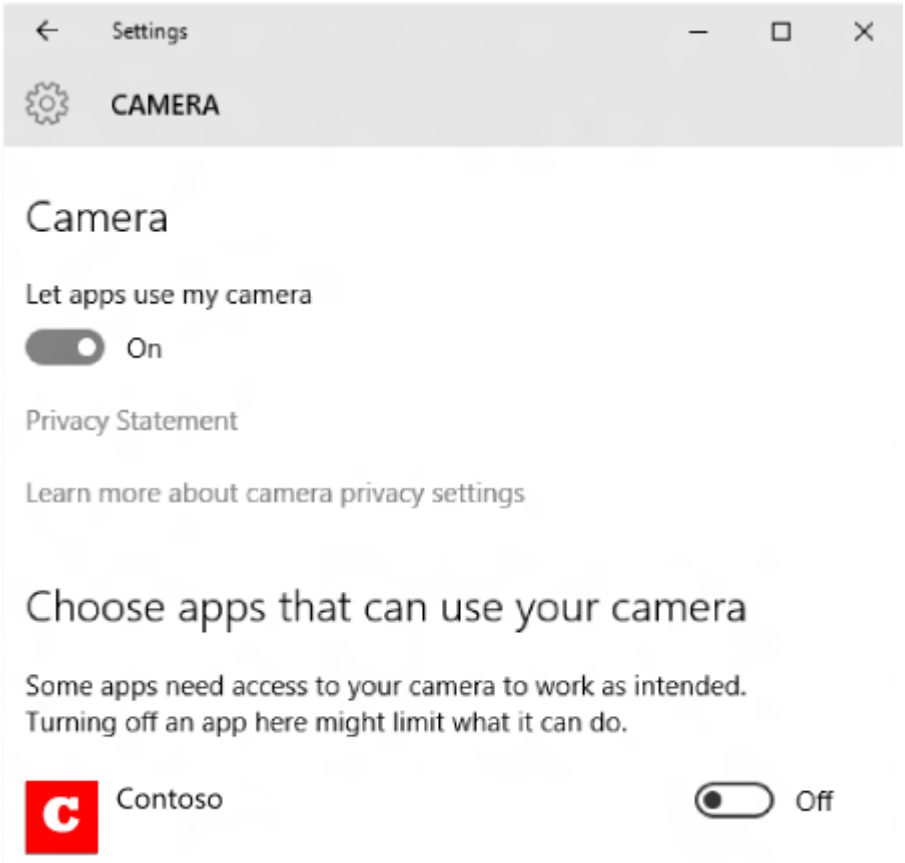
- OneDrive と統合されるアプリを作成する方法のガイドラインについては、「推奨と非推奨」をご覧ください。
- ユーザーの OneDrive から画像、ビデオ、オーディオ ファイルをアップロードする方法について詳しくは、「アルバム、写真、動画、オーディオ、タグ」をご覧ください。
- ユーザーの OneDrive でファイルを保存および更新する方法について詳しくは、「フォルダーとファイル」をご覧ください。

個人データにアクセスするアプリのガイドライン

位置情報、カメラ、マイク、連絡先などは、ユーザーの個人データへのアクセスまたはユーザーへの課金が発生する可能性のあるリソースです。そのため、これらは機密性の高いリソースであると見なされます。 Windows 10 では、プライバシー設定を使うことで、ユーザーは機密性の高いリソースへのアクセスを動的に制御することができます。

プライバシー設定は**設定**アプリで管理します (**プライバシー設定**のページをご覧ください)。機密性の高いリソースは、いつでもオフにできます。また、これらのリソースへのアクセスは、いつでも無効にすることができます。このため、このような変更が発生した場合にアプリが適切に処理できるようにしておく必要があります。アプリでは、機密性の高いリソースに常にアクセスできることを前提にすることはできません。

たとえば、次の図では、Contoso アプリからのアクセスが拒否され、その他すべてのアプリからのアクセスが許可されるように、カメラのプライバシー設定が構成されています。この状態のとき、Contoso アプリでは、Webcam 機能が指定済みであっても、カメラにアクセスできません。Contoso アプリから見て、カメラは存在していない場合と同じ状態になります。



機密性の高いリソースへのアクセス許可は、ユーザーごと、アプリごと、リソースごとに制御します。つまり、2 人のユーザーは同じアプリについて別々にプライバシー設定を指定でき、各ユーザーが同じアプリに別々のデバイスで別々のアクセス許可を付与できます (たとえば、PC でのアクセス許可と電話でのアクセス許可を異なる設定にすることができます)。

重要: 機密性の高い各リソースには、対応するアプリ機能への 1 対 1 のマッピングが存在します。アプリ機能により、アプリでは機密性の高いリソースにアクセスするための API を使うことができます。 アプリ機能と対応するリソースの種類について詳しくは、「アプリ機能の宣言」をご覧ください。

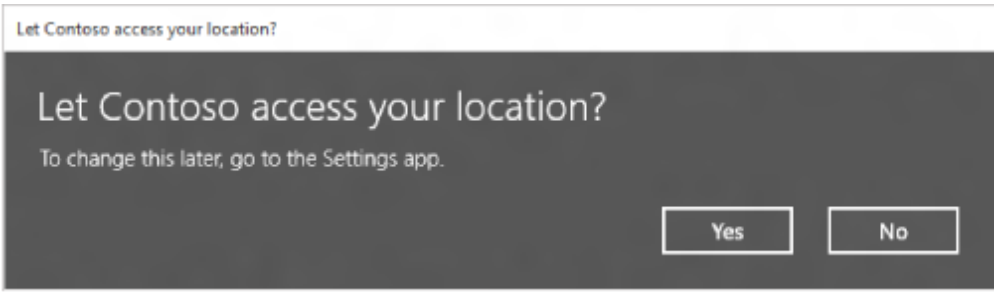
推奨と非推奨

- 独自のプロンプト UI は作成しないでください (Windows Phone 8 では推奨されていました)。同じリソースに関するプロンプト (独自のプロンプト UI と Windows 10 からのプロンプト) が重複するおそれがあります。
- 独自のオン/オフ トグルを作成しないでください (Windows Phone 8 と 8.1 では、機密性の高い一部のリソースについては必須でした)。

- 機密性の高いリソースには、必要になるまでアクセスしないでください。
- 機密性の高いリソースが利用可能であることや、使用するためのアクセス許可がアプリにあることを前提にしないでください。
- 使用を試みる前に、機密性の高いリソースへのアクセスを確認します。
- 機密性の高いリソースへのアクセスが拒否された場合に備えておく必要があります。アクセス拒否の処理は、機能によって異なる場合があります。 詳しくは、「アクセスが拒否された場合の処理」をご覧ください。
- 機密性の高いリソースへのアクセスを要求するための API が存在する場合は、アクセスする前にその API を使用します。詳しくは、「リソースへのアクセス許可を求めるプロンプト」をご覧ください。
- 機密性の高いリソースへのアクセスが拒否された場合は、設定アプリ内の適切な設定ページに移動できるリンクを提示します。詳しくは、「Windows 設定アプリの起動」をご覧ください。

リソースへのアクセス許可を求めるプロンプト

位置情報など、一部のリソースについては、そのリソースにアクセスする前に、アプリがユーザーにアクセス許可を求める必要があります。UI プロンプトは Windows から提供されていますが、アプリで **RequestAccessAsync** または同様の API を呼び出すことによってトリガーする必要があります。ユーザーの位置情報へアクセスを要求している Contoso という名前のアプリの例を次に示します。



この例では、ユーザーの位置情報にアクセスする前に、アプリで **RequestAccessAsync** を呼び出しています。アクセスを要求するときには、アプリをフォアグラウンドで実行し、**RequestAccessAsync** を UI スレッドから呼び出す必要があります。リソースに対するアクセスをユーザーがアプリに許可するまで、アプリは位置情報データにアクセスできません。

ユーザーは、プロンプトでアクセス許可を付与しても、その設定をいつでも変更できます。アクセス許可は、リソースへのアクセスを試みる前に常に確認します。アクセス許可を確認するための API を利用できない場合は、アクセスが拒否された場合のエラー処理を使用します。

プロンプトを必要とする機密性の高いリソース

以下に示す機密性の高いリソースにアプリが初めてアクセスしようとすると、アクセス許可をユーザーに求めるプロンプトがオペレーティング システムによって表示されます。これらの各リソースは 1 つ以上の地域でプロンプトが表示されます。

機密性の高いリソース	アプリ機能	設定アプリの URI スキーム
位置情報	location	ms-settings:privacy-location
カメラ	webcam	ms-settings:privacy-webcam
メッセージング	cellularMessaging	ms-settings:privacy-messaging
録音	microphone	ms-settings:privacy-microphone
連絡先	contacts	ms-settings:privacy-contacts

重要　地域によって、求められるプロンプトの数が異なります。ローカルの地域でプロンプトが必要でない場合も、常にアクセス要求 API を (利用可能であれば) 使用してください (アクセス要求 API は、リソースにアクセスする前に、フォアグラウンド アプリの UI スレッドから呼び出します)。

アクセスが拒否された場合の対処方法

機密性の高い各リソースへのアクセスは、**設定アプリ**で管理します。リソースはいつでもオフにできます。また、アプリからのアクセスは、いつでも拒否できます。アプリからリソースへのアクセスが拒否された場合の動作は、リソースの種類によって異なります。

アクセス拒否の処理は、機能によって異なる場合があります。 アプリのテスト時には、**設定アプリ**を使って、リソースへのアクセスを許可および拒否することで、アプリの反応が適切かどうかを確認してください。

シングル サインオンと接続されているアカウントのガイドライン

このトピックでは、ユニバーサル Windows プラットフォーム (UWP) アプリで Microsoft アカウントを使用するユーザーの認証されたエクスペリエンスについて説明します。

ユーザー認証シナリオ

ユーザーは、Microsoft アカウントの資格情報を使って、Windows 10 を実行しているデバイスにサインインできます。このとき Windows 10 では、Windows ストア アプリを使ってユーザーに対して認証されたエクスペリエンスを有効にできます。これらのエクスペリエンスには次のものがあります。

- ユーザーは最もよく使うオペレーティング システム設定を Microsoft アカウントと関連付けることができます。これらの設定は、Windows 10 を実行していてクラウドに接続しているデバイスに、ユーザーがそのアカウントを使ってサインインするときは常に使用できます。ユーザーがサインインすると、デバイスはユーザー設定をクラウドから自動的に取得することを試みてから、そのユーザー設定をデバイスに適用します。
- Windows アプリではユーザー固有の設定を格納できるため、Windows 10 を実行する任意のデバイスで設定をローミングできます。オペレーティング システム設定と同じように、このようなユーザー固有のアプリ設定は、クラウドに接続している Windows 10 を実行中のデバイスに、ユーザーが同じ Microsoft アカウントを使ってサインインするときは常に使用できます。ユーザーがサインインすると、アプリがインストールされている場合、そのデバイスは設定をクラウドから自動的に取得し、デバイスに適用します。
- Windows 10 では、ユーザーは Microsoft アカウントを任意のアプリまたは Web サイトのサインイン資格情報と関連付けることができるため、Windows 10 が実行されるすべてのデバイスでその資格情報をローミングできます。ユーザーがそのアカウントを使って Windows 10 が実行中のデバイスにサインインして、アプリを実行するか Web サイトを訪問した場合、そのユーザーに対応するサインイン資格情報が格納されていれば、Windows 10 によって自動的にそのユーザーのサインインが試行されます。

- ユーザーが Microsoft アカウントを使って Windows 10 が実行中のデバイスにサインインするとき、やはり認証に Microsoft アカウントを使うデバイスで実行されているアプリやサービスは、そのユーザーの Microsoft アカウントを使ってサインインし、そのユーザーが共有することを同意したデータを取得することができます。

Microsoft アカウント認証 API を使う状況

次の質問に "はい" と答えれば答えるほど、アプリ (およびアプリのコンパニオン Web サイト) と Microsoft アカウント認証 API の統合を検討する必要があります。

- 使っているアプリが Windows アプリか
- アプリはユーザーに合わせたエクスペリエンスを提供するか
- アプリは、独自のクラウド サービスや、Outlook.com、Microsoft OneDrive、Windows Live Messenger のような Microsoft クラウド サービスにアクセスするか
- アプリでユーザー認証システムを提供する必要があるものの、自分で作るための時間、知識、インフラストラクチャがないか

Microsoft アカウント認証 API のメリットを受けられる、アプリとコンパニオン Web サイトのカテゴリがいくつかあります。

- **独自のクラウド サービスにアクセスし、ユーザー認証が必要なアプリ。**アプリがクラウド サービスにアクセスし、ユーザーを認証する必要がある場合、コードはアプリまたは Web サイトがユーザーの代わりにクラウド サービスにアクセスできるようにする認証トークンを要求できます。Microsoft アカウント認証 API は、使用するクラウド サービスに対応する JavaScript Object Notation (JSON) 形式の認証トークンを生成します。それぞれの認証トークンには、アプリ固有のユーザー ID が含まれています。Microsoft アカウント認証 API を使って特定のクラウド サービスで使用する JSON 形式の認証トークンを生成するには、「OnlineIdServiceTicketRequest」をご覧ください。
- **Outlook.com、OneDrive などの Microsoft クラウド サービスにアクセスするアプリとコンパニオン Web サイト。**アプリまたはアプリのコンパニオン Web サイトが Outlook.com や OneDrive のユーザー データにアクセスする場合は、Live

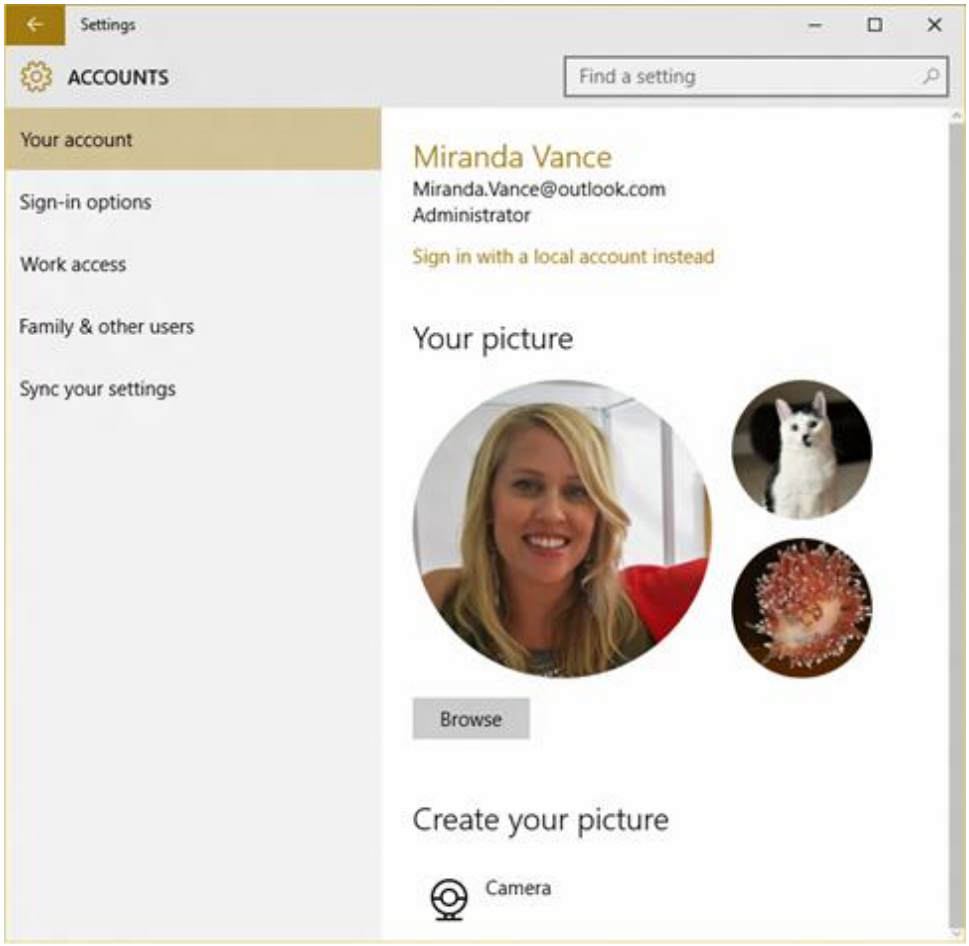
Connect API によって認証トークンの複雑さが整理され、クラウド サービスを操作するコードをある程度簡単に記述できるようになります。

UWP アプリへのユーザー認証機能の追加

Windows アプリ用に一貫したユーザー サインインおよびサインアウトのコードを作るには、「Microsoft アカウントのサインインの要件」と「ログインのガイドライン」をご覧ください。

注意 Windows 10 デバイスでは、ユーザーは Microsoft アカウントに関連付けられた (または接続された) ローカルまたはドメインの Windows アカウントを使ってそのデバイスにサインインできます。その後、ユーザーはサインイン用のこの特定の Microsoft アカウントに依存する Windows ストア アプリを実行できます。その場合、前に示したトピックで説明したガイドラインを使って後でアプリからユーザーをサインアウトしようとしても、ユーザーはそのアプリから正常にサインアウトされません。これを防ぐには、アプリでユーザーがサインアウトされないようにするために、サインアウト コマンドを非表示にする必要があります。代わりに、ユーザーは次の 2 つの方法でアプリからサインアウトすることができます。

- ユーザーは Microsoft アカウントのローカルまたはドメインの Windows アカウントとの関連付けを解除 (または切断) することができます。これを行うには、**[設定]** (次のスクリーン ショットをご覧ください) で **[アカウント]**、**[お使いの アカウント]**、**[ローカル アカウントでのサインインに切り替える]**、**[完了]** の順にタップします。



- ユーザーは他のアカウントを使うように切り替えることができます。これを行うには、**[スタート]** 画面で、アカウントの画像、**[アカウントの切り替え]** の順にタップした後、他のアカウントの資格情報を使ってサインインします。

ユーザーがこのどちらかの操作を実行すると、サインイン用の特定の Microsoft アカウントに依存するすべての UWP アプリと Windows ストア アプリからも自動的にサインアウトされます。

ユーザーに制御を委ねる

Windows 10 を実行中のデバイスに、Microsoft アカウント (または Microsoft アカウントに接続されているローカルまたはドメインのユーザーの Windows アカウント) を使ってサインインすると、ユーザーは Microsoft アカウント認証 API を使うアプリまたはコンパニオン Web サイトがユーザーのために操作できる範囲を制御します。たとえば、ユーザーが Outlook.com や OneDrive のような Microsoft クラウド サービスにアクセスする参加アプリにサインインすると、アプリまたはアプリのコンパニオン Web サイトが特定のクラウド サ

ービスからユーザーの関連データまたはファイルにアクセスすることを許可する (同意する) ように求めるメッセージが表示されます。

しかし、アプリが Outlook.com や OneDrive などの Microsoft クラウド サービスのデータにアクセスするのではなく、独自のクラウド サービスにアクセスする場合は、ユーザーが **[設定]** の **[プライバシー]** でこの設定を明示的に変更していない限り、同意を求めるメッセージは表示されません。それでも、参加アプリおよび Web サイトは、Microsoft アカウント認証 API を使ってアプリまたは Web サイト独自のユーザー ID を取得できます。そのため、アプリまたは Web サイトはそのユーザー ID にデータを関連付けることができます。次にユーザーがサインインすると、アプリまたは Web サイトは同じユーザー ID を取得し、そのユーザー ID を使って既にユーザーに関連付けられているデータをすべて取得できます。

Windows 10 では、ユーザーのアカウントに関連付けられたアプリの名前、画像などのデータに対するアクセスを、ユーザーがすべてのアプリで制限できます。これを行うには、**[設定]** (次のスクリーン ショットをご覧ください) で、**[プライバシー]** をタップし、**[アカウント]** で **「自分の名前、画像、その他のアカウント情報にアプリがアクセスすることを許可する」** を **[オフ]** にスライドします (このオプションは既定ではオンになっています)。

ユーザーのサインインとサインアウト ステータスの変化への応答

Microsoft アカウント認証 API を使うアプリは、ユーザーがその Microsoft アカウントをローカルまたはドメインの Windows アカウントに接続または切断するたびに適切に応答する必要があります。ユーザーがこれを行うと、**ConnectedStateChange** バックグラウンド タスクが開始されます。 このタスクが開始されるたびに、参加アプリはユーザーのアプリ用の ID がクリアされたかどうかをチェックする必要があります。

- ユーザーのアプリ用の ID がクリアされている場合、アプリではまず、ユーザーの関連するすべてのタイル通知をクリアする必要があります。 アプリでユーザーがサインイン中であるかどうかを表示している場合、アプリの状態を変更して、ID がクリアされたユーザーがもうサインインしていないことを示す必要があります。ただし、アプリを既定の状態にリセットしないでください。代わりに、ユーザーがサインアウト状態でなければ、ユーザーはまだアプリを使っていて、作業を中断した時点から再開するつもりであると想定する必要があります。 唯一の違いは、サインアウトしたユーザーはタイル通知などのクラウド サービスにアクセスできなくなる点です。 この方法を使った場合の動作はアプリによって異なることに注意してください。使い続けるためにユーザーに再サインインを求める必要があるアプリもありますが、サインアウト状態でも動作を継続できるアプリもあります。

  注 サインアウトしたユーザーのためにアプリ固有のデータをクラウドに格納しないアプリの場合、動作はアプリによって異なります。そのユーザーのアプリ固有のデータが表示されないように、すべてクリアすることをお勧めします。ただし、ユーザーが Microsoft アカウントとローカルまたはドメインの Windows アカウントの関連付けを再接続する場合に備えて、アプリでこのデータをローカルに保存する必要があります。
- ユーザーのアプリ用の ID がクリアされていない場合、ユーザーはアプリで既に使っているのと同じアカウントに接続されているため、これ以上アプリで実行する必要があることは何もありません。

ConnectedStateChange バックグラウンド タスクについて詳しくは、**OnlineIdConnectedStateChange** システム イベントの **SystemTriggerType** 列挙に関するページをご覧ください。

ユーザーのアプリ用の ID がクリアされたかどうかのチェック方法について詳しくは、**authenticatedSafeCustomerId** プロパティに関するページをご覧ください。

ユーザー名とアカウントの画像のガイドライン

Windows 10 の場合、アプリで現在のユーザーの名前と画像 (アカウントの画像) を取得し、それらを使って、ユーザーを識別し、ユーザーに合わせたエクスペリエンスを作成できます。たとえば、メッセージング アプリで、名前とアカウントの画像を使って、ユーザーを会話の参加者として識別することや、ゲーム アプリで、それらを使って、ゲームのスコアボード ページでユーザーをプレイヤーとして識別することなどができます。

重要な API

UserInformation クラス

推奨と非推奨

- アプリケーションのインターフェイスに適した画像サイズを使います。
- 状態は、ステータス バーを使ってアカウントの画像のごく一部として表示します。
- アカウントの状態のアイコンには、背後に表示されるアカウントの画像に紛れないようなアイコンを使ってください。

その他の使い方のガイダンス

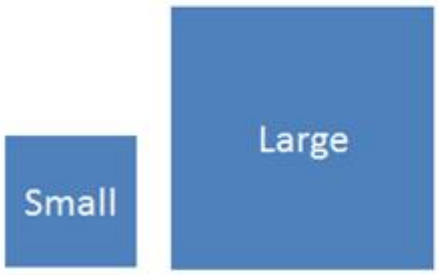
画像を撮影できるアプリの場合は、アプリをアカウントの画像プロバイダーとして宣言することを検討してください。これを行うと、アプリが **[設定]** の **[アカウント]** にある **[お使いのアカウント]** ページに表示されます。ここからアプリを選んで、新しいアカウントの画像を作成できます。

詳しくは、「アプリ コントラクトと拡張機能」と「**UserInformation**」をご覧ください。
アカウントの画像名のサンプルに関するページもご覧ください。

アカウントの画像のサイズ

アプリでは、アカウントの画像を小さな画像、大きな画像、またはビデオ (動的な画像) として取得できます。画像は、最大 DPI プラトーで適切に表示されるサイズに調整されます (1.8 倍)。

Small	Large
96x96	448x448



Plateau	Small	Large
1.0	48	224
1.4	67	314
1.8	86	403

アカウントの画像のビデオは、フレーム サイズが 448x448、最長時間が 5 秒、最大サイズが 5 MB です。

アカウントの画像が特定のエクスペリエンスの中心部分ではない場合、たとえば多数のユーザーを一覧表示するときには、小さな画像を使います。アプリ領域内でごく一部のユーザーを識別する必要がある場合や、各ユーザーを明確に識別する必要がある場合は、大きな画像を使います。たとえば、メッセージング アプリの会話の参加者や、電話の着信の呼び出し元の画像を表示する場合に、大きな画像を使います。

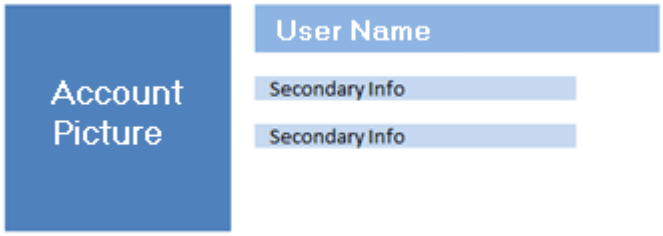
ユーザー名

アプリでは、**UserInformation** クラスのメンバーを使って、ユーザー名を取得できます。ユーザーの姓と名は、ユーザーの表示名と同じように、別々に使うことができます。表示名は、ロケールに適した順番になるように自動的に書式設定されます。

アカウントの画像とユーザー名の同時表示

アカウントの画像とそれに対応するユーザー名を一緒に表示する際は、次のいずれかの形式を使うことをお勧めします。

- **左右に並べる**—十分なスペースがある場合は、この形式がユーザーにとって最も読みやすいため、アカウントの画像とユーザー名を左右に並べて表示します。 次の例は、お勧めの要素の配置を示しています。ユーザー名がアカウントの画像の上端に揃えて配置され、補助的な情報がユーザー名よりも小さいフォント サイズで表示されています。



- **上下に並べる**—スペースが限られている場合は、アカウントの画像の下にユーザー名を表示します。ただし、ユーザー名全体が 1 行に収まらない場合があるため、注意してください。ユーザー名がアカウントの画像の左端に揃えて配置され、補助的な情報がユーザー名よりも小さいフォント サイズで表示されています。

状態の表示

アプリでアカウントの画像と状態情報を組み合わせる必要がある場合は、ステータス バーまたはアイコン オーバーレイ を使うことをお勧めします。ステータス バーの方が画像に被らず、目立つため、お勧めです。ステータス バーには、アカウントの画像と簡単に区別できるだけの幅が必要です。 ステータス バーの例を次に示します。

アイコン オーバーレイを使うと、もっと複雑な状態を伝えることができます。アイコン オーバーレイは、画像の細部に埋もれる可能性があるため、さまざまな背景ではっきり目立つアイコンを使うようにしてください。アイコン オーバーレイの例を次に示します。

起動、中断、再開のガイドライン

このセクションでは、魅力的な起動エクスペリエンスの作成と、ユーザーが切り替えたときに一時停止し、元に戻すと再開するアプリの設計に関するガイドラインを示します。

このセクションの内容

トピック	説明
アプリの起動と中断	ユニバーサル Windows プラットフォーム (UWP) アプリの一時停止と再開の動作を設計するときは、次のガイドラインに従ってください。
スプラッシュ スクリーン	ユーザーに快適な起動エクスペリエンスを提供するために、スプラッシュ スクリーンをカスタマイズし、スプラッシュ スクリーンを拡張するには、以下のガイドラインに従ってください。

アプリの中断と再開のガイドライン

ユーザーが切り替えたときに一時停止し、元に戻すと再開するアプリを設計します。アプリの目的と使用パターンを慎重に検討して、アプリが一時停止および再開されるときに可能な限り最適な操作性を実現できるようにします。 ユニバーサル Windows プラットフォーム (UWP) アプリの一時停止と再開の動作を設計するときは、次のガイドラインに従ってください。

ユニバーサル Windows プラットフォーム (UWP) アプリのライフサイクルの概要については、「アプリのライフサイクル」を参照してください。

重要な API

ApplicationExecutionState 列挙

Windows.ApplicationModel 名前空間

Windows.ApplicationModel.Activation 名前空間

Windows.ApplicationModel.ExtendedExecution 名前空間

注　Windows 8.1 と Windows Phone 8.1 でシステムの応答性を向上させるために、アプリには中断後にリソースへの優先度の低いアクセス権が与えられます。この新しい優先度をサポートするために、中断操作のタイムアウトが延長され、アプリには通常の優先度と同程度の

タイムアウト (Windows では 5 秒、Windows Phone では 1 ～ 10 秒) が与えられます。 このタイムアウトの時間枠を延長したり、変更したりすることはできません。 Windows 10 では、タイムアウトの時間枠を延長する API を提供します。

推奨と非推奨

- 短時間の後に再開した場合は、中断したときのアプリの状態に戻します。たとえば、ユーザーが電子メールを書いている途中で別のアプリに移動し、電子メール アプリに戻った場合、メール アプリのメイン ランディング ページではなく、書きかけのメールのページに戻る必要があります。
- 長時間の後に再開した場合、ユーザーはアプリの既定のランディング ページに戻ります。たとえば、時間がたった記事や以前の天候データの表示に戻るのではなく、ニュースや天気予報アプリの主なランディング ページに戻ります。
- 適切な場合は、アプリの以前の状態を復元するか、アプリを新たに開始するかをユーザーが選択できるようにします。たとえば、ゲームに戻るときに、ゲーム アプリを再開するか、新たに開始するかを選べるようにプロンプトを表示できます。
- アプリを一時停止するときにアプリ データを保存します。一時停止中のアプリは、システムによって終了されても通知を受け取りません。したがって、アプリの状態を復元できるように、データを明示的に保存することが重要です。
  アプリでセカンダリ タイルなどの複数の起動ポイント、トースト通知、ファイルと URI の関連付けをサポートしている場合は、それぞれ起動ポイントに別々のナビゲーション履歴を作成することを検討してください。中断時には、プライマリ起動ポイントに関連付けられた状態を保存し、状態が失われることにユーザーが不満を感じるシナリオでのみセカンダリ起動ポイントの状態を保存します。保存する状態が多すぎると、アプリの再開が遅くなる可能性があります。
- 保存されたアプリ データを使ってアプリを復元します。
- アプリが中断されているときに、排他リソースとファイル ハンドルを解放します。前に説明したように、一時停止中のアプリは終了時に通知を受け取らないため、アプリを一時停止する際は他のアプリからアクセスできるように、リソースとハンドル (Web カメラ、I/O デバイス、外部デバイス、ネットワーク リソースなど) を確実に解放します。

- ユーザーが最後に表示した後にコンテンツが変更された場合は、UI を更新します。再開されたアプリは、ユーザーが切り替えている間も実行されていたように表示されます。
- アプリが画面から消されても終了しないでください。オペレーティング システムでは、ユーザーは一貫した方法でアプリにアクセスして管理できます。アプリは画面から消されると一時停止します。アプリのライフサイクルをシステムにゆだねて、ユーザーができるだけ効率的にアプリに戻れるようにします。これにより、システムのパフォーマンスとデバイスのバッテリ寿命も最適な状態になります。
- ユーザーによって明示的に閉じられたアプリの状態は復元しないでください。回復不能の状態におちいったために、ユーザーがアプリを閉じた可能性があります。アプリがユーザーによって明示的に閉じられた場合は、再開のエクスペリエンスではなく新しいエクスペリエンスを提供します。アプリがユーザーによって閉じられた場合、**PreviousExecutionState** プロパティの値が **ClosedByUser** になります。
- クラッシュが発生したために終了したアプリの状態は復元しないでください。アプリが予期せずに終了した場合、保存されているアプリ データは壊れている可能性があります。この格納されたデータを使ってアプリを以前の状態に復元しないようにします。
- [閉じる] ボタンを表示するなど、ユーザーが UI からアプリを終了できる方法を提供しないようにします。ユーザーは、システムがアプリを管理しているという安心感を持っています。システムはアプリを自動的に終了することで、最良のシステム パフォーマンスと信頼性を確保できます。またユーザーは Windows でジェスチャを使うか、Windows Phone でタスク スイッチャーを使って、アプリを閉じることができます。
- ディープリンクされたページに、ユーザーが取り残されることがないようにします。ユーザーがプライマリ タイル以外の起動ポイントからアプリを起動し、ディープ リンクされたページにランディングした場合でも、ユーザーがアプリのトップ ページに移動できるように UI を用意します。または、プライマリ タイルをタップしてトップ ページにアクセスできるようにします。

スプラッシュ スクリーンのガイドライン

ユーザーに快適な起動エクスペリエンスを提供するために、スプラッシュ スクリーンをカスタマイズし、スプラッシュ スクリーンを拡張するには、以下のガイドラインに従ってください。

重要な API

SplashScreen クラス　　SplashScreen.Dismissed イベント

推奨と非推奨

- スプラッシュ スクリーンをカスタマイズして、アプリを特徴付けます。
  スプラッシュ スクリーンは、画像と背景色で構成され、どちらもカスタマイズできます。スプラッシュ スクリーンが適切にデザインされていると、アプリがより魅力的なものになります。
  画像と背景色を組み合わせてスプラッシュ スクリーンを作ると、アプリがインストールされているデバイスのフォーム ファクターに関係なく、スプラッシュ スクリーンが適切に表示されます。スプラッシュ スクリーンが表示されるとき、背景のサイズだけがさまざまな画面サイズに合わせて変更されます。画像のサイズは常に変わりません。
  このスプラッシュ スクリーンを拡張してカスタマイズする方法については、「クイック スタート: スプラッシュ スクリーンの追加」をご覧ください。
- アプリのランディング ページを表示する前に追加のタスクを完了できるように、スプラッシュ スクリーンを拡張します。
  Windows で表示されるスプラッシュ スクリーンに似たアプリのスプラッシュ スクリーン ページを作成することで、アプリの読み込みエクスペリエンスをさらに制御できます。システムで表示されるスプラッシュ スクリーンに似せることで、スムーズで内容の伝わる読み込みエクスペリエンスをユーザーに提供できます。アプリの UI の準備やネットワーク データの読み込みに時間がかかる場合、アプリがこれらのタスクを完了している間、スプラッシュ スクリーンを拡張することでユーザーにメッセージを表示することができます。

Windows ストアのスプラッシュ スクリーンの拡張ページを次に示します。この画面は初期スプラッシュ スクリーンと同じですが、"不定リング" プログレス コントロールが追加してあり、アプリが読み込み中であることがユーザーに示されます。

不定リングなどのプログレス コントロールの概要については、「プログレス コントロールのガイドライン」をご覧ください。

ヒント フラグメント読み込みを使ってスプラッシュ スクリーンの拡張ページを読み込む場合、Windows のスプラッシュ スクリーンが閉じてから、拡張されたスプラッシュ スクリーン ページが表示されるまでの間に、ちらつきが生じることがあります。このちらつきが生じるのは、**activated** イベント ハンドラーの実行が終了する前に、フラグメント読み込みがスプラッシュ スクリーンの拡張ページの読み込みを非同期的に開始するためです。スプラッシュ スクリーンのサンプルで示されている設計パターンを使うと、この不快なちらつきは生じません。スプラッシュ スクリーンの拡張ページを、フラグメントとして読み込む代わりに、単純にアプリの UI 上に描画します。追加の読み込みタスクが完了すると、スプラッシュ スクリーンの拡張ページの表示をやめてアプリのランディング ページを表示できるようになります。また、スプラッシュ スクリーンの拡張ページをフラグメントとして読み込む場合は、アクティブ化の保留を取得して **activated** イベントに非同期的に応答することで、ちらつきを防ぐこともできます。

activatedOperation.getDeferral メソッドを呼び出して、activated イベントの保留を取得します。

- スプラッシュ スクリーンやスプラッシュ スクリーンの拡張ページを広告の表示に使わないでください。
  スプラッシュ スクリーンの目的は、アプリの読み込み中に、指定したアプリが適切に開始されていることをユーザーに知らせることです。関係ない要素をスプラッシュ スクリーンに含めると、アプリをひとめで識別することが難しくなり、適切なアプリが起動されていないのではないかとユーザーを不安にさせます。
- 複数の異なるスプラッシュ スクリーン画像を表示するメカニズムとして、スプラッシュ スクリーンの拡張ページを使わないでください。
  スプラッシュ スクリーンやスプラッシュ スクリーンの拡張ページの目的は、ユーザーにスムーズで洗練された読み込みエクスペリエンスを提供することです。スプラッシュ スクリーンの拡張ページを使って複数の異なるスプラッシュ スクリーン画像を表示することは、この目的から離れた行為であり、ユーザーに不快な印象を与えたり、混乱を招く可能性があります。スプラッシュ スクリーンの拡張ページは、他のタスクが完了するまで、現在の読み込みエクスペリエンスを続けるためだけに使ってください。
- スプラッシュ スクリーンやスプラッシュ スクリーンの拡張ページをバージョン情報の表示に使わないでください。
  スプラッシュ スクリーンを、バージョン情報やその他のアプリ メタデータの表示に使わないでください。これらの情報は、アプリの Windows ストアの説明や、アプリ自体の中に表示します。

ユーザー エクスペリエンス

- アプリを明確に特徴付ける画像を使います。アプリを明確に特徴付ける画像と配色を使って、ユーザーが正しいアプリを起動したことを確信できるようにします。独特な画面にすると、ブランドを印象付ける効果もあります。
- ビジュアル効果を高めるため、スプラッシュ スクリーンの画像には透過的な PNG を使います。透過的な PNG を使うと、スプラッシュ スクリーンの画像全体に選択した背景色が適用されます。透過的でないと、画像の背景色が異なる場合に、スプラッシュ スクリーンがちぐはぐで魅力のないものになる可能性があります。
- Windows の場合は、3 つの各倍率に合わせたサイズで、スプラッシュ画面の画像のバージョンを用意します。 すべてのアプリのスプラッシュ画面の画像は、デバイ

スが等倍表示の場合、620 x 300 ピクセルにする必要があります。また、1.4 倍と 1.8 倍の追加のスプラッシュ画面画像も含めることをお勧めします。3 つの倍率の各画像を用意すると、異なるデバイスでも、きれいで統一感のある起動エクスペリエンスを提供できます。スプラッシュ画面を設計するときは、画面より小さく、中央に表示されることに注意してください。Windows Phone ストア アプリのスプラッシュ画面のように画面全体に表示されるわけではありません。。

