Firemacs の作り方

山本和彦
(株)インターネットイニシアティブ
kazu@iij.ad.jp

内容

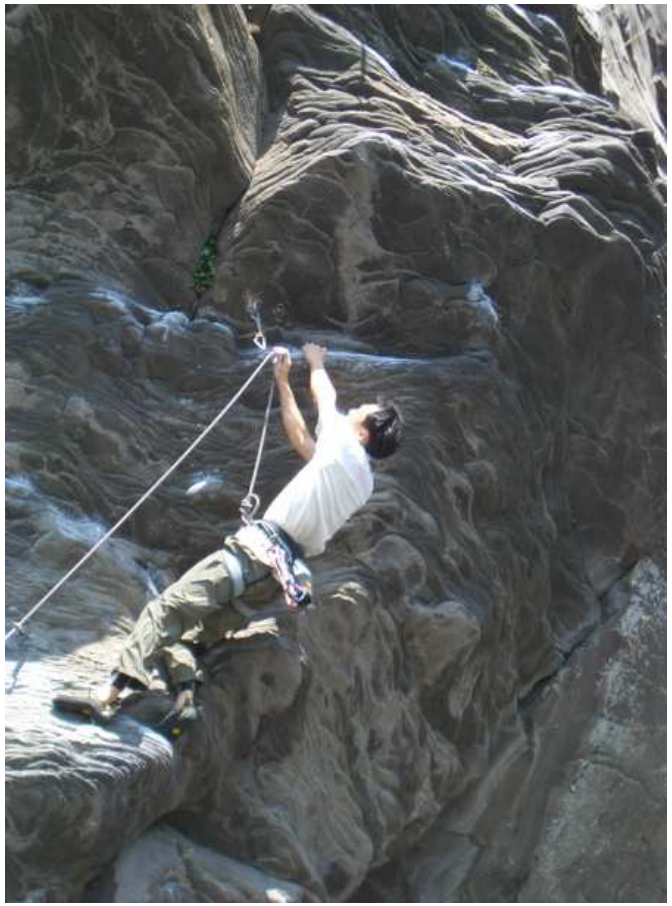
- 自己紹介
- Firemacs とは？ 動機は？
- 作り方
  - ステップ1) 単純な Emacs コマンドを実装してみる
  - ステップ2) 拡張機能としてパッケージ化
  - ステップ3) 開発し易くするための小技
  - ステップ4) 少し複雑な Emacs コマンドを実装してみる
  - ステップ5) Gmail、Vox、Wiki に勝つ
  - ステップ6) キー割り当ての機能を実装してみる

自己紹介

- IIJ の技術研究所に所属
  - 好きなプログラムを書かせて頂いています
- メールリーダ Mew の作者
- プログラミング言語
  - C, Emacs Lisp, JavaScript
- 最近やっていること
  - Firefly -- WYSIWYG のプレゼンテーション・ツール
- 得意な分野
  - TCP/IP、メール、暗号
- 今日は講師をやりますが初心者です
  - 拡張機能、JavaScript、CSS、オブジェクト指向...
- 趣味
  - ロック・クライミング
  - ダイビング

ロック・クライミング

- 伊豆 城ヶ崎 「風に吹かれて」(5.11a)



ダイビング

- モンハナ・シャコ

動機

- Emacs でないと編集能力が極端に下がる
- Mixi、Vox、Gmail では文章を書く気になれない
- でも Wiki は使わないといけない
  - mozex を使って Emacs に飛ばす
    - http://www.mew.org/~kazu/toy/firefox-editor.html
  - やっぱり面倒だ～
- Firefox で直接 Emacs の編集機能を使いたい
  - そういう拡張機能を探してみたが存在しなかった
  - 拡張機能の練習も兼ねて自分で作ることにした
- 名前は Firemacs に決定！

