

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

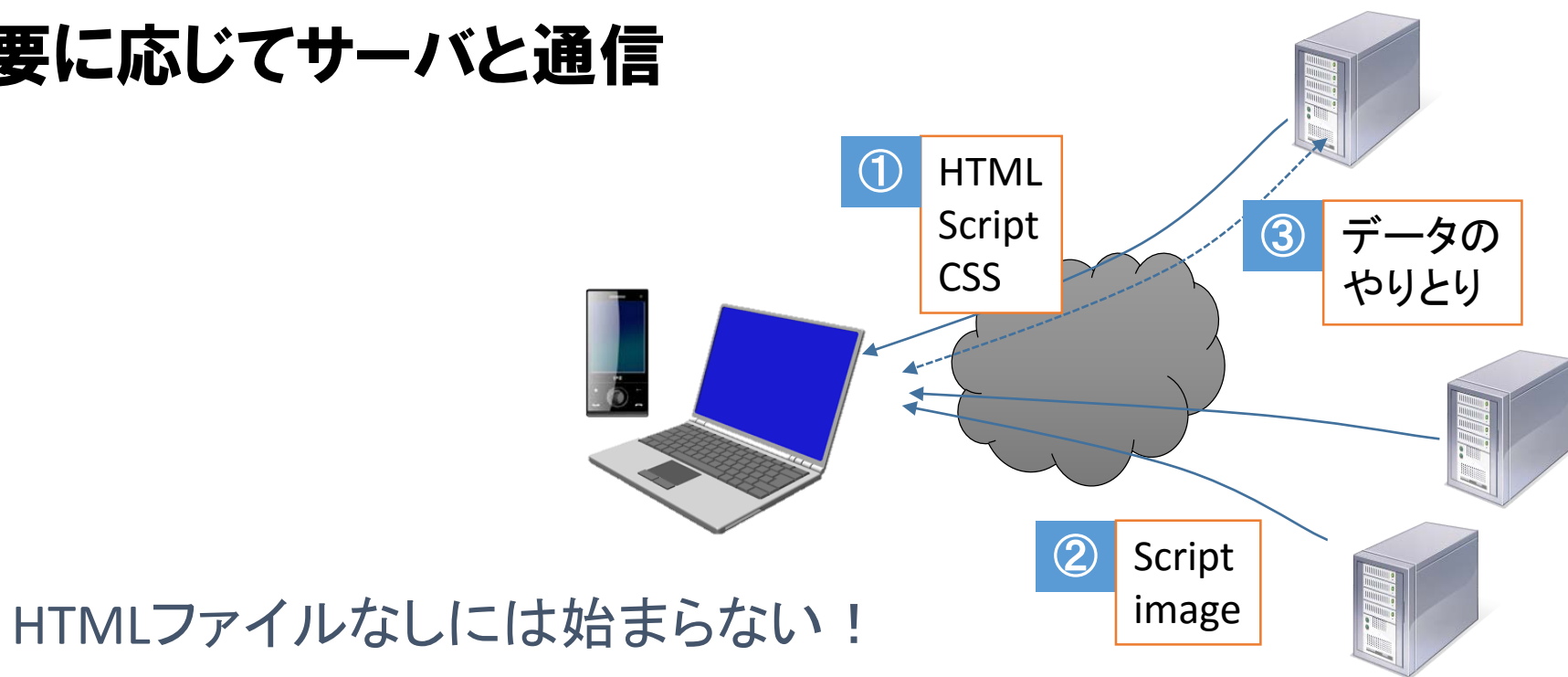
- 9月15日 第1回「TCP/IPプロトコルスイート」
- 9月29日 第2回「ネットワークアプリケーションのプログラミングモデル」
- 10月6日 第3回「アプリケーションプロトコルの設計(1)」
- 10月13日 第4回「アプリケーションプロトコルの設計(2)」
- 10月20日 第5回「アプリケーションプロトコルの設計(3)」 **演習(第3演習室)**
- 10月27日 第6回「アプリケーションプロトコルの設計(4)」 **演習(第3演習室)**
- 11月10日 第7回「ウェブプログラミングについて」
- 11月17日 第8回「クライアントサイドウェブプログラミング(1)」

