JavaScript (2)

二宮 崇
東京大学 情報基盤センター 図書館電子化研究部門 講師

MBC 2008年12月18日

JavaScriptマスターへの道

1. まず、クライアント側のプログラムをいろいろと書けるようになろう!
    1. 構文
    2. オブジェクト指向
    3. DOM
    4. イベント
2. サーバー側も用意してデータをやりとりしながら何か面白いものを作ろう
3. Ajax

お薦めの本

・Shelley Powers, 武舎広幸,武舎るみ「初めてのJavaScript」オライリー

・David Flanagan, 村上列「JavaScript」第5版 オライリー

構文 (1)

- **変数の宣言**

  var a;
- **演算**
  - **代入** a = 8;
  - **足し算** a = 3 + 4;
  - **引き算** a = 8 – 3;
  - **かけ算** a = 8 * 2;
  - **割り算** a = 8 / 2;
  - **余り** a = 8 % 3;
  - 1**足す** a++; **もしくは**++a; (++**の位置によって返値が違うので注意。**var b = ++a;**だと**b**は**a+1。var b = a++**だと**b**は**a。)
  - 1**減らす** a--; **もしくは**--a;
  - **負の値** var b = -a;
  - **演算付き代入** a += 8; a -= 3; a *= 4; a /= 2; **それぞれ**a = a + 8; a = a – 3; a = a * 4; a = a / 2;**とおなじ**

構文 (2)

- if(**条件**) { A } else { B }
  - **条件を満たしていればA**を実行
  - **そうでなければB**を実行
- if(**条件**1) { A } else if (**条件**2) { B } else if(**条件**3) { C } else { D }
  - **条件1を満たす→A**を実行
  - **そうでない場合で条件2を満たす→B**を実行
  - **さらにそうでない場合で条件3を満たす→C**を実行
  - **さらにさらにそうでない場合はD**を実行
- switch(**式**) {
  case **ラベル**1:
  …A
  [break;]
  case **ラベル**2:
  …B
  [break;]
  default:
  …C
  }
  - **式の内容がラベル1ならAを実行。ラベル2ならBを実行。どれでもない場合はC**を実行。break を入れないとAを実行した後にBもCも実行されてしまうので注意

構文 (3)

- while(条件) { A }
  - 条件を満たしている限りAのところを実行
- do { A } while(条件)
  - Aを実行して、条件を満たしていればまたAを実行
- for(A; 条件; B){ C }
  1. Aを実行
  2. 条件を満たしていればCを実行
  3. Bを実行2に戻る
- for(変数名 in オブジェクト) { .. }
  - 配列、連想配列やオブジェクトの全データにアクセスするのに便利

構文 (4)

- **比較**
  - **等価** if (a == 3) … # if(a = 3)**だと代入になります**
  - **不等価** if (a != 3)
  - **大なり** if (a > 3) **小なり** if (a < 3)
  - **大なりイコール** if(a >= 3) **小なりイコール** if(a <= 3)
- **論理演算**
  - AND if ((a > 3) && (a < 10))
  - OR if((a <3) || (a > 10))
  - **否定** if(!(a < 3))

データ型

- データ型
  - 単純型
    - 数値
      - 整数
      - 実数
    - 文字列
    - ブーリアン (true, false)
  - 定数 const HOGE = 3;
  - オブジェクト
  - null (その変数が定義されていない)
    - document.writeln(b); といきなり書くとエラーになる
    - If(x) とやってその変数が定義されているかどうか(=nullかどうか)チェックする
  - undefined (その変数が定義されているけど初期化されていない→undefinedと表示される)
    - var a; document.writeln(a);

関数

- **関数の宣言 (宣言型の関数)**

```
function plus(x, y) { return x + y; }
document.writeln(plus(3, 4));
```