Firemacs とは

- Emacs on Firefox
  - http://www.mew.org/~kazu/proj/firemacs/
- 機能
  - textarea / input での Emacs ライクな編集
    - コピー(C-w)、ペースト(C-y)、リージョン(C-SPC)
    - Mixi, Wiki
  - HTML エディタでの Emacs ライクな編集
    - Vox, Gmail
  - Emacs ライクな検索 (C-s, C-r)
  - ページの Emacs / less ライクなスクロール (C-v, M-v)
  - タブの移動 (C-f, C-b)
  - textarea / input の移動 (C-n, C-p)
  - 補完メニューでの移動 (C-n, C-p)
  - キー割り当てのカスタマイズ

ステップ1

- 単純な Emacs コマンドを実装してみる
- 指針
  - どうやってキーのイベントを捕まえるのか？
  - どうやってどのキーが押されたか調べるのか？
  - どうやって編集コマンドを実装するのか？

キーのイベントを捕まえる

- Mozless が参考になりそう
  - http://mozless.mozdev.org/
  - Less のページ・スクロール機能を付ける
- `mozless-0.1.14.xpi`
  - XPI(ジッピー) = Cross-Platform Installer Module
  - .xpi は、単なる ZIP ファイルらしい
    - `% unzip mozless-0.1.14.xpi`
  - 出てきた `chrome/mozless.jar` も ZIP ファイルらしい
    - `% unzip chrome/mozless.jar`
- `chrome/content/mozless.js` を読む
  - イベント・リスナーで処理しているらしい

```
window.addEventListener("keypress", mozless_onkeypress, false);
```

  - `mozless_onkeypress(event)` で、
    キーに対応するコマンドを呼び出す

押されたキーを特定する

- イベント・リスナーにはイベントが渡される
  - `event.keyCode`
  - `event.charCode`
  - `event.ctrlKey`
  - 詳しくは以下に書いてあった
    - http://developer.mozilla.org/en/docs/Category:Gecko_DOM_Reference
- たとえば、`C-f` は？
  - `event.ctrlKey` → true
  - `String.fromCharCode(event.charCode)` → `"f"`

Emacs の編集コマンドを呼び出す

- Firefox には Emacs ライクな編集機能が実装されているらしい
  - 利用するには、`platformHTMLBindings.xml` を編集する
    - http://kb.mozillazine.org/Emacs_Keybindings_(Firefox)
- たとえば、一文字次へは？
  - `cmd_charNext`
  - 詳しくは以下に書いてあった
    - http://www.mozilla-japan.org/unix/customizing.html
- JavaScript から呼び出すには？
  - Firefox のソースを grep
  - `goDoCommand` を使うらしい
  - `goDoCommand('cmd_charNext');`

firemacs.js

- 拡張機能同士は名前空間を共有するらしい
- 他とぶつからない名前を利用すること

```
function Firemacs() {
}

Firemacs.prototype.charNext = function() {
    goDoCommand('cmd_charNext');
};

Firemacs.prototype.keypress = function(event) {
    if (event.ctrlKey == false) return;
    var char = String.fromCharCode(event.charCode)
    switch(char) {
    case 'f': this.charNext();
    break;
    }
}
(function() {
    var fmx  = new Firemacs();
    window.addEventListener('keypress',
                            function(event){fmx.keypress(event);},
                            false)
})();
```

ステップ2

- 拡張機能としてパッケージ化
- Mozless の必要な部分をコピーしたけど、
  それは遠回りだった
  - install.rdf の書式が古い
  - install.js、contents.rdf はもう不要

今どきのパッケージ化

- 用意するもの
  - `firemacs/install.rdf`
  - `firemacs/chrome.manifest`
  - `firemacs/chrome/content/firemacs.xul`
  - `firemacs/chrome/content/firemacs.js`
  - `firemacs/chrome/skin/icon.png`
- 各ファイルの関係
  - `install.rdf` → `icon.png`
  - `chrome.manifest` → `firemacs.xul` → `firemacs.js`

install.rdf (1)

