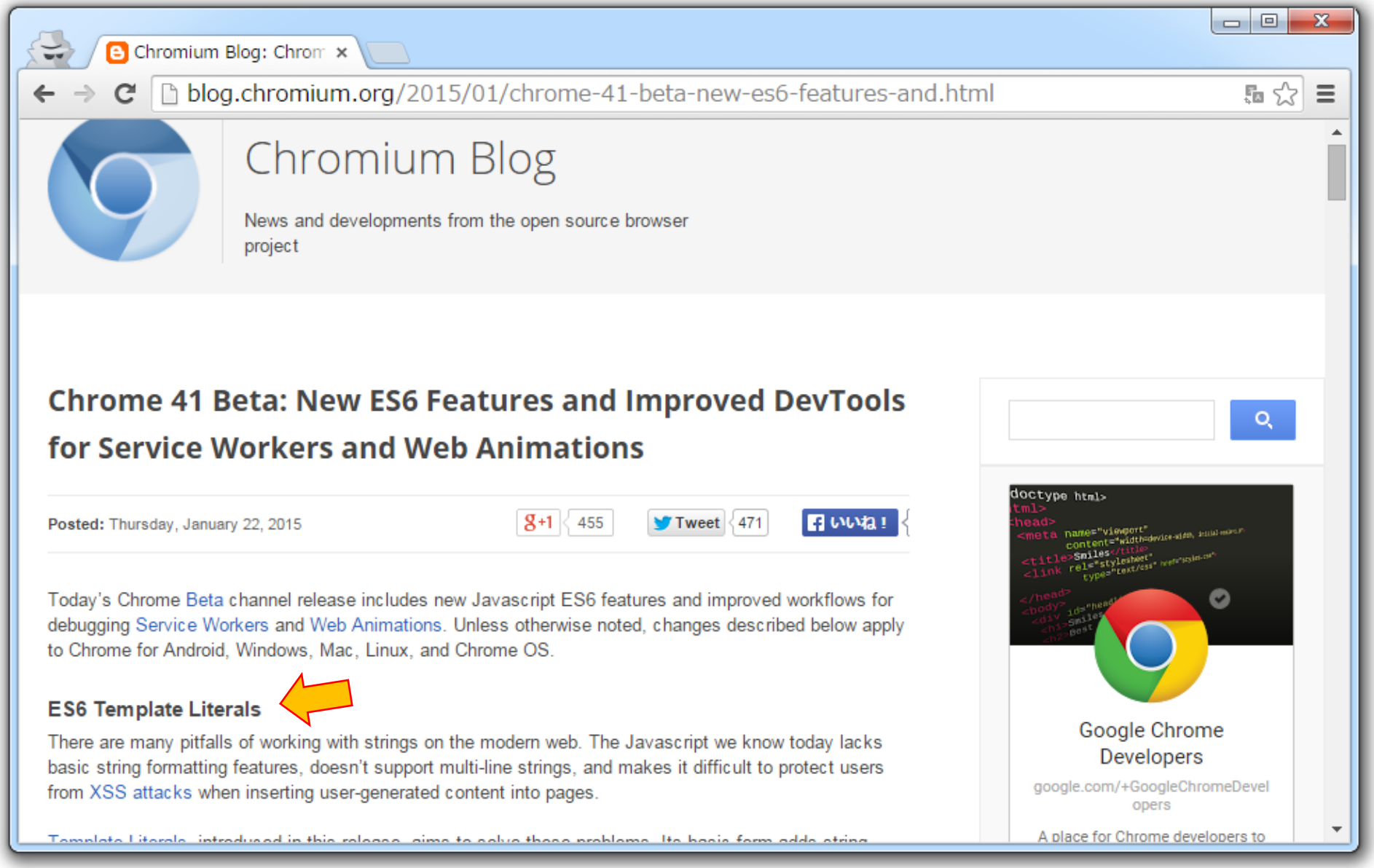
XSS対策としての ES6テンプレートリテラル

Shibuya.XSS
Yosuke HASEGAWA

ES6テンプレートリテラル



ES6テンプレートリテラル

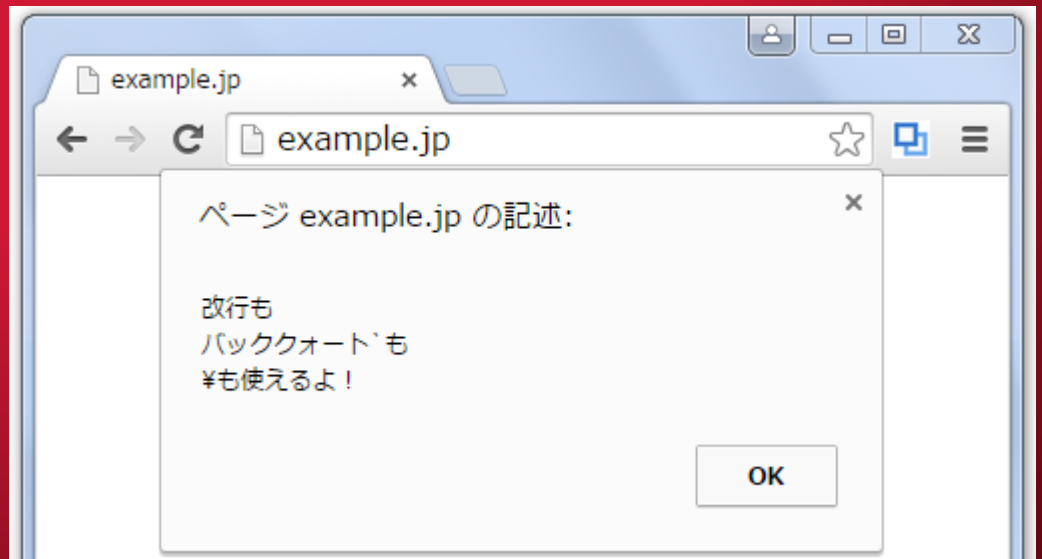
❤バッククォートで囲って改行も含められる

```
var x = `改行も
ダブルクォート「"」も
シングルクォート「'」も
使えるよ！`;
alert( x );
```

ES6テンプレートリテラル

- ♥ヒアドキュメントというには少し残念
  - ♥円記号でのエスケープが生きてる…

```
var x = `改行も￥nバッククォート￥`も
￥￥も使えるよ！`;
alert( x );
```



ES6テンプレートリテラル

❤リテラル内に${...}で埋め込んだ式を評価

```
var name = "hasegawa";
var x = `Hello, ${name}-san`;
console.log( x ); // "Hello, hasegawa-san"

