Slide URL

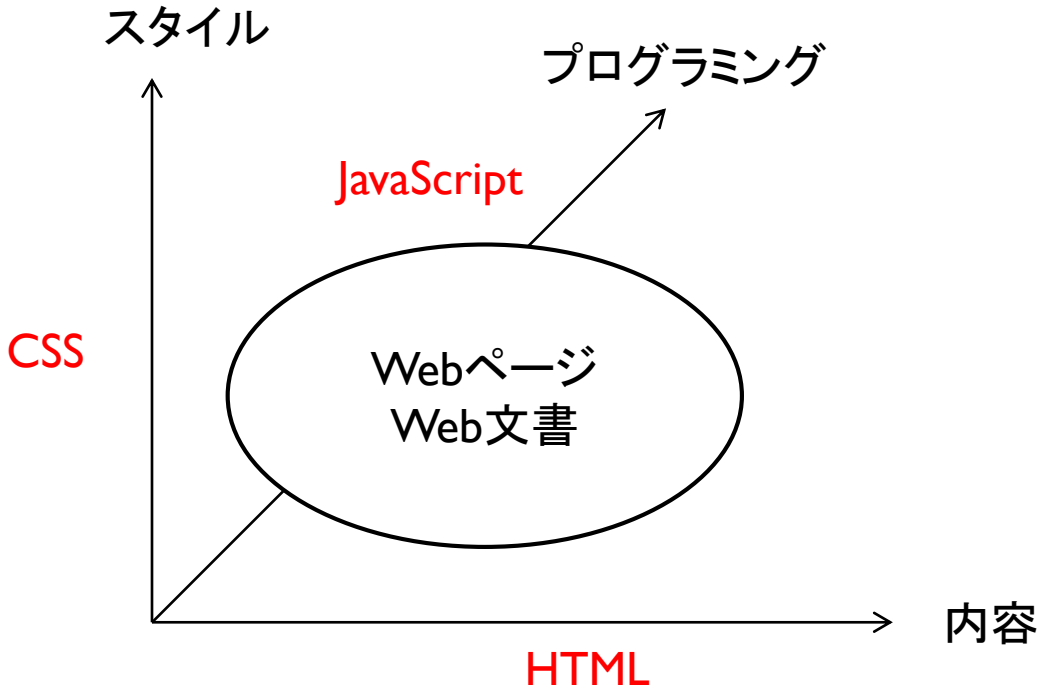
https://vu5.sfc.keio.ac.jp/slide/

Web情報システム構成法
第9回　JavaScript入門

萩野 達也（hagino@sfc.keio.ac.jp）

Webページの構成要素

- 直交技術を組み合わせる
  - 内容
  - スタイル（表現方法）
  - プログラミング

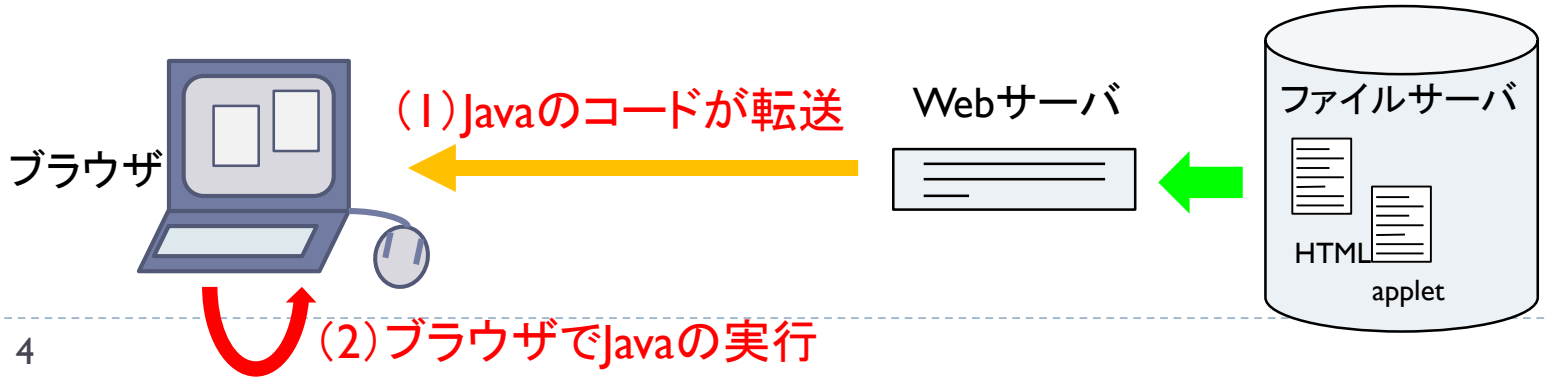


動きのあるページ

- HTMLおよびCSSは静的である.
  - 宣言的であり編集が容易
- 問題点
  - サーバがないと動作しない
  - フォームの入力のチェックはサーバに送られて初めて行われる
  - ページを移行せずに内容を変化させることができない
    - 天気の変化
    - ニュースのアップデート
  - ブラウザでのアニメーションなどがやりにくい
    - GIFアニメーションでは不足

Java Applet

- Java
  - 1995年にサンマイクロシステムズによって開発
  - 本格的なオブジェクト指向プログラミング言語
  - 組み込み機器などへの応用のために考えられた
  - Java仮想機械(JVM)を使うことで，コンパイル後のオブジェクトコードをさまざまなPCや機器で動作させることができる
- Java Applet
  - ブラウザの中でJavaのアプリケーションを動作させる
  - 最初Javaで書かれたHotJavaブラウザ上で動作した
  - Netscapeに組み込まれたことから普及
  - アニメーションなどができることで人気となる
  - セキュリティ的配慮からJava Appletは独立したもので，ブラウザや動作環境にアクセスすることが基本的にできない



JavaScript

- Java Applet問題点
  - ブラウザから独立している
    - HTMLやCSSとは独立
  - 画像のような扱い
- JavaScript
  - ネットスケープコミュニケーションズが開発