各倍率で必要なスプラッシュ スクリーン画像のサイズを判断するには、次の表をご覧ください。

スケール	画像サイズ (ピクセル)
等倍	620 x 300
1.4 倍	868 x 420
1.8 倍	1116 x 540

- Windows Phone ストア アプリの場合は、最低でも 2.4x のアセットを用意します。可能であれば、すべて用意します。画像ファイル アセット自体は、背景が透明である必要があります。アプリ マニフェストで、SplashScreen@Image プロパティの値を "Assets\<assetname>.png" に設定し、VisualElements@BackgroundColor に値を設定します。

  各倍率で必要なスプラッシュ スクリーン画像のサイズを判断するには、次の表をご覧ください。

スケール	画像サイズ (ピクセル)
等倍	400 x 800
1.4 倍	672 x 1120
1.8 倍	1152 x1920

- システムによってスプラッシュ スクリーン画像に割り当てられた領域を使う画像を選びます。スプラッシュ スクリーン画像を選ぶときには、各倍率で割り当てられた領域を活用するようにします。各倍率でのスプラッシュ スクリーン画像のサイズを判断するには、倍率と画像 サイズの表をご覧ください。

  こうすることで、画像の品質を確保し、高品質なスプラッシュ スクリーンを作成できます。
- スプラッシュ スクリーンが消えた後、システムとイベントに関連する UI を表示します。スプラッシュ スクリーンの **dismissed** イベントをリッスンすることで、シ

ステムやイベントに関連する UI を安全に表示するタイミングを識別できます。そうしないと、スプラッシュ スクリーンが消える前に、関連付けられた UI (検索ウィンドウ、メッセージ ダイアログ、Web 認証ブローカーなど) が表示されてしまう場合があります。その場合、期待とは異なるビジュアル効果が発生してしまいます。

- スプラッシュ スクリーンが消えた後、導入アニメーションを開始します。多くのアプリは、アプリのランディング ページが読み込まれるたびに、コンテンツの導入アニメーションを表示しようとします。スプラッシュ スクリーンの **dismissed** イベントをリッスンして、アニメーションを開始するタイミングを識別できます。

スプラッシュ スクリーンの拡張ページ

- スプラッシュ スクリーンの拡張ページの外観は、Windows で表示されるスプラッシュ スクリーンに似せるようにしてください。スプラッシュ スクリーンの拡張ページでは、Windows のスプラッシュ スクリーンと同じ背景色と画像を使う必要があります。一貫性のある画像と背景色を使うと、Windows のスプラッシュ スクリーンからアプリのスプラッシュ スクリーンの拡張ページへの遷移が洗練され、ユーザーに不快感を与えずにすみます。
- スプラッシュ スクリーンの拡張ページの画像を、Windows がスプラッシュ スクリーン画像を表示するときの座標に配置します。

  SplashScreen クラスを使ってスプラッシュ スクリーンの拡張ページの画像を配置する方法については、「スプラッシュ スクリーンを拡張する方法」をご覧ください。
- スプラッシュ スクリーンの拡張ページの画像の位置を調整して、スナップや回転などのサイズ変更イベントに応答します。スプラッシュ スクリーンの拡張ページは、**onresize** イベントをリッスンして、アプリがスナップされたりデバイスが回転したりした場合にスプラッシュ スクリーンの画像の座標に合わせて調整する必要があります。こうすることで、ユーザーが画面上でどのようなデバイスの操作やアプリのレイアウト変更をしていても、アプリの読み込みエクスペリエンスをスムーズで洗練されたものにすることができます。
- スプラッシュ スクリーンの拡張ページを数秒以上表示する場合は、アプリがまだ読み込みを行っていることがユーザーにわかるように、進行状況リングを追加しま

す。進行状況不定リング コントロールを使うと、アプリがクラッシュしたのではなく、間もなく準備ができることをユーザーに知らせることができます。アプリがユーザーのために何をしているかを簡潔に説明する 1 行のテキストを、進行状況リングと一緒に表示することを検討してください。たとえば、スプラッシュ スクリーンの拡張ページには、進行状況リングと、読み込み中であることを示すメッセージを追加できます。

アプリの応答性が高いと感じさせ、ユーザーに情報を提供し続けることは、ユーザーの読み込みエクスペリエンスに対する評価を高めるための優れた方法です。進行状況不定リングとテキストを追加する方法については、「クイック スタート: プログレス コントロールの追加」をご覧ください。

その他の使い方のガイダンス

すべての Windows ストア アプリにはスプラッシュ スクリーンが必要です。スプラッシュ スクリーンは、スプラッシュ スクリーン画像と背景色で構成されています。 その両方をカスタマイズできます。

Windows では、ユーザーがアプリを起動すると、即座にこのスプラッシュ スクリーンが表示されます。これによって、アプリ リソースの初期化中であることがユーザーに示されます。 スプラッシュ スクリーンは、アプリが操作できる状態になるとすぐに、Windows によって閉じられます。

スプラッシュ スクリーンが適切にデザインされていると、アプリがより魅力的なものになります。Windows ストアでは、以下のような装飾の少ないシンプルなスプラッシュ スクリーンを使っています。

以下のスプラッシュ スクリーンは、緑の背景色と透過的な PNG を組み合わせて作られています。

SplashScreen クラスを使って、スプラッシュ スクリーンを拡張し、導入アニメーションをトリガーすることで、アプリの起動エクスペリエンスをカスタマイズできます。

トラブルシューティング

JavaScript: スプラッシュ スクリーン拡張ページへの遷移中のちらつきの回避

「スプラッシュ スクリーンを拡張する方法」には、ちらつきが生じるのを回避するためのアドバイスを記載しています。スプラッシュ スクリーンの拡張ページへの切り替え中にちらつきが生じる場合は、<img> タグに onload="" を追加して、 のようにします。こうすることで、スプラッシュ スクリーンの拡張ページへの切り替え前に画像がレンダリングされるまでシステムを待機させて、ちらつきを防ぐことができます。

C#: スプラッシュ スクリーンの拡張ページへの遷移中のちらつきの回避

「スプラッシュ スクリーンを拡張する方法」の手順に従った場合、スプラッシュ スクリーンの拡張ページへの切り替え中にちらつきが生じることがあります。このちらつきは、ページ コンテンツのレンダリングが終わる前に (Window.Current.Activate を呼び出して) 現在のウィンドウをアクティブ化する場合に生じます。現在のウィンドウをアクティブ化する前にスプラッシュ スクリーンの拡張ページの画像が読み取られていることを確認することで、ちらつきが生じる可能性を減らすことができます。さらにちらつきを防ぐには、現在のウィンドウをアクティブ化する前に、タイマーを使って、少しの間 (たとえば 50 ミリ秒) アプリを待機させる必要があります。残念ながら、ちらつきを防ぐ確実な方法は存在しません。XAML は非同期的にコンテンツをレンダリングするため、レンダリングが完了するときを確実に予測する方法が存在しないからです。

「スプラッシュ スクリーンを拡張する方法」の手順に従った結果、スプラッシュ スクリーンの拡張ページへの切り替え中にちらつきが生じる場合は、現在のウィンドウをアクティブ化する前にアプリを少し待機させ、スプラッシュ スクリーンの拡張ページの画像の読み取りが終わるようにするために、次の手順を実行します。

1. ExtendedSplash.xaml で、スプラッシュ スクリーンの拡張ページの画像のマークアップを更新して、スプラッシュ スクリーンの拡張ページの画像が読み取られたときにユーザーに通知します。

```
<Image x:Name="extendedSplashImage" Source="Assets/SplashScreen.png"
ImageOpened="extendedSplashImage_ImageOpened"/>
```

画像が読み取られた後、**ImageOpened** イベントが発生します。次の例に示すように、**ImageOpened** 属性を追加し、イベント ハンドラー名 (extendedSplashImage_ImageOpened) を指定して、**ImageOpened** イベントを登録する必要があります。

2. ExtendedSplash.xaml では、スプラッシュ スクリーンの拡張ページの画像が読み取られた後で、タイマーに基づいて現在のウィンドウをアクティブ化するコードを ExtendedSplash クラスに追加します。

```
        private DispatcherTimer showWindowTimer;
        private void OnShowWindowTimer(object sender, object e)
        {
            showWindowTimer.Stop();

            // Activate/show the window, now that the splash image has rendered
            Window.Current.Activate();
        }

        private void extendedSplashImage_ImageOpened(object sender,
    RoutedEventArgs e)
        {
            // ImageOpened means the file has been read, but the image hasn't
    been painted yet.
            // Start a short timer to give the image a chance to render, before
    showing the window
            // and starting the animation.
            showWindowTimer = new DispatcherTimer();
            showWindowTimer.Interval = TimeSpan.FromMilliseconds(50);
            showWindowTimer.Tick += OnShowWindowTimer;
            showWindowTimer.Start();
    }
```

この例は、**ImageOpened** イベントに応答する方法と、タイマーを使って、現在のウィンドウをアクティブ化する前にアプリを少しの間待機させる方法を示しています。

3. **OnLaunched** メソッドを次のように変更します。

```
        protected override void OnLaunched(LaunchActivatedEventArgs args)
        {
            if (args.PreviousExecutionState != ApplicationExecutionState.Running)
            {
                bool loadState = (args.PreviousExecutionState ==
    ApplicationExecutionState.Terminated);
                ExtendedSplash extendedSplash = new ExtendedSplash(args.SplashScreen,
    loadState);
                Window.Current.Content = extendedSplash;
```

```
            }

            // ExtendedSplash will activate the window when its initial content has been
        painted.
}
```

この例では、現在のウィンドウに対する **Activate** の呼び出しを削除しました。代わりにアクティブ化は、**ImageOpened** イベントが発生し、スプラッシュ スクリーンの拡張ページの画像が読み取られたことが示されてから、ExtendedSplash オブジェクトによって実行されます。

4. こうしたコードで実際にちらつきを回避できることを確認するために、できるだけ多くの種類のデバイスや状況下でコードをテストしてください。

レイアウトとスケーリングのガイドライン

このセクションでは、アプリの各ページでのアプリ要素のレイアウトと、さまざまなサイズのデバイスやサイズ変更されたウィンドウでユーザーがアプリを操作する際の、アプリのスケーリングに関するガイドラインを示します。

このセクションの内容

トピック	説明
複数のウィンドウ	ユニバーサル Windows プラットフォーム (UWP) アプリで複数のウィンドウをサポートする場合は、次の推奨事項に従ってください。
プロジェクション マネージャー	プロジェクション マネージャーを使うと、アプリの個別のウィンドウを別の画面にプロジェクションできます。

複数のウィンドウのガイドライン

複数のウィンドウのサポートによって、ユーザーはアプリの異なる部分を同時に操作できます。ユーザーは、複数のウィンドウを使って、コンテンツを比較できるほか、コンテンツの複数の特定部分を同時に表示できます。ユニバーサル Windows プラットフォーム (UWP) アプリで複数のウィンドウをサポートする場合は、次の推奨事項に従ってください。

重要な API

CreateNewView メソッド

ApplicationView クラス

ApplicationViewSwitcher クラス

説明

複数のウィンドウをサポートするアプリでは、各ウィンドウが別個のアプリのように動作します。チャームは個別のウィンドウとやり取りします。ユーザーがスタート画面でアプリのタイルをクリックすると、最後に使ったアプリのウィンドウが表示されます。ユーザーはウィンドウごとにサイズを変更したり、ウィンドウを個別に非表示にしたりできるほか、最近使ったアプリの一覧に個々のウィンドウを表示することもできます。

複数ウィンドウの設計

複数のウィンドウをサポートすることがアプリにとって適切な場合、各ウィンドウに表示するコンテンツを決定する必要があります。たとえば、1 個のメイン ウィンドウと、限定された固有の機能セットを持つセカンダリ ウィンドウを作ることも、元のアプリ ウィンドウのコピーとして新しい個別のウィンドウを設計することもできます。ユーザーがアプリを切り替えたときに表示される、セカンダリ ウィンドウのタイトルを指定することもできます。

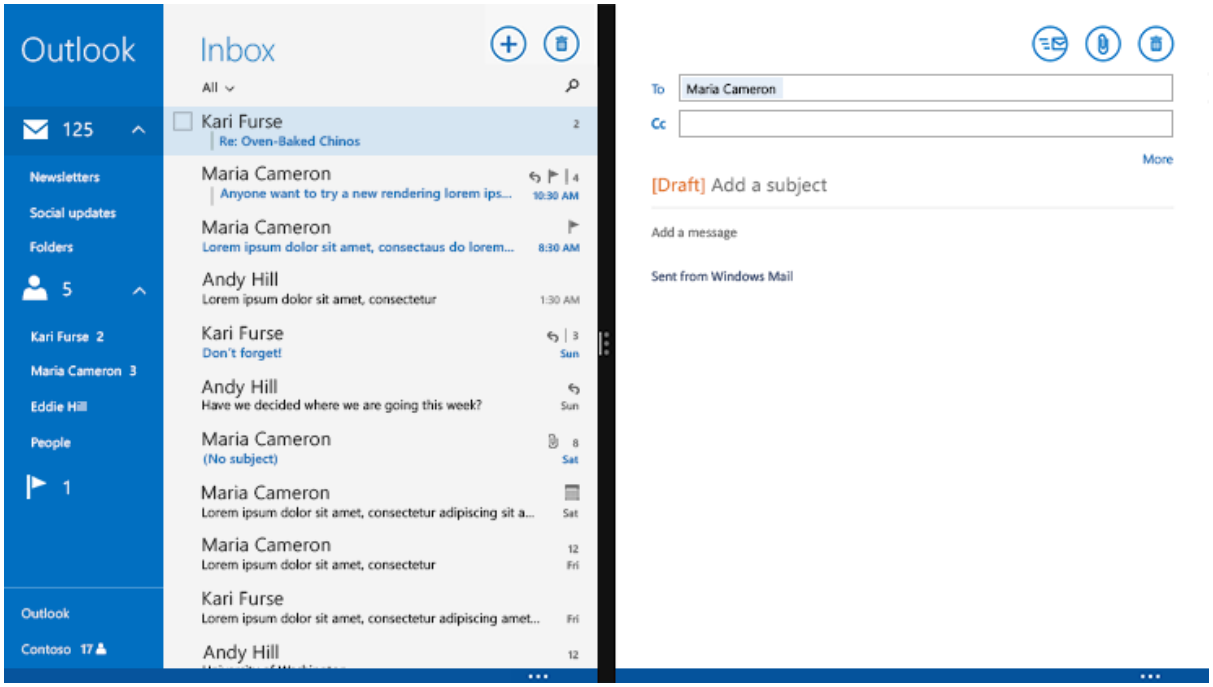
また、(元のアプリ ウィンドウを基準にして) 新しいウィンドウを開く画面上の位置も指定します。新しいウィンドウは、次のいずれかの位置に配置できます。

- 元のウィンドウの隣で、画面の領域を共有する。
- メイン ウィンドウの位置に表示する。
- 画面にまったく表示されない。

セカンダリ ウィンドウが最初に表示された後は、ウィンドウの配置とサイズをユーザーが制御できます。

例

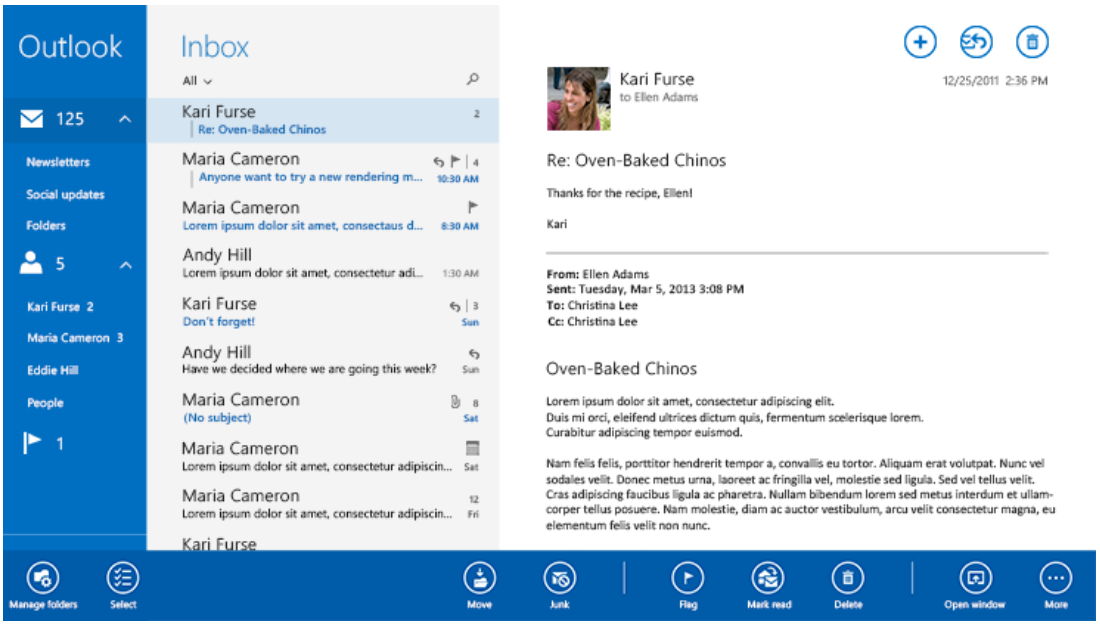
複数のウィンドウを使うアプリの例としてメール アプリがあります。ユーザーは、メイン アプリ ウィンドウでメッセージを表示したり、新しいウィンドウを開いたりすることができます。このことは、たとえば、新しいメッセージを作成しながら、同時にメイン ウィンドウを使って他のメッセージを検索する場合に便利です。



メール アプリで 2 つのウィンドウが開いている場合、最近使ったアプリの一覧が次のように表示されます。

推奨と非推奨

- セカンダリ ウィンドウからメイン ウィンドウに戻る方法をユーザーに提供します。
- ユーザーが新しいウィンドウを開くための明確な操作を提供します。たとえば、新しいウィンドウを開くボタンをアプリ バーに追加します。メール アプリの一番下のアプリ バーに **[開く]** ボタンが表示されます。



- 新しいウィンドウには、必ずそのウィンドウのコンテンツを反映したタイトルを付けます。ユーザーがタイトルに基づいてアプリの各ウィンドウを区別できるようにします。
- **consolidated event** に受信登録し、イベントが発生したときにはウィンドウの内容を閉じます。統合イベントは、最近使ったアプリの一覧からウィンドウが削除されたときや、ユーザーがウィンドウを閉じるジェスチャを実行した場合に発生します。
- 元のアプリ ウィンドウを新しいウィンドウで置き換える場合は、ウィンドウの切り替え時にカスタム アニメーションを提供します。
- 生産性を向上させ、マルチタスクを可能にするシナリオでは、アプリ内で新しいウィンドウを有効にします。
- 新しいウィンドウは、ユーザーがウィンドウ内でタスク全体を完了できるように設計します。
- ユーザーがアプリの別の部分に移動したときに新しいウィンドウを自動的に開かないでください。新しいウィンドウは、常にユーザーの操作によって開くようにします。
- アプリの主要な目的を完了するために、新しいウィンドウを開くことをユーザーに要求しないでください。

プロジェクション マネージャーのガイドライン

プロジェクション マネージャーを使うと、アプリの個別のウィンドウを別の画面にプロジェクションできます。 たとえば、ゲーム アプリでは、大型のモニターにメインのゲーム プレイ画面を表示し、ローカルの画面にゲームのコントロールを表示できます。また、Scrabble などのマルチプレイヤーのワード ゲームでは、プロジェクション画面に共有ゲーム ボードを表示し、ローカルの画面にユーザーのゲーム ピースを表示できます。プレゼンテーション アプリの場合、プロジェクション画面のウィンドウにプレゼンテーションを表示し、ローカルの画面に発表者のノートを表示できます。

重要な API

ProjectionManager クラス　　ApplicationView クラス

既定では、プロジェクション マネージャーを使わない状態で、ユーザーが他の表示デバイスに接続している場合、アプリのウィンドウがサブ画面に複製されます。複製モードのとき、両方の画面に適した解像度が Windows によって自動的に選ばれます。外部画面の画面情報は使われません。しかし、この解像度が動画再生やゲームに最適でないことがあります。プロジェクション マネージャーを使うと、Windows によってプロジェクション画面の解像度と画面の縦横比が取得され、ウィンドウの表示が最適化されます。

プロジェクション マネージャーは、アプリ用に複数のウィンドウを使うことに似ています。この表では、複数のウィンドウとプロジェクション マネージャーの使い分け方について説明します。

シナリオ	複数のウィンドウを使う	プロジェクション マネージャーを使う
ユーザーが両方のウィンドウを操作	推奨	非推奨 (ただし外部ディスプレイが Perceptive Pixel (PPI) by Microsoft のようなタッチ デバイスである場合は除く)
サブ ウィンドウが操作用ではなく表示専用	非推奨	推奨

ユーザーが縦横比や解像度が大きく異なる画面にサブ ウィンドウを表示	非推奨	推奨

推奨と非推奨

- ユーザーがローカルのアプリ ウィンドウからプロジェクションを制御する。
  ユーザーが次の操作を実行できる必要があります。
  ◦ 新しいプロジェクション ウィンドウを起動する。
  ◦ ウィンドウの中断したプロジェクションを再開する。
  ◦ 開始したプロジェクションを停止する。
  ◦ プロジェクション ウィンドウ上のコントロールを使ってローカルとプロジェクションのウィンドウを入れ替える。ウィンドウの自動配置が適切でなく、プロジェクション ウィンドウがローカルの画面に表示される場合、ユーザーはウィンドウを入れ替えることができる必要があります。
- 次のアイコンを使って、プロジェクションの開始、停止、ウィンドウの入れ替えを行う。

コード	アイコン	説明
U+E2B4		プロジェクションの開始または再開
U+E2B3		プロジェクションの停止
U+E13C		プロジェクション ビューの入れ替え

- プロジェクションを自動的に開始または停止しない。プロジェクションはユーザー入力によってのみ開始または停止する必要があります。
  注 "再開" 機能を実装することで、一時停止後や他のアプリへの切り替え後にユーザーが簡単にプロジェクションを再開できるようにすることが可能です。プロジェクション ウィンドウが表示中の画面から切り替わった場合 (通常、別のアプリのプロジェクションのため)、**StartProjectingAsync** を使ってプロジェクション ウィンドウの表示を再開します。**VisibilityChanged** イベントにサブスクライブして、いつプロジェクション ウィンドウが表示中の画面から切り替わったかを調べることができます。**consolidated** イベントにサブスクライブして、いつプロジェ

クション ウィンドウが最近使ったアプリの一覧から削除されたか、いつプロジェクション ウィンドウが閉じられたかを調べることができます。

その他の使い方のガイダンス

スタイルとレイアウト

プロジェクションの開始、停止、入れ替えに使用するアイコンの色やラベル テキストは選ぶことができます。アイコンの配置場所を選ぶことができますが、アイコンは下部のアプリ バーに配置し、アプリ バー ボタンのガイダンスに従うことをお勧めします。

プロジェクション ウィンドウの配置を変更することはできません。自動的に決定されるためです。マウス、キーボード、またはタッチパッドを操作するユーザーはプロジェクション画面を配置後に移動できます。

マップと位置情報のガイドライン

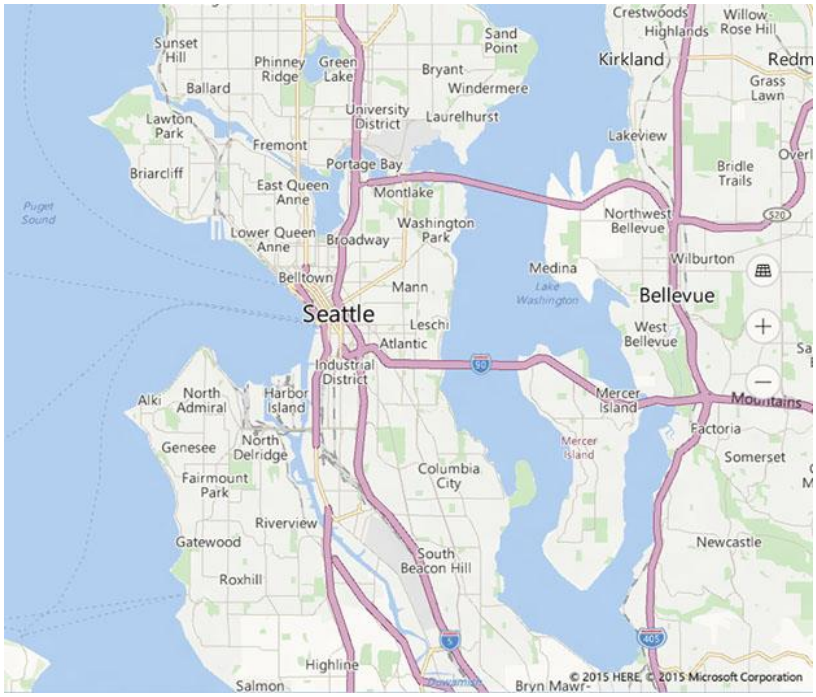
このセクションでは、ジオフェンスと位置情報認識アプリを設計するガイドラインを説明します。

このセクションの内容

トピック	説明
マップ	マップ コントロールでは、地図および上空からの写真、方向、検索結果、トラフィックを表示できます。
ジオフェンス	アプリのジオフェンスについては、次のベスト プラクティスに従ってください。
位置認識アプリ	このトピックでは、ユーザーの位置にアクセスする必要があるアプリを構築する際のパフォーマンス ガイドラインを説明します。

マップのガイドライン

マップ コントロールでは、ビュー、ルート案内、および検索結果をロードマップ、航空写真、および 3D で表示できます。マップ上には、現在地、ルート、関心のあるポイントを表示できます。また、3D の航空写真、Streetside ビュー、交通情報、乗り換え情報、周辺情報を表示することもできます。



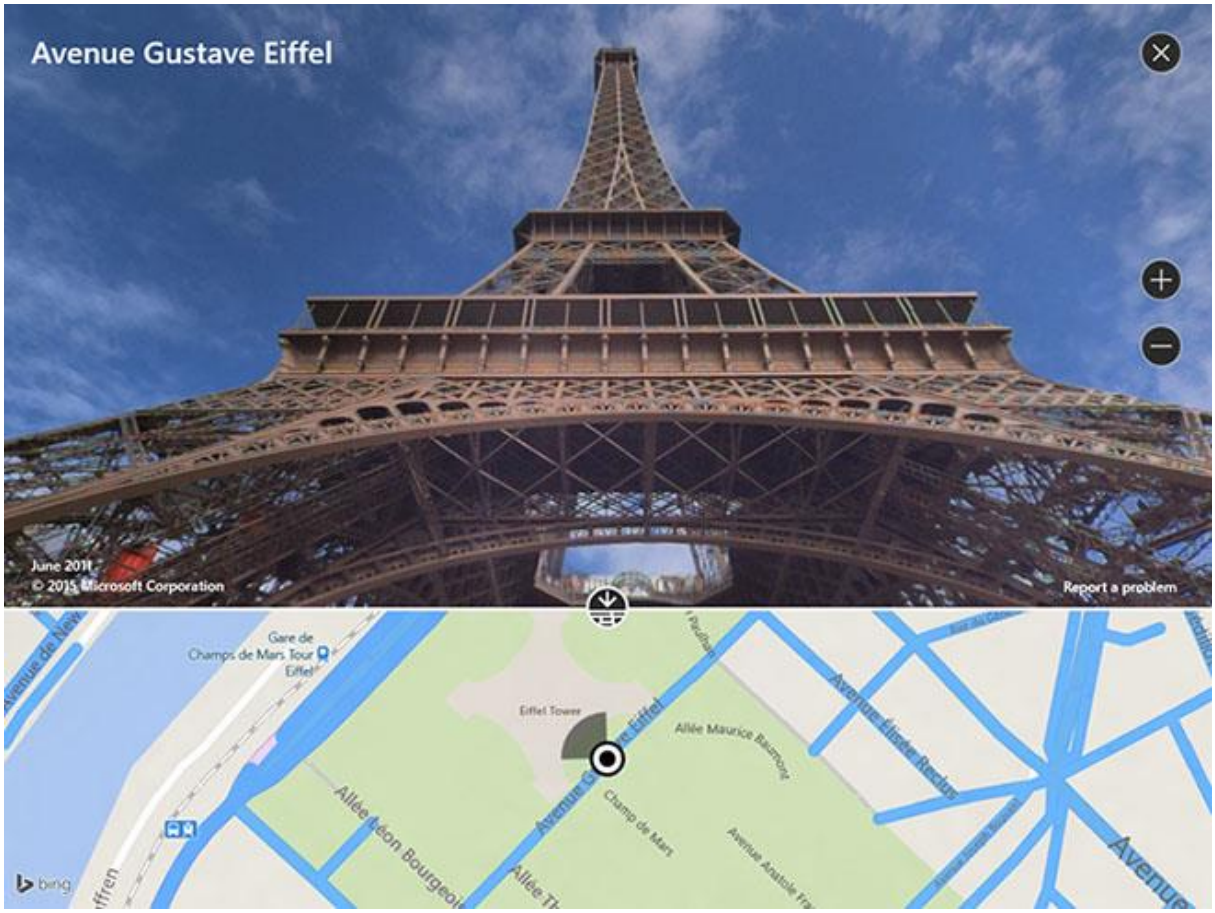
適切なコントロールの使用

アプリ固有の地理情報または一般的な地理情報を表示できるアプリ内でマップを使う場合、アプリにマップ コントロールを含めておくことで、ユーザーはアプリの外部に移動することなく情報を得ることができます。

注　その情報を得るためにユーザーがアプリの外部に移動してもかまわない場合は、Windows マップ アプリを利用することも検討してください。アプリから Windows マップ アプリを起動し、特定の地図、ルート案内、検索結果を表示することができます。詳しくは、「Windows Map アプリの起動」をご覧ください。

例

次の例では、Streetside ビューを使用したマップを示しています。



次の例では、3D 航空写真を使用したマップを示しています。

次の例では、3D 航空写真ビューと Streetside ビューの両方が含まれるアプリを示してます。

推奨と非推奨

- ユーザーが地理情報を表示するためにパンとズームを過度に使用しなくて済むように、十分な画面領域 (または画面全体) を使用してマップを表示します。
- 静的な情報ビューの提示をするためにのみマップを使う場合、小さなマップを使う方が適している場合があります。小さく静的なマップを使う場合は、使いやすさを考えてサイズを決めます。画面上の領域を十分節約できる程度に小さく、判読しにくくならない程度に大きくします。
    - マップ シーンに関心のあるポイントを埋め込むには、MapElements を使います。その他の情報も、マップ シーンのオーバーレイとして表示される一時的な UI に表示できます。

ジオフェンスのガイドライン

アプリの **geofencing** については、次のベスト プラクティスに従ってください。

重要な API

Geofence クラス	Geolocator クラス

推奨事項

- **Geofence** イベントが発生したときにインターネット アクセスが必要な場合は、ジオフェンスを作成する前にインターネット アクセスを確認します。
    - アプリで現在インターネットにアクセスできない場合、ジオフェンスをセットアップする前にユーザーに対してインターネットに接続するようメッセージを表示することができます。
    - インターネット アクセスが不可能である場合は、ジオフェンスの位置確認に必要な電力を消費しないようにしてください。
- ジオフェンス イベントが **Entered** 状態または **Exited** 状態に対する変更を示す場合、タイム スタンプと現在の位置をチェックしてジオフェンス通知の関連性を確

認します。詳しくは、次の「タイム スタンプと現在位置の確認」をご覧ください。

- デバイスで位置情報にアクセスできない場合は、ケースを管理する例外を作成し、必要に応じてユーザーに通知します。アクセス許可がオフになっている、デバイスに GPS 機能が付いていない、GPS 信号がブロックされている、Wi-Fi 信号が弱いなどの理由で、位置情報が利用できない場合があります。
- 一般に、フォアグラウンドとバックグラウンドの両方で同時にジオフェンス イベントをリッスンする必要はありません。ただし、アプリがフォアグラウンドとバックグラウンドの両方で同時にジオフェンス イベントをリッスンする必要がある場合は、次の手順を行います。
  - **ReadReports** メソッドを呼び出して、イベントが発生したかどうかを確認します。
  - ユーザーからアプリが見えなくなったときはフォアグラウンド イベント リスナーの登録を解除し、再び見えるようになったときにもう一度登録します。

  コード例と詳しい情報については、「バックグラウンドとフォアグラウンドのリスナー」をご覧ください。

- 1 つのアプリに 1000 以上のジオフェンスを使わないでください。システムは実際にはアプリごとに数千のジオフェンスをサポートしますが、1000 以下のジオフェンスを使用することによってアプリのメモリ使用量を減らしてアプリの高パフォーマンスを維持できます。
- 半径が 50 m 未満のジオフェンスを作成しないでください。 アプリで小さなジオフェンスを使う必要がある場合は、最高のパフォーマンスを実現するために GPS 機能付きのデバイスでアプリを使うようユーザーに勧めてください。

その他の使い方のガイダンス

タイム スタンプと現在位置の確認

イベントにより **Entered** または **Exited** 状態に変化したときは、イベントのタイム スタンプと現在位置の両方を確認します。 イベントが実際にユーザーによって処理される時期は、システムでバックグラウンド タスクを起動するリソースが不足していたり、ユーザーが通

知に気付かなかったり、デバイスがスタンバイ中であったり (Windows の場合) など、さまざまな要因によって影響を受けます。たとえば、次のような順序で事態が進む可能性があります。

- アプリがジオフェンスを作成して、ジオフェンスの進入イベントと退出イベントを監視します。
- ユーザーがデバイスをジオフェンスの内側に移動して、進入イベントをトリガーします。
- ジオフェンスの内側に入ったという通知をアプリがユーザーに送信します。
- ユーザーが忙しく、通知に気付いたのは 10 分後でした。
- その 10 分間の間に、ユーザーはジオフェンスの外側に移動していました。

タイム スタンプをみれば、アクションが過去に起こったことを判断できます。 現在位置をみれば、ユーザーがジオフェンスの外側に戻ったことが確認できます。アプリの機能によっては、このイベントを無視することもできます。

バックグラウンドとフォアグラウンドのリスナー

一般に、アプリは、フォアグラウンド タスクとバックグラウンド タスクの両方で同時に **Geofence** イベントをリッスンする必要はありません。 両方が必要になる場合に最も明確な処理方法は、バックグラウンド タスクに通知処理を任せることです。 実際にフォアグラウンドとバックグラウンドの両方でジオフェンス リスナーをセットアップした場合、どちらが最初にトリガーされるか不明であるため、常に **ReadReports** メソッドを呼び出してイベントが発生したか確認する必要があります。

また、フォアグラウンドとバックグラウンドの両方でジオフェンス リスナーをセットアップした場合、ユーザーからアプリが見えなくなるたびにフォアグラウンド イベント リスナーの登録を解除し、再び見えるようになったときにアプリを再登録する必要があります。表示イベントに登録するコード例を次に示します。

```
document.addEventListener("visibilitychange", onVisibilityChanged, false);
```

```
    Windows.UI.Core.CoreWindow coreWindow;

    coreWindow = CoreWindow.GetForCurrentThread(); // This needs to be set before
InitializeComponent sets up event registration for app visibility
    coreWindow.VisibilityChanged += OnVisibilityChanged;
```


表示が変わると、ここで示したようにフォアグラウンド イベント ハンドラーの有効または無効を切り替えることができます。

```
function onVisibilityChanged() {
    // NOTE: After the app is no longer visible on the screen and before the app is
suspended
    // you might want your app to use toast notification for any geofence activity.
    // By registering for VisibiltyChanged the app is notified when the app is no longer
visible in the foreground.

    if (document.msVisibilityState === "visible") {
        // register for foreground events

Windows.Devices.Geolocation.Geofencing.GeofenceMonitor.current.addEventListener("geo
fencestatechanged", onGeofenceStateChanged);

Windows.Devices.Geolocation.Geofencing.GeofenceMonitor.current.addEventListener("stat
uschanged", onGeofenceStatusChanged);
    } else {
        // unregister foreground events (let background capture events)

Windows.Devices.Geolocation.Geofencing.GeofenceMonitor.current.removeEventListener("
geofencestatechanged", onGeofenceStateChanged);

Windows.Devices.Geolocation.Geofencing.GeofenceMonitor.current.removeEventListener("
statuschanged", onGeofenceStatusChanged);
    }
}
```

```
private void OnVisibilityChanged(CoreWindow sender, VisibilityChangedEventArgs args)
{
    // NOTE: After the app is no longer visible on the screen and before the app is
suspended
    // you might want your app to use toast notification for any geofence activity.
    // By registering for VisibiltyChanged the app is notified when the app is no longer
visible in the foreground.

    if (args.Visible)
    {
        // register for foreground events
        GeofenceMonitor.Current.GeofenceStateChanged += OnGeofenceStateChanged;
        GeofenceMonitor.Current.StatusChanged += OnGeofenceStatusChanged;
    }
    else
```

```
    {
        // unregister foreground events (let background capture events)
        GeofenceMonitor.Current.GeofenceStateChanged -= OnGeofenceStateChanged;
        GeofenceMonitor.Current.StatusChanged -= OnGeofenceStatusChanged;
    }
}
```

ジオフェンスのサイズ変更

GPS を使うと最も正確な位置情報が得られますが、ジオフェンスでは Wi-Fi などの位置センサーを使ってユーザーの現在位置を判断することもできます。 しかし、GPS とは別のこういった方法を使うと、作成できるジオフェンスのサイズが影響を受けます。 精度が低い場合、小さなジオフェンスを作成しても役に立ちません。 通常、50 m より半径が小さいジオフェンスを作らないことをお勧めします。また、Windows ではジオフェンスのバックグラウンド タスクが周期的にしか実行されないため、小さなジオフェンスを使った場合、**Enter** イベントや **Exit** イベントをまったく認識できない可能性があります。

アプリで小さなジオフェンスを使う必要がある場合は、最高のパフォーマンスを実現するために GPS 機能付きのデバイスでアプリを使うようユーザーに勧めてください。

位置認識アプリのガイドライン

このトピックでは、位置情報認識アプリを構築する際のパフォーマンス ガイドラインを説明します。

重要な API

Geolocation 名前空間　　Geolocator クラス

推奨事項

- location オブジェクトは、アプリで位置データが必要になった場合にのみ使用を開始します。
  ユーザーの位置情報へアクセスする前に、**RequestAccessAsync** を呼び出します。
  RequestAccessAsync メソッドは、必ずアプリの UI スレッドで呼び出す必要があ