x = `abc${
  (function(){ return "def"; })()
}`;
console.log( x ); // "abcdef"
```

ES6テンプレートリテラル

- ❤タグ関数 – テンプレート展開の際に呼び出されるリテラル内に${...}で埋め込んだ式を評価

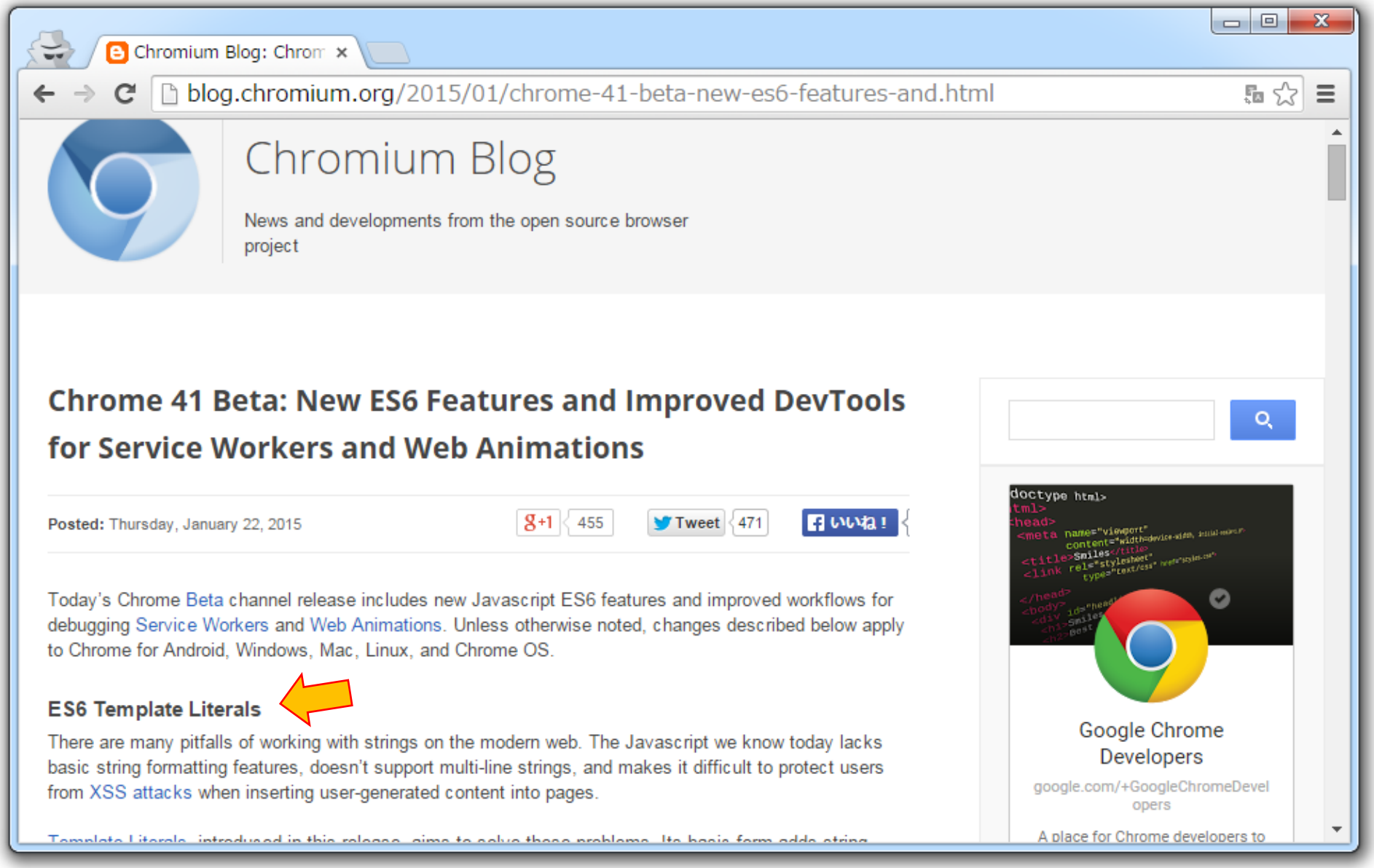
```
function foo( /* strings, ...values */ ){
    var strings = arguments[ 0 ];
    var values = [].slice.call( arguments, 1 );
    console.log( strings );     // [ "a¥nb", "cd", "" ]
    console.log( strings.raw ); // [ "a¥¥nb", "cd", "" ]
    console.log( values );      // [ 123, 456 ]
}

foo`a¥nb${123}cd${456}`;
```

