2015 年度　メディアプロジェクト演習 1
HTML 講座 発展編 4 ~ JavaScript の応用

立命館大学情報理工学部　メディア情報学科

1 プログラミング言語としての JavaScript

HTML 講座 発展編 3 の 2.2 節でも述べたように，JavaScript は C 言語などと同じように，関数や繰り返し，条件分岐などのプログラミング言語としての基本的な機能を備えている．

また，JavaScript は，この演習で後に学ぶ Java 言語と同様に，オブジェクト指向のプログラミング言語でもある（ただし，JavaScript と Java 言語は，名前や文法が似ているが，無関係であることに注意）．

1.1 データ型

JavaScript で扱えるデータ型として，以下のものがある． JavaScript では，変数の型は格納される値によって動的に決められ，宣言も不要である．

基本型	**数値型**	整数や浮動小数点数などの数値
	文字列型	任意の文字の並び
	論理型	true（真），false（偽）のいずれか
	その他	空（null），未定義（undefined）
参照型	**配列**	データの集合
	オブジェクト	名前をキーにアクセスできる配列（連想配列と同じもの）
	関数	一連の処理をまとめたもの

1.2 演算子・制御文

算術・代入・比較・論理演算子などは，ほぼ C 言語と同じものが使用できる．また，if 文，for 文，while 文，select 文などの制御文も C 言語と同様に使用可能である．

算術演算子	+, -, *, /, %, ++, --
代入演算子	=, +=, -=, *=, /=, %= など
比較演算子	==, !=, <, <=, >, >= など
論理演算子	&&（論理積），||（論理和），!（否定）

1.3 配列

JavaScript でも C 言語と同様に配列が使用できる．あらかじめ要素数を指定する必要はなく，必要に応じて動的に領域が確保される．また，要素ごとに型が異なっていても良い．

JavaScript における配列は，Array オブジェクトして実現されている．配列の作成と使用の仕方は以下の通りである．

なお，以降のサンプルプログラムの 3 行目にある <meta>タグは，HTML 文書の文字エンコーディングを指定するためのものであり，文字化けを避けるために必ず記述することを推奨する．

(sample42.html)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>配列のサンプル</title>
</head>
<body>
<script type="text/javascript">
<!--
// 空の配列を作成
a = new Array();

// 要素に値を代入
a[0] = 1;              // 数値型
a[1] = 'two';          // 文字列型
a[2] = true;           // 論理型

// あらかじめ要素の値を指定して配列を作成
b = new Array(1.23, "立命館");

// 配列の内容を表示
for (i = 0; i < a.length; i++) {
    document.write("a[" + i + "]: " + a[i] + "<br>");
}
for (i = 0; i < b.length; i++) {
    document.write("b[" + i + "]: " + b[i] + "<br>");
}
-->
</script>
</body>
</html>
```

1.4 連想配列

通常，配列の添え字は非負の整数であるが，JavaScript では添え字に文字列を使用することもできる．このような配列を「連想配列」と呼ぶ．連想配列の使用例を以下に挙げる．

(sample43.html)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>連想配列のサンプル</title>
</head>
<body>
<script type="text/javascript">
// 空の連想配列を作成
a = new Array();

// 要素に値を代入
a['dog'] = '犬';
```

```
a['cat'] = '猫';
a['mouse'] = '鼠';

// 指定された要素の値を表示
document.write("dog の日本語は「" + a['dog'] + "」です. ");
</script>
</body>
</html>
```

1.5 数学関数

JavaScript では，以下のような数学関数が用意されている（以下に挙げたのは主なもののみ）．これらは Math オブジェクトのメソッドとして実現されており，実際に使用する際は，「Math.メソッド名」の形で，たとえば「a = Math.random();」のように用いる．

メソッド	説明
abs(*num*)	絶対値
ceil(*num*)	切り上げ
floor(*num*)	切り捨て
log(*num*)	自然対数
max(*num1*, *num2*)	最大値
min(*num1*, *num2*)	最小値
pow(*base*, *p*)	べき乗
random()	乱数（0 以上 1 未満の実数）
round(*num*)	小数点以下を丸める
sqrt(*num*)	平方根
sin(*num*※)	正弦（サイン）
cos(*num*※)	余弦（コサイン）
tan(*num*※)	正接（タンジェント）