- **変数に関数を格納できる (関数リテラル)**

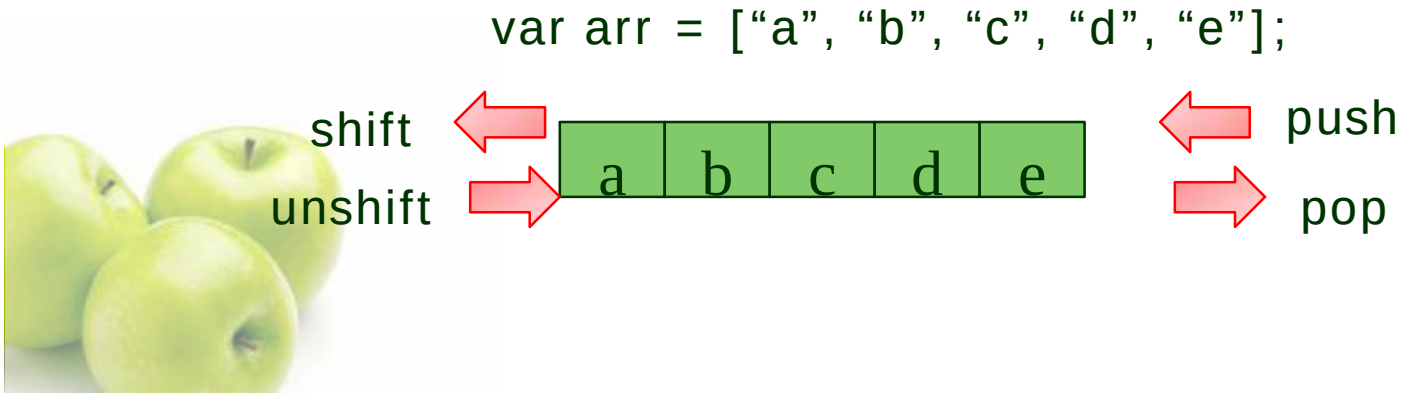
```
var f = function(x, y) { return x + y; }
document.writeln(f(5,6));
```

- **関数のオブジェクトもある (無名関数)**

```
var f = new Function("x", "y", "return x + y");
document.writeln(f(7,8));
```

配列

- var a = new Array(“one”, “two”);
- var a = [“one”, “two”];
- a.push(“three”);
- var elem = a.pop(); / / **配列の最後の要素を返す＆除く**
- Var len = a.length(); / / **配列の長さを返す**



オブジェクト　連想配列

- JavaScript**は正確にはオブジェクト指向言語ではない**
- JavaScript**での連想配列がオブジェクトに対応**

```
var edic = new Object();
edic[“orange”] =“みかん”;
edic[“apple”] = “りんご”;
document.writeln(edic[“orange”]);

var jdic = new Object();
jdic[“みかん”] = “orange”;
jdic[“りんご”] = “apple”;

var edic = {”orange”:“みかん”,”apple”:“りんご”};
document.writeln(edic.orange);
```

メソッド

```
var point = new Object();
point.x = 100;
point.y = 200;
point.move = function(newx, newy) { this.x =
   newx; this.y = newy; }
point.move(300,400);
```

- 連想配列(=オブジェクト)の”move”の値に関数が入っている
- thisはこの連想配列(=オブジェクト)自身を指す特殊なキーワード

クラス (1)

- JavaScriptには正確にはクラスが存在しない
- 代わりにプロトタイプというものがあるけど…
  簡単には理解できないので、パスします

クラス (2)