  - 1996年にInternet Explorer 3.0に搭載されて普及
  - HTMLやCSSにアクセスできる
  - 当初はブラウザ側での簡単なアニメーションやフォームの入力チェックや補助に利用できる
  - クライアント側での本格的なアプリケーションの開発も可能になった
    - Google Map
- 標準化
  - ECMAScript

JavaScript基本

- 構文
  - C言語をまねる（{ }によるブロック構造）
  - 無名関数（ラムダ式）あり
    - 関数クロージャを作ることができる
- データ
  - スクリプト言語なので静的な型チェックは行わない
  - 連想配列による構造体でデータ構造を作る
- プロトタイプベースのオブジェクト指向
  - 静的なクラスが存在しない
  - インスタンスベースとも呼ばれる
  - プロトタイプをコピーすることでオブジェクトが作られる
    - コピーすることで継承する
    - コピー後は新たなプロパティを自由に追加可能
  - クラスチェーンではなくプロトタイプチェーンが存在
    - オブジェクトのプロトタイプチェーンをたどることでメソッドを継承

JavaScriptの基本構文

- 変数宣言
  ```
  var 変数, 変数, … , 変数;
  var 変数 = 式;
  ```
  ```
  let 変数, 変数, … , 変数;
  let 変数 = 式;
  ```
  関数スコープ
  ローカルスコープ
- 代入(代入式)
  ```
  変数 = 式
  ```
- 条件文
  ```
  if (条件式) 文 else 文
  if (条件式) 文
  ```
- 繰り返し
  ```
  while (条件式) 文
  for (初期式; 条件式; 繰り返し式) 文
  do 文 while (条件式)
  ```
  特別な繰り返し
  イテレータ用
  ```
  for (変数 in 連想配列) 文
  for (変数 of イテレータ) 文
  ```
- ブロック(文のグルーピング)
  ```
  { 文; 文; … ; 文; }
  ```