※ sin, cos, tan の引数の単位はラジアン

たとえば，1～100 までの整数の乱数を発生させるには，以下のように記述すれば良い．

```
a = Math.floor(Math.random() * 100) + 1;
```

また，円周率などの定数も定義されており，たとえば円周率は Math.PI で利用できる．

2 イベントハンドラ

HTML 講座 発展編 3 の 2.1 節でマウスイベントについて説明したが，JavaScript では，マウス操作以外にもさまざまな操作のタイミングで処理を行うためのイベントハンドラが用意されている．JavaScript で使用できる主なイベントハンドラを以下に挙げる．

イベントハンドラ	説明
onClick	要素やリンクをクリックしたとき
onDblClick	要素やリンクをダブルクリックしたとき
onMouseOver	マウスカーソルが要素やリンク上に乗ったとき
onMouseOut	マウスカーソルが要素やリンクから離れたとき
onKeyUp	キーボードのキーを離したとき
onKeyDown	キーボードのキーを押したとき
onAbort	画像の読み込みが中断したとき
onError	ページや画像の読み込み中にエラーが発生したとき
onLoad	ページや画像の読み込みが完了したとき
onBlur	ページや要素からフォーカスを失ったとき
onFocus	ページや要素がフォーカスされたとき
onUnload	現在のページから他のページに移るとき
onReset	フォームがリセットされるとき
onSubmit	フォームが送信されるとき
onChange	フォームの内容が変更されたとき
onSelect	フォームのテキストが選択されたとき

3 ブラウザオブジェクトの操作

ブラウザオブジェクトとは，Web ブラウザのウィンドウ，その中に表示されている文書，文書中の HTML フォーム，フォーム中の各要素などのことであり，それらに対してさまざまな操作が可能となっている．

3.1 Window オブジェクト

Window オブジェクトは，ブラウザのウィンドウを表すオブジェクトであり，window という変数で参照できる．ウィンドウの生成・消去，ダイアログボックス（警告・確認・文字入力など）の表示，発展編 3 の 2.3 節でも使用したタイマーの設定などを行うことができる．また，現在のウィンドウの幅・高さを取得したり，これらを新たに設定することも可能である．

Windowオブジェクトで使用できる主なメソッドを以下に示す．

メソッド	説明
open(*url*, *name*, *opt*)	ウィンドウを開く
close()	ウィンドウを閉じる
alert(*str*)	警告ダイアログを表示
confirm(*str*)	確認ダイアログを表示
prompt(*str*)	文字入力ダイアログを表示
setInterval(*func*, *t*)	指定したミリ秒間隔でタイマーの処理を行う
setTimeOut(*func*, *t*)	指定したミリ秒後に 1 度だけタイマーの処理を行う
focus()	ウィンドウを前面に移動
blur()	ウィンドウを背面に移動

(1) 新たなウィンドウの生成

JavaScript では，現在開いているブラウザのウィンドウとは別に，新たなブラウザウィンドウを生成することができる．その際は，window オブジェクトの open()メソッドを用いる．

open()メソッドの引数には，表示する HTML ページもしくは画像ファイルの URL と，ウィンドウのオブジェクト名を指定する．既にあるオブジェクト名を指

定すると，既にそのウィンドウで開いているページが指定したページに置き換わる．存在しないオブジェクト名を指定した場合は，新たなウィンドウが生成される．

オプションで，ウィンドウのサイズなどを指定することも可能である．オプションとして指定可能なプロパティを以下に示す．

プロパティ	説明
width	ウィンドウの幅
height	ウィンドウの高さ
location	アドレスバーの表示/非表示
scrollbars	スクロールバーの表示/非表示
resizable	ウィンドウサイズの変更可能/不可能
toolbar	ツールバーの表示/非表示
status	ステータスバーの表示/非表示
menubar	メニューバーの表示/非表示

ウィンドウを新たに生成するサンプルを以下に示す．なお，<button>タグは，文書中に押しボタンを表示するための HTML 要素である．

(sample44.html)

```
<html><body>
<button onClick="window.open('image1.jpg', 'new',
'width=148,height=154')">ここを押して下さい
</button>
</body></html>
```

発展課題 4：自己紹介ページに，複数の画像のサムネール（縮小版）を表示し，ある画像をダブルクリックすると，その拡大画像が新たなウィンドウに表示される機能を作成して下さい．