ります。位置情報に対するアクセス許可をユーザがアプリに与えるまで、アプリが位置情報データへアクセスできません。

- アプリで位置情報が必須でない場合は、位置情報を必要とするタスクをユーザーが完了することを試みるまではその情報にアクセスしないでください。たとえば、ソーシャル ネットワーキング アプリに、[位置情報を使ってチェックイン] というボタンがある場合、アプリは、ユーザーがそのボタンをクリックするまでは位置情報にアクセスしないようにします。アプリのメイン機能で位置情報が必要な場合は、すぐにアクセスしても問題ありません。
- **Geolocator** オブジェクトの初めての使用はメイン UI スレッドで行い、ユーザーから同意を得るためのプロンプトをトリガーする必要があります。**Geolocator** の初めての使用とは、**getGeopositionAsync** を初めて呼び出すとき、または **positionChanged** イベントのハンドラーの初めての登録にするときです。
- 位置データがどのように使われるかをユーザーに知らせてください。
- ユーザーが現在の位置を手動で更新できる UI を用意します。
- 位置データの取得中は、進行状況バーまたは進行状況リングを表示します。使えるプログレス コントロールとその使い方について詳しくは、「プログレス コントロールのガイドライン」をご覧ください。
- 位置情報サービスが無効または利用不可になっている場合は、適切なエラー メッセージまたはダイアログを表示します。

  ユーザーが設定を使って位置データへのアクセスを許可していない場合は、**設定**アプリ内にある**位置情報に関するプライバシー設定**への使いやすいリンクを示すことをお勧めします。たとえば、ハイパー リンク (Hyperlink) コントロールを使うか、ms-settings:privacy-location URI を使用して **LaunchUriAsyc** メソッドを呼び出すことでコードから**設定アプリ**を起動します。詳細については、「Windows 設定アプリを起動する」をご覧ください。
- 位置情報へのアクセスをユーザーが無効にした場合は、キャッシュされた位置データをクリアし、**Geolocator** オブジェクトを解放します。

  ユーザーが設定を使って位置情報へのアクセスをオフにした場合に、**Geolocator** オブジェクトを解放します。すると、アプリは、あらゆる位置情報 API 呼び出しの結果として **ACCESS_DENIED** を受け取ります。アプリで位置データを保存またはキャッシュしている場合は、ユーザーが位置情報へのアクセスを無効にするときにすべてのキャッシュ データをクリアします。位置情報サービス経由で位置データ

を利用できないときに位置情報を手動で入力するための代替手段を用意してください。

- 位置情報サービスを再び有効にするための UI を用意します。たとえば、**Geolocator** オブジェクトを再インスタンス化して位置情報を取得し直す更新ボタンを提供します。

  位置情報サービスを再び有効にするための UI を提供するー

  - ユーザーが位置情報を無効にした後に再び有効にした場合、アプリには通知されません。**status** プロパティは変更されず、**statusChanged** イベントも発生しません。アプリで、新しい **Geolocator** オブジェクトを作成し、**getGeopositionAsync** を呼び出して更新された位置情報データを取得するか、**positionChanged** イベントの受信登録をもう一度行います。位置情報が再び有効になったことを確認できたら、位置情報サービスが無効であることをユーザーに通知するために表示していた UI をクリアし、新しい状態に対して適切に対応します。
  - アプリをアクティブ化するとき、位置情報が必要な機能をユーザーが明示的に使おうとしたときなど、状況に応じて必要と思われる任意の時点で、位置情報データを取得し直すことをお勧めします。

パフォーマンス

- アプリで位置情報の更新を受け取る必要がない場合は、位置情報の要求を 1 回だけ使います。たとえば、写真に位置情報タグを追加するアプリでは、位置情報更新イベントを受け取る必要はありません。このようなアプリでは、**getGeopositionAsync** を使って現在の位置情報を要求します。詳しくは、「現在位置の検出」をご覧ください。

  1 回限りの位置情報の要求を行う場合は、次の値を設定する必要があります。

  - **DesiredAccuracy** または **DesiredAccuracyInMeters** を設定して、アプリから要求される精度を指定します。これらのパラメーターを使用する場合の推奨事項については、以下をご覧ください
  - **getGeopositionAsync** の最大保存期間のパラメーターを設定して、アプリで有用な位置情報を取得できる期間を指定します。アプリで数秒または数分前の

位置を使用できる場合は、ほとんどすぐに位置を受け取って、デバイスの電力を節約することができます。

- **getGeopositionAsync** のタイムアウト パラメーターを設定します。これが、アプリが返される位置またはエラーを待機することができる長さです。ユーザーへの応答性とアプリが必要とする精度のバランスを理解する必要があります。

- 頻繁に位置を更新する必要がある場合は、連続的な位置情報のセッションを使います。特定のしきい値を超えた移動を検出する場合、または発生時に絶えず位置情報の更新を取得する場合は、**positionChanged** イベントと **statusChanged** イベントを使います。

  位置情報の更新を要求すると、**DesiredAccuracy** または **DesiredAccuracyInMeters** を設定して、アプリから要求される精度を指定する必要があります。また、**MovementThreshold** または **ReportInterval** を使って、位置情報の更新が必要な頻度を設定する必要があります。

  - 移動しきい値を指定します。アプリによっては、ユーザーの移動距離が大きいときにだけ位置情報を更新すれば済むものがあります。たとえば、地域のニュースや天気予報の更新情報を提供するアプリでは、ユーザーの位置が別の都市に変わらない限り位置情報を更新する必要はありません。このような場合は、**MovementThreshold** プロパティを設定して、位置情報更新イベントの発生条件となる最小の移動距離を調整します。このプロパティには **PositionChanged** イベントをフィルター処理する効果があります。
  - アプリのエクスペリエンスと整合し、システム リソースの使用が最小限に抑えられる **ReportInterval** を使います。 たとえば、天気予報アプリでは、15 分ごとにデータを更新するだけでよいと思われます。リアルタイムのナビゲーション アプリを除くほとんどのアプリでは、位置情報の更新について、高い精度のストリームを常に必要とするわけではありません。最大限の精度のデータストリームを必要としない場合や、頻繁に更新する必要がない場合は、**ReportInterval** プロパティを設定して、アプリで位置情報を更新する必要がある最小の頻度を指定します。これにより、必要なときにだけ位置情報を計算することで、位置情報の提供元の電力を節約できます。

リアルタイムのデータを必要とするアプリでは、最短の間隔を指定せずに、**ReportInterval** を 0 に設定する必要があります。既定のレポート間隔は、1 秒またはハードウェアでサポートされる最短間隔 (短い方) です。

位置データを提供するデバイスでは、さまざまなアプリから要求されるレポート間隔を追跡し、要求された最短の間隔でデータをレポートする場合があります。これにより、精度の要件が最も高いアプリに必要なデータを提供できます。そのため、別のアプリで要求された更新頻度の方が高い場合は、要求した頻度よりも頻繁に更新が生成されることがあります。

注 位置情報の提供元からのレポート間隔は、必ずしも要求どおりになるとは限りません。 位置情報取得機能デバイスによってはレポート間隔を追跡しないものもありますが、追跡されるものとして指定しておくことをお勧めします。

- 電力を節約するには、**DesiredAccuracy** プロパティを設定して、アプリで高い精度のデータが必要かどうかを位置情報プラットフォームに示します。高い精度のデータを必要とするアプリがなければ、GPS 位置情報取得機能を無効にして電力を節約できます。
  - GPS でデータを取得するには、**DesiredAccuracy** を **HIGH** に設定します。
  - ターゲティング広告のためにのみ位置情報を使うアプリは、消費電力を最小限に抑えるため、**DesiredAccuracy** を **Default** に設定します。

精度についてアプリに特定のニーズがある場合は、**DesiredAccuracy** を使う代わりに **DesiredAccuracyInMeters** プロパティを使うこともあります。これは、通常、位置情報を移動体通信ビーコン、Wi-Fi ビーコンや衛星に基づいて取得できる Windows Phone に特に役立ちます。より具体的な精度値を選ぶと、システムが位置情報を提供する際に最も低い消費電力で適切なテクノロジを識別するために役立ちます。

たとえば、次のような場合があります

- アプリが広告の調整、天気、ニュースなどのための位置情報を取得している場合は、一般に 5000 m の精度で十分です。
- アプリが地域内のごく近隣を表示する場合は、結果の表示には一般に 300 m の精度が適しています。

        - ユーザーがお勧めの近くのレストランを探している場合は、ブロック内の位置を取得する必要がありますので、100 m の精度で十分です。
        - ユーザーが自身の位置を共有しようとしている場合は、アプリには約 10 m の精度が必要です。
    - アプリに特定の精度の要件がある場合は **Geocoordinate.accuracy** プロパティを使います。たとえば、ナビゲーション アプリでは、**Geocoordinate.accuracy** プロパティを使って、利用可能な位置情報データがアプリの要件を満たしているかどうかを調べます。
- 起動時の待ち時間を考慮します。アプリで初めて位置データを要求したとき、位置情報取得機能が起動するまでに 1 ～ 2 秒の待ち時間が発生することがあります。アプリの UI を設計するときは、この点に注意してください。たとえば、**GetGeopositionAsync** の呼び出しを保留している他のタスクがブロックされないようにしてください。
- バックグラウンドの動作を考慮します。Windows ランタイム アプリにフォーカスがない場合、バックグラウンドで中断されている間は位置情報更新イベントを受け取りません。位置情報の更新をログに記録して追跡する場合は、この点に注意してください。アプリにフォーカスが戻った後は、新しいイベントだけを受け取ります。アプリが非アクティブだったときに発生した更新は取得されません。
- ロー センサーとフュージョン センサーを効率的に使います。センサーには、ローとフュージョンの 2 種類があります。
    - ロー センサーには、加速度計、ジャイロメーター、磁力計が含まれます。
    - フュージョン センサーには、向き、傾斜計、コンパスが含まれます。フュージョン センサーは、ロー センサーの組み合わせからデータを取得します。

    Windows ランタイム API は磁力計以外のすべてのセンサーにアクセスできます。フュージョン センサーの方がロー センサーよりも正確で安定していますが、より多くの電力を使います。用途に適したセンサーを使う必要があります。詳しくは、「センサー」をご覧ください。

コネクト スタンバイ: PC がコネクト スタンバイ状態にある場合、**Geolocator** オブジェクトはいつでもインスタンス化できます。しかし、**Geolocator** オブジェクトは集約する対象のセンサーを見つけることができず、**GetGeopositionAsync** の呼び出しは 7 秒後にタイムアウトします。**PositionChanged** イベント リスナーの呼び出しは行われず、

StatusChanged イベント リスナーは 1 回呼び出され、そのステータスは **NoData** となります。

その他の使い方のガイダンス

位置情報設定の変更を検出する

ユーザーは、**設定**アプリの**位置情報に関するプライバシー設定**を使って、位置情報機能を無効にすることができます。

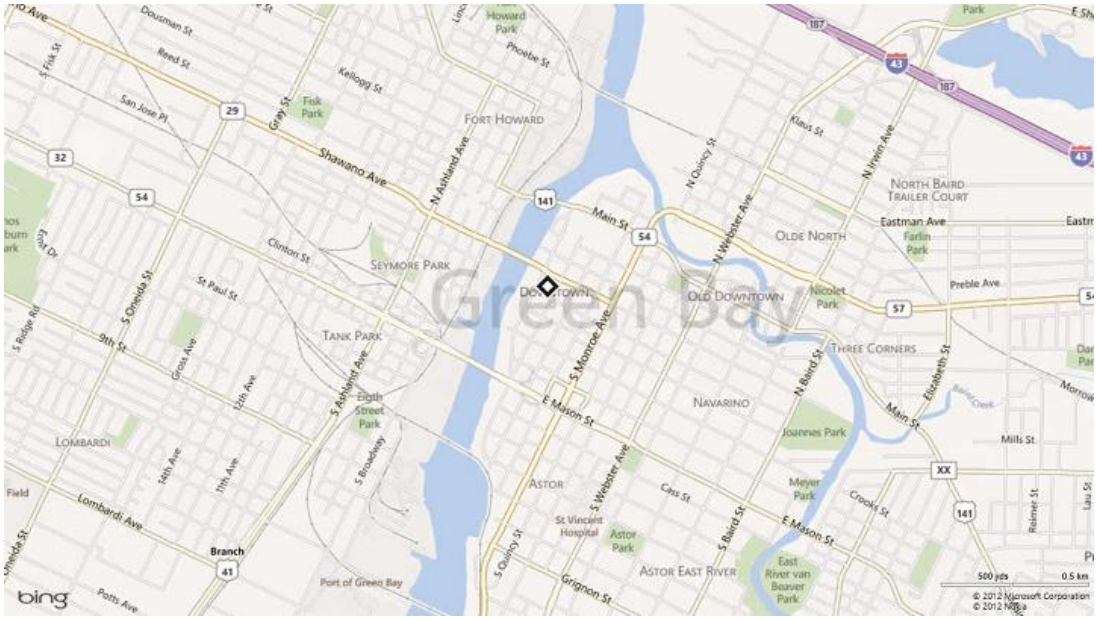
- ユーザーが位置情報サービスを無効にしたり再び有効にしたことを検出するには、次の操作を行います。
  - StatusChanged イベントを処理します。**StatusChanged** イベントの引数である Status プロパティの値は、ユーザーが位置情報サービスを無効にすると **Disabled** になります。
  - GetGeopositionAsync から返るエラー コードをチェックします。ユーザーによって位置情報サービスが無効にされている場合、**GetGeopositionAsync** の呼び出しは **ACCESS_DENIED** エラーで失敗し、LocationStatus プロパティの値は **Disabled** になっています。
- 地図アプリのような、位置情報データが必須のアプリの場合は、必ず次の操作を実行してください。
  - ユーザーの位置情報が変わったときに更新情報を取得できるように、PositionChanged イベントを処理します。
  - 前の説明に従って StatusChanged イベントを処理し、位置設定の変化を検出します。

位置情報 API は、データが利用可能になったときにデータを返します。最初に誤差の範囲が大きい位置情報を返し、より正確な情報が利用可能になったときに位置情報を更新する場合があります。ユーザーの位置情報を表示するアプリでは、通常、より正確な情報が利用可能になったときに位置情報を更新する必要があります。

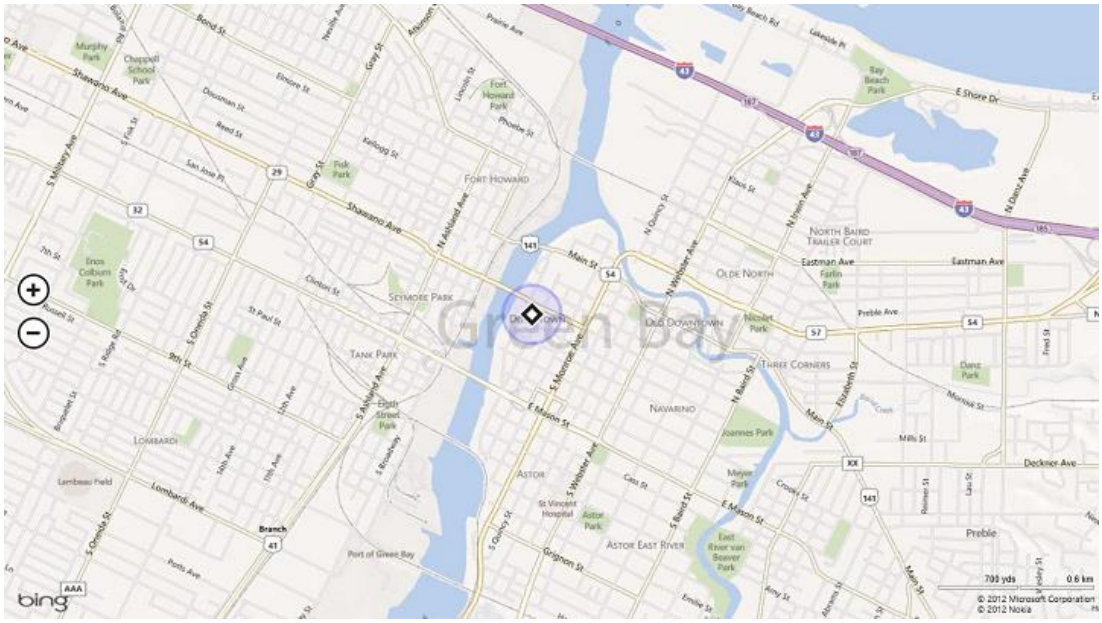
位置情報のグラフィックス表示

アプリでは、**Geocoordinate.accuracy** を使って、ユーザーの現在の位置情報を地図に明確に示すようにします。精度の幅は、主に、誤差の範囲が半径約 10 m、半径約 100 m、半径 1 km 超、という 3 種類があります。精度情報を使うことにより、アプリでは、利用可能なデータの状況に応じて位置情報を正確に表示することができるようになります。マップ コントロールの使い方については、「上空から撮影した立体写真や平面写真、市街地を表示する」を参照してください。

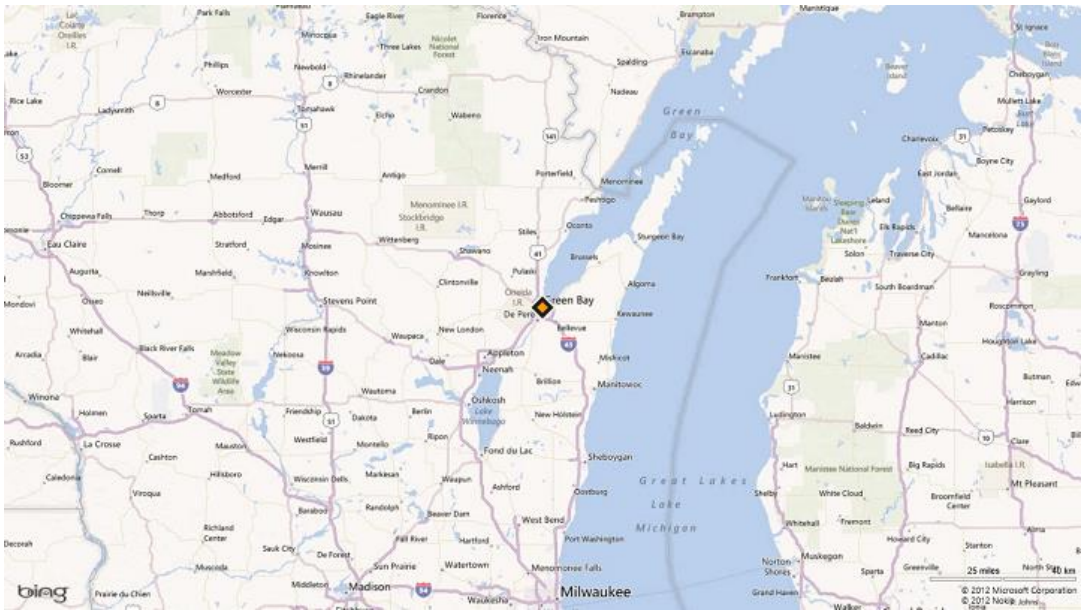
- 約 10 m 相当の精度 (GPS の解像度) の場合、位置情報は点またはピンで地図上に示すことができます。この精度では、経度と緯度の座標、住所の番地も表示できます。



- 10 ～ 500 m (おおよそ 100 m) の精度の場合、位置情報は通常、Wi-Fi による解決で受信されています。移動体通信から取得した位置情報は約 300 m の精度です。この場合、アプリでは誤差を含む範囲を表示することをお勧めします。道順を表示するアプリなど中心点が必要となる場合は、その中心点を誤差を表す範囲で囲むことができます。



- 戻される精度が 1 km より大きい場合は、IP レベルの解決で位置情報を受信することになります。多くの場合、地図上に特定の地点をピンポイントで表示するためにはこのレベルの精度は低すぎます。アプリではピンや円で地理的な場所を表現することは控え、その代わりに、地図を市のレベルまで、または誤差の範囲に応じて適切なエリア (たとえば、地域のレベル) までズームすることをお勧めします。



位置情報の精度が別の精度に切り替わるときは、異なるグラフィックス表示が適切に遷移するようにします。このためには、次のようにします。

- 切り替え時のアニメーションをスムーズにし、切り替えを高速かつ滑らかに保ちます。

- 数回の連続的な報告があるのを待ってから、精度が変わったと判断します。これにより、不要なズームが頻繁に行われるのを防ぐことができます。

位置情報のテキスト表示

天気アプリや地域情報アプリなどアプリの種類によっては、さまざまな精度の位置情報をテキストで表現することが必要になります。位置情報は、データが提供する精度レベルまでに抑えて、明確に表示するようにします。

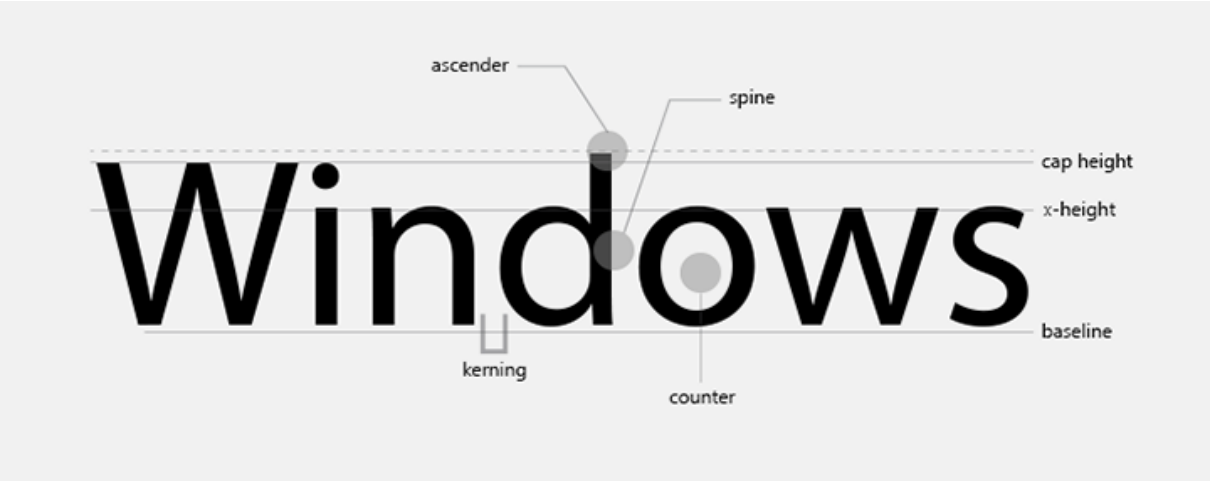
- 精度が約 10 m 相当 (GPS の解像度) の場合、受信した位置情報データは相当に正確であるので、ごく近隣の地名のレベルで情報を伝えることができます。市の名前、都道府県の名前、国/地域の名前も使うことができます。
- 精度が約 100 m 相当 (Wi-Fi の解像度) の場合、受信した位置情報データはある程度正確です。市の名前までの情報を表示することをお勧めします。それより詳しい、近隣地域の地名の使用は避けてください。
- 1 km 超の精度 (IP による解決) の場合は、都道府県の名前、または国/地域の名前のみ表示します。

プライバシーに関する考慮事項

ユーザーの地理的な位置情報は、個人を特定できる情報 (PII) に当たります。ユーザーのプライバシーの保護に関するガイダンスについては、次の Web サイトをご覧ください。

- Microsoft のプライバシー

テキストと入力



メール、本、道路標識、メニューに書かれた価格、タイヤの空気圧のマーキング、道路脇のポールに掲示されたポスターなど、日常生活の中で文字を目にする機会はたくさんあります。私たちの周りには、エンターテインメント性のあるテキスト、情報を提供するテキスト、教育的なテキスト、意味のあるテキストが氾濫しています。その中心となるのが、タイポグラフィとレイアウトです。

ここでは、フォントとタイポグラフィ、アイコン、テキスト ボックス、テキストの形式、テキストのコピーと貼り付けのコマンドについて説明します。さらに、ドキュメント ビューアーやリーダー アプリなどでページの特定のテキストを検索するためのガイドラインも示します。

このセクションの内容

トピック	説明
クリップボード コマンド	クリップボード コマンド (コピー、貼り付け、切り取り) を使うと、ユーザーが使い慣れた方法で、ある場所から別の場所にコンテンツを移動することができます。これらのコマンドは、ユーザーがコンテンツを移動する際に役立ちます。 • 同じアプリ内 • ユニバーサル Windows プラットフォーム (UWP) アプリ間 • クラシック Windows アプリケーション間 • UWP アプリとクラシック Windows アプリケーションの間 Windows 10 では、コンテンツの共有など、アプリが情報を交換するための他の方法もサポートしていますが、コピーと貼り付けのコマンドは、Windows の操作環境として想定されている部分として残ります。可能な場合には常に、アプリでこれらのコマンドをサポートしてください。
ページ内検索	このトピックでは、Windows ストア アプリでのページ内検索機能の実装に関するベスト プラクティスについて説明します。
フォント	フォントを選び、フォントのサイズと色を指定するときには、次のガイドラインに従ってください。
フォーム レイアウト	優れたタッチ エクスペリエンスを提供し、画面上の領域の使用を最適化し、Windows ストア アプリでパンとスクロールを最小限にするフォームを設計します。
パスワード ボックス	パスワード ボックスは、プライバシーの目的で入力文字が非表示になるテキスト入力ボックスです。
Segoe MDL2 アイコンの一覧	このドキュメントでは、Segoe MDL2 Assets フォントが提供する、アイコンとして使うことができる便利なグリフを示します。
スペル チェック	テキストの入力と編集を行っているときに、スペル チェックは単語を赤い波線で強調表示してユーザーに単語のスペルの間違いを知らせ、それを修正する方法を提供します。
テキスト入力	JavaScript を使った Windows アプリにテキスト入力コントロールを追加するためのガイドライン。

クリップボード コマンドのガイドライン

クリップボード コマンド (コピー、貼り付け、切り取り) を使うと、ユーザーが使い慣れた方法で、ある場所から別の場所にコンテンツを移動することができます。これらのコマンドは、ユーザーがコンテンツを移動する際に役立ちます。

- 同じアプリ内
- ユニバーサル Windows プラットフォーム (UWP) アプリ間
- クラシック Windows アプリケーション間
- UWP アプリとクラシック Windows アプリケーションの間

Windows 10 では、コンテンツの共有など、アプリが情報を交換するための他の方法もサポートしていますが、コピーと貼り付けのコマンドは、Windows の操作環境として想定されている部分として残ります。可能な場合には常に、アプリでこれらのコマンドをサポートしてください。

重要な API

PopupMenu クラス

推奨事項

- ドキュメントや画像のサブセットなど、ユーザーが明示的に選択できる編集可能なすべてのコンテンツに対して、コピーと貼り付けをサポートします。
- ユーザーがコピーと貼り付けコマンドを使いたいと考える可能性のあるコンテンツに対して、コマンドのサポートを検討します。

  例：
  - フォト ギャラリー アプリケーション内の画像
  - 電卓の計算結果
  - レストラン検索アプリケーションでのレストランの住所
- 権利の管理や、コピーと貼り付けのコマンドの使用に制限があると思われるその他の要因に留意してください。たとえば、閲覧権利制限付きメールをサポートするアプリの場合、そのようなコンテンツの全部または一部をユーザーがコピーすることをポリシーによって制限する場合があります。

- ユーザーがコピーする対象、またはユーザーがコンテンツを貼り付けることができる場所が明確であるか確認します。
- アプリケーション内の編集可能な領域およびキャンバスに対してのみ、貼り付けのサポートを提供します。
- コピーと貼り付けを使うとコンテンツが削除または置換される場合があるため、元に戻すコマンドの実装を検討します。
- コピーと貼り付けをサポートするコントロールが既にある場合は、そのコントロールの実装を使います。コピーと貼り付けの実装を独自にビルドする必要がある場合は、作る操作環境とそれらのコントロールの一貫性を保つようにします。
- コピーもサポートしている場合は、共有のサポートも検討します。
- ユーザーがコンテキスト メニューとコマンド バーのどちらを使ってコピーと貼り付けのコマンドにアクセスするかを決定します。次の場合は、コンテキスト メニューを使います。
  - ハイパーリンクや埋め込み画像など、タップしたまま押さえるジェスチャでのみ選ぶことができる項目たとえば、アプリでユーザーにアドレスを表示し、ユーザーがそのアドレスをコピーできるようにするとします。最適なユーザー エクスペリエンスは、ユーザーがアドレスを右クリックするか、タップしたまま押さえるとアクセスできるアドレスのコピー コマンドを作ることです。このコマンドによってアドレスがクリップボードにコピーされ、そのクリップボードからユーザーは希望するアプリに貼り付けることができます。

  - テキスト選択 (編集可能と読み取り専用の両方)
  - カーソル位置や表のセルなど、対象が明確に定義されている場合の貼り付け操作

前のガイドラインに該当しない場合は、コマンド バーを使います。次のような例があります。

  - 複数の項目の選択をアプリでサポートする場合
  - ユーザーが画像の一部を選ぶことができる場合

- スクリーン ショットをキャンバスに貼り付けるなど、貼り付けコマンドの対象が明確な場合
- クリップボード コマンドのキーボード ショートカットをサポートすることを強くお勧めします。
- 明示的に選ぶことができない、またはコンテキスト メニューを使って選ぶことができないコンテンツのコピーはサポートしないでください。
- クリップボードが空であるか、クリップボードにアプリでサポートされないコンテンツが含まれている場合は、貼り付けコマンドを有効にすることは避けてください。

ページ内検索のガイドライン

ページ内検索により、ユーザーは現在のテキスト本文から一致を検索できるようになります。ページ内検索が提供される最も一般的なアプリは、ドキュメント ビューアー、リーダー、ブラウザーです。

推奨事項

- ユーザーがページ内のテキストを検索できるように、ページ内検索機能を備えたコマンド バーをアプリ内に配置します。配置について詳しくは、「例」をご覧ください。
  - ページ内検索を提供するアプリでは、必要なすべてのコントロールがコマンド バーに含まれている必要があります。
  - ページ検索以外に多くの機能をアプリに含める場合は、ページ内検索のすべてのコントロールが含まれる別のコマンド バーへのエントリ ポイントとしてトップ レベルのコマンド バーに **[検索]** ボタンを追加できます。
  - ユーザーがタッチ キーボードを操作しているときもページ内検索のコマンド バーが表示されるようにします。タッチ キーボードは、ユーザーが入力ボッ

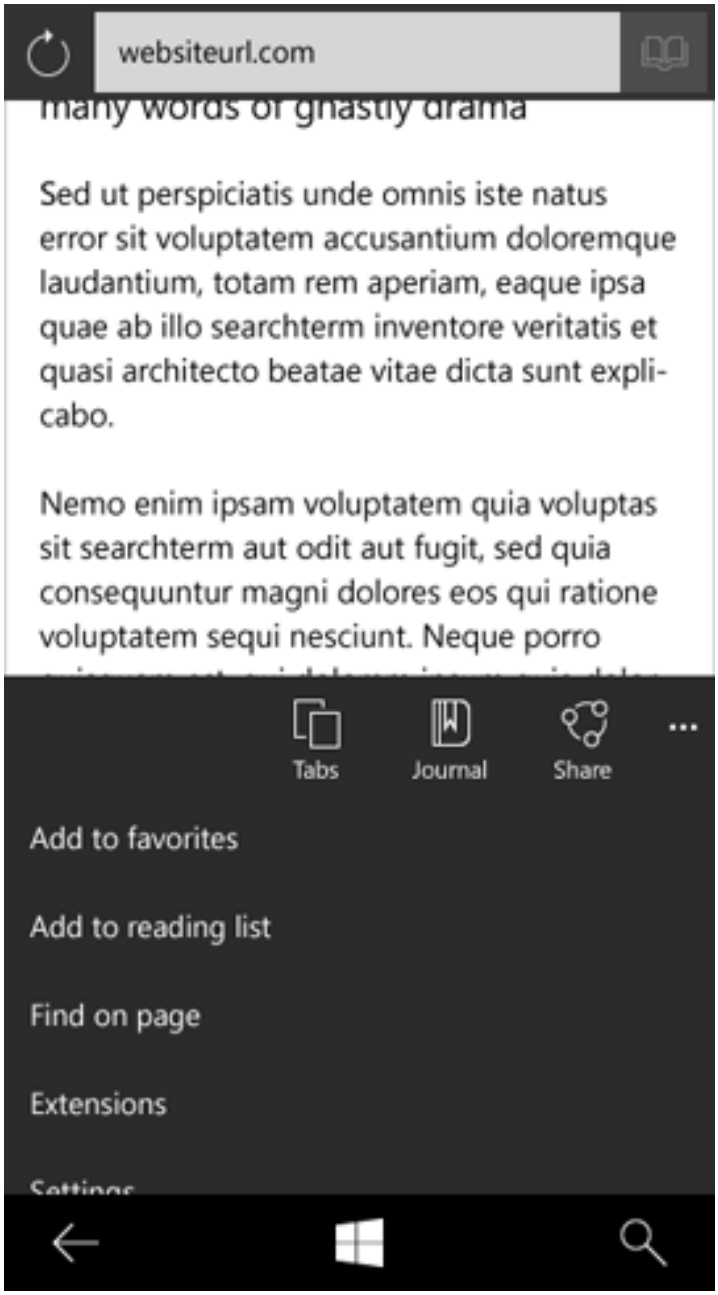
クスをタップすると表示されます。ページ内検索のコマンド バーは、タッチ キーボードに隠れないように上へ移動する必要があります。

- ユーザーが表示を操作しているときもページ内検索を利用できるようにする。ユーザーは、ページ内検索を使いながら表示内のテキストを操作する必要があります。たとえば、ユーザーはテキストを読むために、ドキュメントを拡大表示または縮小表示したり、表示をパンしたりすることがあります。ユーザーがページ内検索を使い始めたら、コマンド バーはページ内検索を終了するための [**閉じる**] ボタンと共に表示されたままにする必要があります。
- キーボード ショートカット (Ctrl + F) を有効にする。キーボード ショートカット Ctrl + F を実装し、ページ内検索のコマンド バーをユーザーがすぐに呼び出すことができるようにします。
- ページ内検索機能の基本要素を含める。ページ内検索を実装するために必要な UI 要素を次に示します。
  - 入力ボックス
  - [前へ] ボタンと [次へ] ボタン
  - 一致数
  - 閉じる (クラシック Windows アプリのみ)
- 表示で一致した結果が強調表示され、スクロールして次の一致が画面に表示されるようにする。ユーザーは、**[前へ]** ボタンと **[次へ]** ボタンの使用、スクロール バーの使用、またはタッチによる直接操作によってドキュメントをすばやく移動できます。
- 検索と置換の機能が基本的なページ内検索機能と共に機能するようにする。検索と置換の機能があるアプリでは、ページ内検索が検索と置換の機能の妨げにならないようにします。

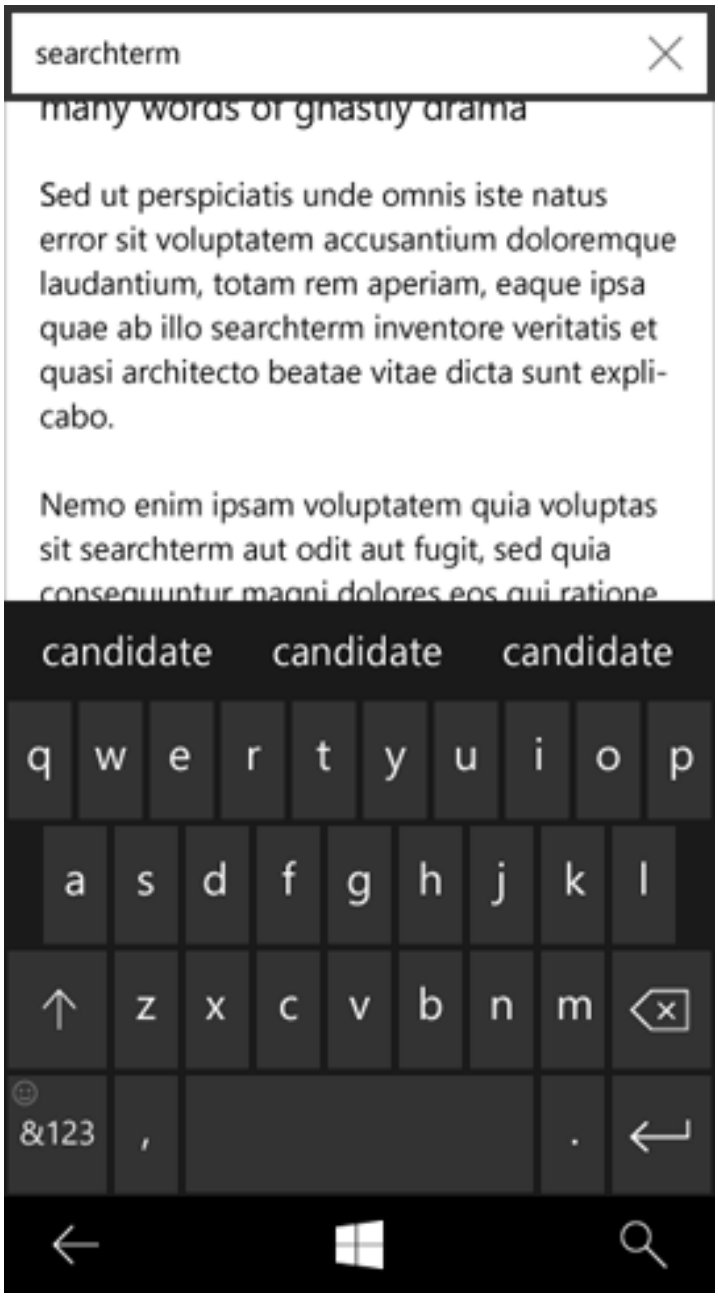
- ページ上にあるテキスト一致数がユーザーにわかるように、一致カウンターを追加します。
- キーボード ショートカット (Ctrl + F) を有効にする。