JavaScriptの基本データ

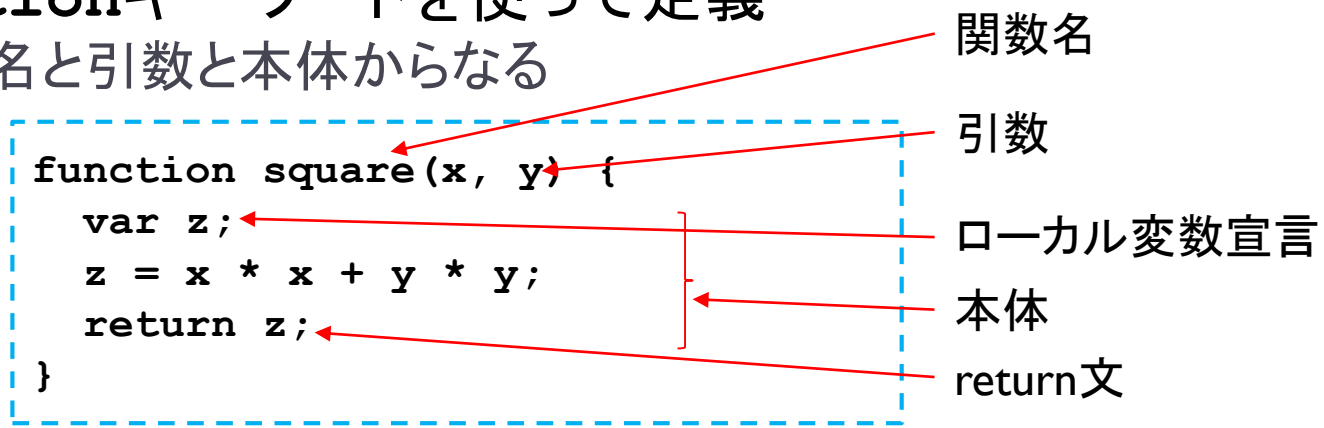
- 真偽値
  - trueとfalseのみ
- nullとundefined
  - 未定義の場合undefined
- 数字
  - 整数，浮動小数点
  - 2進，8進，16進表記も可能
- 文字列
  - 「'」あるいは「"」で囲まれた文字列
  - 「+」で文字列の結合が可能
    - 数字の加算との混在に注意
  - 文字列と数値の変換は自動的に行われる
- オブジェクト
  - オブジェクトは連想配列
  - 添え字（インデックス）と値の対の集合
  - 通常の配列も連想配列の特別な場合
    - 添え字が自然数
    - Arrayオブジェクト

```
var x = '333';
var y = '111';

x + y  ----->  "333111"
x - y  ----->  222
```

```
var obj = {width: 100, height: 80 };

var obj = {};
obj['width'] = 100;
obj.height = 80;

var a = [100, 80];

var b = {0: 100, 1: 80};
b.__proto__ = Array.prototype;

var c = new Array(100, 80);
```

関数の宣言

- **functionキーワードを使って定義**
  - 関数名と引数と本体からなる



- 無名関数(ラムダ式)
  - 名前を付けない関数
  - 値として使う

```
var square;

square = function (x, y) {
  var z;
  z = x * x + y * y;
  return z;
}
```

```
function counter() {
  var x = 0;
  return function () {
    return x++;
  };
}
```

オブジェクトの例

- Array
  - 配列
  - `new Array(…)` 以外に `[…, …]` でも生成
  - `length` プロパティが大きさを保持
  - `push`, `pop`, `shift` など配列操作のメソッドを保持
- Math
  - newすることはない
  - `Math.abs`, `Math.max`, `Math.random` など数値に関するメソッドを保持するオブジェクト
- RegExp
  - 正規表現
  - 正規表現による文字列の検索や置換などのメソッドを保持
  - `/ab+c/i` あるいは `new RegExp('ab+c', 'i')` で生成
- Object
  - すべてのオブジェクトの元
  - プロトタイプチェーンの一番上流にある