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



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

MDN > Web technology for developers > JavaScript

言語 編集 設定

JavaScript

JavaScript® (しばしば JS と略される)は軽量なオブジェクト指向言語です。Web ページで使用されるスクリプト言語として最もよく知られていますが、[ブラウザ以外の多くの環境においても使用されています。](#) (JavaScript についてさらに詳しく読む。)

JavaScript「再」入門
世界で最も誤解されているであろうプログラミング言語「JavaScript」の本当の姿

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

JavaScript開発環境

- Dreamweaver
- Atom Editor
- jsdo.it

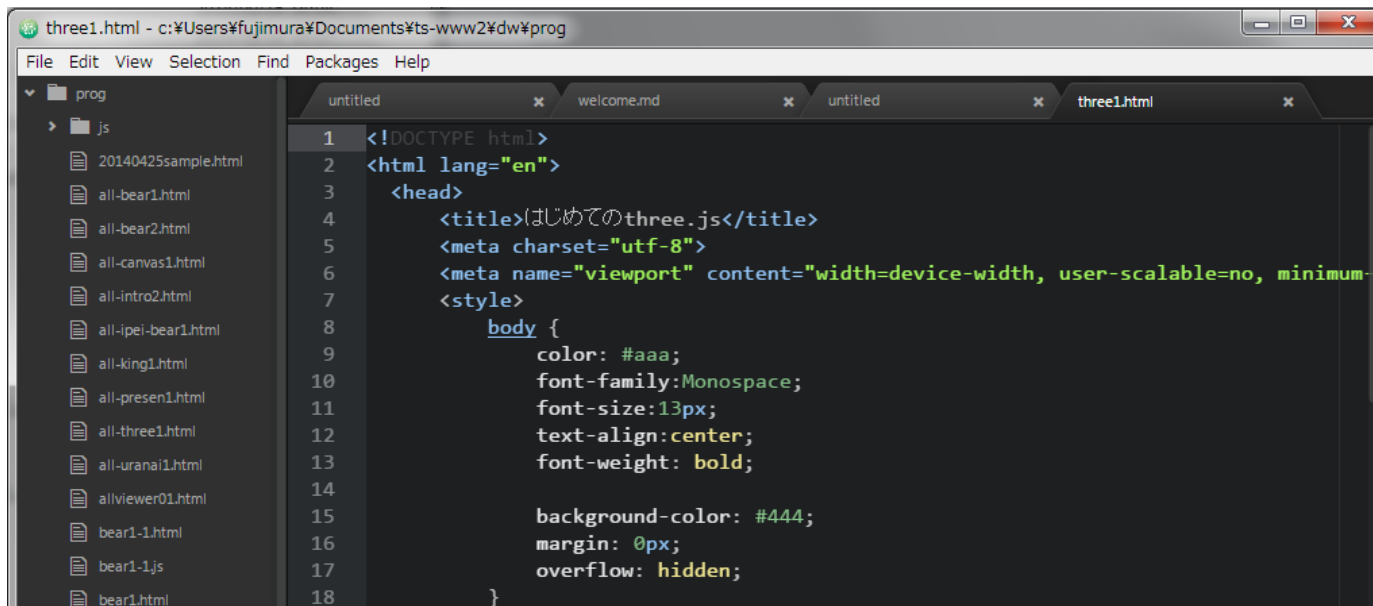
-
- Microsoft Visual Web Studio
 - Eclipse JSDT
 - Aptana Studio
 - Firefox with Firebug + エディタ(Emacs等)