- **コンストラクタ**

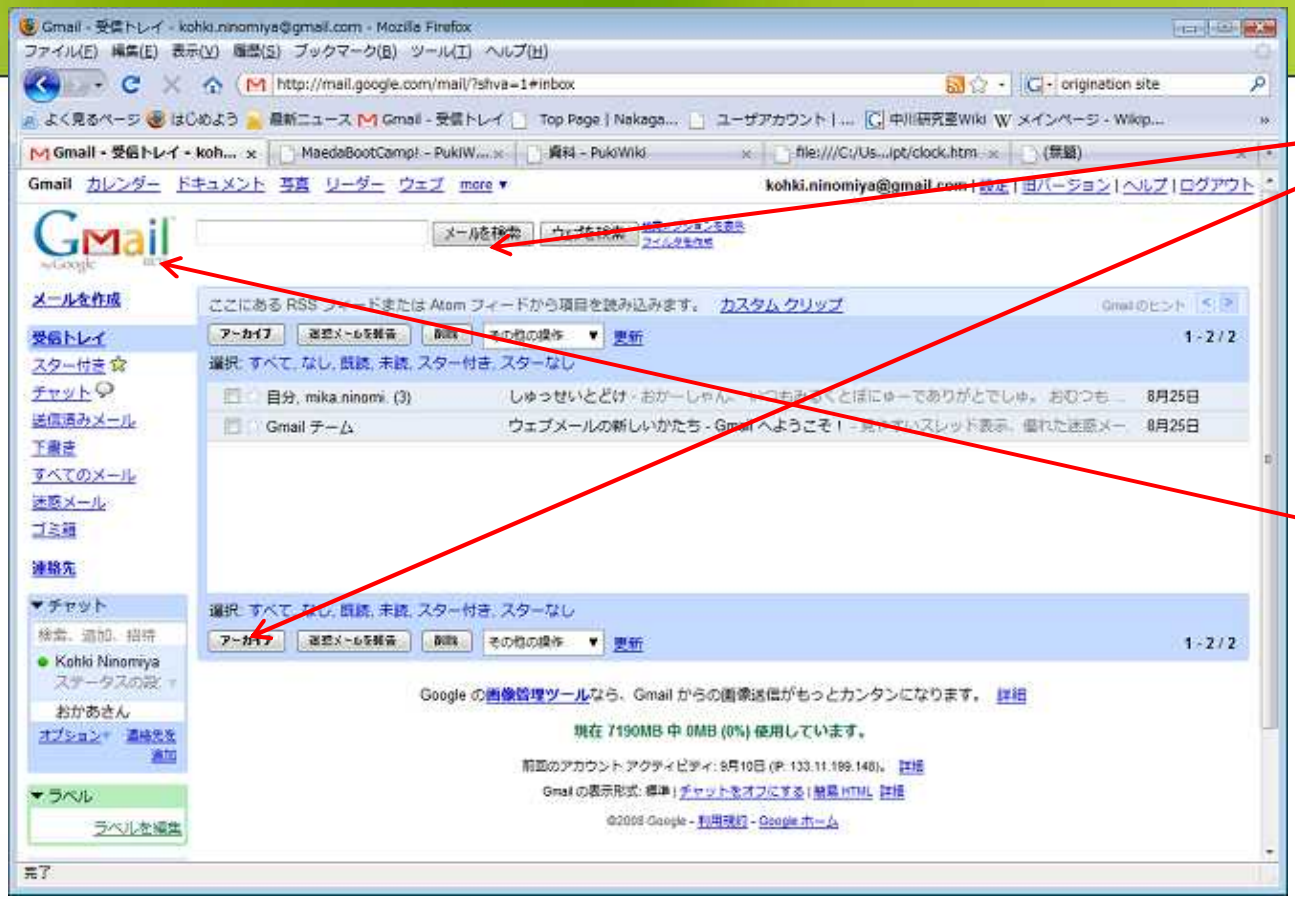
```
function Point(x, y) {
   this.x = x;
   this.y = y;
   this.move = function(newx, newy) { this.x = newx; this.y =
     newy; };
}
var p = new Point(3, 4);
```

効率を考えると最後のmoveメソッドは

```
Point.prototype.move = function(newx, newy) { this.x = newx;
    this.y = newy; };
```