(2) 各種ダイアログボックスの表示

JavaScript では，警告（alert），確認（confirm），文字入力（prompt）の 3 種類のダイアログボックスを表示することができる．

confirm ダイアログボックスでは，メッセージとともに「OK」ボタンと「キャンセル」ボタンが表示され，利用者が押したボタンによってそれぞれ true，false が返される．

prompt ダイアログボックスでは，メッセージとともに，文字を入力するフィールドと「OK」ボタン，「キャンセル」ボタンが表示され，「OK」ボタンを押した時点で入力フィールドに入力された文字列を取得することが可能である．

confirm ダイアログボックスの利用例を sample45.html に，promprt ダイアログボックスの利用例を sample46.html にそれぞれ示す．

(sample45.html)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>confirm ダイアログボックスのサンプル
</title>
</head>
<body>
<script type="text/javascript">
<!--
ret = confirm("どちらかのボタンを押して下さい．
");
document.write("<h2>" + ret + "が選択されました．
</h2>");
//-->
</script>
</body></html>
```

(sample46.html)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>prompt ダイアログボックスのサンプル
</title>
</head>
<body>
<script type="text/javascript">
<!--
str = prompt("名前を入力して下さい．");
document.write("<h2>こんにちは，" + str + "さん．
</h2>");
//-->
</script>
</body></html>
```

発展課題 5：自己紹介ページの中に，決められた合い言葉をダイアログボックスに入力しないと表示されないようなページを作成して下さい．

3.2 Document オブジェクト

Document オブジェクトは，ブラウザのウィンドウ上に表示されている文字列・画像・フォーム・リンクなどを扱うオブジェクトであり，document という変数で参照できる．

すでに発展編 3 で，2.1 節の画像の切り替えや 2.2 節の文字列の表示に Document オブジェクトが使用されている．また，この発展編 4 の 1.3～1.4 節の配列の使用例（sample46～47.html）や，3.2 節(2)のダイアログボックスの表示の例（sample49～50.html）では，文字列の表示に document.write()メソッドを使用している．

文書中の文字列・画像・フォーム・リンクなどの特定の要素を示すには，次節で述べる DOM を用いる．

3.3 DOM

DOM（Document Object Model）とは，HTML 文書中の特定の要素にアクセスするために用いる仕組

みである.

発展編 3 の 2.1～2.2 節などで使用したように, HTML 文書中のある特定の画像や, あるタグで囲まれた文字列を指定し, それらに対して, 属性値を変更したり, 文字列を書き換えるなどの操作が可能である.

DOM にはさまざまな使用方法があるが, ここでは HTML の id 属性を用いた簡単な使用例を示す.

(1) HTML 要素に ID を付与する

HTML 文書中の要素(タグで囲まれた部分)には, id 属性を指定することにより文書中で一意な ID を付与することができ, この値を用いてその要素を一意に識別できる.

発展編 3 の 2.1 節の sample43.html では, 「JavaScript!」という文字列を囲む font 要素に id 属性の値として"word1"が付与されている.

同様に, 発展編3の2.2節のsample44.htmlでは, 文字を追加表示する部分の font 要素の id 属性の値として"area1"が付与されている.

(2) ID を用いて HTML 要素を参照・操作する

(1)で述べたように HTML 要素に ID を付与しておくと, Document オブジェクトの getElementID()メソッドを用いることで, その要素を参照することができる.

発展編3の2.1節のsample43.htmlでは, "word1"の ID を持つ font 要素に対して onClick イベントハンドラでその要素の文字色を変更している.

同様に, 発展編 3 の 2.2 節の sample44.html では, "area1"の ID を持つ font 要素に対して, この innerHTML プロパティ(要素の内側の HTML)を書き換えて(文字を追加して)いる.

DOM で用いることができる主なプロパティとメソッドを以下に示す.

プロパティ／メソッド	説明
innerHTML	要素内の HTML 部分
innerText	要素内のテキスト部分
getElementById(*id*)	特定の ID の要素を取得
createElement(*name*)	要素を作成
createTextNode(*text*)	テキストノードを作成
appendChild(*elem*)	要素を子ノードとして追加
removeChild(*elem*)	子ノードの要素を削除
createAttribute(*name*)	指定した属性名の属性を作成
removeAttribute(*name*)	指定した属性を削除
getAttribute(*name*)	属性値を取得
setAttribute(*name, value*)	属性名と属性値を設定

DOM を用いて, テキスト入力フィールドに入力された文字列を取得し, 表の中の文字列を入力された文字列で書き換える例を以下に示す.