例

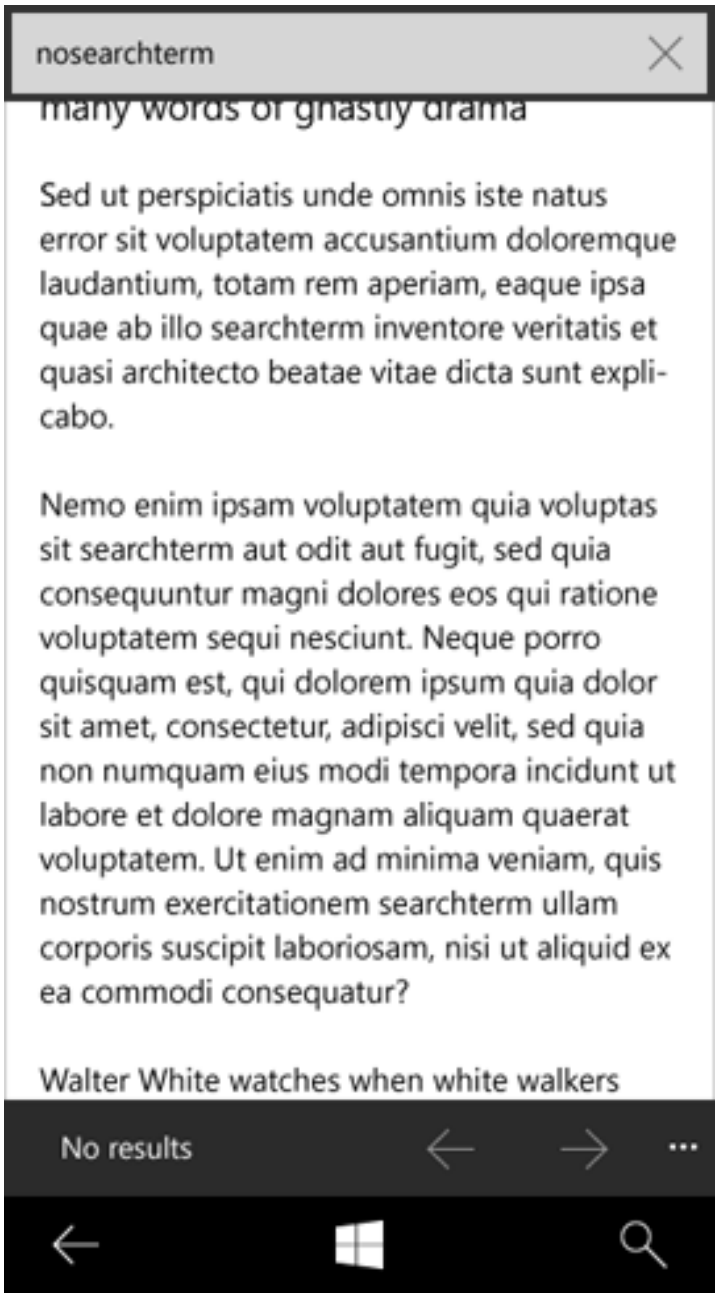
ページ内検索機能にアクセスする簡単な方法を提供します。ここに示すモバイル UI の例では、展開可能なメニューで、2 つの追加コマンドの後に [ページ内を検索] が表示されています。



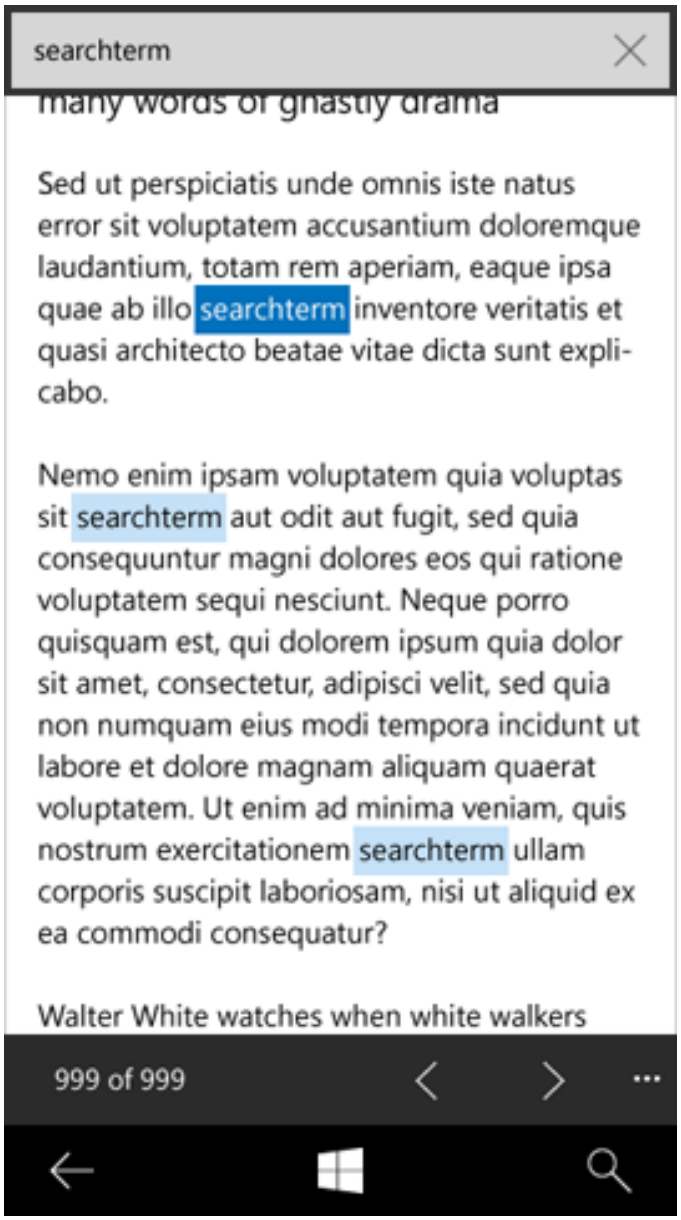
ユーザーは、[ページ内を検索] を選択してから検索語句を入力します。検索語句の入力中に、テキスト入力候補を表示できます。



検索で一致するテキストがなかった場合は、結果ボックスに "検索結果が見つかりませんでした" というテキストを表示します。



検索で一致するテキストがあった場合は、最初の語句を区別できる色で強調表示します。以降の一致は、例に示されているように、同じカラー パレットのもう少し淡いトーンで強調表示します。



ページ内検索には、一致カウンターがあります。

その他の使い方のガイダンス

ページ内検索の実装

- ドキュメント ビューアー、リーダー、ブラウザーは、ページ内検索が提供される最も一般的なアプリの種類であり、全画面での表示/読み取りエクスペリエンスをユーザーに提供します。
- ページ内検索機能は補助的な機能であり、コマンド バーに配置する必要があります。

フォントのガイドライン

フォントのサイズ、太さ、色、トラッキング、間隔を適切に設定すると、外観がきれいに整い、ユニバーサル Windows プラットフォーム (UWP) アプリが使いやすくなります。フォントを選び、フォントのサイズと色を指定するときには、次のガイドラインに従ってください。

Segoe UI Symbol アイコンの一覧については、「Segoe UI MDL2 Aseets アイコンのガイドライン」をご覧ください。

重要な API

FontFamily プロパティ (XAML)

font-family プロパティ (HTML)

font-size プロパティ (HTML)

Windows 10 の書体見本

書体見本 (type ramp) は、ヘッドラインからの本文までの重要なデザインの関係を確立し、異なるレベル間の明快でわかりやすい階層を保証します。ユーザーは、情報を見つける場所やページの解析方法をすぐに理解できます。

UWP アプリ用にお勧めする書体見本を次に示します。

テキスト スタイル	書体	太さ	サイズ (epx)	行間 (epx)	単語の間隔	トラッキング (1/1000em)	XAML スタイル キー
ヘッダー	Segoe UI	細い	46	56	100%	0	Header TextBlockStyle
サブヘッダー	Segoe UI	細い	34	40	100%	0	Subheader TextBlockStyle
タイトル	Segoe UI	Semi light	24	28	100%	0	Title TextBlockStyle
サブタイトル	Segoe UI	通常	20	24	100%	0	Subtitle TextBlockStyle
ベース	Segoe UI	Semi bold	15	20	100%	0	Base TextBlockStyle
本文	Segoe UI	通常	15	20	100%	0	Body TextBlockStyle
キャプション	Segoe UI	通常	12	14	100%	0	Caption TextBlockStyle

推奨フォント

すべてに対して、必ずしも Segoe UI フォントを使う必要はありません。読み取りや英語以外の言語でのテキストの表示など、特定のシナリオでは、他のフォントを使うことができます。

ここでは、UWP アプリをサポートするすべての Windows 10 エディションで利用可能なことが保証されているフォントの一覧を示します。

注 この一覧に含まれていないフォントを使う場合、アプリは Microsoft サービスのフォント データの自動ダウンロードをトリガーすることができます。これは、特にモバイル デバイスでは、パフォーマンスやその他の影響が懸念の対象となる可能性があります。具体的には、これによりユーザーのモバイル データ プランの一部が使用されたり、モバイル データ

使用コストが発生したりする可能性があります。モバイル デバイスで利用できる UWP アプリでは、UI コンテンツにこのリスト以外のフォントを使用することはできません。

Font-family	スタイル	コメント
Arial	通常、斜体、 太字、太字斜体、 黒	
Calibri	通常、斜体、 太字、太字斜体、 中細、中細斜体	
Cambria	通常	
Cambria Math	通常	
Comic Sans MS	通常、斜体、 太字、太字斜体	
Courier New	通常、斜体、 太字、太字斜体	
Ebrima	通常、太字	アフリカのスクリプト (エチオピア文字、ンコ文字、オスマニア文字、ティフィナグ文字、ヴァイ文字) 用のユーザー インターフェイス フォント
Gadugi	通常	北アメリカ スクリプト (カナダ音節文字、チェロキー文字) 用のユーザー インターフェイス フォント
Georgia	通常、斜体、 太字、太字斜体	

ジャワ文字のテキスト ジャワ文字のスクリプト用の標準フォールバック フォント	通常	ジャワ文字のスクリプト用のフォールバック フォント
Leelawadee UI	通常、Semilight、太字	東南アジアのスクリプト (ブギス文字、ラオス文字、クメール文字、タイ文字) 用のユーザー インターフェイス フォント
Lucida Console	通常	
Malgun Gothic	通常	韓国語用のユーザー インターフェイス フォント
Microsoft Himalaya	通常	チベット文字のスクリプト用のフォールバック フォント
Microsoft JhengHei	通常	
Microsoft JhengHei UI	通常	繁体字中国語用のユーザー インターフェイス フォント
Microsoft New Tai Lue	通常	新タイ ロ文字のスクリプト用のフォールバック フォント
Microsoft PhagsPa	通常	パスパ文字のスクリプト用のフォールバック フォント
Microsoft Tai Le	通常	タイ ロ文字のスクリプト用のフォールバック フォント
Microsoft YaHei	通常	
Microsoft YaHei UI	通常	簡体字中国語用のユーザー インターフェイス フォント
Microsoft Yi Baiti	通常	イ文字のスクリプト用のフォールバック フォント
Mongolian Baiti	通常	モンゴル文字のスクリプト用のフォールバック フォント
MV Boli	通常	ターナ文字のスクリプト用のフォールバック フォント

Myanmar Text	通常	ミャンマー文字のスクリプト用のフォールバック フォント
Nirmala UI	通常、Semilight、太字	南アジア言語のスクリプト (バングラ文字、デーバナーガリー文字、グジャラート文字、グルムキー文字、カンナダ文字、マラヤーラム文字、オディア文字、オル チキ文字、シンハラ文字、ソラング ソンペング文字、タミール文字、テルグ文字) 用のユーザー インターフェイス フォント
Segoe MDL2 アセット	通常	アプリ アイコン用のユーザー インターフェイス フォント
Segoe Print	通常	
Segoe UI	通常、斜体、太字、斜体、太字斜体、中細、Semilight、Semibold、黒	ヨーロッパおよび中東のスクリプト (アラビア文字、アルメニア文字、キリル文字、ジョージア文字、ギリシャ文字、ヘブライ文字、ラテン文字) およびリス文字のスクリプト用のユーザー インターフェイス フォント
Segoe UI Emoji	通常	Windows Phone に付属しているバージョンでは、各絵文字の周りに白い線が示されており、どのような色の背景にでも表示されます。Windows に付属するバージョンと測定的に互換性があります。
Segoe UI Historic	通常	歴史上のスクリプト用のフォールバック フォント
Segoe UI Symbol	通常	記号用のフォールバック フォント
SimSun	通常	
Times New Roman	通常、斜体、太字、太字斜体	
Trebuchet MS	通常、斜体、太字、太字斜体	

Verdana	通常、斜体、太字、太字斜体	
Webdings	通常	
Wingdings	通常	
Yu Gothic	中	
Yu Gothic UI	通常	日本語用のユーザー インターフェイス フォント

フォーム レイアウトのガイドライン

優れたタッチ エクスペリエンスを提供し、画面上の領域の使用を最適化し、ユニバーサル Windows プラットフォーム (UWP) アプリでパンとスクロールを最小限にするフォームを設計します。

推奨事項

- コンテンツとアプリに適したフォーム レイアウトを使います。
- フォーム内のすべてのコントロールで、ラベルの配置スタイルを同じにします。
- フォームのコンテンツが単純またはわかりやすい場合は、インライン プレース ホルダーを使います。
- 垂直方向に広いパンがある場合は、複数の列を使わないでください。
- ラベルの長さがさまざまに異なる場合は、左側にラベルを配置しないでください。
- タッチ入力が行われない場合、タッチ キーボードを自動的に起動しないでください

その他の使い方のガイダンス

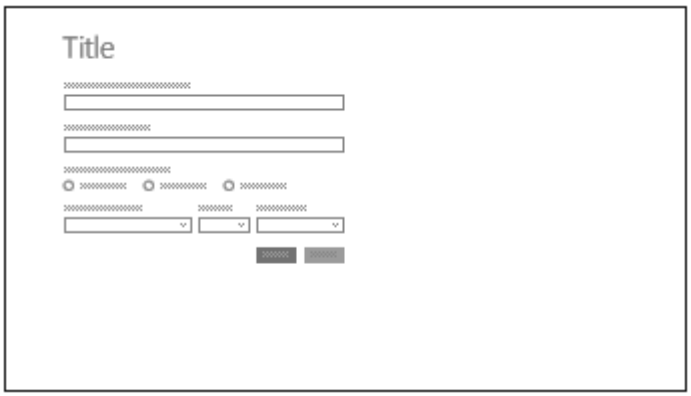
フォームとコントロールのレイアウトを設計するときは、ユーザーにフォームでどのように入力してもらうかと、パンとスクロールがエクスペリエンスにどのような影響を与えるかについて考慮します。 使用する場合は、タッチ キーボード (横長で、画面の最大 50% を使用できます) とインライン エラー通知の影響も考慮します。

1 列のレイアウト

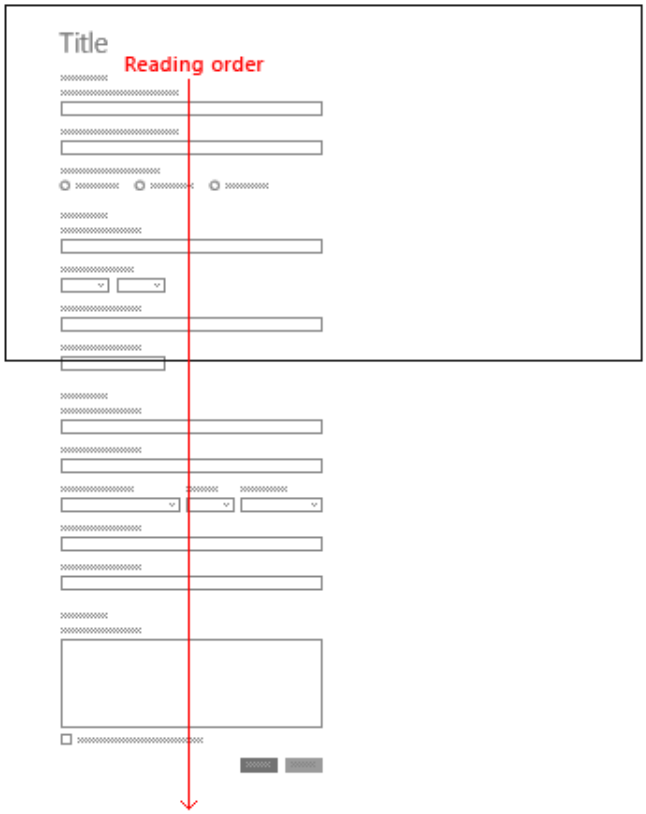
次の場合には、フォームで 1 列のレイアウトを使います。

- ユーザーに特定の順序でフォームに入力してもらいたい場合。
- フォームが複数のページにわたる場合。
- アプリは縦長で幅の狭いレイアウトに合わせてサイズ変更されます。
- アプリが追加のコメント、情報、指示、ブランド情報、広告を表示する場合。

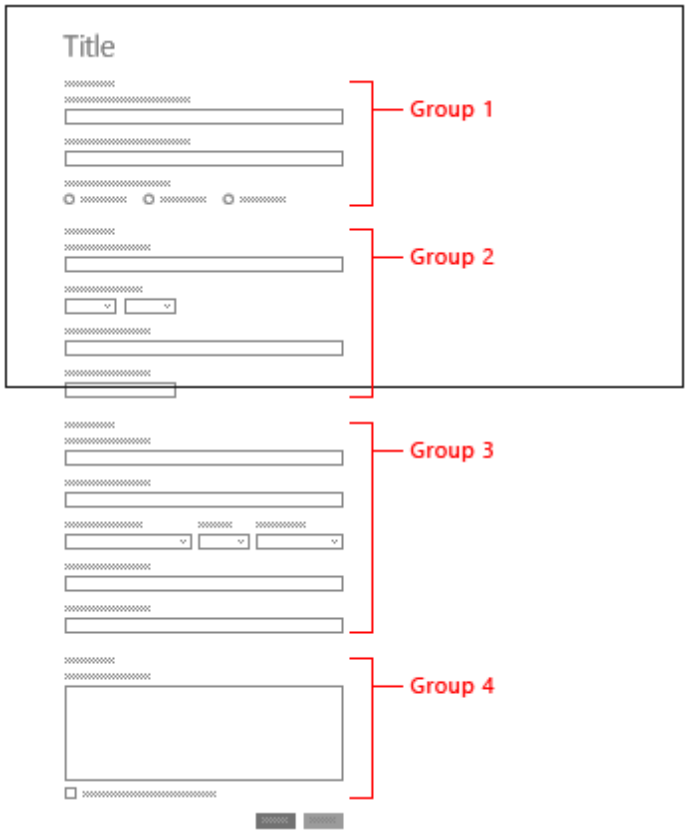
1 列のレイアウトを使った短いフォームの例を次に示します。



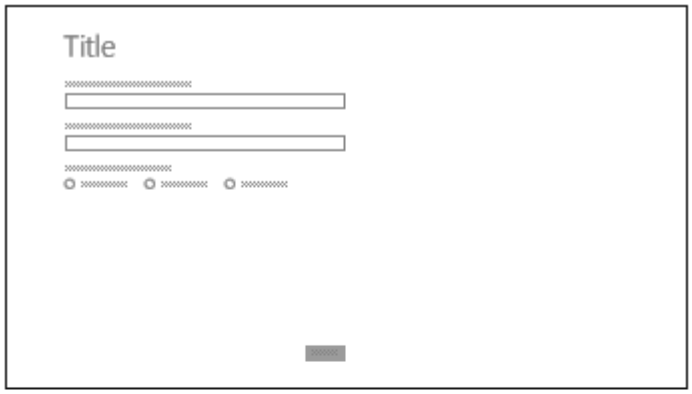
1 列のレイアウトを使った長いフォームの例を次に示します。

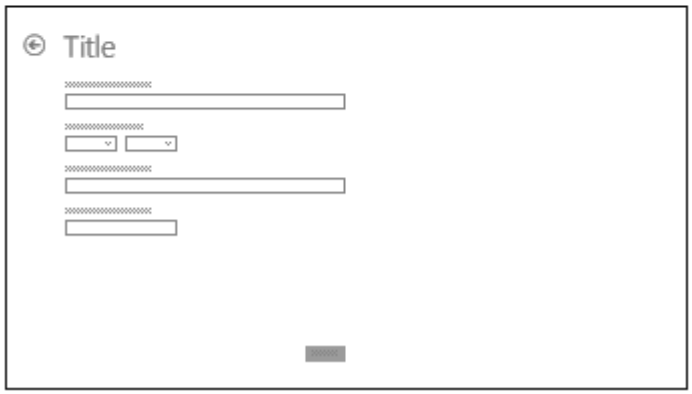


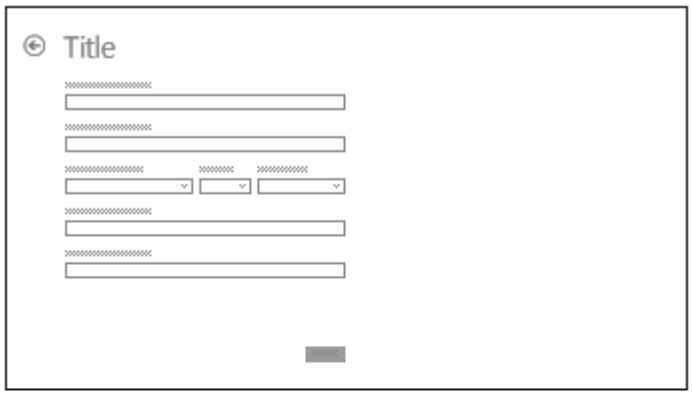
多数のコントロールを 1 つの非常に長いフォームに収めようとするのではなく、タスクをグループや一連の複数のフォームに分けることを検討してください。グループに分けた長いフォームの例を次に示します。

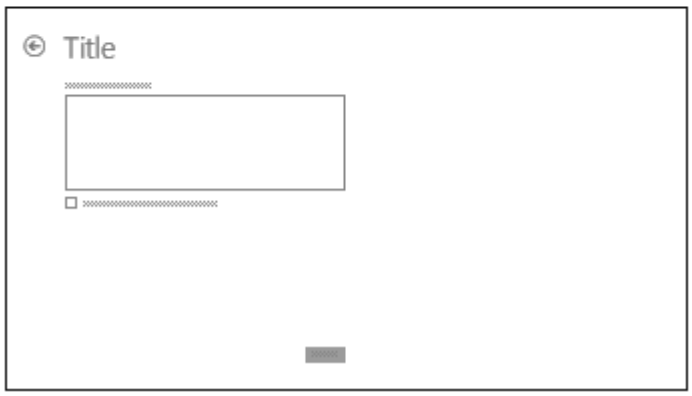


複数のページに分けた長いフォームの例を次に示します。

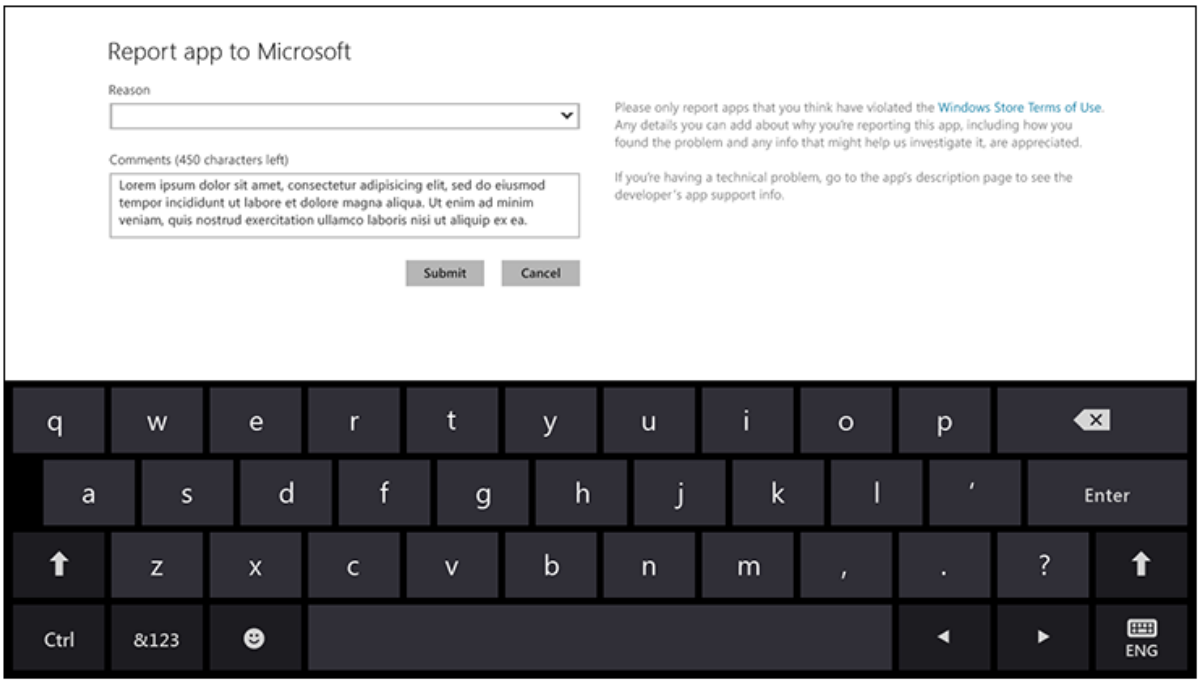








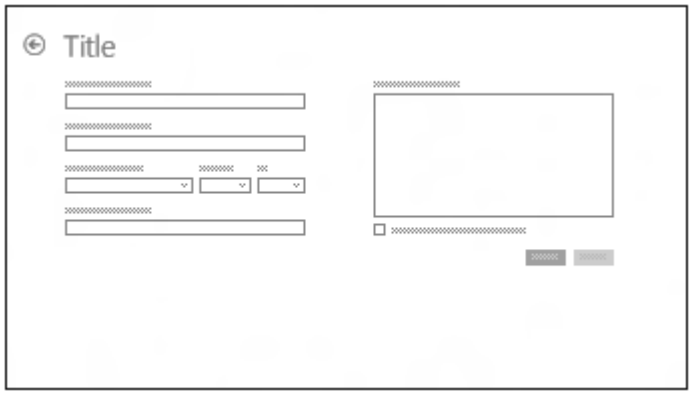
UI に追加のコメントと情報が含まれている場合に 1 列のレイアウトを使ったフォームの例を次に示します。



2 列のレイアウトを使う場合

垂直方向のパンに制限がある短いフォームの場合は、2 列のレイアウトを使います。 2 列のレイアウトでは、横方向の画面領域を最大限に活用します。 2 つの列の間には必ず十分な空間を確保してください。

2 列を使ったフォームの例を次に示します。



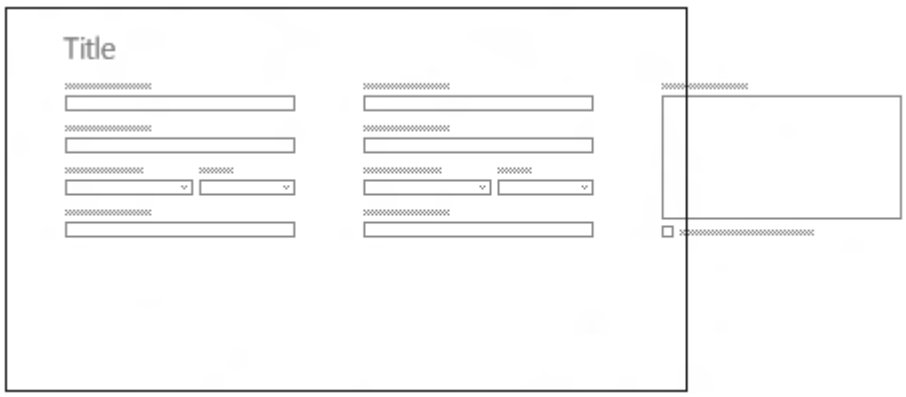
垂直方向に広いパンがある場合は、複数の列を使わないでください。フォームに入力するために、ユーザーは最初の列の一番下まで移動し、2 番目の列の先頭に戻って、また下に移動

しなければなりません。 このエクスペリエンスは、タッチ キーボードが表示されている場合、さらに扱いにくいものになります。

2 列を不適切に使ったフォームの例を次に示します。

3 列以上のレイアウトを使う場合

3 列以上のレイアウトは、横長で表示できる画面領域を最大限に活用するために使います。このレイアウトは、固定の画面または水平方向にパンする画面ではうまく機能しますが、垂直方向にパンする画面では扱いにくくなります。 入力順序が重要でない場合にのみ、このレイアウトを使ってください。



ラベルの配置

ほとんどのコントロールにはラベルが必要です。

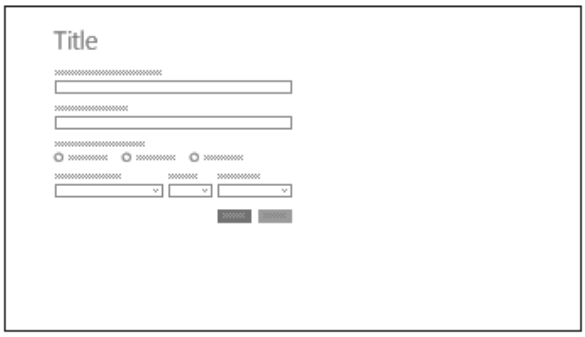
- コントロールの上側またはコントロールの左側にラベルを配置します。
- フォーム内のすべてのコントロールで、ラベルの配置スタイルを同じにします。

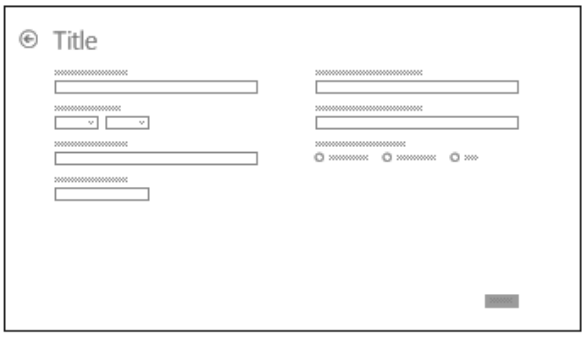
上側にラベルを配置する場合

通常は、コントロールの上側にラベルを配置します。段組フォーム レイアウトを使う場合は、常にコントロールの上側にラベルを配置します。

コントロールの上側にラベルを配置すると、ラベルとコントロールの関係を確立しやすくなり、すべてのラベルとコントロールを左揃えにできるので、すっきりしたレイアウトになります。コントロールの上側にラベルを配置すると、長い文字列を表示することや、ローカライズとアクセシビリティの問題に対処することが容易になります。

以下に、上側のラベル配置の例を 2 つ示します。



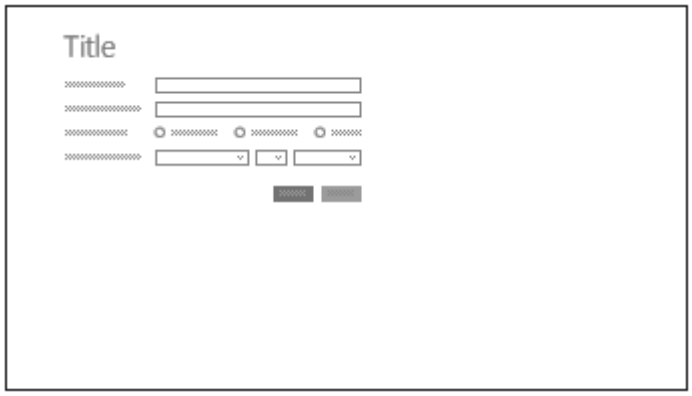


左側にラベルを配置する場合

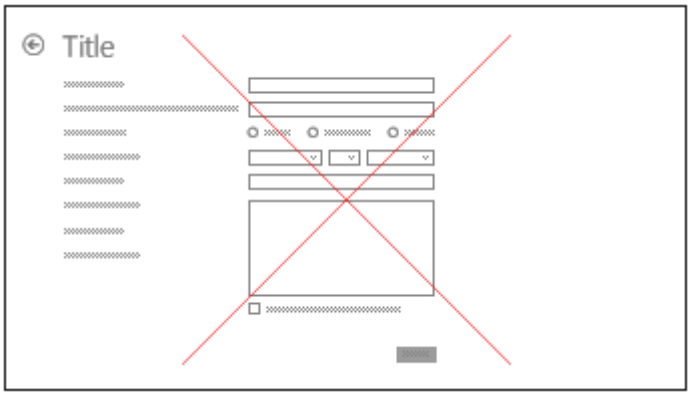
1 列のフォームで垂直方向の領域を節約する必要がある場合、次のときにはコントロールの左側にラベルを配置します。

- それぞれのラベル文字列が短く、長さがほぼ同じである。
- どのロケールでも、最も長いラベル文字列を表示するのに十分な横方向の領域がある。

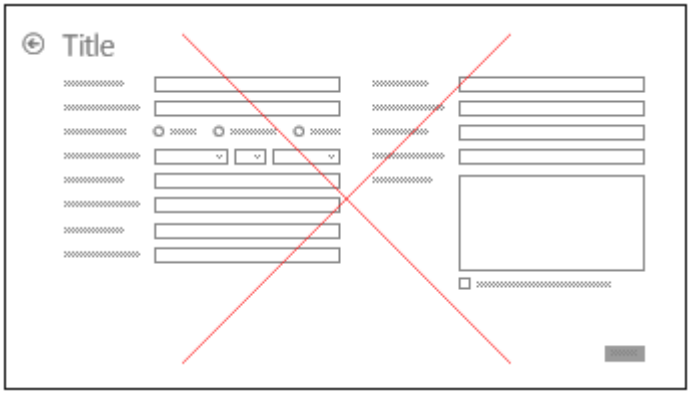
左側にラベルを配置したフォームの例を次に示します。



ラベルの長さがさまざまに異なる場合は、一部のラベルがコントロールから離れすぎた位置になるため、左側にラベルを配置しないでください。



段組フォーム レイアウトでは、左側のラベル配置を使わないでください。ラベル自体が別の列のように見える場合があります。

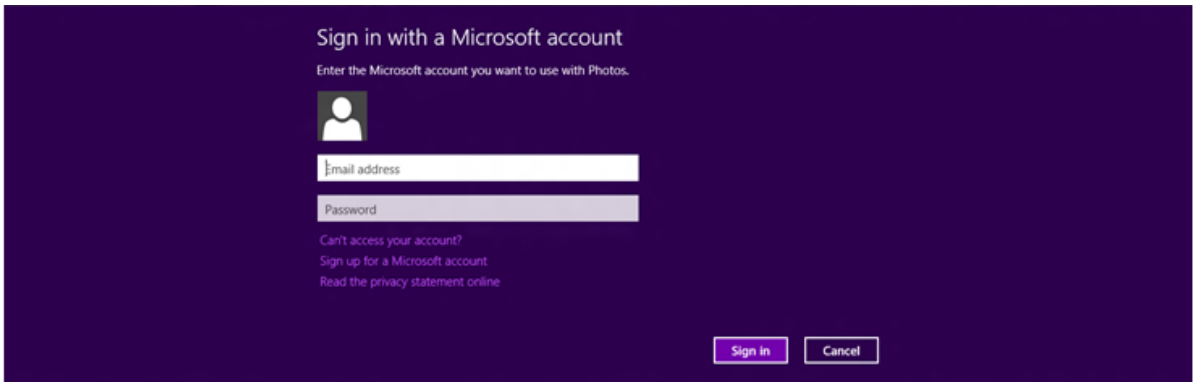


インライン プレースホルダー テキスト

ラベルの代わりにインライン プレースホルダー テキストを使うと、レイアウトが簡単になることがあります。インライン プレースホルダー テキストを使うのは次のような場合です。

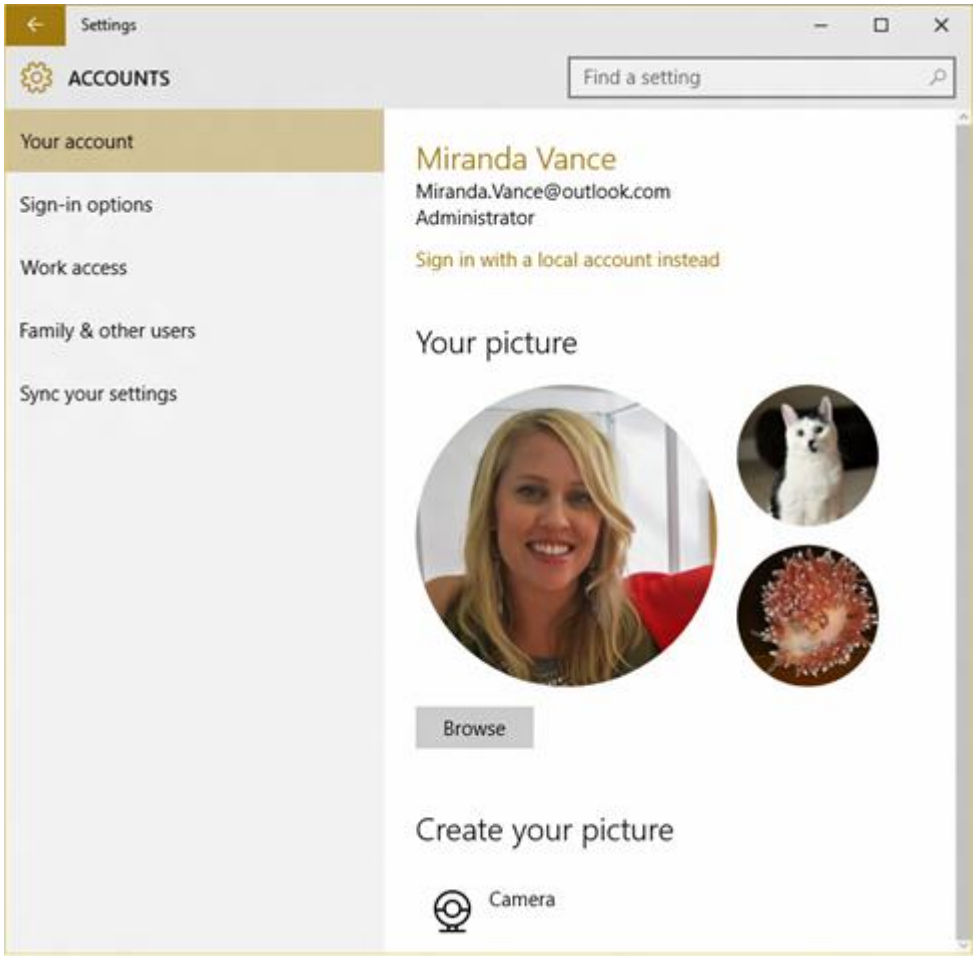
- フォームのパターンが多数のユーザーに一般的に理解されている場合 (たとえば、ユーザーのログオン制御やパスワード入力のフィールド)。
- 入力フィールドへのデータ入力後に簡単にラベルを推量したり解釈したりできる場合 (データの入力後にツールチップのテキストが非表示になるため)。
- 入力フィールドの数に制限がある場合。

