ネットワークアプリケーション

第8回 クライアントサイドウェブプログラミング（1）

石井 健太郎
（423研究室・オフィスアワー水3限）

スケジュール

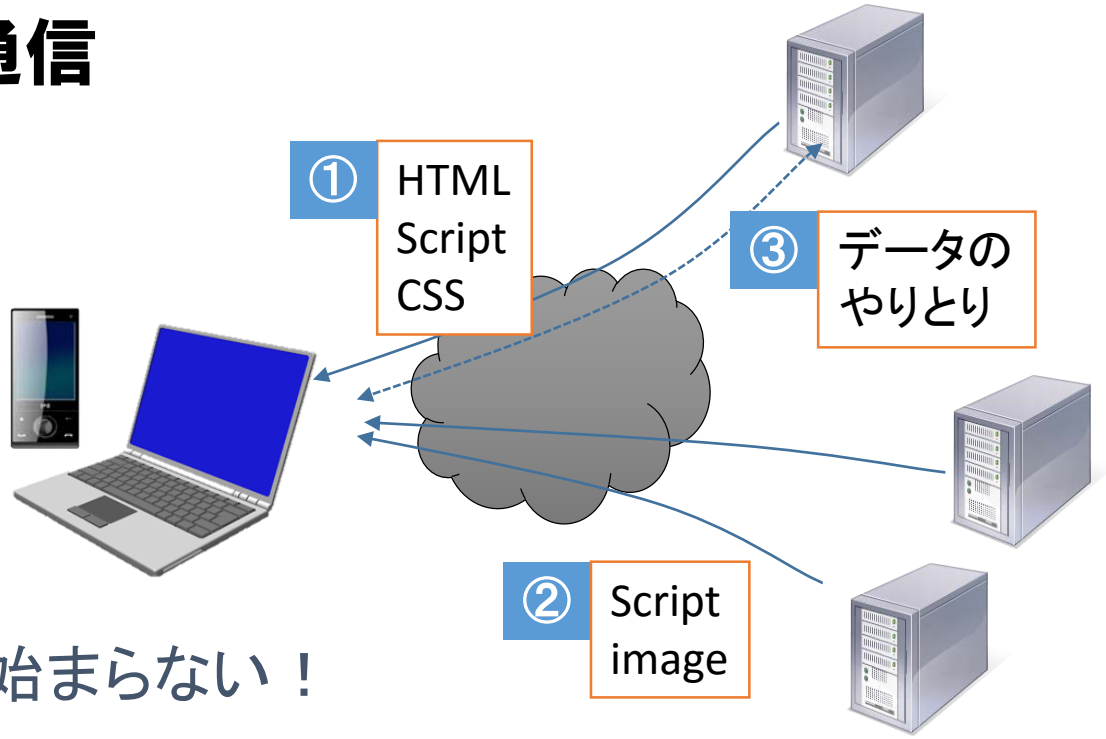
9月15日　第1回「TCP/IPプロトコルスイート」
9月29日　第2回「ネットワークアプリケーションのプログラミングモデル」
10月6日　第3回「アプリケーションプロトコルの設計（1）」
10月13日　第4回「アプリケーションプロトコルの設計（2）」
10月20日　第5回「アプリケーションプロトコルの設計（3）」　演習（第3演習室）
10月27日　第6回「アプリケーションプロトコルの設計（4）」　演習（第3演習室）
11月10日　第7回「ウェブプログラミングについて」
11月17日　第8回「クライアントサイドウェブプログラミング（1）」

スケジュール

11月24日	第9回「アプリケーションプロトコルの設計（5）」	演習（第3演習室）
12月1日	第10回「クライアントサイドウェブプログラミング（2）」	演習（第3演習室）
12月8日	第11回「クライアントサイドウェブプログラミング（3）」	
12月15日	第12回「クライアントサイドウェブプログラミング（4）」	
12月22日	第13回「クライアントサイドウェブプログラミング（5）」	演習（第3演習室）
1月12日	第14回「クライアントサイドウェブプログラミング（6）」	演習（第3演習室）
1月19日	第15回「まとめと演習」	演習（第3演習室）

JavaScriptの実行モデル

1. サーバからHTMLが配布されてブラウザ上で実行
2. 追加のスクリプトや画像を必要に応じロードされ実行
3. 必要に応じてサーバと通信



HTMLファイルなしには始まらない！

JavaScriptの特徴

- ウェブブラウザ上で実行するデファクトスタンダードの言語
- 様々な端末（PCもスマートフォンも）に対応できる
- 実行環境を端末にインストールしなくても実行できる
- ブラウザの更新にともないソフトウェアの更新される
- ネイティブアプリに変換することもできる
- HTML5により端末のグラフィック機能が自在に使えるようになった
- ライブラリが急速に充実
  - ソースが見えるので、良いコードを開発者がマネしやすい（技術の伝搬が早い）