ES6テンプレートリテラル

- タグ関数 – 「foo`string`」の形式で呼び出し可能
  - 「()」が不要

```
alert`xss`;
eval.call`${alert}``1`;
```

ES6テンプレートリテラル



ES6テンプレートリテラル

文字列を処理するのにはたくさんの落とし穴がある。…略…ユーザーの生成したコンテンツをページに挿入する際にXSSを防ぐのは難しい。…略…テンプレートリテラルにはXSSを防ぐためにHTMLをエスケープするのに便利なタグ付きテンプレートが導入される。

テンプレートリテラルでXSS対策

❤タグ関数でエスケープ

```
function escapeHtml( strings, ...values ){
    var result = strings[ 0 ];
    for( var i = 0; i < values.length; i++ ){
        result += values[i].replace( /&/g, "&amp;" )
            .replace( /</g, "&lt;" )
            .replace( />/g, "&gt;" )
            .replace( /"/g, "&quot;" )
            .replace( /'/g, "&#x27;" )
            + strings[ i + 1 ];
    }
    return result;
}

var name = "<img src='#' onerror='alert(1)'>";
elm.innerHTML = escapeHtml`<div>${name}</div>`;
```

テンプレートリテラルでXSS対策

❤タグ関数でエスケープ

```
function escapeHtml( strings, ...values ){
    var result = strings[ 0 ];
    for( var i = 0; i < values.length; i++ ){
        result += values[i].replace( /&/g, "&amp;" )
            .replace( /</g, "&lt;" )
            .replace( />/g, "&gt;" )
            .replace( /"/g, "&quot;" )
            .replace( /'/g, "&#x27;" )
        + strings[ i + 1 ];
    }
    r
}

var name = "<img src='#' onerror='alert(1)'>";
elm.innerHTML = escapeHtml`<div>${name}</div>`;
```

```
<div>&lt;img src=&#x27;#&#x27;
onerror=&#x27;alert(1)&#x27;&gt;</div>
```

テンプレートリテラルでXSS対策

- ❤HTMLテンプレート的な使い方ができる
- ❤ただしJSのコードとテンプレート部分が不可分
  - ❤あくまでもリテラル

```
var str = escapeHtml`<div>${name}</div>`;
elm.innerHTML = str;
```

テンプレートリテラルでXSS対策

♥あるいはeval等が必要

```
<template id="tmpl">
    <div>${name}</div>
</template>
<script>
    var t =
document.getElementById("tmpl").innerHTML;
    var name = "<img src='#' onerror='alert(1)'>";
    elm.innerHTML = eval( "escapeHtml`" + t + "`" );
</script>
```

♥「XSS対策」といいつつevalはない

♥CSP使いたいし

まとめ

- ❤テンプレートリテラルはHTMLテンプレートとは無関係
- ❤テンプレートリテラルをXSS対策に利用することは不可能ではないがあまり綺麗ではない
- ❤文字列置換ではない、DOM-basedなHTMLテンプレートエンジン欲しい…