インライン プレースホルダー テキストを使ったフォームの例を次に示します。



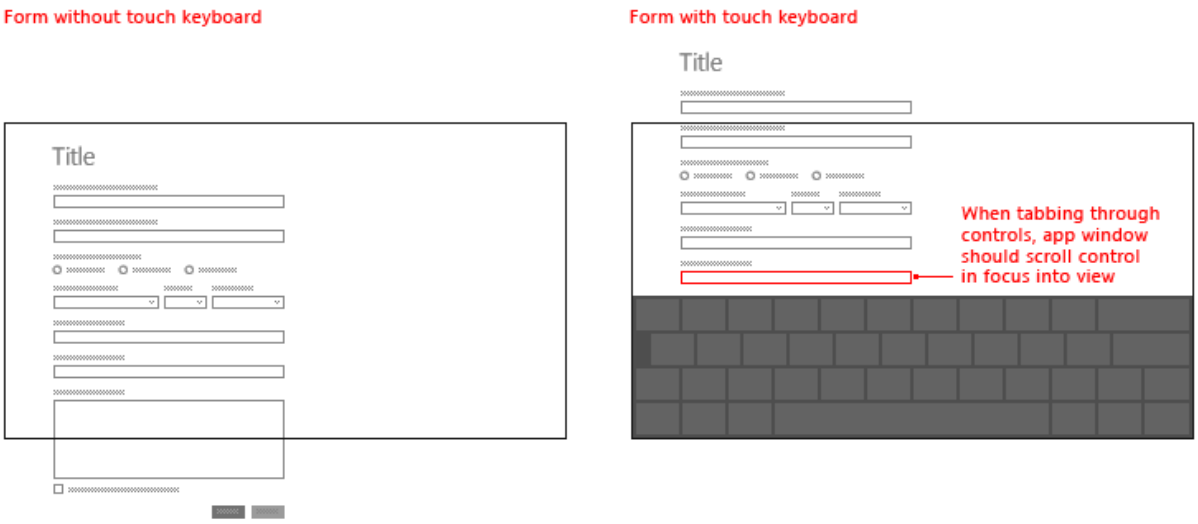
ボタンの配置

複数のボタンが他のコントロールに埋め込まれている場合以外は、フォームのボタンは右揃えにします。 フォームをより整理された外見にするには、ボタンを他のコントロールの配置と一致させる必要があります。 たとえば、**[設定]** の画面では、埋め込まれたボタンは左揃えになっています。

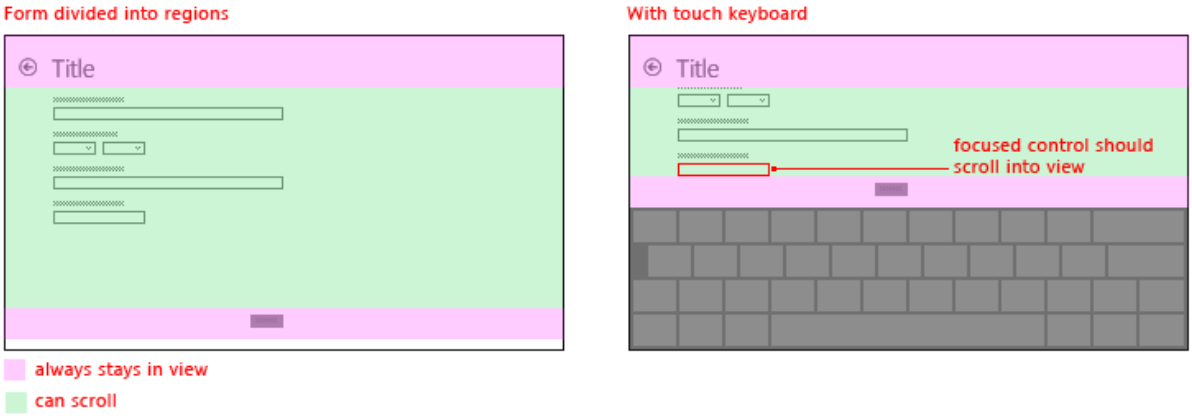


フォーカスとナビゲーション

ユーザーがフォームで操作しているコントロールにはフォーカスがあります。 ユーザーは、キーボードの Tab キーを使って、フォームのコントロール (現在は表示されていないコントロールを含む) 間を移動します。 フォーカスのあるコントロールが含まれるパン領域は、コントロール全体が表示されるように自動的にパンされる必要があります。 フォーカスがあるコントロールと、画面の端部またはタッチ キーボード (表示されている場合) の上部との間は 30 ピクセル以上ある状態にして、さまざまなエッジ ジェスチャー、UI、テキスト選択グリッパー用にスペースを残しておく必要があります。



場合によっては、画面にずっと表示されたままであることが必要な UI 要素もあります。 フォーム コントロールがパン領域に含まれ、重要な UI 要素が静的であるように UI を設計します。 たとえば、次のような場合です。



タッチ キーボードの起動と終了

タッチ操作をサポートするデバイスの場合は、ユーザーがテキスト入力フィールドをタップすると、タッチ キーボードが表示されます。 この操作が行われない場合は、アプリがタッチ キーボードを自動的に起動しないようにしてください。

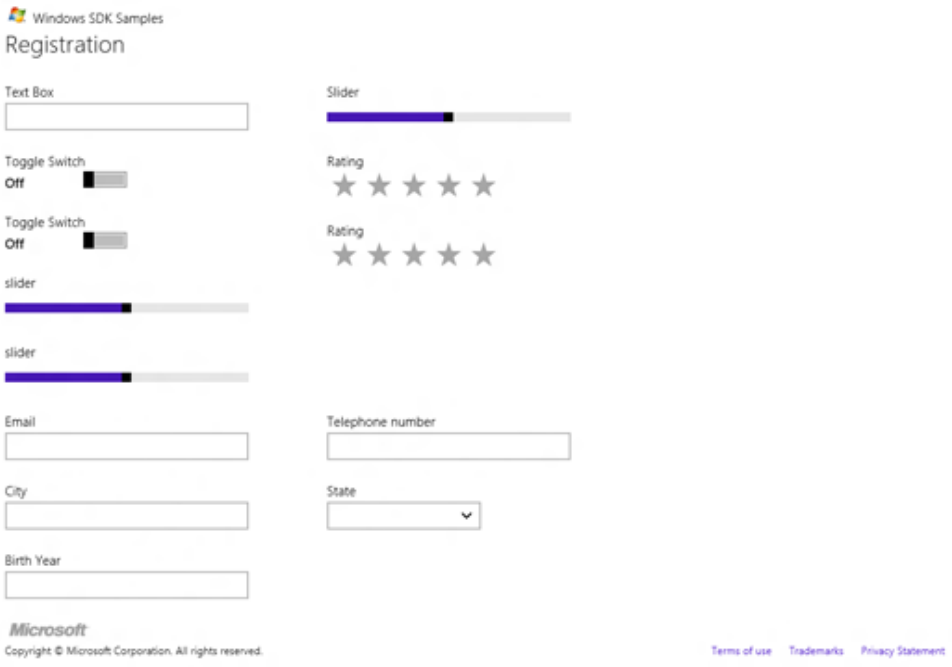
キーボードは次のいずれかの方法で終了します。

- フォームを送信したとこ。
- 次に示すように、[キーボードを隠す] コマンドが呼び出されます。

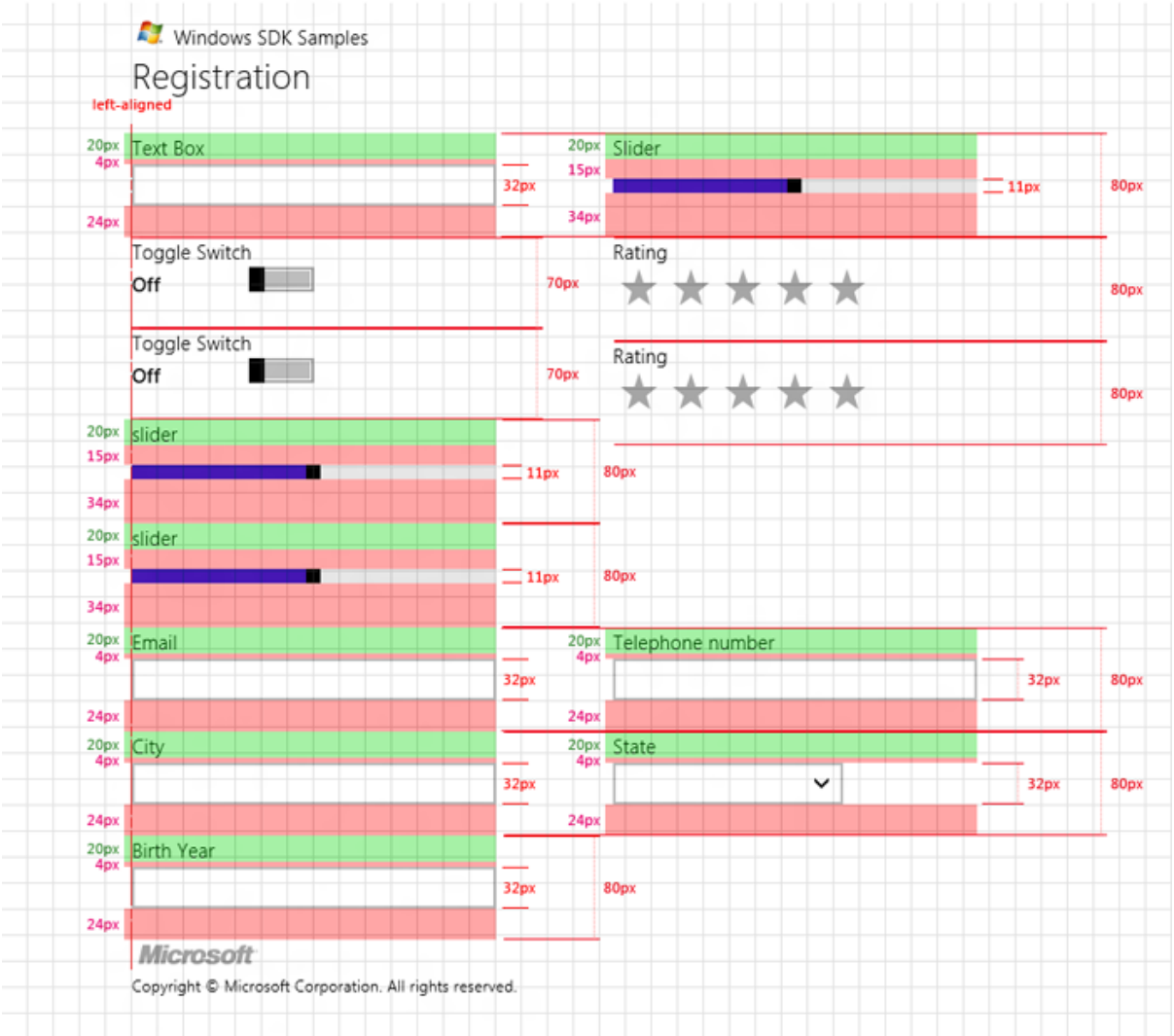
通常、ユーザーがフォームでコントロール間を移動している間は、タッチ キーボードは表示されたままです。 この動作は、フォーム内の他のコントロールの種類に基づいて異なります。

レイアウトの例

このドキュメントのガイドラインに従った登録フォームの例を次に示します。



次に、さまざまな要素の推奨サイズと、間隔のガイドラインを示します。



パスワード ボックスのガイドライン

パスワード ボックスは、プライバシーの目的で入力文字が非表示になるテキスト入力ボックスです。パスワード ボックスは、入力されたテキストの代わりに記号が表示される点を除けば、テキスト入力ボックスに似ています。この記号は、カスタマイズできます。

ユーザーがテキスト入力を開始したときや、変更が保存されたときも、記号は引き続き表示されます。ただし、引き続き表示される記号の数は、必ずしも入力文字の数には一致しません。

入力済みのパスワード ボックスをユーザーがタップすると、既にある記号が強調表示され、新たに入力した文字で元の値を置き換えることができるようになります。

重要な API

PasswordBox クラス (XAML)　　input type="password" 要素 (HTML)

適切なコントロールの使用

パスワード ボックスでは、パスワードに加えて、社会保障番号などその他の機密データも収集できます。2 ～ 3 語以上の単語を入力する場合は、ユーザーが入力内容を一時的に見ることができるように、パスワード表示ボタンを有効にすることを考えます。

例

パスワード ボックスは、これから説明するいくつかの状態を持っています。

パスワード ボックスは、ユーザーがパスワード入力であることを理解しやすいようにヒント テキストを表示することができます。

無効状態のパスワード ボックスでも、ヒントのテキストを表示できます。

ユーザーがパスワード ボックスに入力すると、既定の動作では、入力中のたテキストを隠す記号が表示されます。

右側にある "表示" ボタンを押すと、入力中のパスワード テキストを一時的に表示できます。

推奨事項

- アカウントの作成時は、新しいパスワードの入力用および新しいパスワードの確認用として、2 つのパスワード ボックスを提示することを検討します。
- ログオン時は 1 つのパスワード ボックスのみを表示します。
- PIN の入力にパスワード ボックスを使う場合は、確認ボタンを使う代わりに、最後の数値が入力されたらすぐに応答を返すことを検討します。

Segoe MDL2 アイコンのガイドライン

このドキュメントでは、**Segoe MDL2 Assets** フォントに付属しているグリフの一覧と、その使い方のガイダンスが含まれています。このフォントを入手するには Windows 10 をインストールする必要があります。

重要な API

Symbol 列挙 (XAML)　　AppBarIcon 列挙 (WinJS)

推奨事項

- これらのグリフは、**Segoe MDL2 Assets** フォントを明示的に指定できる場合にのみ使います。

その他の使い方のガイダンス

Windows 8/8.1 の **Segoe UI Symbol** アイコンのフォントは、Windows 10 のリリースで有効な **Segoe MDL2 Assets** フォントに置き換えられました。以前のフォントとほとんど同じ方法で使えますが、多くのグリフは、アイコンがタイポグラフィのベースライン上に合わせられるのではなくフォントの em スクエア内で調整されるようにフォントのメトリックが設定された、Windows 10 のアイコン スタイルに作り替えられました。

注　**Em** は、フォントの測定単位です。フォントの 1 Em は、72 ppi で指定したポイント値の 100% に相当します。たとえば、16 ポイントは 72 ppi で 16 ピクセルに相当します (100% のプラトーとも言います)。新しい MDL2 フォントはアイコン領域の面積が正方形の Em となるように設計されています。このため、コードで幅と高さを 16 ピクセルに指定すると、アイコンの面積は 16 x 16 ピクセルになります。これは、常にアイコンがこの面積いっぱいに表示されるということではありません。

Segoe UI Symbol は、"レガシ" リソースとして引き続き使うことができます。ただし、アプリケーションはすべて、新しい **Segoe MDL2 Assets** を使うように更新することをお勧めします。

Segoe MDL2 Assets フォントに含まれるアイコンや UI コントロールのほとんどは、Unicode の私用領域 (PUA) にマップされます。フォント開発者は PUA を使って、既にあるコード ポイントにマップされないグリフにプライベート Unicode 値を割り当てることができます。これは、記号フォントを作成するときに役立ちますが、相互運用性の問題が生じます。フォントが利用できない場合、グリフは表示されません。これらのグリフは、**Segoe MDL2 Assets** フォントを指定できる場合にのみ使います。

タイルを使っている場合は、タイルのフォントを指定できず、フォントのフォールバックで PUA グリフを使うことができないため、これらのグリフは使うことができません。

Segoe UI Symbol とは異なり、**Segoe MDL2 Assets** フォントのアイコンは、テキスト内で使用することを想定していません。これは、段階的表示の矢印のような一部の古い方法が利用できなくなったことを意味します。さらに、新しいアイコンはすべて同じ場所に同じ大きさで表示されるため、幅を 0 にして作成する必要はありません。一組で機能することが確認済みです。一組として設計された 2 つのアイコンは、ぴったり重ねることができ、正しい位置に収まることが理想的です。これにより、コード内の色付けが可能になります。たとえば、スタート タイルのバッチ ステータス用に、U+EA3A と U+EA3B が作成されました。これらは既に中央揃えされているため、ステータスが変わった場合に円を色で塗りつぶすことができます。

同様に、**Segoe UI Symbol** は、重ね合わせたり色付けする際に “幅が 0” のグリフに依存していたため、次の例のように、黒い枠 (U+E006) を幅が 0 の赤いハート (U+E00B) に重ねて描画することができました。

Segoe MDL2 Assets のグリフにはすべて、一貫した高さと、左を原点とした同一の固定幅が設定されているため、重ね合わせや色付けの効果はグリフどうしを直接重ねて描画することで表現できます。

また、アイコンの多くは、アラビア語、ペルシア語、ヘブライ語などの右から左に書く文字を使う言語でも利用できるように、左右が反転した形式も作成されています。

C#/VB/C++ と XAML を使ってアプリを開発している場合は、Unicode ID の代わりに Symbol 列挙値を使って **Segoe MDL2 Assets** フォントのグリフを指定できます。

JavaScript と HTML を使ってアプリを開発している場合は、Unicode ID の代わりに AppBarIcon 列挙値を使って **Segoe MDL2 Assets** フォントのグリフを指定できます。

さらに、**Segoe MDL2 Assets** フォントには、以下に示すアイコンもあります。ここで紹介するアイコンの多くは、特殊な目的のために使用されるもので、それ以外の場合は通常使用しません。

ハート

コード	Symbol	列挙値	説明
U+E006		HeartLegacy	輪郭のみのハート
U+E0A5		HeartFillLegacy	塗りつぶされたハート
U+E007		HeartBrokenLegacy	ひびの入ったハート
U+E00B		HeartFillZeroWidthLegacy	塗りつぶされたハート (幅が 0)
U+E00C		HeartBrokenZeroWidthLegacy	ひびの入ったハート (幅が 0)

評価用の星

コード	Symbol	列挙値	説明
U+E224		RatingStartLegacy	星 (輪郭のみ)
U+E0B4		RatingStartFillLegacy	塗りつぶされた星
U+E00A		RatingStartFillZeroWidthLegacy	塗りつぶされた星 (幅が 0)
U+E082		RatingStartFillReduced PaddingHTMLLegacy	塗りつぶされた星 (HTML で使用するためにパディングを小さくしたもの)
U+E0B5		RatingStartFillSmallLegacy	小さい星
U+E07C6		HarlfStartLeft	半分の星 (左側)
U+E07C7		HalfStarRight	半分の星 (右側)

チェック ボックス コンポーネント

コード	Symbol	列挙値	説明
U+E001		CheckMarkLegacy	チェック マーク

U+E002		CheckboxFillFillLegacy	塗りつぶされたチェック ボックス
U+E003		CheckboxLegacy	チェック ボックス
U+E004		CheckboxIndeterminateLegacy	不確定の状態
U+E005		CheckboxCompositeReversedLegacy	反転
U+E008		CheckMarkZeroWidthLegacy	チェック マーク (幅が 0)
U+E009		CheckboxFillZeroWidthLegacy	塗りつぶし (幅が 0)
U+E0A2		CheckboxCompositeLegacy	コンポジット
U+E739		Checkbox	チェック ボックス
U+E73A		CheckboxComposite	コンポジット チェック ボックス
U+E73B		CheckboxFill	塗りつぶされたチェック ボックス
U+E73C		CheckboxIndeterminate	不確定の状態
U+E73D		CheckboxCompositeReversed	反転したコンポジット
U+E73E		CheckMark	チェック マーク

その他

コード	Symbol	列挙値	説明
U+E134		CommentLegacy	コメント
U+E113		FavoriteLegacy	
U+E195		UnfavpriteLegacy	
U+E734		FavoriteStar	お気に入り (輪郭のみ)
U+E735		FavoriteStarFill	
U+E8D9		Unfavoraite	
U+E19F		LikeLegacy	
U+E19E		DislikeLegacy	
U+E19D		LikeDislikeLegacy	
U+E116		VideoLegacy	

U+E714		Video	
U+E20B		MailMessageLegacy	メール (レガシ)
U+E248		ReplyLegacy	返信する
U+E249		Favorite2Legacy	塗りつぶされたお気に入り
U+E24E		Unfavorite2Legacy	お気に入りから外す
U+E25A		MobileContactLegacy	携帯電話の連絡先
U+E25B		BlockedLegacy	ブロックされた連絡先
U+E25C		TypingIndicatorLegacy	入力インジケーター
U+E245D		PresenceChickletVideoLegacy	ビデオのプレゼンス情報のアイコン
U+E25E		PresenceChickletLegacy	プレゼンス情報のアイコン

スクロール バーの矢印

コード	Symbol	列挙値
U+E00E		ScrollChevronLetterLegacy
U+E00F		ScrollChevronRightLegacy
U+E010		ScrollChevronUpLegacy
U+E011		ScrollChevronDownLegacy
U+E016		ScrollChevronLeftBoldLegacy
U+E017		ScrollChevronRightBoldLegacy
U+E018		列挙値
U+E019		CommentLegacy

戻るボタン

従来の戻るボタンのグリフは 2 つの異なるサイズのものを使うことができるため、20 ポイントと 42 ポイントの両方で外側の輪の太さの一貫性を確保できます。2 つの新しいプロポーショナル クロムの戻るボタンも使用できます。これらのグリフは重ねることができます。

コード	Symbol	列挙値	説明

U+E0C4		BackBttnArrow20Legacy	戻るボタンの矢印、20pt
U+E0A6		BackBttnArrow42Legacy	戻るボタンの矢印、42pt
U+E0AD		BackBttnMirroredArrow20Legacy	左右と白黒が反転して矢印も反転した戻るボタン、20pt
U+E0AB		BackBttnMirroedArrow42Legacy	左右が反転した戻るボタンの矢印、42pt
U+E830		ChromeBack	Chrome の戻るボタン
U+EA47		ChromeBackMirroed	Chrome の左右が反転した戻るボタン

HTML の戻るボタン

これらのグリフの周囲に円を作成するには別のコードを追加します。

コード	Symbol	列挙値	説明
U+E0D5		ArrowHTMLLegacy	HTML 用の矢印
U+E0AE		ArrowHTMLMirroredLegacy	U+E0D5 の左右が反転したバージョン

AppBar のグリフ

アプリ バーでは、次の一覧のグリフを使います。慣例として、これらは列挙名で参照されます。これらは、丸で囲まれていない 20 x 20 ピクセルのアイコンとして設計されています。

コード	Symbol	列挙値	説明
U+E8FB		Accept	
U+E910		Accounts	
U+E710		Add	
U+E8FA		AddFriend	
U+E7EF		Admin	
U+E8E3		AlignCenter	
U+E8E4		AlignLeft	
U+E8E2		AlignRight	

U+E71D		AllApps
U+E723		Attach
U+E8A2		AttachCamera
U+E8D6		Audio
U+E72B		Back
U+E73F		BackToWindow
U+E8F8		BlockContact
U+E8DD		Bold
U+E8A4		Bookmarks
U+E7C5		BrowsePhotos
U+E8FD		BulletedList
U+E8EF		Calculator
U+E787		Calendar
U+E8BF		CalendarDay
U+E8F5		CalendarReply
U+E8C0		CalendarWeek
U+E722		Camera
U+E711		Cancel
U+E8BA		Caption
U+E7F0		CC
U+E8EA		Cellphone
U+E8C1		Characters
U+E894		Clear
U+E8E6		ClearSelection
U+E89F		ClosePane
U+E753		Cloud
U+E90A		Comment
U+E77B		Contact
U+E8D4		Contact2
U+E779		ContactInfo

U+E8CF		ContactPresence
U+E8C8		Copy
U+E7A8		Crop
U+E8C6		Cut
U+E74D		Delete
U+E8F0		Directions
U+E8D8		DisableUpdates
U+E8CD		DisconnectDrive
U+E8E0		Disklike
U+E90E		DockButton
U+E90C		DockLeft
U+E90D		DockRight
U+E8A5		Document
U+E896		Download
U+E70F		Edit
U+E899		Emoji
U+E76E		Emoji2
U+E728		FavoriteList
U+E734		FavoriteStar
U+E735		FavoriteStarFill
U+E71C		Filter
U+E11A		FindLegacy
U+E7C1		Flag
U+E8B7		Folder
U+E8D2		Font
U+E8D3		FontColor
U+E8E7		FontDecrease
U+E8E8		FontIncrease
U+E8E9		FontSize
U+E72A		Forward

U+E908		FourBars
U+E740		FullScreen
U+E774		Globe
U+E8AD		Go
U+E8FC		GoToStart
U+E8D1		GoToToday
U+E778		Hangup
U+E897		Help
U+E8C5		HideBCC
U+E7E6		Highlight
U+E80F		Home
U+E8B5		Import
U+E8B6		ImportAll
U+E8C9		Important
U+E8DB		Italic
U+E765		KeyboardClassic
U+E89B		LeaveChat
U+E8F1		Library
U+E8E1		Like
U+E8DF		LikeDislike
U+E71B		Link
U+EA37		List
U+E81D		Location
U+E715		Mail
U+E8A8		MailFill
U+E89C		MailForward
U+E8CA		MailReplay
U+E8C2		MailReplayAll
U+E912		Manage
U+E8CE		MapDrive
U+E707		MapPin

U+E77C		Memo
U+E88D		Message
U+E720		Microphone
U+E712		More
U+E8DE		MoveToFolder
U+E90B		MusicInfo
U+E74F		Mute
U+E8F4		NewFolder
U+E78B		NewWindow
U+E893		Next
U+E905		OneBar
U+E8E5		OpenFile
U+E8DA		OpenLoacl
U+E8A0		OpenPane
U+E7AC		OpenWith
U+E8B4		Orientation
U+E7EE		OtherUser
U+E1CE		OutlineStarLegacy
U+E7C3		Page
U+E77F		Paste
U+E769		Pause
U+E716		People
U+E8D7		Permissions
U+E717		Phone
U+E780		PhoneBook
U+E718		Pin
U+E768		Play
U+E8F3		PostUpdate
U+E8FF		Preview
U+E8A1		PreviewLink
U+E892		Previous

U+E8D0		Priority
U+E8A6		ProtectedDocument
U+E8C3		Read
U+E7A6		Redo
U+E72C		Refresh
U+E8AF		Remote
U+E738		Remove
U+E8AC		ReName
U+E90F		Repair
U+E8EE		RepeatAll
U+E8ED		RepeatOne
U+E730		ReportHacked
U+E8EB		ReShare
U+E7AD		Rotate
U+E89E		RotateCamera
U+E74E		Save
U+E78C		SaveLocal
U+E8FE		Scan
U+E8B3		SelectAll
U+E724		Send
U+E7B5		SetLockScreen
U+E97B		SetTile
U+E713		Settings
U+E72D		Share
U+E719		Shop
U+E8C4		ShowBCC
U+E8BC		ShowResults
U+E8B1		Suffle
U+E786		SlidesShow
U+E1CF		SolidStarLegacy
U+E8CB		Sort

U+E71A		Stop
U+E913		Street
U+E8AB		Switch
U+E8F9		SwitchApps
U+E895		Sync
U+E8F7		SyncFolder
U+E8EC		Tag
U+E907		ThreeBars
U+E7C9		TouchPointer
U+E78A		Trim
U+E906		TwoBars
U+E89A		TwoPage
U+E8DC		Underline
U+E7A7		Undo
U+E8D9		UnFavorite
U+E77A		UnPin
U+E8F6		UnSyncFolder
U+E74A		Up
U+E898		Upload
U+E8AA		VideoChat
U+E890		View
U+E8A9		ViewAll
U+E767		Volume
U+E8B8		Webcam
U+E909		World
U+E904		ZeroBars
U+E71E		Zoom
U+E8A3		ZoomIn
U+E71F		ZoomOut

バッテリーのグリフ

コード	Symbol	列挙値	コード	Symbol	列挙値
E996		BatteryUnknown	EC02		MobBatteryUnKnown
E850		Battery0	EBA0		MobBattery0
E851		Battery1	EBA1		MobBattery1
E852		Battery2	EBA2		MobBattery2
E853		Battery3	EBA3		MobBattery3
E854		Battery4	EBA4		MobBattery4
E855		Battery5	EBA5		MobBattery5
E856		Battery6	EBA6		MobBattery6
E857		Battery7	EBA7		MobBattery7
E858		Battery8	EBA8		MobBattery8
E859		Battery9	EBA9		MobBattery9
E83F		Battery10	EBA0		MobBattery10
E85A		BatteryCharging0	EBAB		MobBatteryCharging0
E85B		BatteryCharging1	EBAC		MobBatteryCharging1
E85C		BatteryCharging2	EBAD		MobBatteryCharging2
E85D		BatteryCharging3	EBAE		MobBatteryCharging3
E85E		BatteryCharging4	EBAF		MobBatteryCharging4
E85F		BatteryCharging5	EBB0		MobBatteryCharging5
E860		BatteryCharging6	EBB1		MobBatteryCharging6
E861		BatteryCharging7	EBB2		MobBatteryCharging7
E862		BatteryCharging8	EBB3		MobBatteryCharging8
E83E		BatteryCharging9	EBB4		MobBatteryCharging9
EA93		BatteryCharging10	EBB5		MobBatteryCharging10
E863		BatterySaver0	EBB6		MobBatterySaver0
E864		BatterySaver1	EBB7		MobBatterySaver1
E865		BatterySaver2	EBB8		MobBatterySaver2
E866		BatterySaver3	EBB9		MobBatterySaver3
E867		BatterySaver4	EBBA		MobBatterySaver4

E868		BatterySaver5	EBBB		MobBatterySaver5
E869		BatterySaver6	EBBC		MobBatterySaver6
E86A		BatterySaver7	EBBD		MobBatterySaver7
E86B		BatterySaver8	EBBE		MobBatterySaver8
EA94		BatterySaver9	EBBF		MobBatterySaver9
EA95		BatterySaver10	EBC0		MobBatterySaver10

スペル チェックのガイドライン

テキストの入力と編集を行っているときに、スペル チェックは単語を赤い波線で強調表示してユーザーに単語のスペルの間違いを知らせ、それを修正する方法を提供します。

重要な API

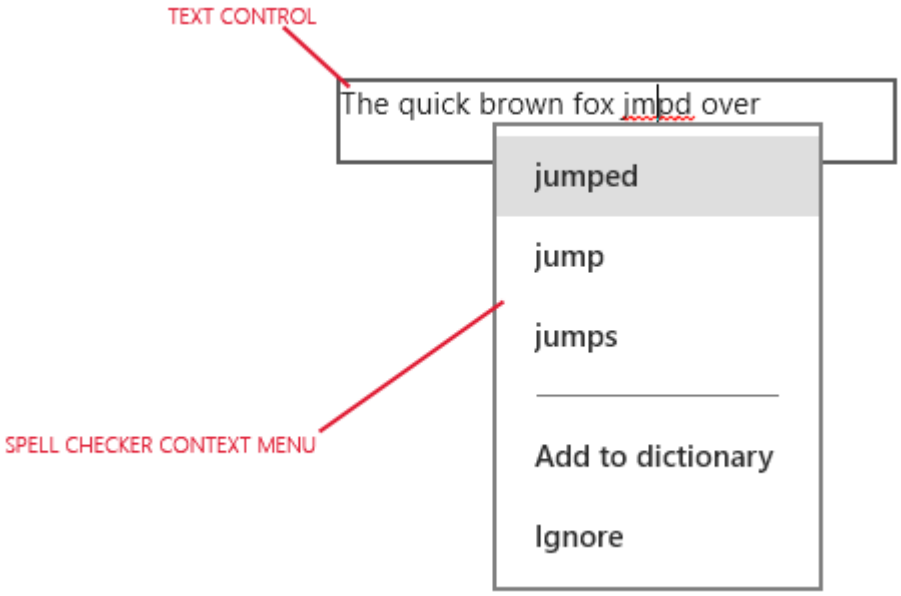
IsSpellCheckRnabled プロパティ (XAML)　　spellcheck プロパティ (HTML)

推奨事項

- スペル チェックは、テキスト入力コントロールに単語や文を入力するときにユーザーを補助するために使います。スペル チェックは、タッチ、マウス、キーボード入力で機能します。
- 単語が辞書になさそうな場合や、ユーザーがスペル チェックを重視しない場合は、スペル チェックを使わないでください。たとえば、パスワード、電話番号、名前などの入力ボックスでは、スペル チェックを有効にしません。これらのコントロールでは、スペル チェックが既定で無効になっています。
- 現在のスペル チェック エンジンがアプリの言語をサポートしていないという理由だけで、スペル チェックを無効にしないでください。スペル チェックでその言語がサポートされていない場合は、何も行われないので、有効にしたままで何も問題がありません。また、一部のユーザーは Input Method Editor (IME) を使ってアプリに他の言語を入力する場合があり、その言語はサポートされているかもしれません。 たとえば、日本語のアプリを構築している場合、現在はスペル チェック エンジンが日本語を認識していなくても、スペル チェックを無効にしないでください。ユーザーが英語 IME に切り替え、アプリに英語を入力する場合があります。スペル チェックが有効になっていれば、英語のスペル チェックが行われます。

その他の使い方のガイダンス

Windows アプリでは、**contentEditable** プロパティが **true** に設定されている要素のために組み込みのスペル チェックが用意されています。 組み込みスペル チェックの例を次に示します。



詳しくは、JavaScript については input type=text と textarea、Extensible Application Markup Language (XAML) については TextBox クラスをご覧ください。

テキスト入力コントロールでのスペル チェックは、次の 2 つの目的で使います。

- スペル ミスを自動修正する
  スペル チェック エンジンは、スペルを間違えている単語を、修正が確実であれば自動的に修正します。たとえば、エンジンは自動的に "teh" を "the" に変更します。
- 代わりのスペルを示す
  修正が確実でないとスペル チェック エンジンが判断した場合、スペル ミスのある単語には赤い下線が引かれ、ユーザーがその単語をタップするか右クリックすると、コンテキスト メニューに修正候補が表示されます。

JavaScript コントロールの場合、複数行テキスト入力コントロールでは、既定でスペル チェックが有効になっており、単一行コントロールでは無効になっています。単一行コントロールでスペル チェックを手動で有効にするには、コントロールの **spellcheck** プロパティを **true** に設定します。 コントロールのスペル チェックを無効にするには、**spellcheck** プロパティを **false** に設定します。

XAML TextBox コントロールの場合、スペル チェックが既定で無効になっています。**IsSpellCheckEnabled** プロパティを **true** に設定することによって有効にできます。

テキスト入力のガイドライン

テキスト入力ボックスでは、ユーザーは物理的なキーボードやスクリーン キーボードを使ってテキストや数値を入力および編集できます。テキストの折り返しを使うことで、テキスト入力ボックスに単一行または複数行のテキストを入力できるように構成できます。

重要な API

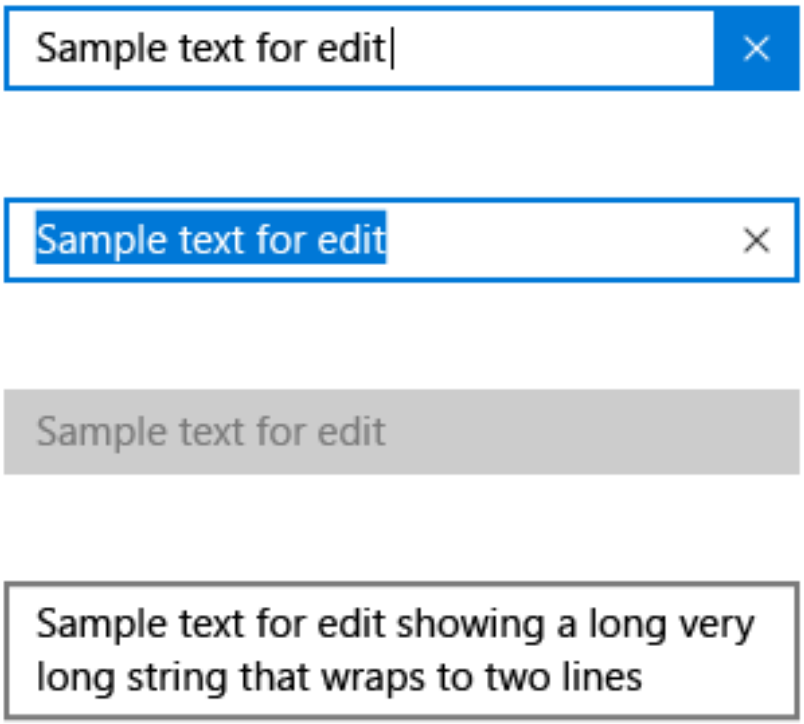
TextBox クラス (XAML)

input 要素 (HTML)

textArea 要素 (HTML)

例

ここでは、標準的なテキスト ボックスを 4 つの異なる状態 (単一行のテキストを入力中の状態、テキストが選択された状態、無効になった状態、複数行のテキストが入力された状態) で示しています。