マニュアル

- Mozilla Developer Networkの資料がおすすめ

- https://developer.mozilla.org/ja/docs/Web/JavaScript

JavaScript開発環境

- Dreamweaver
- Atom Editor
- jsdo.it

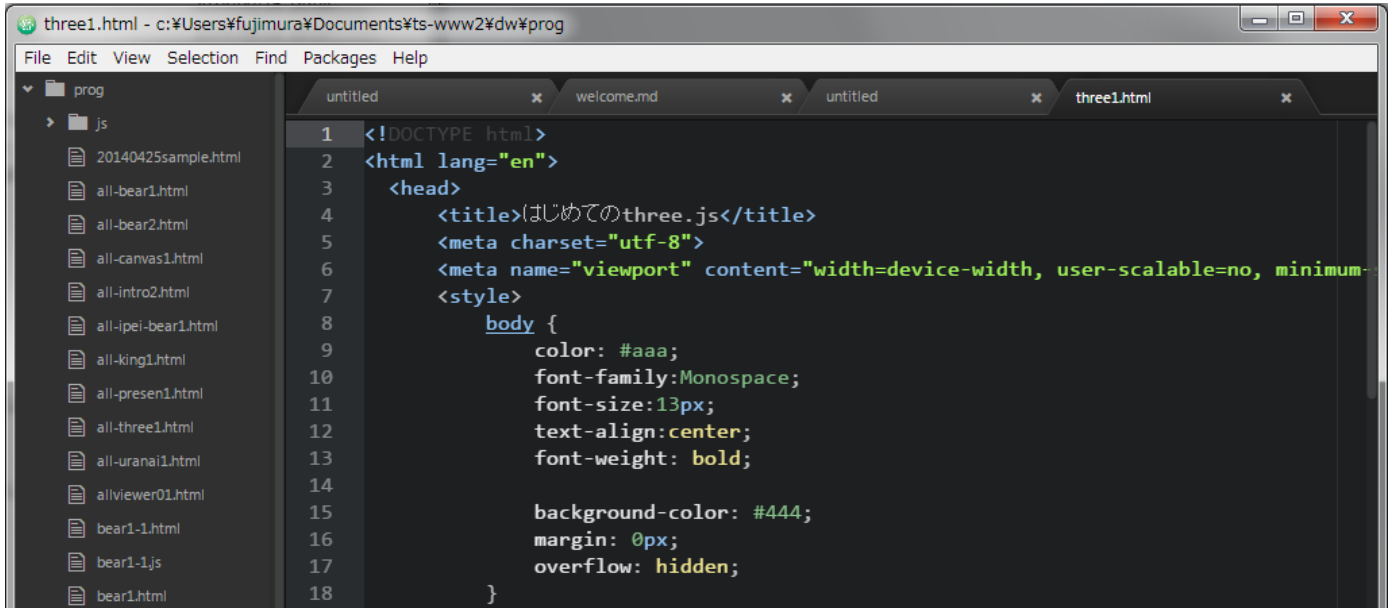
- Microsoft Visual Web Studio
- Eclipse JSDT
- Aptana Studio
- Firefox with Firebug + エディタ(Emacs等)

Dreamweaver

- CSSのデザインのサポートが強力
- PhotoshopやIllustrator等の連携がしやすい
- JavaScirptの入力支援機能
- Webサーバへのコンテンツのアップロードが容易
  - １クリックでサーバと同期できる
- デメリット
  - 機能が多すぎて少し難しい
  - 高価（自宅では使えない）

Atom Editor

- 高機能で無料の注目のエディタ
- https://atom.io/
- 使いやすくカスタマイズできる



jsdo.it

- JavaScriptで書かれたJavaScriptのための開発環境
- ブラウザでプログラムを書く

http://jsdo.it

Visual Web Studio

- Microsoftの開発環境

  http://www.microsoft.com/japan/msdn/vstudio/express/

結論

- 授業ではDreamweaverを採用
- 自宅でプログラミングを行う場合には，
  Atom Editor または jsdo.it を利用するとよい

JavaScript

- C言語と異なるところ
  - インタープリタ言語
  - コンパイルをあらかじめ行わない
    （実行時にプログラムの意味を解釈し、実行される）
  - 変数の宣言時に変数の型（char, string, int, float, double等）を指定しない
    - 変数の宣言自体は必要
      - すべての変数の宣言はvarによって行う
  - 型が自動的に変換される（楽だがミスがわかりにくい）
  - イベントドリブン
    （同時に複数のプログラムコードの箇所が並行して動く）

データ型

- 文字列
- 数値
- 真偽値(true / false)
- オブジェクト
  - 配列
  - 関数
  - 組み込みオブジェクト
- undefined 定義されていない
- null なにもない

数値と演算子

- **数値**
  10
  2.5
  -2.5
- **演算子**
  + - * / %
  += -= *= /=
  ++ --

【注意】+ は「文字列の結合の演算子」でもある。左右のどちらかが文字のとき、文字に変換される

```
var x;
x = "1" + 2;
alert(x);
x = +"1" + 2;
alert(x);
```

のように単項演算子として使うと文字列を数値に変換する関数として使うこともできる

文字列

- 表現方法と特殊文字
- 「”」か「’」で囲む

```
x = ”hello world”;
x = ’hello world’;
x = ”it’s a pen”;
x = ’it¥’s a pen’;
```

- エスケープ文字は「¥」(円マーク)
- 特殊文字の改行やタブは「¥n」「¥t」

制御構造

- if文 C言語と同じ
- for文 C言語と同じ（変数の宣言はvarに変わる）

```
for (var i=0; i<20; i++) {
  console.log(i);
}
```

- while文、do while文　　C言語と同じ
- break文、continue文　　C言語と同じ
- switch文
- 条件分岐の中にある比較演算子は二種類ある

=== !==　型も含めて一致するか比較

== !=　異なる型の場合には変換してから比較

関数

- C言語と異なり、返り値の型の宣言は不要
- その代わり、functionというキーワードを書く
- 引数の型宣言も不要

```
var x = 100;
var y = 200;

function max(x, y) {
  if (x>=y) return x; else return y;
}

alert(max(x,y));
```

標準的な書き方

```
var x = 100;
var y = 200;

var max = function (x, y) {
  if (x>=y) return x; else return y;
}

alert(max(x,y));
```

このような書き方もできる

- 来週11月24日（火）は**第3演習室**に集まってください