(sample47.html)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>DOM のサンプル</title>
</head>
<body>

<script type="text/javascript">
<!--
function rewrite() {
    // 入力された文字列を取得
    name = document.getElementById("name").value;

    // テーブルのセルの内容を書き換え
    document.getElementById("cell").innerHTML =
        '<font color="red">' + name + '</font>';
}
-->
</script>

名前を入力して下さい:
<input type="text" id="name"/>
<input type="button" value="送信"
 onclick="rewrite()">

<table border="1">
<tr><td>名前</td></tr>
<tr><td id="cell">
<font color="grey">未入力</font>
</td></tr>
</table>

</body></html>
```

発展課題 6:DOM を用いて, テキスト入力フィールドに入力した文字列が, 送信ボタンを押すごとに, ページの末尾に追加されていくようなページを作成して下さい.

参考文献

JavaScript に関する Web 上の資料として, 講義の Web ページに挙げてある他にも, 以下のサイトなどが参考になる.

[1] JavaScript 入門
http://www.pori2.net/js/
[2] JavaScript サンプル集 & HTML の基礎
http://www9.plala.or.jp/oyoyon/html/

JavaScript についてより詳しく学習したい場合は, 以下の書籍などを参考にすると良い.

[3] 独習 JavaScript 第2版, 高橋和也, 竹添直樹, 里見知宏著, 翔泳社, 2013.
[4] JavaScript 本格入門, 山田祥寛著, 技術評論社, 2010.

UNIX の便利な使い方　（メディアプロジェクト演習 1　補助資料）

Emacs で便利なキーバインド（ショートカット）

「C-何々」というのは「コントロールキー（Ctrl）を押しながら何々ボタンを押す」という意味。
「M-何々」というのは「エスケープキー（Esc）を押した後で何々ボタンを押す」という意味。
「Esc を押した後で」が面倒な場合は、「Ctrl を押しながら[を押した後で」でもよい。

C-f　1 文字進む (forward)　C-b　1 文字戻る (backward)　C-n　次の行 (next)　C-p 前の行 (previous)
C-a　行頭に移動（a はアルファベットの最初の字なので）
C-e　行末に移動（end）
C-k　カーソルの現在位置から行末までを削除 (kill)。しかし内容は kill-リングという場所に保存されている。
C-Space カーソルの現在位置を「切り取り」「コピー」の起点として保存
C-w　C-Space で保存した起点からカーソルの現在位置までを切り取り
M-w　C-Space で保存した起点からカーソルの現在位置までをコピー
C-y　保存してある文字列をカーソルの現在位置に貼り付ける（yank）
C-/　元の状態に戻す。（直前の操作を取り消す）
C-s　検索。ミニバッファ（Emacs の一番下の行）に語を入力して Enter。C-s で次の候補。C-r で前の候補。
M-%　置換。ミニバッファに置換したい文字列を入れて Enter。
　　置換後の文字列を入れて Enter。y を押すたびに置換される。（yes の y）
C-x C-s 保存（save）
C-x C-w 名前を付けて保存（write）

【よく使う組み合わせ】ある行を別の場所にコピーしたい場合。
1. C-a でその行の先頭に移動。
2. C-k でその行を削除。これによってその行の内容が kill-リングに保存される。
3. C-y で貼り付け。（先ほど削除した行を復活させる）
4. C-n や C-p、C-f や C-b を使ってコピー先の場所にカーソルを移動させる。
5. C-y で行を貼り付け。kill-リングの中身は何回貼り付けても保存されているので。

コントロールキーとエスケープキーの対応の法則

「コントロールを押しながら」→文字単位
「エスケープキーを押した後で」→単語単位
例：　C-f は 1 文字右に移動。M-f は 1 単語右に移動。C-b は 1 文字左に移動。M-b は 1 単語左に移動。
　　C-d は 1 文字削除。M-d は 1 単語削除。これでプログラムのコマンドやファイル名を簡単に削除できる。

UNIX のシェル　便利なコマンド集

ls -lrt　　ファイルを新しく編集した順で並べてリスト。（list の ls）
cat　kadai5-1.c　ファイルの中身を表示。（concatenate の cat）
diff　kadai5-1.c　kadai5-2.c　　二つのファイルの間の違いを表示。（difference の diff）
man ls　　任意のコマンド（この場合 ls）にどういう機能があるかを表示してくれる。（manual の man）
gcc --help　コマンドの後に--help をつけて Enter を押すと、簡単な説明が出る。この場合、gcc の説明が出る。
　　-o の後に続く文字列が出力先ファイル名になる、といったことが分かる。
pwd　　現在いるディレクトリのパスを表示。（present working directory）

USB メモリのマウント方法（RAINBOW GUIDE 2015 Linux 操作入門編 p.169～170）

Linux では，USB メモリを接続すると自動的にマウントされる．MS-DOS ファイルシステム（FAT 形式とも呼ばれる）でフォーマットされた USB メモリが使用可能である．オートマウントによりマウントされると，デスクトップにマウント先にアクセスできるアイコンが表示される．

USB メモリのマウント先は/media/以下となる．
その後，以下のコマンドを実行することでファイルやディレクトリ（この場合 datadir）をコピーすることができる．