適切なコントロールの選択

標準テキスト ボックスかそれ以外か、テキスト入力に最適なコントロールを決定する際には、次の点を考慮します。

- **有効なすべての値を効率的に列挙することが現実的か?** そうである場合は、代わりにいずれかの選択コントロールを使うことを検討してください。考えられる選択コントロールは、チェック ボックス、ドロップダウン リスト、リスト ボックス、ラジオ ボタン、スライダー、トグル スイッチ、DatePicker、または TimePicker です。
- **有効な値は比較的少数か?** そのは、ドロップダウン リストまたはリスト ボックス (値の文字数が多い場合)をお勧めします。
- **有効なデータに、何も制約がないか? または、形式の制約 (長さや文字の種類による制約) だけがあるか?** そうである場合は、テキスト入力コントロールを使います。このコントロールでは、入力できる文字数を制限できるほか、入力値の種類の一覧から選んで、入力できる値を特定の文字セットや文字形式 (数値、URL、通貨など) のみに制限できます。
- **値は専用のコモン コントロールがあるデータ型か?** そうである場合は、テキスト入力コントロールではなく、適切なコントロールを使います。たとえば、データ入力を受け付けるには、テキスト入力コントロールの代わりに DatePicker を使います。
- 数値データのみに制限されている場合:
  - **入力される値は、近似値や同じページの別の数量に対する相対値か?** そうである場合は、スライダーを使います。
  - **設定の変更による影響をすぐに確認できることがユーザーにとって便利か?** そうである場合は、スライダーを使い、必要であれば付随するコントロールも使います。
  - **入力した結果の確認後 (たとえば、音量や明るさを設定した後など)、入力された値を調整する可能性が高いか?** そうである場合は、スライダーを使います。

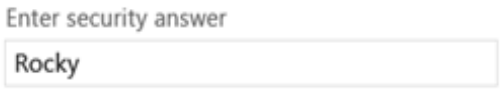
推奨事項

- テキスト ボックスの目的がわかりにくい場合は、ラベルまたはプレース ホルダー テキストを使用します。ラベルは、テキスト入力ボックスに値が存在するかどうかに関係なく表示します。プレースホルダー テキストはテキスト入力ボックスの内側に表示され、値を入力すると非表示になります。
- テキスト ボックスは、入力できる値の範囲に適した幅になるようにします。単語の長さは言語によって異なるため、アプリを世界対応にする場合は、ローカライズを考慮に入れて幅を調整します。
- テキスト入力ボックスは、通常、単一行です (テキストの折り返しがオフ)。ユーザーが長い文字列を入力または編集する必要がある場合は、テキスト入力ボックスを複数行 (テキストの折り返しがオン) に設定します。
- 通常、テキスト入力ボックスは編集可能なテキストに対して使います。ただし、読み取り、選択、コピーはできるが編集はできないように、テキスト入力ボックスを読み取り専用に設定することもできます。
- 画面をすっきりと見せる必要がある場合は、制御するチェック ボックスがオンの場合のみ、一連のテキスト入力ボックスを表示することをお勧めします。また、有効な状態のテキスト入力ボックスをチェック ボックスなどのコントロールにバインドすることもできます。
- 既に値が入力されているテキスト入力ボックスをユーザーがタップしたときの動作を検討します。既定の動作では、単語の間に挿入ポイントが配置され、何も選択されません。この動作は値の置き換えよりも編集に適しています。対象のテキスト入力ボックスが、編集よりも主に置き換えに使われる場合は、フォーカスを受け取ったときにフィールドのすべてのテキストを選択し、入力によって選択内容が置き換えられるように設定できます。
- 入力できる値を特定の文字セットまたは形式 (数値、URI、通貨など) に制限するには、入力値の種類を適切な値に設定します。これにより、使われるスクリーン キーボードが決まります。

単一行入力ボックス

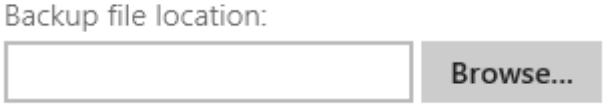
- 少量のテキスト情報を多数取得する場合は、いくつかの単一行テキスト ボックスを使います。 それらのテキスト ボックスに本来の関連性がある場合は、グループ化します。
- 単一行テキスト ボックスのサイズを、予測される最長の入力より少し広くします。それによってコントロールが広くなり過ぎる場合は、2 つのコントロールに分割します。たとえば、1 つの住所の入力を "Address line 1" と "Address line 2" に分割できます。
- 入力できる最大文字数を設定します。バッキング データ ソースが長い入力文字列を許可しない場合は、入力を制限し、制限に達したら検証ポップアップを使ってユーザーに知らせます。
- ユーザーから少量のテキストを収集するには、単一行テキスト入力コントロールを使います。

  次の例は、セキュリティの質問への回答を取得する単一行テキスト ボックスを示しています。 回答には短い文字列が想定されるので、単一行テキスト ボックスが適しています。 収集される情報は、Windows で認識される専用の入力型のいずれとも一致しないので、汎用の "Text" 型が適しています。

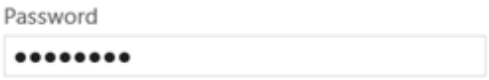


- 特定の書式のデータを入力するには、短い固定サイズの単一行テキスト入力コントロールのセットを使います。

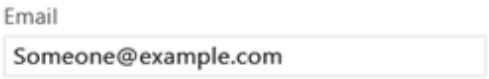
- 文字列を入力または編集するには、単一行の、制約のないテキスト入力コントロールと、ユーザーが有効な値を選択できるように補助するコマンド ボタンを組み合わせて使います。



- 数値を入力または編集する場合は、単一行の数値入力コントロールを使います。
- パスワードと PIN を安全に入力するには、単一行のパスワード入力コントロールを使います。



- メール アドレスを入力する場合は、単一行のメール入力コントロールを使います。



メール入力コントロールを使う場合は、次の機能を無料で利用できます。

◦ ユーザーがテキスト ボックスに移動すると、タッチ キーボードがメール専用のキー レイアウトで表示されます。

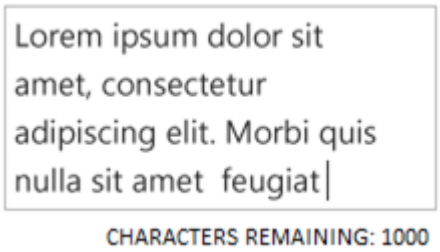
◦ ユーザーが無効なメール形式を入力すると、それを知らせるダイアログが表示されます。

- Web アドレスを入力するには、URL 入力コントロールを使います。
- 電話番号を入力するには、電話番号入力コントロールを使います。
- 単一行テキスト ボックスを作成するために、行の高さを 1 にしたテキスト領域は使いません。代わりに、テキスト ボックスを使います。
- プレースホルダー テキストは、テキスト コントロールを事前設定するためには使いません。ユーザーがコントロールを使うと、プレースホルダー テキストはクリアされます。代わりに、"value" 属性を使います。
- パスワード入力ボックスのすぐ横に、他のコントロールを配置しないようにします。パスワード入力ボックスには、入力したパスワードをユーザーが確認するための、パスワード表示ボタンがあります。他のコントロールをすぐ横に配置すると、ユーザーが他のコントロールを操作しようとしたときに、誤ってパスワードを表示してしまう可能性があります。 これを防ぐには、パスワード入力ボックスと他のコントロールの間には少し間隔をおくか、他のコントロールを次の行に配置します。

複数行テキスト入力コントロール

- リッチ テキスト ボックスを作る場合は、スタイル設定ボタンを用意し、その動作を実装します。
- アプリの雰囲気に合ったフォントを使います。

- テキスト コントロールの高さは、典型的な入力が十分に入るように設定します。
- 最大文字数または単語数を設定して長いテキストを取得する場合、プレーン テキスト ボックスを使い、ライブ実行のカウンターを用意して、制限までの残りの文字数または単語数をユーザーに示します。カウンターは自分で作成する必要があります。テキスト ボックスの下に配置し、ユーザーが文字や単語を入力するたびに動的に更新します。



- ユーザーの入力中にテキスト入力コントロールの高さが増加するようにはしません。
- ユーザーが 1 行しか必要としていない場合は、複数行テキスト ボックスを使いません。
- プレーン テキストで十分な場合は、リッチ テキスト コントロールは使いません。

その他の使い方のガイダンス

テキスト入力ボックスを使うと、ユーザーがテキスト値を入力し、既に入力した値を編集することができます。また必要に応じて、入力できる値を制限できます。入力できる文字数を制限できるほか、入力値の種類の一覧から選んで、入力できる値を特定の文字セットや形式 (数値、URI、通貨など) のみに制限できます。

適切な複数行テキスト入力コントロールの選択

ユーザーが長い文字列を入力または編集する必要がある場合は、複数行テキスト コントロールを使います。複数行テキスト入力コントロールには、プレーン テキスト入力コントロールとリッチ テキスト コントロールの 2 つの種類があります。

- 複数行テキスト ボックスの主な目的がドキュメントの作成 (ブログのエントリ、メール メッセージのコンテンツなど) であり、ドキュメントでリッチ テキストが必要な場合は、リッチ テキスト ボックスを使います。
- ユーザーがテキストを書式設定できるようにする場合は、リッチ テキスト ボックスを使います。
- 内部的に使うだけで、後でユーザーに再表示しないテキストを取得する場合は、プレーン テキスト入力コントロールを使います。
- 他のすべてのシナリオでは、プレーン テキスト入力コントロールを使います。

タイルと通知のガイドライン

このセクションのガイドラインを使って、タイルや通知を使った、ユーザーに合わせた魅力的なエクスペリエンスの作成方法を学習します。

このセクションの内容

トピック	説明
タイルとアイコン アセット	Windows 10 オペレーティング システム全体でさまざまな形式で表示される、アプリ アイコン アセットは、ユニバーサル Windows プラットフォーム (UWP) アプリの代名詞です。このガイドラインでは、システム内でアプリ アイコン アセットが表示される場所の詳細について説明し、最も洗練されたアイコンを作成する方法に関して詳細なデザインのヒントを提供します。
ロック スクリーン	ユーザーは、Windows デバイスがロックされているときのロック スクリーンのプロバイダーとしてアプリを使うようにロック スクリーンをカスタマイズできます。また、ロック スクリーンに表示される基本的な通知 (メールやテキストなど) を、アプリで提供される通知を反映するように変更することもできます。このトピックでは、ロック スクリーンの背景と通知の実装に関するベスト プラクティスについて説明します。
定期的な通知	このトピックでは、ユニバーサル Windows プラットフォーム (UWP) アプリで定期的な (ポーリングされた) 通知を使う際のガイドラインを示します。
プッシュ通知	このトピックでは、Windows アプリでプッシュ通知を使う際の一般的なガイドラインとコーディングのガイドラインを示します。
直接通知	このガイドラインでは、有効な直接プッシュ通知を作成する方法について説明します。
スケジュールされた通知	Windows アプリにスケジュールされたタイル通知やトースト通知を追加するときには、次のガイドラインに従ってください。

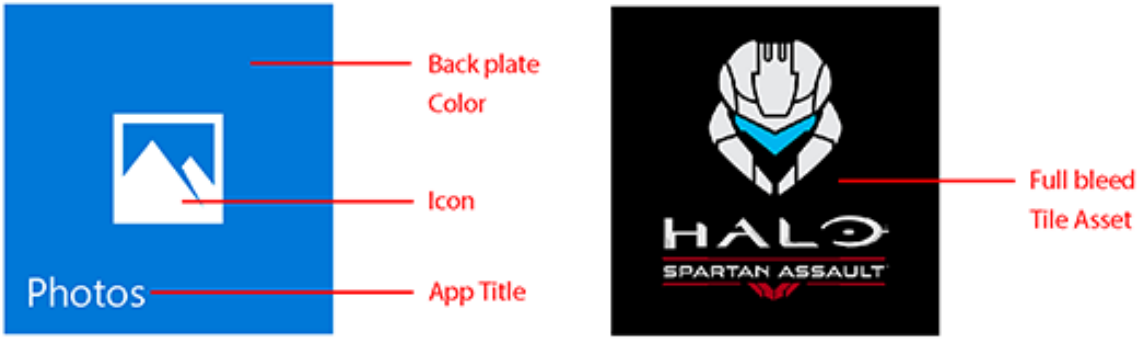
タイルとバッジ	ここでは、スタート画面とロック画面の両方における、アプリのタイルの作成と更新に関するベスト プラクティスとグローバリゼーション/ローカライズの推奨事項について説明します。また、アプリが Windows ストアで承認されるために満たす必要のあるタイルに関連する特別な要件も示します。
トースト通知	このトピックでは、トースト通知の用途を説明し、トースト通知の作成方法と送信方法について推奨事項を示します。

タイルとアイコン アセットのガイドライン

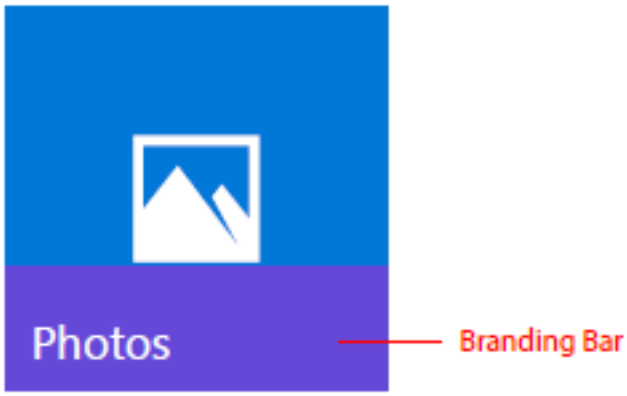
Windows 10 オペレーティング システム全体でさまざまな形式で表示される、アプリ アイコン アセットは、ユニバーサル Windows プラットフォーム (UWP) アプリの代名詞です。このガイドラインでは、システム内でアプリ アイコン アセットが表示される場所の詳細について説明し、最も洗練されたアイコンを作成する方法に関して詳細なデザインのヒントを提供します。

タイル要素

要素は、スタート画面のタイルの基本コンポーネントであり、バック プレート、アイコン、アプリのタイトルで構成されます。



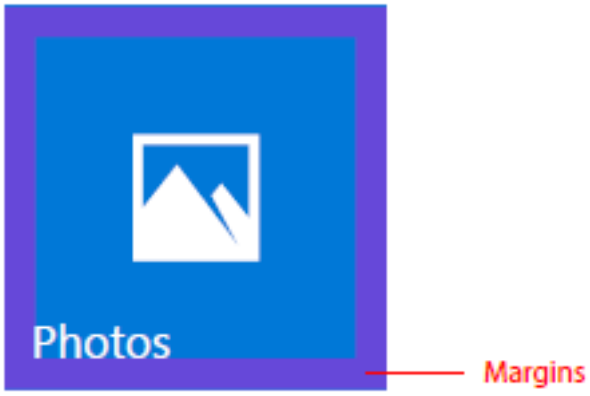
タイルの下部にあるブランド バーは、アプリの名前、バッジ、カウンター (使用する場合) が表示される場所です。



ブランドのバーの高さは、表示されているデバイスのスケール ファクターに基づいています。

スケール ファクター	有効なピクセル (epx)
100%	32
125%	40
150%	48
200%	64
400%	128

タイルには、固定された余白があり、ほとんどのコンテンツは余白の内側に表示されます。



余白の幅は、表示されているデバイスのスケール ファクターに基づいています。

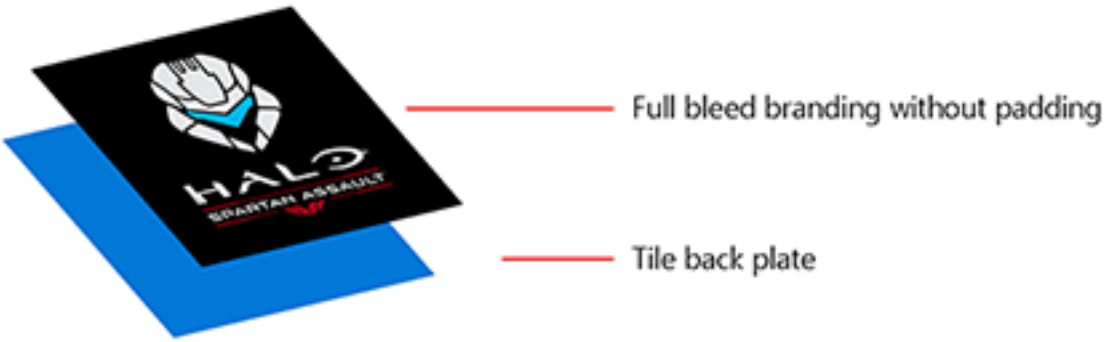
スケール ファクター	有効なピクセル (epx)
100%	8
125%	10
150%	12
200%	16
400%	32

タイル アセット

各タイル アセットは、配置されるタイルと同じサイズです。アセットの 2 つの異なる表示によって、アプリのタイルをブランド化できます。

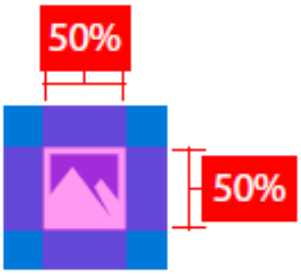
1. パディングによって中央に配置されたアイコンまたはロゴ。この場合、バック プレートの色が透けて見えます。

2. パディングのない、フルブリードのブランド化されたタイル。

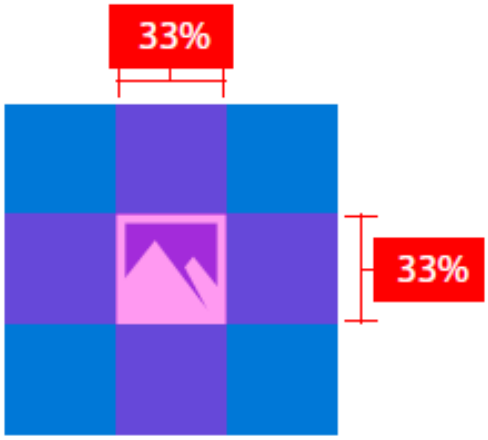


デバイス間で一貫性を確保するために、各タイル サイズ (小、普通、ワイド、大) には独自のサイズの関係があります。タイルの間で一貫したアイコンの配置を実現するために、以下のタイル サイズについて、パディングの基本的なガイドラインに従うことをお勧めします。紫色の 2 つのオーバーレイが交差する領域が、アイコンの最適な面積を表しています。アイコンがこの面積の内側に収まらない場合もありますが、アイコンの表示領域は用意されている例とほぼ同じである必要があります。

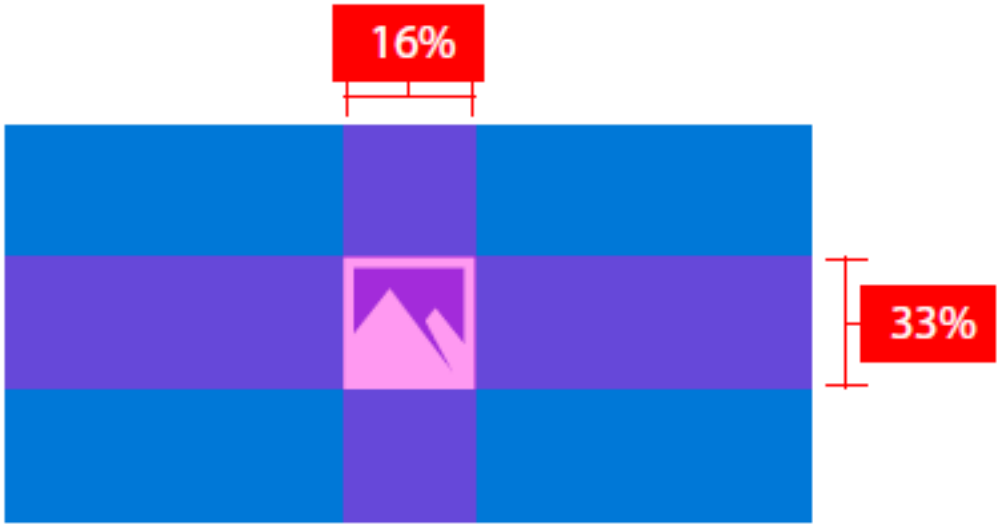
小さいタイルのサイズ調整:



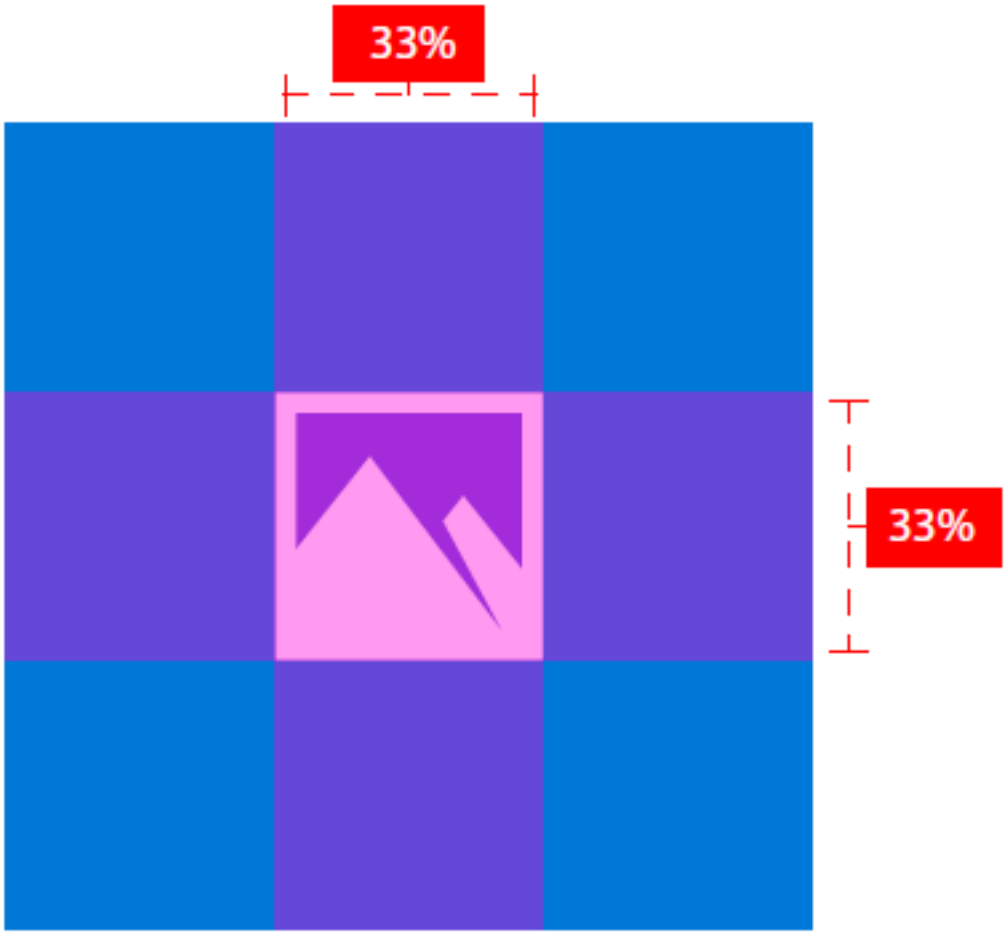
普通サイズのタイルのサイズ調整:



ワイド タイルのサイズ調整:



大きいタイルのサイズ調整:



この例では、アイコンがタイルに対して大きすぎます。

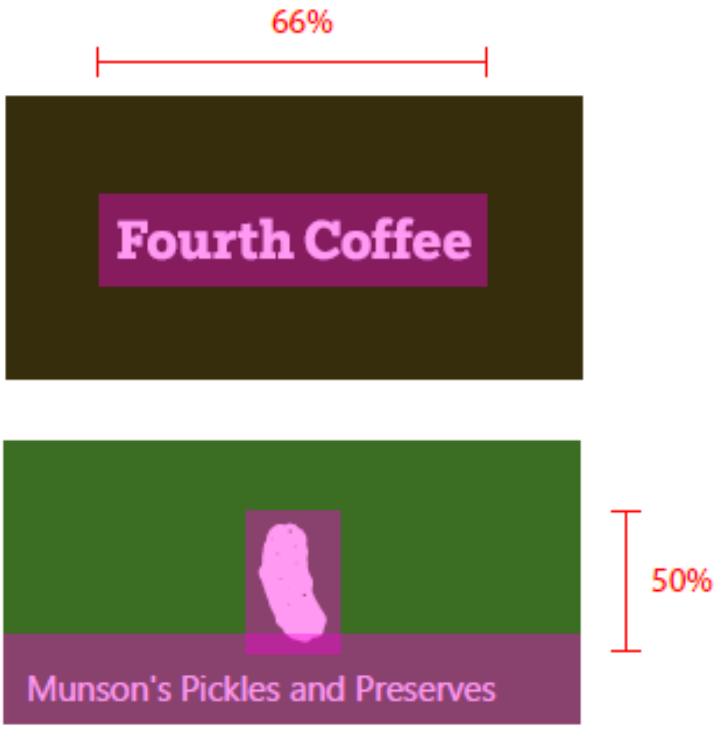
この例では、アイコンがタイルに対して小さすぎます。

水平方向または垂直方向のアイコンに最適なパディングの比率は次のとおりです。

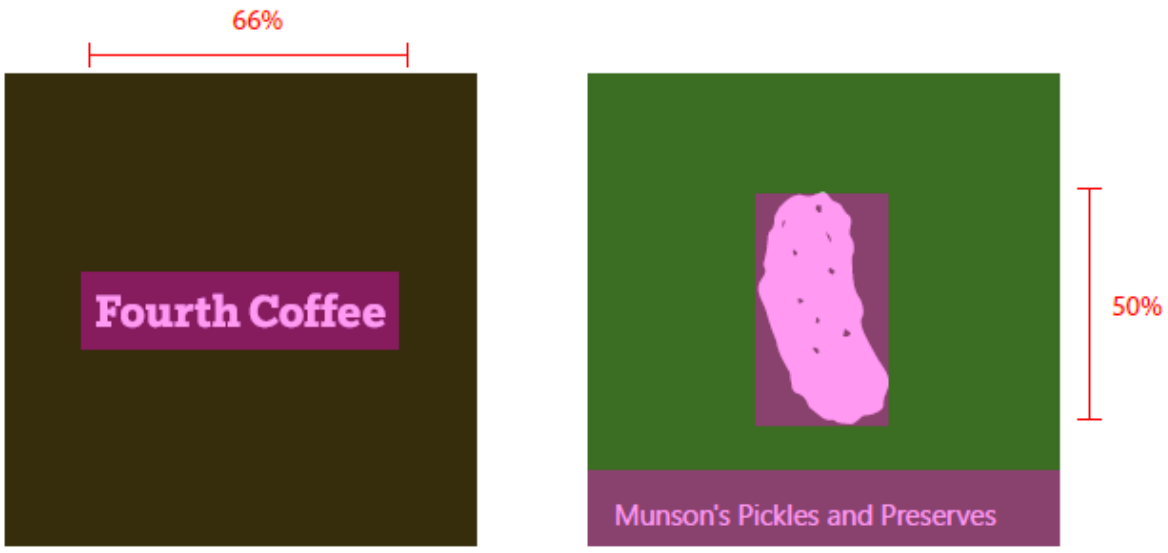
小さいタイルでは、アイコンの幅と高さをタイル サイズの 66% に制限します。

普通サイズのタイルでは、アイコンの幅をタイル サイズの 66% に、高さをタイル サイズの 50% に制限します。これによって、ブランド バー内の要素と重ならないようにします。

ワイド タイルでは、アイコンの幅をタイル サイズの 66% に、高さをタイル サイズの 50% に制限します。これによって、ブランド バー内の要素と重ならないようにします。



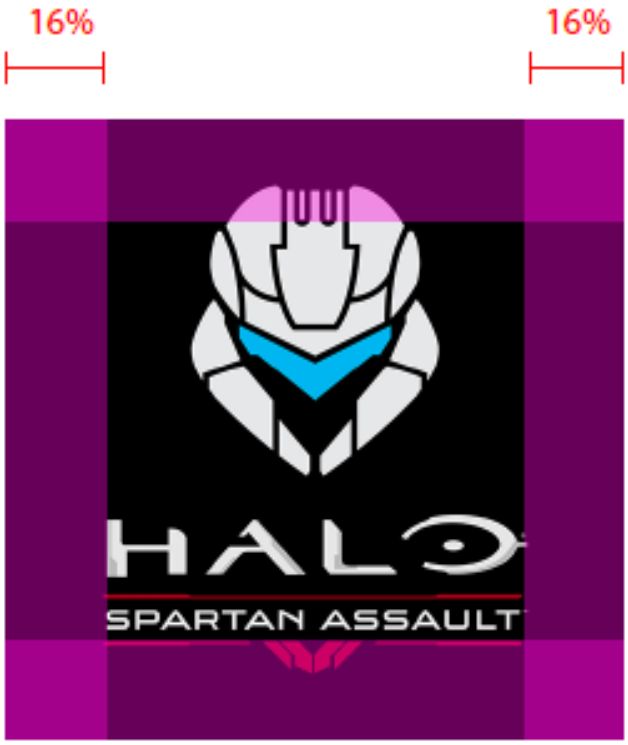
大きいタイルでは、アイコンの幅と高さをタイル サイズの 50% に制限します。



水平方向または垂直方向にデザインされたアイコンがある一方で、ターゲット サイズの正方形に収まらない、より複雑な形状のアイコンもあります。中央に配置されるアイコンの一方の辺に重みを付けることができます。この場合、アイコンの視覚的な重みが正方形に収ま

るアイコンと同じであれば、アイコンの一部が推奨される面積の外側にはみ出していてもかまいません。

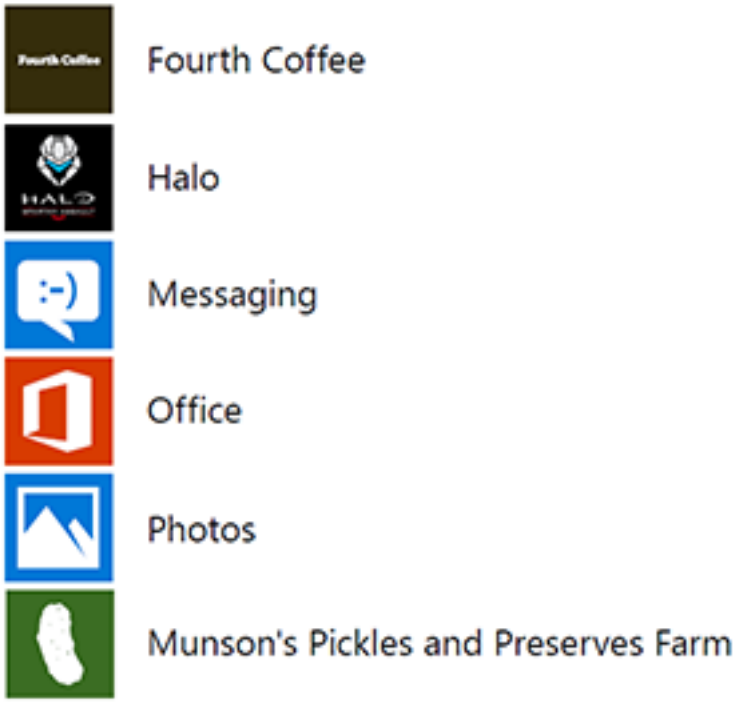
フルブリード アセットでは、要素が余白およびタイルの端の内側に接するように考慮します。タイルの高さまたは幅の 16% 以上の余白を維持します。この割合は、最小タイル サイズでの余白の幅の 2 倍を表しています。



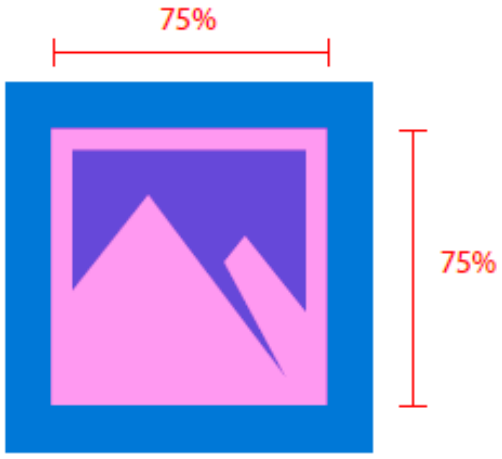
次の例では、余白が狭すぎます。

リスト ビューでのタイル アセット

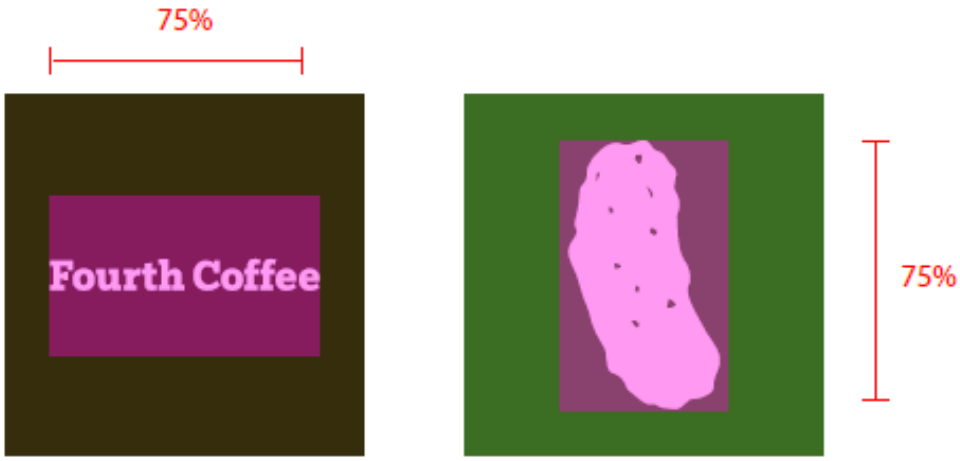
タイルはリスト ビューにも表示されます。リスト ビューに表示されるタイル アセットのサイズ調整に関するガイドラインは、先ほど説明したタイル アセットとは少し異なります。このセクションでは、これらのサイズ調整の詳細について説明します。



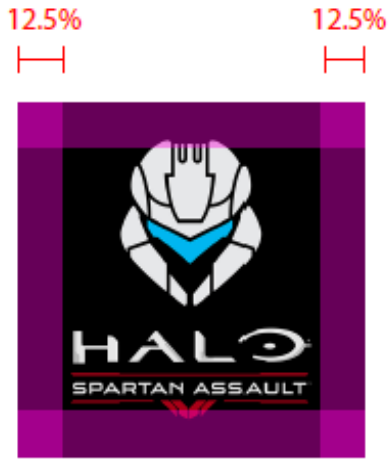
アイコンの幅と高さをタイル サイズの 75% に制限します。



垂直方向および水平方向のアイコンでは、幅と高さをタイル サイズの 75% に制限します。



重要なブランド要素のフルブリード アートワークの場合、12.5% 以上の余白を維持します。

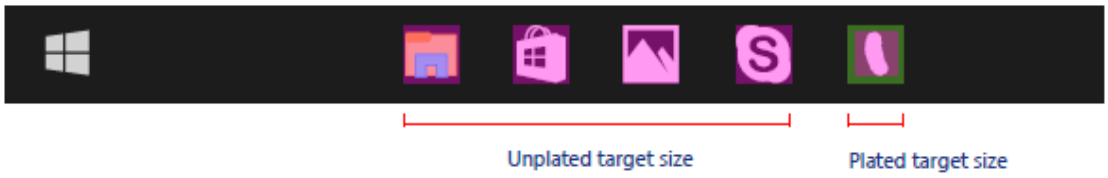


次の例では、アイコンがタイル内で大きすぎます。

次の例では、アイコンがタイル内で小さすぎます。

ターゲット ベースのアセット

ターゲット ベースのアセットは、Windows タスク バー、タスク ビュー、[Alt] + [Tab] キーを押したとき、スナップ アシスト、およびスタート画面のタイルの右下に表示されます。これらのアセットにパディングを追加する必要はありません。必要な場合は、Windows によってパディングが追加されます。これらのアセットは、16 px の最小面積を占めている必要があります。Windows タスク バーに表示されたこのようなアセットの例を以下に示します。



これらの UI では、既定で色付きバック プレート上でターゲット ベースのアセットが使用されますが、ターゲット ベースのプレートなしのアセットを使用することもできます。プレートなしのアセットは、さまざまな背景色で表示される可能性があることを考慮して作成する必要があります。

Unplated white　　Unplated color　　Plated version

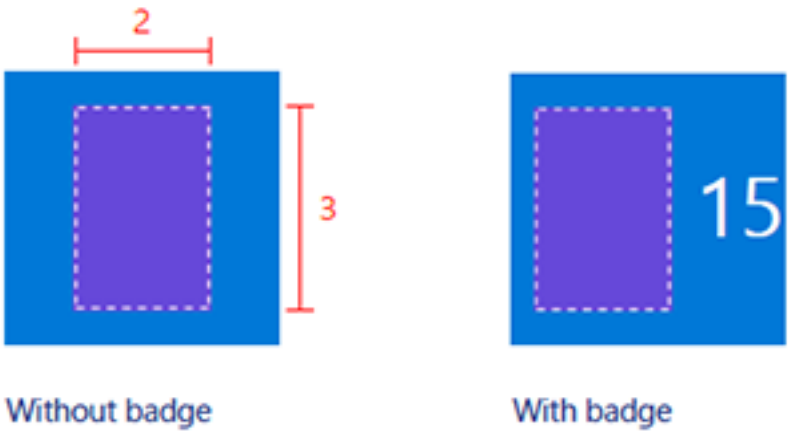
100% スケールでのターゲット ベースのアセットのサイズに関する推奨事項を次に示します。

Unplated 24px icon

Default Plated 16px icon
on a 24px plate

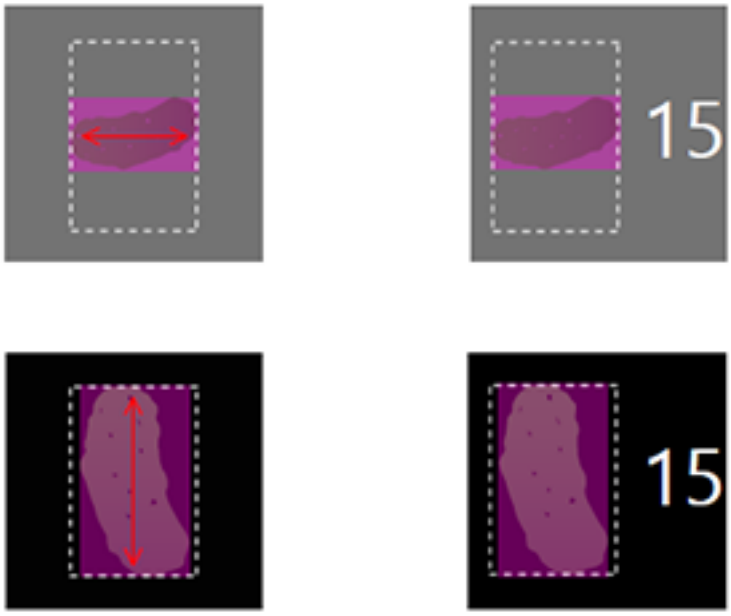
アイコン テンプレート アプリのアセット

メッセージング、電話、ストアなど、アイコン テンプレートを使ったアプリでは、ターゲット ベースのアセットでバッジ (ライブ カウンターを含む) を表示できます。他のターゲット ベースのアセットと同様にパディングは必要ありません。アイコン アセットは、アプリ マニフェストの一部ではなく、ライブ タイルのペイロードの一部です。アセットは 3:2 の比率のコンテナーに収まるように拡大縮小され、中央に配置されます。