- 以下を見て作る
  - http://developer.mozilla.org/ja/docs/install.rdf
- <em:id> には GUID を指定する
  - http://developer.mozilla.org/en/docs/Generating_GUIDs
  - % uuidgen

    e98b7313-167d-48c6-89be-bc514d6de8d9
- アイコンの指定
  - <em:iconURL>chrome://firemacs/skin/icon.png</em:iconURL>

install.rdf (2)

- Firefox の GUID
  - http://firefox.geckodev.org/index.php?GUID%E4%B8%80%E8%A6%A7
  - `ec8030f7-c20a-464f-9b0e-13a3a9e97384`
- Firefox のバージョン
  - http://developer.mozilla.org/en/docs/Toolkit_version_format
  - `<em:minVersion>1.5</em:minVersion>`
  - `<em:maxVersion>2.0.0.*</em:maxVersion>`

chrome.manifest

```
overlay chrome://browser/content/browser.xul #継続
        chrome://firemacs/content/firemacs.xul
content firemacs jar:chrome/firemacs.jar!/content/
skin    firemacs classic/1.0 jar:chrome/firemacs.jar!/skin/
```

firemacs.xul

```
<?xml version="1.0"?>
<overlay xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <script type="application/x-javascript"
          src="chrome://firemacs/content/firemacs.js"/>
</overlay>
```

パッケージにする

```
% cd chrome
chrome% zip firemacs.jar content/firemacs.xul \
                         content/firemacs.js \
                         skin/icon.png

chrome% cd ..
% zip firemacs-0.1.xpi install.rdf \
                       chrome.manifest \
                       chrome/firemacs.jar
```

- 面倒なので、シェル・スクリプトを書く

インストール

- `firemacs-0.1.xpi` を「ファイルで開く」で開けばよい
- インストールされる場所
  - プロファイル/`extentions/$GUID`
- プロファイルの場所
  - http://www.mozilla-japan.org/support/firefox/profile

ステップ3

- 開発し易くするための小技
- インストールの手間をなくす
- エラーを見る
- 値を表示する

プログラムの編集

- 開発手順
  - firemacs.js を編集する
  - パッケージを作る
  - Firemacs のインストール
  - Firefox の再起動
- 面倒だ～
- Firefox を再起動するだけでよくす方法
  - `chrome.manifest` を変更する
  - Firefox から開発環境を参照する

chrome.manifest を変更する

- プロファイル/extentions/$GUID の chrome.manifest を変更する
- 配布用

```
overlay chrome://browser/content/browser.xul #継続
        chrome://firemacs/content/firemacs.xul
content firemacs              jar:chrome/firemacs.jar!/content/
skin    firemacs classic/1.0 jar:chrome/firemacs.jar!/skin/
```

- 開発用

```
overlay chrome://browser/content/browser.xul #継続
        chrome://firemacs/content/firemacs.xul
content firemacs              chrome/content/
skin    firemacs classic/1.0 chrome/skin/
```

Firefox から開発環境を参照する

- プロファイル/extentions/$GUID の
  - install.rdf
  - chrome/

  を開発環境へのシンボリック・リンクにする

- 新たな開発手順
  - firemacs.js を編集
  - Firefox を再起動

※指摘

- シンボリック・リンクを使いにくい Windows だと chrome.manifest にパスを直接書くのがよい。

エラーを見る

- 「ツール」メニューのエラーコンソール
- 以下を true へ変えること!
  - javascript.options.showInConsole
  - javascript.options.strict

値を表示する

- 拡張機能のデバッグに Firebug は使えない
- alert() で頑張る ^^;
- Q) 他によい方法はありますか？
  - Console2 は便利でしょうか？

※指摘

- Firebug のコンソール機能が使える
- Venkman も使える
  - http://www.mozilla.org/projects/venkman/