JavaScriptオブジェクトの仕組み

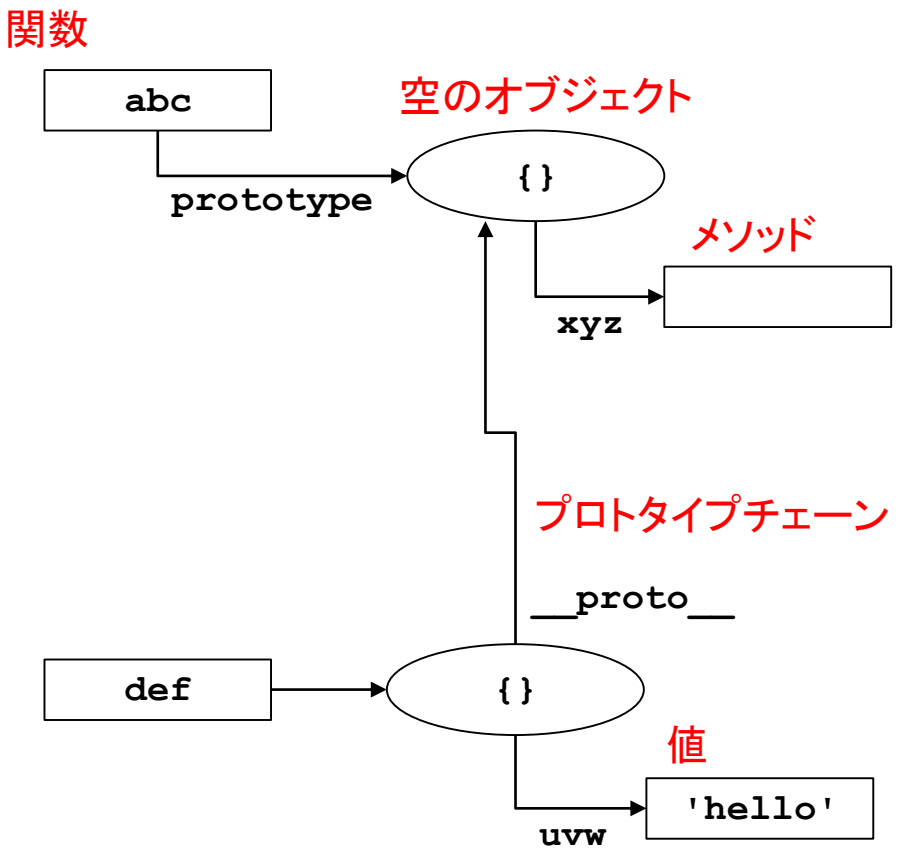
```
function abc(x) { … }
```

```
abc.prototype.xyz =
  function(y) { … };
```

```
var def = new abc(123);
```

```
def.xyz(567);
```

```
def.uvw = 'hello';
```



HTMLとJavaScript

- script 要素
  - HTMLへのJavaScriptの埋め込み
  - head内およびbody内で利用できる
  - defer属性がない場合には，即実行される
    - deferが指定されると文書をすべて読み終えてから実行される

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>JavaScriptの実験</title>
    <script src="myscript.js"></script>
  </head>
  <body>
    <h1>JavaScript</h1>
    <script>
      window.alert("Hello, World!");
    </script>
    <p>テスト</p>
    <script src="mysecond.js"></script>
  </body>
</html>
```

外部のJavaScriptファイルを参照

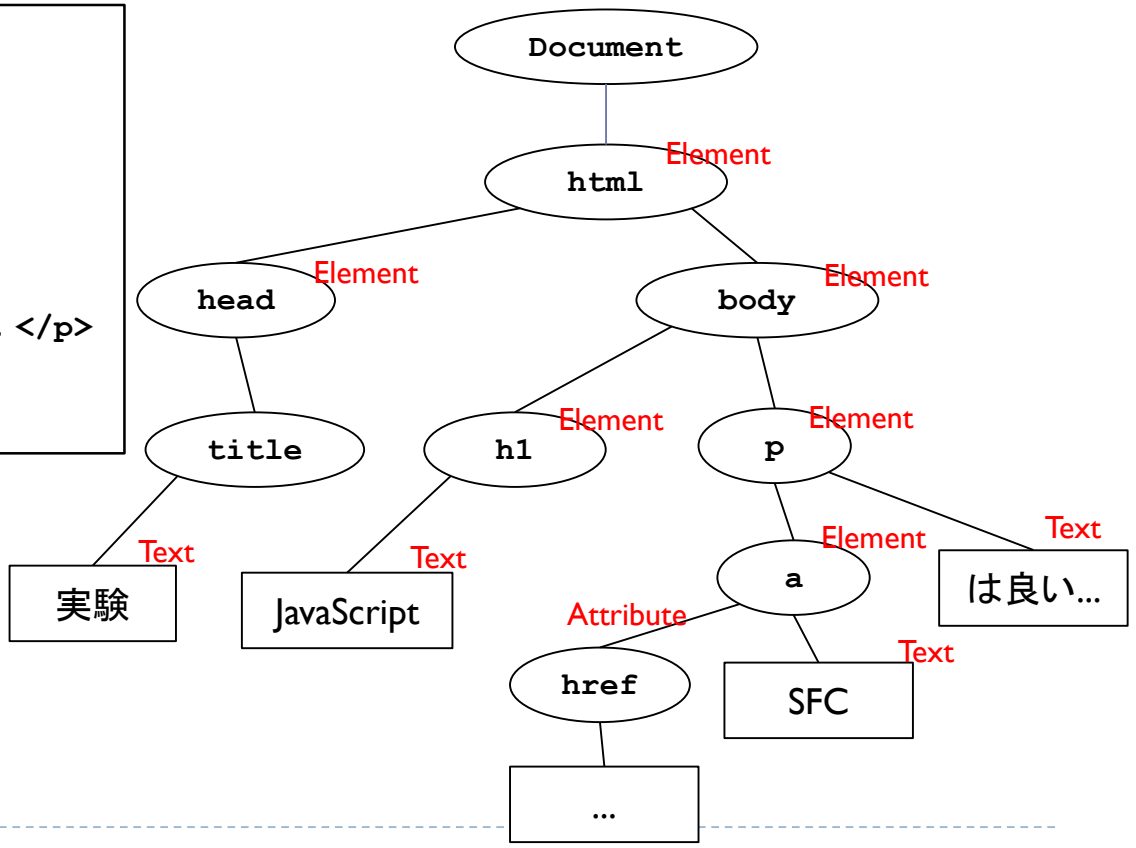
JavaScriptを直接埋め込む

<script や </script や </ を内部に含んではいけない

DOM

- Document Object Model
  - HTMLはブラウザで構文解析（パース）され，オブジェクトの木構造となる．