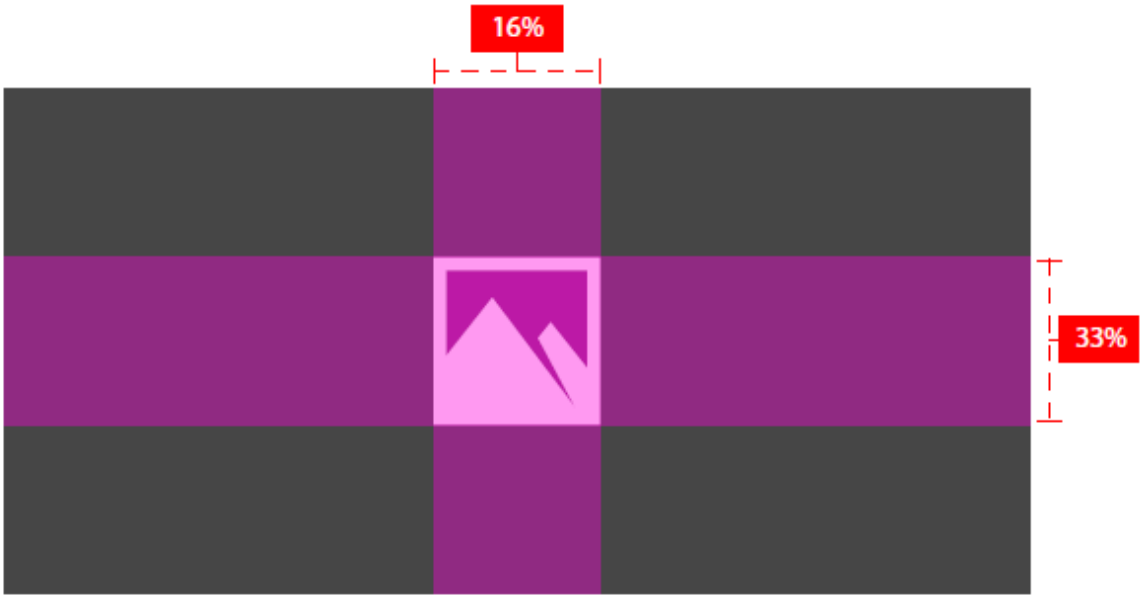
正方形のアセットの場合、コンテナー内で自動的に中央に配置されます。

正方形以外のアセットの場合、自動的に水平方向/垂直方向の中央に配置され、コンテナーの幅/高さに合わせてスナップされます。



スプラッシュ スクリーンのアセット

スプラッシュ スクリーンのアセットは、表示されるすべてのデバイスで中央に配置されます。



ハイ コントラストのアセット

ハイ コントラスト モードでは、別のハイ コントラスト白 (白の背景に黒のテキスト) とハイ コントラスト黒 (黒の背景に白のテキスト) のアセットを使用します。アプリでハイ コントラストのアセットが提供されない場合、標準アセットが使用されます。

アプリの標準アセットをモノクロの背景にレンダリングしたときに許容できる表示エクスペリエンスが提供される場合、アプリはハイ コントラスト モードでも最低限満足できる表示になります。標準アセットをモノクロの背景にレンダリングしたときに、許容できる表示エクスペリエンスが得られない場合は、明確にハイ コントラストのアセットを含めることを検討します。以下の例は、2 種類のハイ コントラストのアセットを示しています。

ハイ コントラストのアセットを用意する場合は、黒地に白と白地に黒の両方のセットを含める必要があります。これらのアセットをパッケージに含める場合は、黒地に白のアセット用に "contrast-black" フォルダーを、白地に黒のアセット用に "contrast-white" フォルダーを作成できます。

アセット サイズ一覧

少なくとも、100、200、400 の倍率のアセットを提供することを強くお勧めします。すべてのスケールのアセットを提供することによって、最適なユーザー エクスペリエンスを提供できます。

スケール ベースのアセット

カテゴリ	要素名	100%	125%	150%	200%	400%
小	Square71x71Logo	71x71	89x89	107x107	142x142	284x284
普通	Square150x150Logo	150x150	188x188	225x225	300x300	600x600
ワイド	Square310x150Logo	310x150	388x188	465x225	620x300	1240x600
大 (デスクトップのみ)	Square310x310Logo	310x310	388x388	465x465	620x620	1240x1240
アプリ一覧 (アイコン)	Square44x44Logo	44x44	55x55	66x66	88x88	176x176

スケール ベースのアセットのファイル名の例

カテゴリ	要素名	100%	125%	150%	200%	400%
小	Square 71x71Logo	AppName SmallTile. scale-100. png	AppName SmallTile. scale-125. png	AppName SmallTile. scale-150. png	AppName SmallTile. scale-200. png	AppName SmallTile. scale-400. png
普通	Square 150x150 Logo	AppName MedTile. scale-100. png	AppName MedTile. scale-125. png	AppName MedTile. scale-150. png	AppName MedTile. scale-200.png	AppName MedTile.sc ale-400.png
ワイド	Square 310x150 Logo	AppName WideTile. scale-100. png	AppName WideTile. scale-125. png	AppName WideTile. scale-150. png	AppName WideTile.s cale-200.png	AppName WideTile.s cale-400.png

大 (デスクトップのみ)	Square 310x310 Logo	AppName LargeTile.scale-100.png	AppName LargeTile.scale-125.png	AppName LargeTile.scale-150.png	AppName LargeTile.scale-200.png	AppName LargeTile.scale-400.png
アプリ一覧 (アイコン)	Square 44x44Logo	AppName LogoSmallTile.scale-100.png	AppName LogoSmallTile.scale-125.png	AppName LogoSmallTile.scale-150.png	AppName LogoSmallTile.scale-200.png	AppName LogoSmallTile.scale-400.png

ターゲット ベースのアセット

ターゲット ベースのアセットは、複数のスケール ファクターで使用されます。ターゲット ベースのアセットの要素名は **Square44x44Logo** です。最低でも以下のアセットを提出することを強くお勧めします。

16 x 16、24 x 24、32 x 32、48 x 48、256 x 256

次の表は、すべてのターゲット ベースのアセットのサイズと対応するファイル名の例を示します。

アセット サイズ	ファイル名の例
16x16*	AppNameAppList.targetsize-16.png
24x24*	AppNameAppList.targetsize-24.png
32x32*	AppNameAppList.targetsize-32.png
48x48*	AppNameAppList.targetsize-48.png
256x256:	AppNameAppList.targetsize-256.png
20x20	AppNameAppList.targetsize-20.png
30x30	AppNameAppList.targetsize-30.png
36x36	AppNameAppList.targetsize-36.png
40x40	AppNameAppList.targetsize-40.png
60x60	AppNameAppList.targetsize-60.png
64x64	AppNameAppList.targetsize-64.png
72x72	AppNameAppList.targetsize-72.png
80x80	AppNameAppList.targetsize-80.png
96x96	AppNameAppList.targetsize-96.png

* ベースラインとしてこれらのサイズのアセットを提出します

アセットの種類

ここでは、すべてのアセットの種類、その用途、推奨されるファイル名の一覧を示します。

タイル アセット

- 中央に配置されるアセットは、通常、スタート画面にアプリを表示するために使用されます。
- ファイル名の形式: *Tile.scale-*.PNG
- 影響を受けるアプリ: すべての UWP アプリ
- 用途：
  - 既定のスタート画面のタイル (デスクトップとモバイル)
  - アクション センター (デスクトップとモバイル)
  - タスク スイッチャー (モバイル)
  - 共有ピッカー (モバイル)
  - ピッカー (モバイル)
  - ストア

プレート付きのスケーラブルな一覧のアセット

- これらのアセットは拡大縮小が必要なサーフェスで使われます。アセットのバック プレートはシステムによって提供されるか、アプリに含まれている場合は独自の背景色のプレートが使用されます。
- ファイル名の形式: *AppList.scale-*.PNG
- 影響を受けるアプリ：すべての UWP アプリ
- 用途：
  - スタート 画面のすべてのアプリ の一覧(デスクトップとモバイル)
  - スタート 画面のよく使うアプリの一覧 (デスクトップ)
  - タスク マネージャ (デスクトップ)
  - Cortana の検索結果
  - 設定

プレート付きのターゲット サイズの一覧のアセット

- これらはプレート付きで、拡大縮小されない固定サイズのアセットです。ほとんどの場合、従来のエクスペリエンスに使用されます。アセットはシステムによって確認されます。
- ファイル名の形式: * AppList.targetsize-*.PNG
- 影響を受けるアプリ：すべての UWP アプリ
- 用途：
  - スタート画面のジャンプ リスト (デスクトップ)
  - スタート画面のタイルの下隅 (デスクトップ)
  - ショートカット (デスクトップ)
  - コントロール パネル (デスクトップ)

プレートなしのターゲット サイズの一覧アセット

- これらは、システムによってプレートの追加や拡大縮小が行われないアセットです。
- ファイル名の形式: * AppList.targetsize-*_altformunplated.PNG
- 影響を受けるアプリ：すべての UWP アプリ
- 用途：
  - タスク バーと拡張 UI (デスクトップ)
  - タスク バーのジャンプ リスト
  - タスク ビュー
  - [Alt] + [Tab] キー

ファイル拡張子アセット

- これらはファイル拡張子に固有のアセットです。エクスプ ローラーで Win32 スタイルのファイルの関連付けアイコンの横に表示され、テーマに依存しません。サイズ調整は、デスクトップ プラットフォームとモバイル プラットフォームで異なります。
- ファイル名の形式: *LogoExtensions.targetsize-*.PNG
- 影響を受けるアプリ：ミュージック、ビデオ、フォト、Microsoft Edge、Microsoft Office
- 用途：

- ファイル エクスプローラー
- Cortana
- さまざまな UI サーフェス (デスクトップ)

スプラッシュ スクリーン

- アプリのスプラッシュ スクリーンに表示されるアセット。デスクトップとモバイルの両方のプラットフォームで自動的に拡大縮小されます。
- ファイル名の形式: *SplashScreen.screen-100.PNG
- 影響を受けるアプリ：すべての UWP アプリ
- 用途
  - アプリのスプラッシュ スクリーン

アイコン タイル アセット

- これらは、アイコン テンプレートを使用するアプリ用のアセットです。
- ファイル名の形式: 該当なし
- 影響を受けるアプリ：メッセージング、電話、ストアなど
- 用途
  - アイコン タイル

ロック スクリーンの設計ガイドライン

ユーザーは、Windows デバイスがロックされているときのロック スクリーンのプロバイダーとしてアプリを使うようにロック スクリーンをカスタマイズできます。また、ロック スクリーンに表示される基本的な通知 (メールやテキストなど) を、アプリで提供される通知を反映するように変更することもできます。このトピックでは、ロック スクリーンの背景と通知の実装に関するベスト プラクティスについて説明します。

ロック スクリーンのオプション

ユーザーがデバイスのロック スクリーンのプロバイダーとしてアプリを使う場合、ロック スクリーンでは、指定されたバッジとタイルのデータを使ってアプリに関する情報を表示します。

- アプリはロック スクリーンにバッジを表示できます。バッジのデザインに関する推奨事項については、「タイルとバッジのガイドライン」をご覧ください。
- アプリはロック スクリーンに詳細ステータスを表示できます。この詳細ステータスは、スタート画面の普通サイズ タイルに指定したのと同じコンテンツから取得されます。設計ガイダンスについては、「タイルとバッジのガイドライン」をご覧ください。

注 アプリがアダプティブ タイルを使う場合、ロック スクリーンには詳細ステータスを表示できません。

ロック スクリーンの背景の変更

次に示すように、ロック スクリーンの背景は、デバイスがロックされているときに表示される静止画像です。

PC やノート PC で実行されるユニバーサル Windows プラットフォーム (UWP) アプリではロック スクリーンの背景を変更できます。ロック スクリーンの背景を変更するかどうかはユーザーが確認します。

定期的な通知のガイドライン

製品通知は、クラウド サービスをポーリングして新しいコンテンツの有無を調べて、タイルとバッジを一定の間隔で更新します。Windows では、各ポーリング間隔の開始時にサービスに要求を送り、サービスが返すコンテンツをダウンロードして、アプリのタイルに新しいコンテンツを表示します。 詳しくは、「定期的な通知の概要」をご覧ください。このトピックでは、ユニバーサル Windows プラットフォーム (UWP) アプリで定期的な (ポーリングされた) 通知を使う際のガイドラインを示します。

重要な API

StartPeriodicUpdate メソッド　　StartPeriodicUpdateBatch メソッド

アプリに定期的な通知を含めるかどうか

定期的に一定間隔で更新する必要のあるコンテンツをアプリで提供している場合に、定期的な通知を使います。たとえば、この通知の種類は次のような場合に適しています。

- 現在の予報を表示するライブ タイルを 30 分ごとに更新する天気予報アプリ。
- 毎朝、新しい日々のキャンペーンをユーザーと共有するアプリ。

定期的な通知をトースト通知で使うことはできないことに注意してください。時間を争う差し迫ったアラート (ニュース速報の更新など) やトースト通知を使うスケジュールされたアラームを共有する場合は、プッシュ通知またはスケジュールされた通知を使います。4 種類の利用できる通知オプション (ローカル、スケジュール、プッシュ、定期的) を比較するには、「通知配信方法の選択」をご覧ください。

推奨事項

全般

- 定期的な通知は、該当しなくなったら期限切れにします。たとえば、真夜中に終了する特別なオンライン オファーは、期限が切れた後、表示してはいけません。
- サーバーに更新を要求する頻度は、30 分ごとに 1 回を超えないようにします。この間隔であれば、ユーザーに負担をかけずにタイルの状態を最新に保つことができます。

- 通知コンテンツは、ホーム ページやランディング ページなど、アプリ内の目立つ場所に表示します。こうすると、ユーザーがタイル通知を受けてアプリを起動したときに、ユーザーの目をひいた通知のコンテンツを簡単に見つけることができます。
- ニュース速報など、ユーザーが即座に受け取ることを求めているコンテンツに定期的な更新を使わないでください。時間を争う更新を配布するには、プッシュ通知サービスを使います。
- 定期的な通知を使って、ライブ タイルに広告を表示しないでください。タイルに広告を表示するのは禁止です。

効果的なタイルとバッジの設計に関する推奨事項については、「タイルとバッジのガイドライン」をご覧ください。

コード作成

- アプリを起動したりフォーカスを移動したりするたびに **StartPeriodicUpdate** または **StartPeriodicUpdateBatch** メソッドを呼び出します。これにより、アプリを起動したり切り替えたりするたびにタイルのコンテンツが更新されます。
- クライアントのポーリング頻度に合わせて、Web サービスのタイルとバッジの XML コンテンツを更新します。たとえば、タイルが 30 分間隔でポーリングするように設定されている場合は、Web サービスのコンテンツも 30 分おきに更新します。
- クラウド サービスにアクセスできなくなった場合や、ユーザーがネットワークに長時間接続していない場合は、古いコンテンツや無効になったコンテンツをタイルから削除します。たとえば、深夜 0 時に期限が切れるショッピング キャンペーンの場合は、有効期限を深夜 0 時に設定します。有効期限の設定について詳しくは、「定期的な通知の概要」をご覧ください。
- 特定の時刻に更新を行うには、**StartPeriodicUpdate** または **StartPeriodicUpdateBatch** の startTime パラメーターを使います。startTime は、初回ポーリングの時刻のみを指定します。それを基に、後続のポーリングのタイミングが決まります。繰り返し間隔を 24 時間として startTime を 2:00 PM に設定した場合は、毎日午後 2 時になると更新が行われます。

注 タイルは、任意の時間に、最大 5 件の通知を順番に表示できます。キューに通知が 5 件ある場合、既定では、次の新しい通知によってキューの最も古い通知が置き換えられます。ただし、**StartPeriodicUpdateBatch** を使うと、サービス側で X-WNS-Tag という HTTP 応答ヘッダーを使って通知にタグを付けて、キューの置き換えポリシーを変更することができます。キュー内の既にある 5 件の通知とタグが合致する新しい通知を受け取ると、(最も古い通知が自動的に置き換えられる代わりに) タグが合致する古い方の通知が新しい通知で置き換えられます。タグと通知キューの使用について詳しくは、「ローカル通知で通知キューを使用する方法 (HTML)」または「ローカル通知で通知キューを使用する方法 (XAML)」をご覧ください。

プッシュ通知のガイドライン

プッシュ通知はクラウド サーバーから送られて、アプリのライブ タイルを更新すると共に、トースト通知を送ります。このトピックでは、Windows ストア アプリでプッシュ通知を使う際の一般的なガイドラインとコーディングのガイドラインを示します。

重要な API

PushNotification 名前空間

プッシュ通知を使うかどうか

プッシュ通知の配信方法では、アプリが実行されていなくても、いつでもユーザーがアプリから通知を受け取ることができます。

プッシュ通知は、アプリで次の情報を共有する場合に便利なオプションです:

- リアルタイム更新 (ゲーム内のスポーツ スコアなど)
- 予測不可能なタイミングで生成されるコンテンツ (ニュース速報、受信メール、ソーシャル ネットワークの更新など)

Windows アプリで利用できる 4 種類の配信方法 (ローカル、スケジュール、プッシュ、定期的) の比較については、「通知配信方法の選択」をご覧ください。

推奨事項

- タイル通知とトースト通知の全体的なガイドラインに従います。タイル通知またはトースト通知をローカルとクラウドのどちらで生成するかに関係なく、同じユーザー ガイドラインに従う必要があります。詳しくは、次のトピックをご覧ください。
  - タイルのガイドライン
  - トースト通知のガイドライン
- ユーザーのバッテリの寿命を考慮します。デバイスが低電力状態でも、ユーザーは通知をいつでも受け取ることができます。送る通知が増えると、必要なリソースが増え、デバイスのスリープ状態を解除する頻度も増えることになります。通知の頻度を決めるときにはこの点に注意してください。
- 快適なユーザー エクスペリエンスを実現できる最低限の通知頻度を選びます。通知の頻度を増やすことで、必ずしもアプリの価値が高まるわけではありません。たとえば、タイルのコンテンツの更新頻度が多すぎると、一部の更新データをユーザーに見てもらえなくなります。
- プッシュ通知を使って機密データや重要なデータを送らないでください。たとえば、銀行の口座番号やパスワードを通知で送らないでください。
- 重要な通知に Windows プッシュ通知サービス (WNS) を使わないでください。WNS は信頼性の高いサービスですが、通知の配信は保証されていません。
- プッシュ通知を広告やスパムに使わないでください。WNS にはユーザーを保護する権利があります。アプリで通知を使う方法が不適切だと考えられる場合、WNS はそのアプリがプッシュ通知を利用できないようにブロックできます。ユーザーからアプリに悪意があるという報告があった場合、そのアプリは Windows ストアの削除ポリシーの対象となる可能性があります。

開発者向け

- WNS を使うことができるように、アプリをダッシュボードに登録します。アプリサーバーでは、ダッシュボードで提供される特定の資格情報を使って認証を行い、通知を送る必要があります。

- アプリを起動するたびにチャネルを要求します。チャネルの URL は期限切れになる場合があり、URL を要求するたびに常に同じであることが保証されているわけではありません。返されたチャネルの URL がこれまで使っていた URL と異なる場合は、アプリ サーバーで参照を更新します。
- チャネルの URL が WNS から提供されたものであることを確認します。WNS 以外のサービスに通知をプッシュしようとしないでください。チャネルの URL で "notify.windows.com" (Windows または Windows Phone) か、"s.notify.live.net" (Windows Phone のみ) のいずれかのドメインが使われていることを確認します。
- アプリ サーバーへのチャネル登録のコールバックを常にセキュリティで保護します。アプリがチャネルの URL を受け取り、その URL をアプリ サーバーに送るときには、セキュリティで保護された方法でこの情報を送る必要があります。チャネルの URL の送受信に使うメカニズムには、認証と暗号化を適用します。
- どのデバイスに URL が割り当てられているかアプリ サーバーが追跡できるように、チャネルの URL とデバイス ID の両方をアプリ サーバーに送信します。URL が変更された場合、アプリ サーバーは、そのデバイス ID に関連付けられた以前の URL を置き換えることができます。
- アクセス トークンを再利用します。アクセス トークンを使って複数の通知を送ることができるため、サーバーでは通知を送るたびに再認証を行わなくて済むように、アクセス トークンをキャッシュする必要があります。トークンが期限切れになっている場合は、アプリ サーバーにエラーが返されます。この場合、アプリ サーバーの認証を行い、通知を再試行する必要があります。
- パッケージ セキュリティ識別子 (PKSID) と秘密鍵を第三者と共有しないでください。これらの資格情報は、セキュリティで保護された方法でアプリ サーバーに保存します。秘密鍵が侵害されていると思われる場合は、新しいキーを生成します。攻撃者の標的になりにくくするために、新しい秘密鍵を定期的に生成します。

直接通知のガイドライン

直接通知はプッシュ通知の一種ですが、他の 3 種類のプッシュ通知 (トースト、タイル、バッジ) と異なり、関連付けられた UI がありません。他のプッシュ通知と同様に、Windows プッシュ通知サービス (WNS) 機能は、クラウド サービスからアプリに直接通知を配信します。このガイドラインでは、有効な直接プッシュ通知を作成する方法について説明します。

重要な API

RowNotification クラス

直接通知を使うかどうか

直接通知は、ユーザーがアプリに権限を与えている場合にアプリによるバックグラウンド タスクの実行をトリガーするなど、さまざまな目的で使うことができます。直接通知はまた、アプリでクラウド サービスからダウンロードできるデータがある場合に通知を受け取る優れた方法でもあります。アプリは次のことができます。

- 直接通知を、ファイル ダウンロードを開始するトリガーとして使う。たとえば、ユーザーがネット上で電子ブックを購入する場合、そのユーザーのリーダー アプリに直接通知を送り、その新しい電子ブックのダウンロードをトリガーするというものです。
- 直接通知を使い、インスタント メッセージや電話の受信を通信アプリに知らせる。通知を受けた通信アプリは、接続を確立し、ローカル トースト通知を使ってユーザーに知らせることができます。
- 直接通知を使い、クライアントとクラウド サービス間の同期操作を調整する (リーダー アプリで最後に読まれたページの同期をトリガーするなど)。

アプリとの通信に Windows プッシュ通知サービス (WNS) を使うことで、固定ソケット接続の作成、HTTP GET メッセージの送信、サービスとアプリ間でのその他の接続などに伴う処理のオーバーヘッドを回避できます。

詳しくは、「直接通知の概要」をご覧ください。

推奨事項

- 直接通知で送信する情報の量はできる限り抑えてください。WNS では直接通知で 5 KB を超えるデータを送信できないことに注意してください。
- 直接通知は、アプリがクラウド サービスからダウンロードできる情報を通知内に含めるのではなく、そのような情報があるということを知らせるために使います。
- 通知内のバイナリ データは、直接通知に含める前に base64 にエンコードしてください。これにより、転送中にコンテンツが不適切にエンコードされることがなくなり、クライアントによって確実に取得されます。
- 快適なユーザー エクスペリエンスを実現できる最低限の通知頻度を選びます。
- アプリを起動するたびにチャネルを要求します。チャネルの URL は期限切れになる場合があり、URL を要求するたびに常に同じであることが保証されているわけではありません。返されたチャネルの URL がこれまで使っていた URL と異なる場合は、アプリ サーバーで参照を更新します。
- チャネルの URL が WNS から提供されたものであることを確認する: WNS 以外のサービスに通知をプッシュしようとしないでください。チャネルの URL で "windows.com" ドメインが使われていることを確認します。
- アプリ サーバーへのチャネル登録のコールバックを常にセキュリティで保護します。アプリがチャネルの URL を受け取り、その URL をアプリ サーバーに送るときには、セキュリティで保護された方法でこの情報を送る必要があります。チャネルの URL の送受信に使うメカニズムには、認証と暗号化を適用します。
- アクセス トークンを再利用します。アクセス トークンを使って複数の通知を送ることができるため、サーバーでは通知を送るたびに再認証を行わなくて済むように、アクセス トークンをキャッシュする必要があります。トークンの有効期限が切れると、アプリ サーバーがエラーを受け取ります。アプリ サーバーを認証し、通知をもう一度試みます。
- 直接通知を利用して、連続した通知に情報を少量ずつ含める方法でアプリに情報をストリーミングしないでください。直接通知の送信は、クラウド サービスでトリガーされたイベントに応答する場合だけに限定する必要があります。
- バックグラウンド タスクの実行を継続させるという目的でクラウド サービスから直接通知を送信しないでください。このような処理は、ユーザーのバッテリ寿命を無駄に消費することになります。直接通知は、有用な情報をアプリに伝える必要があります。

- 関連付けられたバックグラウンド タスクによるリソース クォータの超過を引き起こす速度で直接通知を送らないでください。詳しくは、「バックグラウンド タスクのガイドライン」をご覧ください。
- 重要な通知の送信に WNS を使わないでください。WNS は信頼性の高いサービスですが、通知の配信は保証されていません。
- 通知を広告やスパムに使わないでください。WNS にはユーザーを保護する権利があります。アプリで通知を使う方法が不適切だと考えられる場合、WNS はそのアプリが通知を利用できないようにブロックできます。ユーザーからアプリに悪意があるという報告があった場合、そのアプリは Windows ストアの削除ポリシーの対象となる可能性があります。
- 直接通知にはサイズがゼロのペイロード コンテンツを含めないでください。ペイロードが含まれない直接通知は WNS によってドロップされ、アプリには配信されません。
- 直接通知を使って機密データや重要なデータを送らないでください。
- パッケージ セキュリティ識別子 (PKSID) と秘密鍵を第三者と共有しないでください。これらの資格情報は、セキュリティで保護された方法でアプリ サーバーに保存します。新しい秘密鍵を定期的に生成します。秘密鍵が侵害されている場合は、すぐに新しいキーを生成します。

その他の使い方のガイダンス

アプリで直接通知によってトリガーされるバックグラウンド タスクを使う前に、他の通信手段を利用できないか検討してください。ユーザーはアプリに対し、バックグラウンド タスクを実行するアクセス許可を明示的に付与する必要があります。このアクセス許可は、同時に最大 7 つのアプリに付与できます。アプリで他の通信メカニズム (標準のプッシュ通知、トースト更新など) を使えば、ユーザーの許可がなくてもバックグラウンド タスクを実行できます。

バックグラウンド タスクに替わる手段として、次の方法を利用できます。

- ユーザーに知らせるには、トースト プッシュ通知を送信します。
- タイルを更新するには、タイル プッシュ通知を利用します。

スケジュールされた通知のガイドライン

スケジュールされた通知を使って、定期的にアプリのタイルを更新したり、ユーザーにトースト通知を送ったりすることができます。Windows アプリにスケジュールされたタイル通知やトースト通知を追加するときには、次のガイドラインに従ってください。

重要な API

ScheduledTileNotification クラス　　ScheduledToastNotification クラス

アプリでスケジュールされた通知を使うかどうか

アプリ内からコンテンツを使ってアプリのタイル、バッジ、またはトースト通知を定期的に更新する場合は、スケジュールされた通知を使います。スケジュールされた通知は、タイルやバッジを更新する時刻またはトースト通知を表示する時刻を指定できる点を除いて、ローカル通知と同じです。

通知の内容が即時性を求める場合 (ニュース速報など)、予測不可能なタイミングで表示される場合 (受信メールなど)、アプリ外部からのデータに依存する場合、またはアプリが実行されていないときに更新する必要がある場合、別の形式の通知配信を使う必要があります。

Windows アプリで利用できる 4 種類の通知配信の概要については、「通知配信方法の選択」をご覧ください。

推奨事項

- タイル通知またはトースト通知の内容と各通知の更新頻度を決定する場合は、「タイルのガイドライン」と「トースト通知のガイドライン」の推奨事項に従います。
- **MaintenanceTrigger** クラスを使ってスケジュールを定期的に更新するために、**background tasks** を使うことを検討します。たとえば、最初に 1 週間の通知をあらかじめスケジュールしておき、**MaintenanceTrigger** クラスを使って以降の週のスケジュールを引き続き設定します。こうすると、ユーザーがある週にアプリを起動しなくても、継続的にスケジュールを設定できます。
- **timeZoneChange** システム トリガーを使って、システム クロックの変更 (夏時間など) に対応することを検討します。既定では、スケジュールされた通知は協定世

界時 (UTC) でトリガーされます。システム クロックの変更に対応して自動的に更新されるわけではありません。たとえば、アラーム アプリでは、システム時刻が変更された場合にアラームの予定時刻を変更する必要があります。そのためには、アプリで **timeZoneChange** トリガーに応答してタイミングを適切に調整するバックグラウンド タスクを使います。

タイルとバッジのガイドライン

このトピックでは、スタート画面とロック スクリーンの両方でのアプリのタイルの作成時または更新時に使用するベスト プラクティスおよびグローバリゼーション/ローカライズの推奨事項について説明します。

推奨事項

- 一般的なガイドライン
- 既定のタイル
- プレビュー テンプレート
- バッジ
- タイル通知

一般的なガイドライン

- アプリでタイル通知を使ってユーザーに更新を送らない場合は、小と普通サイズのタイルだけを使います。ワイドと大サイズのタイルのコンテンツは、定期的に更新し、新鮮な状態を保つ必要があります。ライブ タイルを使わない場合は、マニフェストでワイドまたは大サイズ ロゴを指定しないでください。
- アプリで、短い要約通知を使うシナリオだけをサポートする場合は、バッジと小または普通サイズのタイルだけを使います。要約通知とは、**badge image** または 1 つの数字でのみ表すことができる通知です。たとえば、SMS アプリで、受け取っ

た新しいテキストの数だけを伝えるために通知を使う場合などが、このシナリオに該当します。マニフェストにワイド ロゴを指定しないでください。

- ユーザーに表示する新しい興味深いコンテンツがアプリにあり、その通知が頻繁に更新される (少なくとも週 1 回) 場合にのみ、ワイドまたは大サイズのタイルを使います。
- 1 つの通知で複数の記事を同時に表示したり、長い一覧を表示したり、ユーザーがスタート画面で大きなサイズで見たがるような画像を表示したりするために、大きいタイルを使います。
- 既定のタイル イメージは、アプリのブランドを表すために基本的にアプリのロゴのキャンバスとして使います。
- ユーザーに合わせた興味を引く新しいコンテンツがない場合は、ライブ タイルは使わないでください。たとえば、電卓アプリなどにはそのようなコンテンツはありません。
- ユーザーへのスパムの送信や広告の表示にライブ タイルを使わないでください。Windows ストアから除外されることになります。
- 広告の表示にライブ タイルを使わないでください。

既定のタイル

- ワイド ロゴを含めるか、ワイド ロゴと大サイズ ロゴの両方を含める場合は、指定する普通サイズ、ワイド、大サイズのタイル イメージとのデザインの関係を検討します。ユーザーはサポートされている任意のサイズでタイルを表示でき、いつでもそのサイズを変更できることを忘れないでください。ここでは、一般的な規則をいくつか示します。
  - ロゴは、タイルの中央 (垂直方向および水平方向)に配置します。
  - 同じ高さの正方形タイルとワイド タイルの両方で、ロゴの縦方向の位置を同じに保ちます。大サイズ タイルで、ロゴの同じ比率の縦方向の位置を保ちます。
  - ロゴ イメージ自体にアプリ名が含まれていない場合は、タイルの下部にアプリ名を含めます。ただし、小サイズ タイルにはアプリ名を表示するオプションがありません。次の例は、両方のタイルの状態を示しています。
    マニフェストに定義されているアプリ名の要素を使ったタイル

ロゴ イメージにアプリ名を含んだタイル

- 長い名前のアプリの場合、名前が 2 行に折り返される可能性があるため、ロゴ イメージと名前が重ならないことを確認します。たとえば、普通サイズとワイドのタイルでは、100% の画像リソースで約 80 x 80 ピクセルにロゴの大きさを制限するのが、確実な方法です。
- 画像内でロゴ自体の周りのスペースを透明にすると、Windows 8 以降の外観の一部として、アプリのブランドの色 (マニフェストで宣言済み) が、事前に割り当てられたグラデーションを伴い、透けて表示されます。この方法は、先に示したメール アプリのタイルなどのロゴで使われます。

- 既定のタイルは、アプリの起動を露骨に要求するようなテキストを含んだデザイン ("クリックしてください!" と表示されたタイルなど) にしないでください。
- ロゴにアプリ名を含める場合は、名前フィールドでアプリ名を繰り返さないでください。ここで示すように、どちらか一方のみを使ってください。

プレビュー テンプレート

- シナリオに、それぞれが独立しているイメージとテキスト コンテンツが含まれる場合は、プレビュー テンプレートを使います。たとえば、テンプレートの上部に旅行先の写真を表示し、下部に場所の名前を表示する場合があります。

- プレビュー テンプレートでアニメーションを表示すると、ユーザーの注意を引き付けます。よって、魅力的なコンテンツを提供するようにしてください。コンテンツが魅力的でないと、ユーザーを不快にするだけです。
- プレビュー テンプレートを使うと、表示をサイクルのいずれかの終わり (フレーム) (テキストが一番下にある状態またはテキストが一番上にある状態) で開始し、もう一方のフレームに上がっていくまたは下がっていくアニメーションを表示できます。したがって、各フレームのコンテンツが単独で動作できることを確認します。
- プレビュー テンプレートを使ってユーザーが既に知っていることに関する情報を表示しないでください。たとえば、ビデオが一時停止中であるという通知をタイルに表示する場合は、プレビュー テンプレートを使わないでください。
- 概念的にグループ化されていない通知には、プレビュー テンプレートを使わないでください。たとえば、写真がテキストとまったく関係がない場合は、プレビュー テンプレートを使わないでください。
- プレビュー アニメーションによって通知の最も重要な部分が画面から消される可能性がある場合は、プレビュー テンプレートを使わないでください。たとえば、気温と付随する画像 (笑っている太陽、または雲) を表示する天気予報アプリでは、プレビュー テンプレートを使うと、気温 (通知するポイント) がいつでも表示されるとは限りません。画像と気温を同時に表示する静的テンプレートの方が、ユーザーにとっては便利です。
- ニュース記事の場合のように、画像にコンテキストを与えるためにテキストが必要な場合は、プレビュー テンプレートを使わないでください。

バッジ

- アプリで要約通知を使うシナリオだけをサポートする場合は、バッジを表示する普通サイズのタイルだけをサポートします。たとえば、ショート メッセージ サービス (SMS) アプリで、受け取った新しいテキストの数だけを表示する場合などです。ユーザーがタイルを小サイズに変更しても、バッジは表示されることに注意してください。
- シナリオで伝える情報を小さな数値で表すことができる場合は、バッジに数値を表示します。バッジに常に 50 以上の数値を表示する可能性が高い場合は、システム グリフの使用を検討します。バッジの数値が大きくなりすぎないようにする方法と

して、絶対数ではなく、ユーザーがアプリを前回起動した時点からの数を示すことが挙げられます。たとえば、アプリがインストールされた時点からの不在着信の合計件数を示すよりも、ユーザーがアプリを前回起動した時点からの不在着信件数を示す方が実用的です。

- 数値が役に立たない場合や非常に大きくなる場合は、用意されているシステム グリフのいずれかを使って変化を示します。たとえば、大量の RSS フィードの新しい未読記事は膨大な数になる可能性があります。この場合、数値ではなく、newMessage システム グリフを使います。
- 数値に意味がない場合はグリフを使います。たとえば、タイルに再生リストの "一時停止" 通知を表示する場合、数値では意味がわからないため、paused グリフを使います。
- 数値では意味があいまいになる場合は、newMessage グリフを使います。たとえば、ソーシャル メディア タイルのバッジに "10" と示されている場合、10 個の新しいリクエスト、10 件の新しいメッセージ、10 個の新しい通知、これらの組み合わせなど、複数の解釈が可能となります。
- タイルのバッジに最大値の "99+" が常に表示される可能性のある大量シナリオ (メールやソーシャル メディアなど) では、newMessage グリフを使います。常に最大値が表示される可能性がきわめて高い場合、同じ数値が表示された状態が続くことになり、ユーザーにとって役に立たない情報を伝えることになります。
- タイルの本文コンテンツとは別の場所でバッジの数値を繰り返さないでください。これは、2 つのインスタンスにずれが生じる場合があるためです。
- グリフでユーザーに伝える内容が変わることがない場合は、グリフを使わないでください。グリフは通知や過渡的な状態を表します。永続的なブランド情報や状態を表すものではありません。

タイル通知

- ユーザーに関する情報を基に、タイルを使ってそれぞれのユーザーに合わせた通知を送ります。タイル通知は、対象のユーザーに関連している必要があります。利用する必要がある情報の大部分は特定のアプリの内部にあり、ユーザーのプライバシー選択によって制限される場合があります。たとえば、テレビ ストリーミング サービスでは、最も視聴率の高い番組に関する最新情報をユーザーに示すことができ

ます。また、交通情報アプリでは、ユーザーの現在の場所に基づいて (ユーザーが現在の場所を知らせることを許可した場合) 最適な地図を表示できます。

- アプリが接続され、最新のライブ コンテンツを受け取っているとユーザーが感じるように、タイルに更新データを頻繁に送ります。タイル通知の更新間隔は、特定のアプリのシナリオによって異なります。たとえば、活発に利用されるソーシャル メディア アプリの場合は 15 分おきに更新し、天気予報アプリは 2 時間おき、ニュース アプリは 1 日に数回、毎日情報を提供するアプリは 1 日に 1 回、雑誌アプリは月に 1 回更新します。アプリでの更新が週に 1 回未満の場合は、古いコンテンツが表示されないように、シンプルな普通サイズ タイルとバッジを併用することを検討します。
- ユーザーが十分な情報に基づいてアプリの起動が必要であるかを判断できるように、魅力的で役立つタイル通知を提供します。一般に、通知とは詳細を知るために、またはアクションを実行するためにアプリを起動するようユーザーを促すものです。たとえば、通知によってユーザーはソーシャル メディアの投稿に返信したり、ニュース記事の全文を読んだり、セールの詳細を知ったりする場合があります。
- アプリのホーム ページまたはランディング ページでホストされるコンテンツについての通知を送信します。このようにして、ユーザーは通知に応答してアプリを起動するときに、通知の対象となったコンテンツを簡単に見つけることができます。