```
% cp -rf datadir /media/USB メモリのフォルダ名
```

コピーが完了したら，デスクトップの USB メモリのアイコンを右クリックして「取り出す」を選択することでアンマウントする．アンマウントが正常に終了すれば USB メモリを抜いて良い．

よくある問題とその対処法　（メディアプロジェクト演習 1　補助資料）

1. 作成した個人ホームページにアクセスできない場合の対処

ホームディレクトリが自分のアカウントでしか見られないような設定になっている場合，public_html ディレクトリ以下にファイルを作成しても，サーバ経由（http://www.ritsumei.ac.jp/~アカウント名）でアクセスすることができません．

ホームディレクトリを誰でも見られるようにし，かつ他のファイルを他人から見られないように設定するには，以下のコマンドを実行します．

```
$ chmod go-rwx -R ~
$ chmod o+rx -R ~/public_html
$ chmod o+x ~
```

アクセス権限の設定について詳しくは，RAINBOW GUIDE 2015 Linux 操作入門編 p.13〜16 を参照して下さい．

2. ホームディレクトリの容量を超えてしまった場合の対処

RAINBOW では，学部生のホームディレクトリの容量が 100MB に制限されています．これを超えた場合，emacs で作成したファイルを保存できない，Java プログラムのコンパイルができないなど，様々な不具合が発生します．自分のホームディレクトリが容量を超えているかどうかの確認と対処の方法は以下の通りです（詳しくは RAINBOW GUIDE 2015 Linux 操作入門編 p.12〜13 を参照）．

まず，自分のホームディレクトリを現在どの程度利用しているかは，以下のコマンドで確認できます．

```
$ du -sh ~
```

もしホームディレクトリの容量（100MB）を超えていた場合は，以下の対処を行って下さい．

I. 容量の大きなファイルを探して削除

以下のコマンドを実行することで，ホームディレクトリ内にあるすべてのディレクトリ・ファイルを，容量の大きい順に表示します．

```
$ du ~| sort -rn | less
```

この一覧から，容量が大きく不要なファイルを削除するか，USB メモリなどに移動することで，容量を減らすことができます．

II. Firefox のキャッシュファイルの削除

上記を行っても容量の大きなファイルが見つからない場合は，Firefox のキャッシュファイルがホームディレクトリを圧迫していることがあります．Firefox のキャッシュは以下のディレクトリにあります．

```
~/.mozilla/firefox/xxxxxxxx.default/Cache （xxxxxxxx は任意の英数字）
```

このキャッシュディレクトリの容量を確認するには，上記の Cache があるディレクトリに移動し，以下のコマンドを実行します．

```
$ du -sh Cache
```

キャッシュを削除するには以下のコマンドを実行します．

```
$ rm -rf Cache
```

また，今後キャッシュが作成されないように，Firefox の設定でキャッシュ容量を 0MB に設定することを推奨します．その方法は，まずメニューバーの [編集] から [設定] を選択します．そこで，開いた設定ウインドウ上の [詳細] タブの [ネットワーク] を選択し，「キャッシュサイズを制限する」にチェックを入れ，その下の部分を0MBにして「閉じる」をクリックします（RAINBOW GUIDE 2015 Linux 操作入門編 p.16～17 を参照）．