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**
- **JavaScript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



```
three1.html - c:\Users%fujimura\Documents\ts-www2\dw\prog
File Edit View Selection Find Packages Help
└─ prog
  └─ js
    └─ 20140425sample.html
    └─ all-bear1.html
    └─ all-bear2.html
    └─ all-canvas1.html
    └─ all-intro2.html
    └─ all-ipei-bear1.html
    └─ all-king1.html
    └─ all-presen1.html
    └─ all-three1.html
    └─ all-uranai1.html
    └─ allviewer01.html
    └─ bear1-1.html
    └─ bear1-1.js
    └─ bear1.html
  1 <!DOCTYPE html>
  2 <html lang="en">
  3   <head>
  4     <title>はじめてのthree.js</title>
  5     <meta charset="utf-8">
  6     <meta name="viewport" content="width=device-width, user-scalable=no, minimum-
  7     <style>
  8       body {
  9         color: #aaa;
 10        font-family:Monospace;
 11        font-size:13px;
 12        text-align:center;
 13        font-weight: bold;
 14
 15        background-color: #444;
 16        margin: 0px;
 17        overflow: hidden;
 18      }
```

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>

The screenshot displays the jsdo.it interface for a project titled "Recursive tree using raphael.js". The page includes a header with the user "blogranger", a "Follow" button, the date "2012-12-27 12:09:30", and the license "MIT License". There are also buttons for "ブラウザの通知を使用する: 設定", "Fav 0", "View 1164", and a red "Fork 2" button. Below the header, there are tabs for "Readme", "JavaScript 55 lines", "HTML 11 lines", and "CSS 24 lines". The main content area is split into two panes: the left pane shows the JavaScript code for a recursive tree, and the right pane shows a rendered tree with a blue sky and a dark ground. The code defines a function `tree` that recursively generates a tree structure using Raphael.js. The tree is rendered with a dark green trunk and branches, and small green leaves. The background is a light blue sky with a dark grey ground at the bottom.

```
16 // Definition of leaf shapes.
17 var leaf_style = ["#4e724c", "#9eb296", "#5e7153", "#142a1e", "#2f563b", "#244f33"];
18
19 Raphael.fn.leaf = function (x, y, scale, rotate) {
20   return this.path("M-4.6-3.7C-8.2-6.2-23-16-23.8-25.2s1.2-15.1,11.1-27.6s8.1-27.8,8.1-27.8s3.1,17.3
21 ");
22 };
23
24 function tree(x,y,scale,rotate,depth) {
25   if (scale < 0.02 || depth > 10) {
26     paper.leaf(x, y, scale, Math.floor(180*Math.random()));
27     paper.leaf(x, y, scale, Math.floor(180*Math.random()));
28     return 0;
29   } else {
30     var scale1 = 1 - Math.exp(-0.02 * Math.min(y-1.5*x+500,y+1.5*x-700));
31     scale = scale * scale1;
32     var s1 = scale * (0.5 + 0.6 * Math.random()); // Growth ratio of right branch
33     var s2 = scale * (0.5 + 0.6 * Math.random()); // Growth ratio of left branch
34     var r1 = rotate + Math.random()*30 + depth; // Rotation of right branch
35     var r2 = rotate - Math.random()*30 - depth; // Rotation of left branch
36     var x1 = x + s1 * BRANCH_LENGTH * Math.sin(r1 * Math.PI / 180.0);
37     var y1 = y - s1 * BRANCH_LENGTH * Math.cos(r1 * Math.PI / 180.0);
38     var x2 = x + s2 * BRANCH_LENGTH * Math.sin(r2 * Math.PI / 180.0);
39     var y2 = y - s2 * BRANCH_LENGTH * Math.cos(r2 * Math.PI / 180.0);
40     depth++;
41     paper.branch(x, y, 0.5*s1, r1).animate({opacity:1.0},0,function () {tree(x1, y1, s1, r1, depth
42 // Next level of right branch
43     paper.branch(x, y, 0.5*s2, r2).animate({opacity:1.0},0,function () {tree(x2, y2, s2, r2, depth
44 // Next level of left branch
45   }
46 }
47
```

Visual Web Studio

- Microsoftの開発環境

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



Microsoft
Visual Web Developer 2010
Express

Microsoft Web Platform Installer (Web PI) は、Visual Web Developer や IIS などの Microsoft Web Platform の最新コンポーネントのダウンロード、インストール、最新版への更新が簡単に可能になる無償ツールです。さらに、人気のあるオープンソースの ASP.NET と PHP Web アプリケーションをインストールします。

» [Web PI のダウンロード ページへ](#)

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
- その代わりに、**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