その他の使い方のガイダンス

- タイルの設計哲学
- さまざまなタイル サイズの選択
- 既定タイルの使用
- プレビュー テンプレートの使用
- デザインに関するその他の考慮事項
- タイル通知の更新

タイルは、スタート画面でのアプリの表示です。タイルにより、アプリが実行されていないときでも、スタート画面に魅力的なコンテンツを表示できます。タイルをタップまたはクリックすると、アプリが起動します。タイルには、3 つの正方形サイズ (小、普通、大) と 1

つのワイド サイズがあります。普通、ワイド、大サイズに対して複数のテンプレートのバリエーションがあり、テキスト、画像、またはテキストと画像の組み合わせを使うことができます。"プレビュー テンプレート" と呼ばれる一部のテンプレートは、タイルのスペース内をスクロールして移動できる、2 つの積み重ねられたフレームで構成されています。プレビュー テンプレートは、普通サイズとワイド サイズのタイルで利用できます。

タイルをライブにするか (通知による更新)、または静的のままにすることができます。最初に使うタイルは既定のタイルで、アプリのマニフェストで定義されます。静的なタイルには、必ず既定のコンテンツ (一般に、タイル全体を占めるロゴ イメージ) が表示されます。ライブ タイルは既定のタイルを更新して新しいコンテンツを表示できますが、更新が期限切れになるか、または削除された場合に、既定の状態に戻すことができます。タイルには、数字またはグリフで表すことができる状態バッジも表示できます。

普通サイズのタイル、ワイド タイル、大きいタイルでは、オプションで下隅にアプリ名 (既定のタイルまたはライブ タイル) または小さいアイコン (ライブ タイルのみ) でブランドを表示できます。

以下に、常に注意すべき非常に重要なポイントを 2 つ示します。

- ユーザーはタイルのサイズを、タイルでサポートされている任意のサイズに変更できます。ユーザーのスタート画面で現在どのサイズが表示されているかを知る方法はありません。すべてのタイルで小と普通サイズをサポートする必要がありますが、オプションでワイドと大サイズもサポートできます。大サイズをサポートする場合は、ワイド サイズもサポートする必要があるので、大サイズをサポートするには 4 つのすべてのタイル サイズをサポートしなければならないことになります。大きいタイルとワイド タイルは、タイルでライブ更新をサポートする場合にのみ使うようにしてください。
- タイルがライブ タイルをサポートしている場合、ユーザーはタイル通知のオンとオフをいつでも切り替えることができます。タイル通知がオフの場合、タイルは静的です。

タイルの設計哲学

目標は、アプリの魅力的なタイルを作成することです。ライブ タイルを使う場合は、アプリの起動を促す訴求力があり、かつユーザーがスタート画面に表示する価値があると思うような魅力的で新しいコンテンツを提供することが目標です。そのためには、派手な色を使いすぎないようにします。騒ぎ立てる子供のように、注意を引くために派手に飾り立てたタイルよりも、シンプルで見やすい洗練されたデザインのタイルの方が成功します。

アプリを設計するときに、わざわざライブ タイルを開発する価値があるかどうか、疑問に感じるかもしれません。その理由はいくつかあります。

- タイルはアプリへの "入り口" です。魅力的なライブ タイルは、アプリが実行されていないときに、ユーザーを引き付けることができます。ユーザーが頻繁に使うアプリを重視する傾向は強まりつつあります。
- ライブ タイルは、Windows ストアの他のアプリ (ユーザーは、似たようなアプリでも、静的なタイルを持つものよりも、便利なライブ タイルを持つものを好む傾向にあります) や、ホーム画面に静的なタイルとアイコンしか表示できないオペレーティング システムのアプリとの差別化につながるセールス ポイントです。
- ユーザーがライブ タイルを気に入った場合、スタート画面にその目立つタイルがあると、アプリを繰り返し使ってもらえます。そのタイルから素敵なアプリ コンテンツを偶然に発見すると、ユーザーは楽しい気持ちになるでしょう。
- ライブ タイルを使うと、ライブ更新を見られるように、ユーザーがアプリをアプリ ビューからスタート画面にピン留めする可能性が高くなります。
- ユーザーがタイルを気に入らなければ、スタート画面の一番後ろに置く、ピン留めを外す、更新を無効にする、アプリをアンインストールするといった結果になりかねません。

次のような特性がライブ タイルの魅力を高めます。

- 頻繁に更新される新鮮なコンテンツ。アプリが実行されていなくても、ユーザーはアプリがアクティブだと感じます。
  例: 最新のヘッドラインや、新しいメールの数の表示。
- ユーザーに関する情報 (アプリの設定を通じてユーザーが設定できるようにした興味の対象など) に基づく、カスタマイズされた更新。
  例: ユーザーの趣味に合わせてカスタマイズされた、1 日限定キャンペーン。

- ユーザーの現在の状況に合ったコンテンツ。
  例: ユーザーの現在位置を使って該当する交通情報地図を表示する、交通情報アプリ。

さまざまなタイル サイズの選択

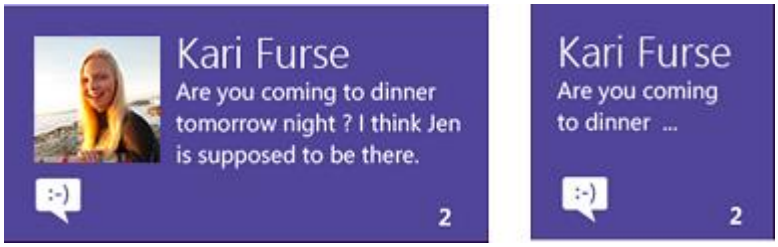
アプリには常に小サイズと普通サイズのタイルを用意する必要があります。アプリのマニフェストに、少なくとも普通サイズ タイルの画像アセットを用意する必要があります。小サイズ タイルのアセットも用意できますが、用意しない場合は、普通サイズ タイルのアセットの縮小バージョンが使われます。

また、ワイドまたは大サイズ タイルでも同様にするかどうかも決める必要があります。

- ワイド タイルをサポートするには、アプリのマニフェストに、既定のタイルの一部としてワイド (wide310x150) ロゴ イメージを含めます。その既定のワイド ロゴ イメージを含めなかった場合は、小 (square70x70) と普通 (square150x150) サイズだけがサポートされます。ユーザーがワイド サイズに変えることはできず、ワイド通知を受け取ることもできません。
- 大サイズ (square310x310) タイルをサポートするには、アプリのマニフェストに、既定のタイルの一部としてワイド ロゴ イメージと大サイズ ロゴ イメージを含めます。その既定の大サイズ ロゴ イメージを含めなかった場合は、ユーザーがタイルを大サイズに変えることはできず、大サイズ テンプレートを使う通知を受け取ることもできません。大サイズのタイルをサポートするにはワイド タイルのサポートが必要なので、既定の大サイズ ロゴ イメージを含めても既定のワイド ロゴ イメージを含めないと、両方を含めないのと同じ結果になります。

アプリが現在サポートしているタイル サイズ以外のサイズもサポートするには、追加の既定ロゴ イメージを含む更新されたマニフェスト付きでアプリの新しいバージョンをリリースする必要があります。

- 普通サイズ タイルに表示されるコンテンツは、ワイドや大サイズ タイルの場合よりも少なくなるので、コンテンツに優先順位を付けてください。ワイド タイルに表示できるコンテンツをすべて普通サイズ タイルに収めようとしないでください。さらに小さい小サイズ タイルでサポートされている唯一のライブ コンテンツは、バッジ通知です。



ワイド タイルのコンテンツに画像とテキストを含む場合は、正方形のプレビュー テンプレートを使って、コンテンツを 2 つのフレームに分割できます。ただし、画像自体で記事の概要を十分に伝えられない場合は、プレビュー テンプレートを使わないでください。

通知は、現在のタイルのサイズを知ることができないため、小サイズ タイル以外のサポートされているすべてのタイル サイズのテンプレート コンテンツを用意する必要があります。ワイド テンプレートのみを使って通知を定義し、タイルを普通サイズに表示している場合や、普通サイズ テンプレートのみを使って通知を定義し、タイルをワイドで表示している場合には、通知が表示されません。

規定のタイルの使用

アプリの既定のタイルは、アプリのマニフェストに定義されています。既定のタイルは静的で、一般的にはシンプルなデザインとなっています。アプリによっては、既定のタイルを使えば十分な場合があります。アプリのインストール後、ユーザーがタイルをアプリ ビューからスタイル画面にピン留めすると、既定のタイルが、通知を受け取るまでスタート画面に表示されます。ワイド ロゴ イメージを用意する場合は、タイルをスタート画面にピン留めする際の初期サイズを普通にするかワイドにするかを指定できます。マニフェストで指定されているワイド ロゴ画像によりワイド タイル サイズがアプリでサポートされている場合、アプリのタイルは既定でワイド タイルとしてピン留めされます。それ以外の場合、タイルは普通サイズでピン留めされます。ユーザーは、ピン留めされたタイルのサイズを、サポートされている任意のサイズに変更できます。期限切れになっていない新しい通知がなく、表示する必要がない場合は、ライブ タイルは既定の状態に戻ることがあります。

プレビュー テンプレートの使用

プレビュー テンプレートは、タイルのスペース内で 2 つのフレーム間の情報を切り替えるタイルのコンテンツを提供します。上のフレームは画像またはイメージ コレクションで、

下のフレームはテキストまたはテキストと画像です。例については、「タイル テンプレート カタログ」をご覧ください。

デザインに関するその他の考慮事項

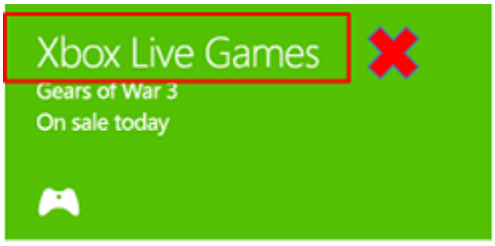
- タイルでアプリのブランド情報を伝える方法を決めるときは、次に示すようにアプリの名前を選びます。

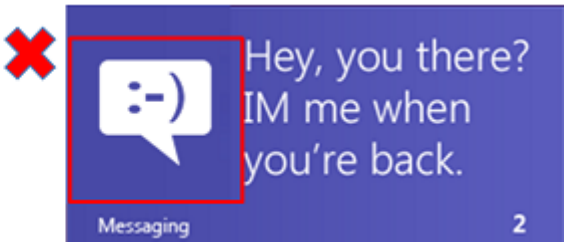
または、次に示すようにロゴ イメージを選びます。

これらの項目は最初はアプリ マニフェストで定義されます。開発者は、以降の各通知で 2 つのうちのどちらを表示するかを選ぶことができます。ただし、名前またはロゴを選んだら、常にそれを使って一貫性を保つ必要があります。スペースの制約により、一部のテンプレートでは名前を表示できず、ロゴを表示するか非表示にするかのオプションしかないことに注意してください。

- 画像要素またはテキスト要素を使って、タイルの通知にアプリのブランド情報を表示しないでください。ユーザーに対してアプリのブランドを印象付け、一貫性を保つには、アプリ名 (短い名前) またはロゴ イメージなど、その目的のために提供されたテンプレートの要素を使ってブランドを表す必要があります。ライブ タイルの外観は通知ごとに大きく変わることがありますが、名前/ロゴの場所には一貫性があります。このことにより、その情報が各タイルの同じ場所に表示されるため、ユーザーはすばやく見るだけでお気に入りのアプリを探すことができます。提供さ

れたブランド要素 (名前とロゴ) をアプリで利用しない場合は、ユーザーにとってすばやくアプリのタイルを特定することが難しくなる可能性があります。
次の画像は、ブランドの伝え方が不適切なテンプレートのテキスト要素と画像要素を使ったタイルを示しています。どちらの場合も、タイルは名前またはロゴをデザインされたまま使っています。つまり、さらにブランド化しても余分な情報となります





- オプションの "短い名前" で指定されたスペースにアプリの名前が収まらない場合は、短縮バージョンまたは意味のある頭文字を使います。たとえば、いつでも使いたくなる "Contoso Fun Game Version 3" の代わりに "Contoso Game" を使うことができます。最大ピクセル数を超える名前は途中で切り捨てられ、省略記号が付けられます。英語の場合、名前の最大長は 2 行で約 40 文字ですが、これは名前に含まれる文字によって異なります。デザインの観点から、短いアプリ名をお勧めします。また、マニフェストでさらに長いアプリの名前 ("表示名") を指定することもできます。この名前は、アプリ ビューとツールチップで使われますが、タイルでは使われません。
- タイルを広告に使わないでください。
- タイルで派手な色を使いすぎないようにします。騒ぎ立てる子供のように、注意を引くために派手に飾り立てたタイルよりも、シンプルで見やすい洗練されたデザインのタイルの方が成功します。
- テキスト コンテンツでテキストと画像を併用しないでください。テキスト コンテンツには、テキスト フィールドが含まれたテンプレートを使います。画像内のテキストは、レンダリングされたタイルのテキストほど鮮明には見えません。現在の表示に適した画像アセットが指定されていない場合は、画像は拡大される可能性があり、さらに読みにくくなることがあります。
- 緊急のリアルタイム情報をユーザーに送るときは、タイルに頼らないでください。たとえば、通信アプリでユーザーに着信を伝える場合、タイルは適切なサーフェスではありません。リアルタイム性のあるメッセージに適したメディアはトースト通知です。

- ハイパーリンク、ボタン、その他のコントロールのように見える画像コンテンツを含めないようにします。タイルではこれらの要素をサポートしておらず、タイル全体がシングルクリックの対象となります。
- 相対的なタイム スタンプや日時 (たとえば "2 時間前" など) は、時間が経過しても静的で、メッセージを不正確にするため、タイル通知で使わないでください。"11:00 A.M." などの絶対的な日時を使います。
- アプリのタイルからはアプリをホーム画面でしか起動できないため、タイルの更新は、そのホーム画面から簡単にアクセスできるアプリの要素に関するものにしてください。たとえば、ニュース アプリのタイルには、ユーザーがタイルをクリックしてアプリのホーム ページで簡単に見つけることができる記事のみを表示する必要があります。

タイル通知の更新

タイルを更新するための正しい通知方法の選択

ライブ タイルの更新には、以下の複数のメカニズムを利用できます。

- ローカル API 呼び出し
- ローカル コンテンツを使った、1 回限りのスケジュールされた通知
- クラウド サーバーから送信されるプッシュ通知
- 一定の間隔でクラウド サーバーから情報を取得する定期的な通知

使うメカニズムの選択は、表示するコンテンツとコンテンツを更新する頻度によって大きく左右されます。大部分のアプリは、起動時またはアプリ内での状態の変更時に、ローカル API 呼び出しを使ってタイルを更新すると考えられます。このようにすると、アプリの起動時と終了時に、タイルが必ず最新の状態になります。ローカル通知、プッシュ通知、スケジュールされた通知、またはポーリング通知のどれを使い、さらに単独または組み合わせて使うかは、アプリによってまったく異なります。たとえば、ゲームではプレーヤーが新しいハイ スコアに達したときに、ローカル API 呼び出しを使ってタイルを更新できます。同時に、同じゲーム アプリで、プッシュ通知を使って友人が獲得した新しいハイ スコアをプレイヤーに送信できます。

タイルの更新頻度

ライブ タイルを使う場合は、タイルの更新頻度を検討します。

- メッセージ数やゲームのプレイ順など各ユーザーに合わせたコンテンツでは、特にユーザーがタイルのコンテンツの遅れ、不正確さや不足に気付く可能性がある場合は、情報が利用可能となった時点でタイルを更新することをお勧めします。
- 天気予報の更新など、各ユーザーに合わせていないコンテンツでは、30 分に 1 回以下の頻度でタイルを更新することをお勧めします。このようにすると、ユーザーに負担をかけずにタイルが最新の状態になっていることを感じることができます。

タイル通知とバッジ通知の有効期限

タイルのコンテンツは、意味がなくなっても保持されることがないようにする必要があります。すべてのタイル通知とバッジ通知には、アプリにとって適切な有効期限を設定します。既定では、ローカルとスケジュールのタイルとバッジには期限がなく、プッシュ通知または定期的な通知を通じて送られたタイルとバッジのコンテンツの期限は送信後 3 日です。通知の有効期限が切れると、タイルまたはキューからコンテンツが削除され、ユーザーに表示されなくなります。

通知のコンテンツの期限として、特定の日時を設定できます。明示的な有効期限は、コンテンツの存続期間が決まっている場合に特に役立ちます。また、クラウド サービスが通知の送信を停止したり、アプリが長い間実行されなかったり、ユーザーがネットワークに長期間接続しなかったりする場合、明示的に有効期限を設定しておくと、システムの接続状態に関係なく、古いコンテンツが確実に削除されます。

たとえば、株式市場の取引が活発な日は、株価の更新の有効期限を送信間隔の有効期限の 2 倍に設定することをお勧めします (30 分ごとに通知を送っている場合は有効期限を通知の送信後 1 時間にするなど)。また、ニュース アプリの場合、毎日のニュースを表示するタイルの更新の有効期限は 1 日が適しています。

有効期限をどのように設定するかは、配信方法によって異なります。プッシュ通知と定期的な通知では、通知を配信するクラウド サービスとの通信に使われる HTTP ヘッダーで設定

します。ローカル通知とスケジュールされた通知では、API 呼び出しの一部として設定できます。

ロック スクリーンでのタイルとバッジの表示

- 良いロック スクリーンの表示の特徴
- ロック スクリーン要求 API を使うケース

アプリがロック スクリーンでの表示に適しているかどうかを判断するには、ロック スクリーンの操作と制限を理解する必要があります。ここではロック スクリーンの要約を示します。詳しくは、「ロック スクリーンの概要」をご覧ください。

- 最大 7 つのアプリ バッジをロック スクリーンに表示できます。バッジの情報には、アプリのスタート画面のタイルに表示されるバッジの情報が反映されます。バッジ (グリフまたは数値のどちらか) には、バッジが関連付けられたアプリを特定できるモノクロのアイコン (ロゴ イメージ) が付随します。
- これら 7 つのアプリのうち 1 つだけが詳しい状態スロットを占有し、アプリの最新のタイルの更新をテキスト コンテンツで表示できます。
- ロック スクリーンの詳しい状態タイルには、タイルの更新に含まれた画像は表示されません。
- ユーザーは、ロック スクリーンに情報を表示できるアプリと、詳しい状態を表示できるアプリを選びます。
- ロック スクリーンに表示されるすべてのアプリは、バックグラウンド タスクを実行することもできます。バックグラウンド タスクを実行できるすべてのアプリは、ロック スクリーンに表示されます。アプリは、同時にロック スクリーンのスロットを要求しないでバックグラウンド タスクを使うことはできません。
- 通知キューは、ロック スクリーンの詳しい状態タイルではサポートされません。最新の更新のみが表示されます。
- ロック スクリーンに表示されるアプリでは、そのアプリのマニフェストで **[トースト対応]** オプションを "はい" に設定している限り、ロック スクリーンの表示中、受け取ったトースト通知がロック スクリーンに表示されます。ロック スクリーンに表示されるトーストは、別の場所で表示されるトーストと同じです。

- タイルの更新、バッジの更新とトースト通知は、ロック スクリーン専用に設計されていたり、またはロック スクリーンだけに送られるわけではありません。送信者には、デバイスが現在ロックされているかどうかはわかりません。ロック スクリーンに表示されるアプリの場合、通知はスタート画面とロック スクリーンの両方に反映されます。

良いロック スクリーンの表示の特徴

アプリをロック スクリーンに表示するには、必ずユーザーが明示的な許可を与える必要があります。それには、ユーザーがアプリからの要求に応えるか (ユーザーに要求できるのは 1 回限り)、**[設定]** で手動で許可します。アクセス許可を与えることで、ユーザーはアプリから送られる情報が自分にとって重要であることを示し、アプリはそれに応える必要があります。そのため、アプリ自体がロック スクリーンでの表示に適しているかどうかを検討する必要があります。

ロック スクリーンでの表示に適した候補には、次のような特性があります。

- 情報がすぐに理解できる
- 情報は常に最新にする
- 追加のコンテキストがなくても、情報が理解できる
- ユーザーに合わせた便利な情報
- 役立つ情報に絞った的確な表示
- トースト通知でのみ到着時にサウンドを再生する

情報がすぐに理解できる

ロック スクリーンが表示されると、ユーザーが現在デバイスを操作していないことを表します。そのため、アプリがロック スクリーンに表示する更新情報は、ユーザーの注意を引き、ひとめで理解できるものにする必要があります。似たような例として、携帯電話の着信を考えてみます。電話をひとめ見れば発信者がわかり、応答するか、そのままボイス メールに送ることができます。ロック スクリーンに表示される情報も、携帯電話の表示と同じように簡単に理解でき、処理されるものにしてください。他のすべての特徴を使って、これを実現できます。

情報は常に最新にする

良いバッジの更新、タイルの更新、トースト通知は、スタート画面またはロック スクリーンアプリのどちらに表示されても、すべてアクション可能です。このような通知が提供する情報に基づいて、ユーザーは新しいメールを読む、ソーシャル メディアの投稿にコメントするなど、アプリを起動して応答するかどうかを判断できます。ロック スクリーンでは、デバイスのロックの解除も必要です。そのため、ユーザーが十分な情報に基づいて判断できるよう、情報は最新にする必要があります。ユーザーは、ロック スクリーンに表示されるアプリの情報が最新でないことに気付くようになると、アプリを信頼しなくなり、より信頼性の高い役立つアプリを探してロック スクリーンのスロットに配置するようになるでしょう。

良い例: 最新の情報

- メッセージング アプリは、新しいメッセージが届いたときに通知を送ります。その通知が無視されると、アプリは未読のメッセージ数でバッジを更新します。ユーザーが操作中である場合、画面をオンにしてメッセージの重要度を判断し、すぐに応答するか、後で応答するかを選ぶことができます。ユーザーが操作中でない場合は、戻ったときに正確な未読メッセージの数を確かめます。
- メール アプリはバッジを使って未読メールの数を表示します。 新しいメールが到着すると、バッジがすぐに更新されます。ユーザーはすぐに画面をオンにして未読メールの数を確かめることができます。また、その数が正確であることが保証されます。ユーザーには、デバイスのロックを解除してメールを読むかどうかを判断する情報があります。

悪い例: 古い情報

- メッセージング アプリは、30 分に 1 回だけ未読のメッセージ数でバッジを更新します。ユーザーは、デバイスのロックを解除するかどうかを判断する際に、バッジの数値は参考になりません。
- 詳しい状態スロットを使う天気予報アプリが、重大な天気予報の警報が期限切れになった後も、その警報を表示し続けます。これはユーザーに誤った情報を与えるだけでなく、テキストに警報の期限が示されていると、古い情報であることがユーザーにはっきりわかるため、非常に良くない例です。アプリで常に適切な情報を知る

ことができるというユーザーからの信頼が失われます。情報が期限切れになったとき、アプリはその情報をクリアしておく必要があります。

- カレンダー アプリが、期日を過ぎた予定を表示し続けています。この場合も、情報が期限切れになったときに、アプリはその情報をクリアしておく必要があります。

追加のコンテキストがなくても、情報が理解できる

ロック スクリーンには、次のようなコンテキスト情報は表示されません。

- アプリに詳しい状態の表示が許可されていない場合の、バッジに対応するタイル。詳しい状態が表示される場合でも、バッジはタイルからは物理的に切り離されています。バッジの横のロゴ イメージが、そのアプリを示す唯一の要素です。
- タイル更新時の画像。更新のテキスト部分だけが、詳しい状態スロットに表示されます。
- 通知キュー。最新の更新だけが、詳しい状態スロットに表示されます。

そのため、更新は、利用できる追加のコンテキストがスタート画面になくても、ユーザーが理解できるようにする必要があります。繰り返しますが、通知をロック スクリーン専用にすることはできません。したがって、アプリによる更新のやり取りは、必ず "追加のコンテキストがなくても理解できる" というルールに従う必要があります。

注 詳しいタイルとは異なり、トーストでは画像 (存在する場合) とテキストの両方が表示されます。ロック スクリーンに表示されるトーストはそれ以外の場所に表示されるトーストと同じであるため、コンテキストは失われません。

良い例: 追加のコンテキストがなくても理解できる

- メール アプリはバッジを使って未読メールの数を表示します。スタート画面のタイルには、最新のメールからのテキスト スニペットや送信者の画像など、詳しい情報が表示されることがありますが、バッジではこのような追加情報がなくても理解できる情報を伝えます。
- ソーシャル ネットワーキング アプリが、詳しい状態スロットを使って、ユーザーに友人の最近の活動を知らせます。友人がユーザーにメッセージを送ると、通知のテキストに友人の名前が含まれます (たとえば、"カイルから新しいメッセージが届

きました!" など)。スタート画面では、通知に表示される友人の画像によって、ユーザーに優れたエクスペリエンスを提供できます。ロック スクリーンでは、画像は表示されませんが、だれがメッセージを送信したかがテキストからはっきりとわかります。

悪い例: 追加のコンテキストがないと理解できない

- メッセージング アプリが最新の受信メッセージでタイルを更新し、送信者の画像とメッセージのテキストだけを表示します。スタート画面では、ユーザーは、だれがメッセージの送信者であるかはっきりわかります。ロック スクリーンでは、送信者の画像がないため、ユーザーはだれがメッセージを送信したのかわかりません。
- ソーシャル ネットワーキング アプリがフォト コラージュでタイルを更新します。テキストはありません。スタート画面では、これは楽しく、生き生きとしたタイルです。ロック スクリーンでは、タイルの更新にテキストが含まれていないため、何も表示されません。

ユーザーに合わせた便利な情報

ロック スクリーンの主な目的のうち 2 つは、ユーザーにカスタマイズされたサーフェスを提供することと、アプリの更新プログラムを表示することです。アプリがロック スクリーンでの表示に適しているかどうかを判断する際に、これらの目的を両方とも検討してください。

ロック スクリーンに表示されるアプリは非常に特殊です。同時に 7 個だけロック スクリーンに表示できます。貴重なロック スクリーンのスロットの 1 つを割り当てることにより、ユーザーは、頻繁にデバイスを使っていない場合でも、そのアプリから送られる情報が重要で確認する必要があることを示しています。そのため、アプリが提供する情報は、ユーザーに合わせると同時に便利である必要があります。

注 デバイスがロックされるとロック スクリーンが表示されるというのが決まりです。ロック スクリーンのコンテンツを表示するうえで、ログインなどのセキュリティ上の操作は必要ありません。したがって、ロック スクリーンに表示される情報は個人向けにカスタマイ

ズされるのが理想的ですが、だれでもその情報を見ることができるということに注意してください。

良い例: ユーザーに合わせた情報

- メール アプリは、ユーザーのアカウントにある未読メールの数を表示します。
- メッセージング アプリは、ユーザーに送信された未読メールの数を表示します。
- ニュース アプリは、ユーザーがお気に入りとしてフラグを付けたカテゴリにあるニュース記事の数を表示します。

悪い例: ユーザーに合わせていない情報

- ニュース アプリは、ユーザーが示した好みをまったく考慮しないで、ニュース サービスから送られるすべての新しい記事の合計数を表示します。
- ショッピング アプリは、セールの通知を送りますが、ユーザーが示した好みの商品やカテゴリに基づいていません。

役立つ情報に絞った的確な表示

情報は、変更が発生した場合だけ表示する

前に説明したように、目的は、ロック スクリーンで情報がひとめで理解されるようにすることです。そのために、アプリが現在バッジを表示していない場合は、ロック スクリーン上でバッジが本来表示される場所にすき間が残されます。このことにより、ユーザーは注意が必要な情報に気付きやすくなります。イベント後のバッジとロゴの外観は、新しい情報を伝えることなく最初からその場所に表示されていた場合よりも目立ちます。

状態を示すという目的のためだけに、状態を表示しないでください。長期間続く状態や変更されることのない状態は、ロック スクリーンを煩雑にするだけで、より重要な情報を目立たなくします。ユーザーが気付くべきイベントが発生したときだけ、バッジを表示します。タイルの更新も同じです。古くなった通知コンテンツはタイルから削除します。削除すると、スタート画面ではタイルは既定の画像に戻り、ロック スクリーンでは何も表示されません。

良い例: 有益な場合にのみ表示される情報

- メール アプリは、未読メールがあるときにのみバッジを表示します。新しいメールが到着すると、バッジは更新されて表示されます。
- メッセージング アプリは、ユーザーがメッセージを受信できないときだけ接続状態を表示します。"接続された" 状態は、アプリの既定の状態であるという前提のため、その情報を伝える意味はありません。"すべて順調" という通知は、アクション可能な通知とはいえません。ただし、メッセージを受信できないときにユーザーに通知することは有益で、アクション可能な情報といえます。

悪い例: 長期間続く状態

- メール アプリまたはメッセージング アプリで、未読メールの数が表示されないと、新しいメールまたはメッセージが到着するまで接続状態を表示します。このようにすると、バッジが常に表示されているため、ユーザーは新しいメッセージがあるかどうかをひとめで確認しにくくなります。
- カレンダー アプリが、予定がないことを示すメッセージを表示します。この場合も、常に何か情報が詳しい状態スロットに表示されるため、ひとめ見て確認できるという詳しい状態スロットの操作性が低下します。

トースト通知でのみ到着時にサウンドを再生する

バッジまたはタイルが更新されたときにサウンドを再生するコードをアプリに埋め込まないでください。トーストについては、到着時にサウンドを再生するように設計できます。

この記事で説明しているガイダンスに従うと、ロック スクリーンに正しい情報を正しい方法で表示するアプリを作成できるようになり、アプリに対するユーザーの満足度と信頼を高めることができます。

ロック スクリーン要求 API を使うケース

アプリが正常に機能するためにバックグラウンド権限が必要な場合にのみ、ロック スクリーン要求 API (**RequestAccessAsync**) を呼び出してください。利用できるバックグラウンド スロットは 7 つだけであるため、ユーザーは、正常に機能するにはバックグラウンド権限

が必須であるアプリと、(バックグラウンド権限があれば機能が追加される場合でも) バックグラウンド権限がなくても正常に機能するアプリを見分ける必要があります。

アプリがユーザーの要求を満たすうえでバックグラウンド権限が不可欠な場合にのみ、要求 API を使って、アプリをロック スクリーンに配置するようユーザーに求めることをお勧めします。

バックグラウンド権限がなくてもユーザーの要求を満たすことができる場合は、アプリをロック スクリーンに配置するようユーザーに対して明示的に求めないようにしてください。代わりに、ユーザーが **[設定]** の **[パーソナル設定]** ページを通じてアプリをロック スクリーンに配置できるようにします。

要求 API を呼び出す必要のあるアプリの例:

- アプリがフォアグラウンドに存在しないときにメッセージを受け取るためにバックグラウンド権限が必要なメッセージング アプリ
- アプリがフォアグラウンドに存在しないときにユーザーの受信トレイを同期するためにバックグラウンド権限が必要なメール アプリ

要求 API を呼び出さないようにするアプリの例:

- 予報を更新するためにバックグラウンド アクティビティではなく定期的な通知を使う天気予報アプリ
- バッジで示される新しい記事の数を特定の時刻に更新するニュース アプリ

注 アプリには、ロック スクリーンにそのアプリを追加するようユーザーに求めるダイアログを実装できません。アプリでロック スクリーンへのアクセスが正常に動作する必要がある場合は、ロック スクリーン要求 API によって表示されるダイアログを使う必要があります。以前にユーザーがこのダイアログでアプリに対するロック画面の権限を拒否した場合は、このダイアログが再表示されないことがあります。この場合は、アプリのインライン テキストを使って、**[設定]** の **[パーソナル設定]** ページにユーザーを誘導し、手動でアプリをロック スクリーンに追加してもらうこともできます。

トースト通知のガイドライン

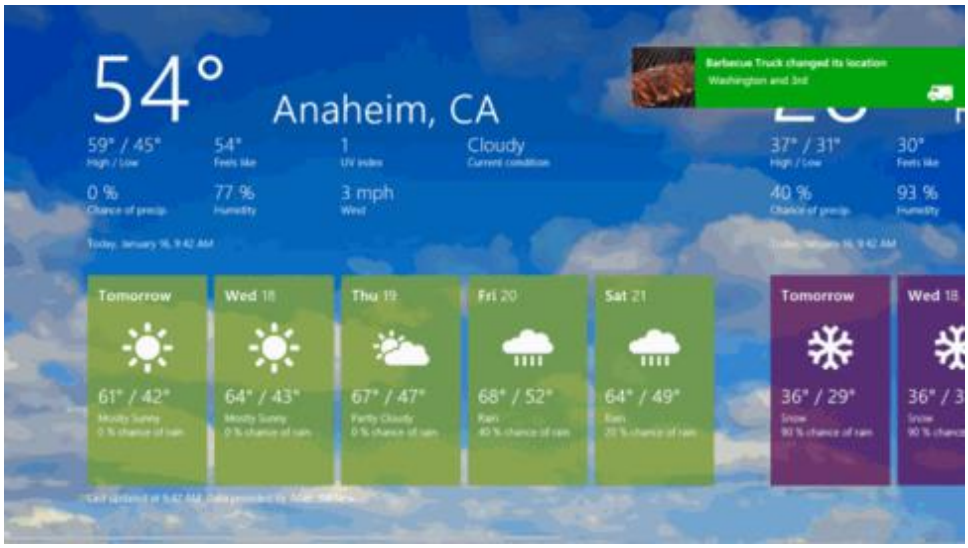
このトピックでは、トースト通知の用途を説明し、トースト通知の作成方法と送信方法について推奨事項を示します。

重要な API

ToastNotification クラス

例

お気に入りの移動式屋台が出店場所を変えたときにトースト通知を受け取るようにしています。



アプリにトースト通知を含めるかどうか

トースト通知を使うと、即時性が必要な事項や個人的に関係のある事項について、別のアプリを使っている場合や、スタート画面、ロック画面、デスクトップが表示されている場合でも関係なく、ユーザーに対して通知できます。たとえば、トースト通知を使ってユーザーに次のようなことを通知できます。

- VOIP の着信呼び出し
- 新着インスタント メッセージ
- 新着テキスト メッセージ

- カレンダーの予定やその他のアラーム
- その他のユーザーが要求した個人的に重要な通知

トースト通知を受け取るには、ユーザーがオプトインする必要があり、また、ユーザーはいつでもトースト通知を無効にできることを忘れないでください。

推奨奨事項

アプリにトースト通知を追加するときは、以下の推奨事項を考慮してください。

- ユーザーがトースト通知をクリックしたときに、アプリの適切な移動先に移動します。通知は厳密な情報更新ではなく、コンテキストの切り替えをユーザーに促すものと考えてください。
- 情報が重要な場合は、トースト通知で得られる情報をユーザーが入手できる代替方法を用意します。たとえば、アプリのライブ タイルやアプリ内に関連情報を表示することができます。
- 短期間に複数の関連する更新が発生した場合は、まとめて 1 つのトースト通知にします。たとえば、3 つの新しい更新が同時に到着した場合、アプリまたはアプリ サーバーは、3 つの個別の通知ではなく、3 つの新しい更新があることを示す 1 つの通知を表示する必要があります。
- 情報はできるだけシンプルな形で提供します。コンテンツでヘッドラインが不要な場合は、ヘッドラインを省略します。"ダウンロードが完了しました" などのメッセージはこれで完結しているので、追加の表示は不要です。
- 画像はメッセージに明確な付加価値を付ける場合に使います (メッセージの送信者の写真など)。
- 有効でなくなった通知は表示しないようにします。たとえば、相手が電話を切ったり、ユーザーが別のデバイスで既に通話している場合は、着信呼び出しを非表示にします。アプリが実行中のときしか通知を非表示にできないことに注意してください。
- 重大な情報をユーザーに通知するためにトースト通知を使わないでください。代わりに、重大な通知が確実に目にとまるように、フライアウト、ダイアログ、アプリバー、その他のインライン要素を使ってアプリ内でユーザーに通知します。

- "... するためにここをクリックする" ことをユーザーに伝えるテキストを含めないでください。すべてのトースト通知は、クリックまたはタップ操作によって関連アプリに移動することを前提としています。
- トースト通知を使って、一時的な障害やネットワーク イベント (接続の切断など) をユーザーに通知しないでください。
- 株価情報など、大量の通知を伴うものにトースト通知を使わないでください。
- トースト通知を使って、定期メンテナンス イベント (ウイルス スキャンの完了など) をユーザーに通知しないでください。
- アプリがフォアグラウンドで動作していて、コンテキストにより適したサーフェス (インライン要素、フライアウト、ダイアログ、アプリ バーなど) を使うことができる場合は、トースト通知を表示しないでください。たとえば、ビュー内で進行中の会話に関連する追加のインスタント メッセージは、個々の新しいメッセージが含まれたトースト通知を継続して表示するのではなく、会話をインラインで更新する必要があります。アプリケーションが実行中のとき、**PushNotificationReceived** イベントをリッスンして、プッシュ通知を中断します。
- 通知の画像フィールドにアイコンやアプリのロゴなどの汎用画像を追加しないでください。
- 通知のテキストにアプリ名を含めないでください。ユーザーはアプリのロゴでアプリを識別します。アプリのロゴはトースト通知に自動的に表示されます。
- ユーザーがトースト通知を無効にすることを選んでいる場合は、アプリを使って、トースト通知を有効にすることをユーザーに要求しないでください。アプリはトースト通知なしで動作することを求められています。
- バルーン通知のシナリオをトーストに自動的に移行しないでください。ユーザーが全画面表示のエクスペリエンス (デスクトップ スタイル アプリのみ) に没入していないときには、バルーン通知の方が適している場合もあることを考慮します。
- その日の概況など、リアルタイムでない情報にトースト通知を使わないでください。
- どうしても必要な場合以外は、トースト通知を非表示にしないでください。
- ユーザーが通知を求めていないものを通知しないでください。たとえば、知り合いがオンラインになるたびに通知されることをすべてのユーザーが望んでいるわけではありません。