とするほうが良い。
なぜこんなことになっているのかを理解するのはあとさらに1時間は必要だ。

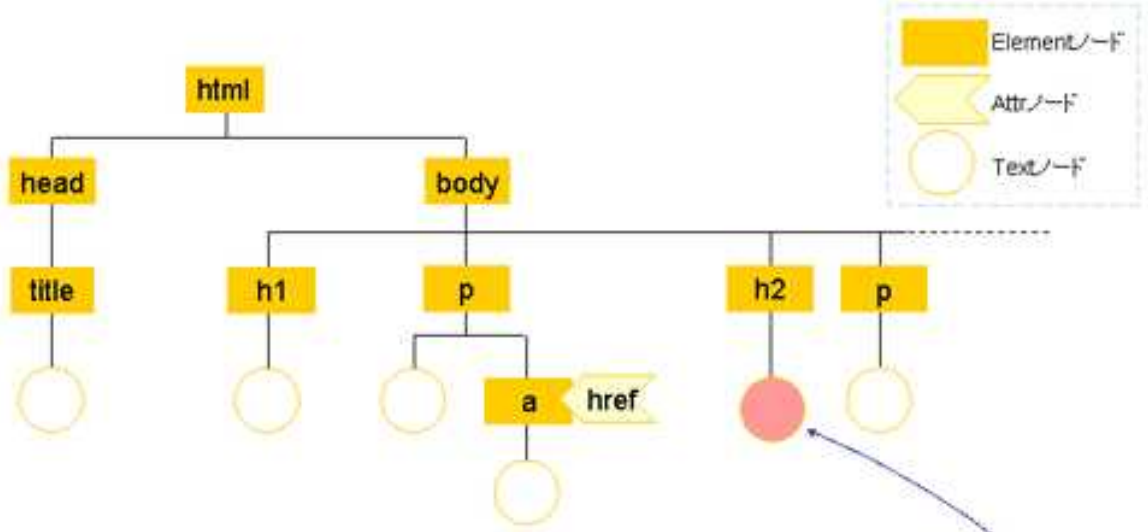
ブラウザのオブジェクト



こういうフォームやボタンもオブジェクト

画像やリンクもオブジェクト

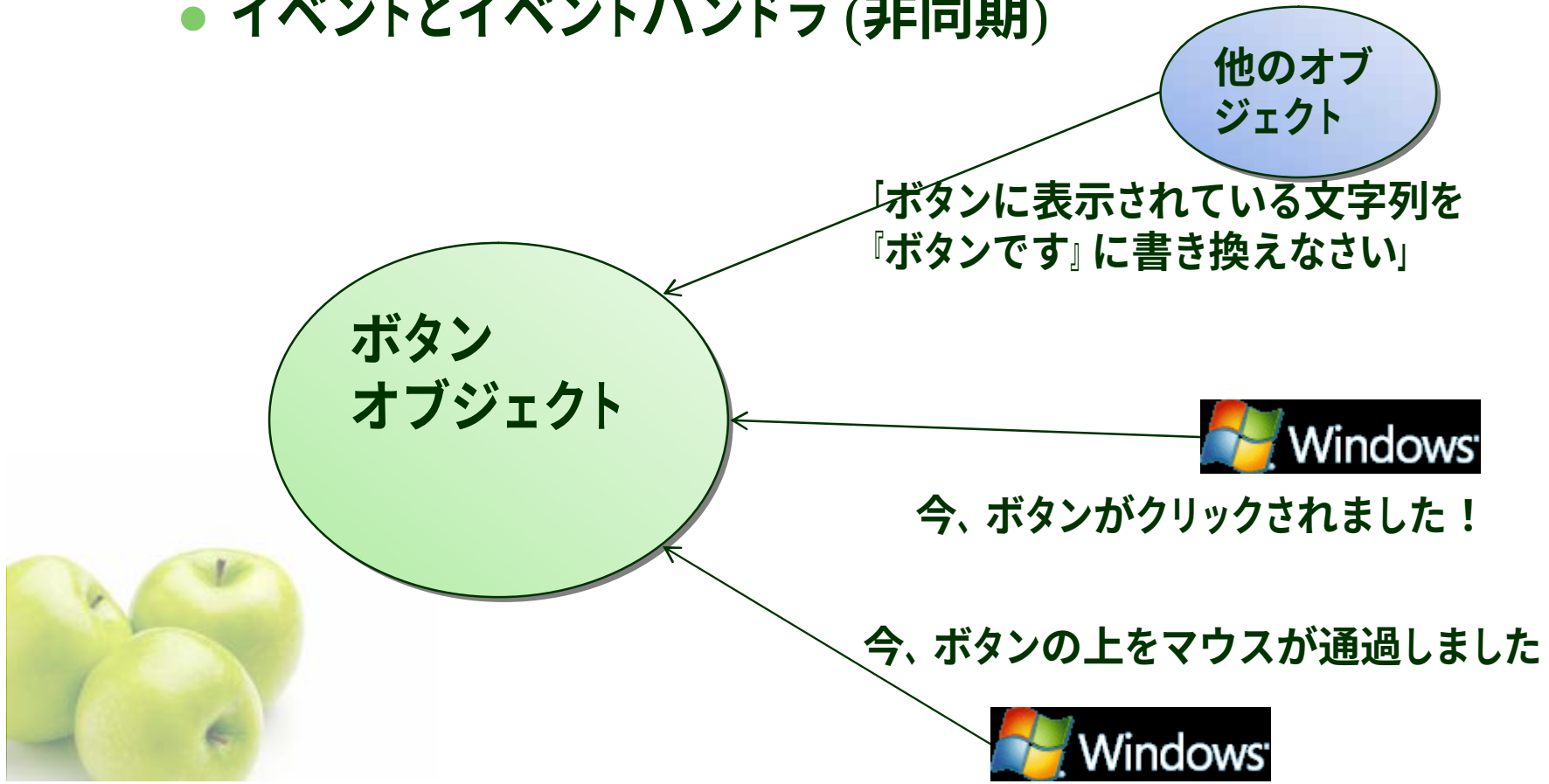
DOM




```
<html>
  <head>
    <title>らららのへへへ</title>
  </head>
  <body>
    <h1>「らららのへへへ」の真実</h1>
      <p>「らららのへへへ」について初めての方は<a href="07.html">前回の記事</a>参照</p>
    <h2>今声高に「へへへ」と叫ばれているのか</h2>
      <p>近年の・・・・。しかし、・・・・</p>
...
```