ステップ4

- 少し複雑な Emacs コマンドを実装してみる
- リージョン (C-SPC, C-w)
- 文字の挿入 (C-m)
- less ライクなコマンド
- 検索 (C-s, C-r)
- Firemacs とは 1% の Google と
  99% の grep である

```
% grep -r -i KEYWORD $MOZILLA_DIR

% find . -name "*.js" -print0 | \
  xargs -0 grep -i KEYWORD
```

※指摘

- http://lxr.mozilla.org/mozilla1.8/ を使ってみては？

リージョン

- cmd_charNext に対し、
  cmd_selectCharNext があるらしい
  - 移動前と移動の後の間の文字列を自動的に選択する
- C-SPC でマークを打ったというフラグを立てる
- C-f
  - マークされているなら cmd_selectCharNext
  - マークされていないなら cmd_charNext
- C-w で cmd_cut を呼び出す

ステータス・バーへの表示

- C-SPC で "Set Mark" と表示したい
- 通常の JavaScript
  - `window.status = 'Set Mark'`
- 拡張機能では利用できない
  - ひたすら grep し、ついに発見！

```
display_message = function(msg, time) {
    var statusbar = document.getElementById('statusbar-display');
    if (!statusbar) return;
    statusbar.label = msg;
    if (time) setTimeout(display_message, time, '', 0);
}
```

文字の挿入

- C-m で RET を挿入したい
  - goDoCommand では、呼び出すコマンドに引数を渡せない
- Google のお告げ
  - http://kb.mozillazine.org/Inserting_text_at_cursor
  - すでに内容が書き変わっている orz

```
insert_text = function(aText) {
  var command = 'cmd_insertText';
  var controller = document.commandDispatcher.getControllerForCommand(command);
  if (controller && controller.isCommandEnabled(command)) {
    controller = controller.QueryInterface(Components.interfaces.nsICommandController);
    var params = Components.classes['@mozilla.org/embedcomp/command-params;1'];
    params = params.createInstance(Components.interfaces.nsICommandParams);
    params.setStringValue('state_data', aText);
    controller.doCommandWithParams(command, params);
  }
}
```

- XPCOM(cross platform component object model) を使っているらしい
  - C++ のモジュールを JavaScript から利用できるようになる

less ライクなコマンド

- 「表示のみの領域」と「編集可能領域」が区別できるようになった

```
function isWritable() {
  var command = 'cmd_insertText';
  var dspt = document.commandDispatcher;
  var controller = dspt.getControllerForCommand(command);
  return (controller && controller.isCommandEnabled(command));
}
```

- 表示のみの領域では less ライクなコマンドを割り当てる
  - 'j' → `cmd_scrollLineDown`
  - 'k' → `cmd_scrollLineUp`

ページ内検索

- これもひたすら grep!

- C-s

```
// 検索バーを表示
gFindBar.onFindCmd();

// 検索文字列があれば検索する
var findField = document.getElementById('find-field');
if (findField && findField.value && findField.value != '')
  gFindBar.onFindAgainCmd();
```

- C-g

```
// 検索バーを隠す
gFindBar.closeFindBar();
```

ステップ5

- Gmail、Vox、Wiki に勝つ
- Gmail と Vox
  - 独自の HTML エディタを持つ
  - C-b で太字になってしまう...
  - どうやら、イベント・リスナーを使っているらしい
- Wiki
  - どうやら XHTML で accesskey を定義しているらしい

Gmail と Vox に勝つ

- イベントのバブルアップ
  - イベントはフォーカスのあるノードから
    親の方向に向かって順に渡されていく
  - Gmail と Vox は HTML エディタにイベント・リスナーを
    設定
  - Firemacs は `window` にイベント・リスナーを設定
  - 負けるのは当然
- `addEventListener` の第三引数を `true` にすると、
  上位のノードがイベントを横取りできる

```
window.addEventListener("keypress", mozless_onkeypress, true);
```