```
<!DOCTYPE html>
<html>
  <head>
    <title>実験</title>
  </head>
  <body>
    <h1>JavaScript</h1>
    <p><a href="...">SFC</a>は良いところ. </p>
  </body>
</html>
```



documentオブジェクト

- documentオブジェクト
  - HTMLページを表すオブジェクト
  - DOMの必要な要素を取得することが可能

メソッド	説明
`document.getElementById(id)`	idにより要素を取得
`document.getElementsByTagName(tagName)`	要素名での要素の集合の取得
`document.getElementsByClassName(className)`	class名での要素の集合の取得

- 要素の内容・属性の変更

`element.innerHTML = "content";`	要素の内容の変更
`element.textContent = "content";`	要素のテキスト内容の変更
`element.attribute = "value";`	属性の変更
`element.setAttribute("attribute", "value")`	属性の変更
`element.style.property = "value";`	スタイル属性の変更

- 要素の追加変更

`document.createElement(tagName)`	新しい要素を生成
`document.createTextNode(text)`	テキストノードを作成

要素の挿入・削除

- document.createElement で作成した要素を挿入

メソッド	説明
parent.appendChild(*element*)	要素を親要素の最後に挿入
parent.insertBefore(*sibling*, *element*)	指定要素の前に挿入

- 要素の削除・置き換え

parent.removeChild(*element*)	親要素から指定された子要素を削除
parent.replaceChild(*old*, *element*)	親要素から指定された子要素を置き換える

- DOMノード関係のプロパティ

element.childNodes	すべての子要素（ノード）のリスト
element.firstChild	最初の子要素（ノード）
element.lastChild	最後の子要素（ノード）
element.parentNode	親要素（ノード）

例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>test for JavaScript</title>
  </head>
  <body>
    <h1>JavaScript</h1>
    <p id="id1">JavaScriptは難しい. </p>
    <p id="id2"></p>
    <script>
      var p1 = document.getElementById("id1");
      var t1 = document.createTextNode("いや簡単. ");
      p1.appendChild(t1);

      var p2 = document.getElementById("id2");
      p2.textContent = "ためしてみよう. ";
      p2.style.color = "red";
      p2.style['text-align'] = "center";

      var plist = document.getElementsByTagName('p');
      for (var i = 0; i < plist.length; i++) {
        plist[i].style['font-size'] = '20px';
      }
    </script>
  </body>
</html>
```

要素を探し出し
テキストノードを追加

要素を探し出し
テキストの変更
スタイルの変更

複数の要素を探し出し
スタイルの変更

課題：JavaScriptでページを作成

- 作っている自分が好きな海外の街を紹介のトップページと同じものをJavaScriptを使って作成しなさい.
  - innerHTMLは利用しないこと
  - CSSのスタイルもJavaScriptで指定すること

- 提出
  - URLではなくHTML（JavaScript）を提出
  - JavaScriptはHTMLに埋め込むこと

towm-js.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>海外の好きな街</title>
  </head>
  <body id="body">
   <script>
      var body = document.getElementById("body");
      var h1 = document.createElement('h1');
      h1.textContent = "好きな街";
      body.appendChild(h1);
      ....
   </script>
  </body>
</html>
```

まとめ

- JavaScript
  - 基本構文
  - 基本データ
  - オブジェクト
    - 連想配列
  - プロトタイプベース

- DOM
  - DOM木
  - documentオブジェクト