オブジェクトへのメッセージ

- 二種類のメッセージ
  - メソッドとプロパティ (同期)
  - イベントとイベントハンドラ (非同期)



マウスの位置

```
<head>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
<script type="text/javascript">
document.onmousemove = mouseMove;
function mouseMove(evnt) {
  var x = parseInt(evnt.clientX);
  var y = parseInt(evnt.clientY);
  var screen = document.getElementById("screen");
  screen.innerHTML="x: "+x+" y: "+y;
}
</script>
</head>

<body>
<div id="screen"></div>
</body>
```

ブロックを動かそう

- スタイルを使うとDOMのオブジェクトの位置、色、大きさ等を変えられる。
- スタイルオブジェクト

```
var screen =
   document.getElementById(“screen”);
var scrsty = screen.style;
scrsty.position=“absolute”;
scrsty.top=100 +”px”;
sctsty.left=200+”px”;
```

数字入力ボタンを作ろう

```
var amount = 0;
function changeAmount(x) {
  document.getElementById("inputscreen").value=x;
  amount = x;
}
</script>
</head>
<body onload="changeAmount(0)">
<form>
<input type="text" width="20" id="inputscreen" /><br />
<input type="button" value="7" onclick="changeAmount(10*amount+7)" />
<input type="button" value="8" onclick="changeAmount(10*amount+8)" />
<input type="button" value="9" onclick="changeAmount(10*amount+9)" />
<br />
<input type="button" value="4" onclick="changeAmount(10*amount+4)" />
<input type="button" value="5" onclick="changeAmount(10*amount+5)" />
<input type="button" value="6" onclick="changeAmount(10*amount+6)" />
<br />
<input type="button" value="1" onclick="changeAmount(10*amount+1)" />
<input type="button" value="2" onclick="changeAmount(10*amount+2)" />
<input type="button" value="3" onclick="changeAmount(10*amount+3)" />
<br />
<input type="button" value="0" onclick="changeAmount(10*amount)" />
<input type="button" value="c" onclick="changeAmount(0)" />
<input type="button" value="d" onclick="changeAmount(Math.floor(amount / 10+0.01))" />
</form>
```

雛形

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN“
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>

<head>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
</head>

<body>
<script type="text/javascript">
</script>
</body>

</html>
```