  - 他のノードへイベントが渡らないようにする

```
event.stopPropagation();
event.preventDefault();
```

Wiki に勝つ

- DOM から AccessKey を取り除く
- 今扱っているページのルートノードは
  - `getBrowser().contentDocument.body`
- 木構造を再帰的に渡り歩く

```
function killAccessKey(node) {
  if (node.nodeType == node.ELEMENT_NODE ||
      node.noteType == node.DOCUMENT_NODE) {
    var accesskey = node.getAttribute('accesskey');
    if (accesskey) {
      var clone = node.cloneNode(true);
      clone.removeAttribute('accesskey');
      node.parentNode.replaceChild(clone, node);
    }
    if (node.localName == 'FRAME' || node.localName == 'IFRAME')
      node = node.contentDocument;
    if (node.hasChildNodes()) {
      var children = node.childNodes;
      var length = children.length;
      for (var i = 0; i < length; i++)
        killAccessKey(children[i]);
    }
  }
}
```


※指摘

- XPATH を使う方が速いだろう

ステップ6

- キー割り当ての機能を実装してみる
- キー割り当てモジュール
  - キー割り当てウインドウの作成
  - プリファレンスの保存
- Firemacs
  - プリファレンスの読み込み
  - キー割り当ての反映

キー割り当てウインドウの作成

- `install.rdf` へ以下を追加

```
<em:optionsURL>chrome://firemacs/content/config.xul</em:optionsURL>
```

- `chrome/content/config.xul` の作成
  - http://developer.mozilla.org/ja/docs/XUL_Tutorial

```
<script src="chrome://firemacs/content/config.js"/>

<row>
  <textbox id="NextChar" value="C-f"/>
  <description value="to the next char"/>
</row>


<button label="Save" oncommand="Config.save(); window.close();" />
```

プリファレンスの保存

- chrome/content/config.js
  - Config.save

```
Config.pref = Components.classes['@mozilla.org/preferences;1']
              .getService(Components.interfaces.nsIPrefBranch);

var elt = document.getElementById('NextChar');
var str = elt.value;
var pref_name = 'Firemacs.NextChar';
Config.pref.setCharPref(pref_name, str);
```

プリファレンスの読み込み

- `getCharPref` を使うと
  コマンド名とキー割り当ての対応が分かる
  - `NextChar` → `C-f`
  - Firemacs の初期化の際に読み込む
- キー割り当てからコマンド名への DB を作る
  - `C-f` → `NextChar`
  - これでキーから Emacs の編集コマンドを呼び出せる
- この仕組みでは Firefox を再起動しないと
  キー割り当てが反映されない

キー割り当ての動的な変更

- キー割り当てウインドウから Firemacs へ変更を伝えたい
  - 異なるコンテキスト間で情報を伝達できるか？
- 両方ともプレリファレンスを参照できる
  - 変更もプリファレンスを使う
- キー割り当てウインドウ
  - `Config.save`

```
Config.pref.setBoolPref('Firemacs.changed', true);
```

- Firemacs
  - イベントを拾うたびに `Firemacs.changed` を検査
  - `true` になっていたら、プリファレンスを読み込んで、DB を再構築

※指摘

- Preference Observer を使ってみては？

整合性の保証

- 整合性の問題
  - コマンドを追加するたびに、Firemacs と
    キー割り当てウインドウを変更しなければならない
  - 変更忘れが起こる
- XML データベースから自動生成する
  - コマンド名
  - コマンド (JavaScript のコード)
  - デフォルトのキー
  - 説明
- `firemacs.js`
  - 「コマンド名」「コマンド」「デフォルトのキー」から生成
- `Config.xul`
  - 「コマンド名」「デフォルトのキー」「説明」から生成

おしまい

要望

- 矢印キーでも移動やリージョン管理ができる
  ようにしてほしい