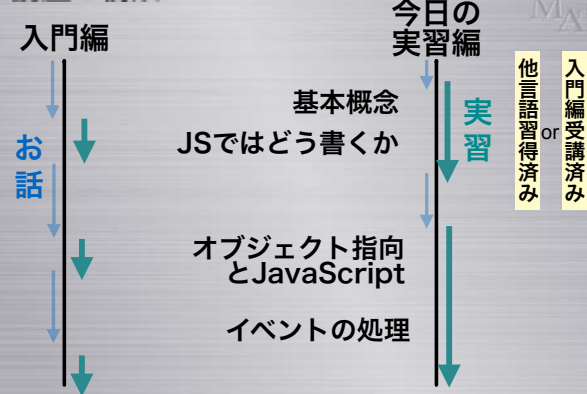


実習編 JavaScriptで学ぶ プログラミング入門 丸1日コース

マーリンアームズ株式会社
むしや 武舎広幸



講座の構成



教科書

- ◆『実践JavaScript!』
 - ・講座の内容をより詳しく解説
- www.musha.com/scjs



今日の時間割

Part I 基本概念の紹介 (入門編の復習)

JavaScriptではどう書く

実習

お昼

Part II タイマー

オブジェクト指向とJS

イベント処理

実習

JavaScriptのコード

第1章

ch0101.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>hello, world!</title>
</head>
<body>
  <h1>hello, world</h1>
  <p>
    ダイアログボックスに「hello, world」と表示します。
  </p>
  <script>
    alert("hello, world");
  </script>
</body>
</html>
```

JavaScriptのプログラム

<script>...</script>で囲む

別ファイルにして読み込む

p. 266

hello.html

```
<!DOCTYPE html>
<html>
<head>...</head>
<body>
  ...
  <script src="hello.js">
  </script>
</body>
```

hello.js

```
"use strict";
alert("hello, world");
```

- ・本格的なプログラムは外部ファイルにするのが普通 (HTMLとJavaScriptは別ファイルにする)
- ・</body>の前に置くのが基本 (表示が早くなる)
- ・教科書ではしばらく単一のHTMLファイルに書く

1. JSの基本構文

ch0407c.html

```
let html = ""; 変数宣言 代入 型の指定は不要 (cf. 他言語)
for (let i=0; i<16; i++) { 論理演算子 算術演算子
  let ファイル名 = `pictures/picture00${i}.jpg`; 日本語識別子
  if (10<=i) { //数字2桁の場合
    ファイル名 = `pictures/picture0${i}.jpg`;
  }
  const 画像タグ = ``;
  html += 画像タグ; 文字列の連結
}
document.write(html); document領域にHTMLコードとして出力



...



...

```

変数宣言letと定数宣言const

p. 46

let html = ""; varも使えるが... 変数は最初に一度だけ宣言。「型」指定は不要

const 画像タグ = ``; 値が変わらない「変数」 — constで宣言したほうがよい

p. 86

画像タグ += "... ←エラー。値を変更できない

- ・constの代わりにletを使っても結果は変わらないが、値が変わらないことを自分で (読む人も) 意識する
- ・「どこにあって同じ」が保証される (単純な変数の場合)

変数宣言letと定数宣言const

第4章 p. 87

```
const x = ...;
...
x = ...
...
y = x + z;

let x = ...;
...
x = ...
...
y = x + z;
```

図 4.9 constとletの比較

文字列の連結とテンプレートリテラル

p. 20

- ◆ "... " あるいは '...' あるいは `...` で「文字列」を表す
- ◆ 演算子+は文字列の連結にも
- ◆ `... \${...} ...` でテンプレートリテラル (可変部分付きの文字列)

```
let ファイル名 = `pictures/picture00${i}.jpg`;
const 画像タグ = ``;
```

ファイル名	picture000.jpg
画像タグ	

10

+ = で自分の後ろに追加

p. 90 問題4-3

```
let html="";
for (let i=0; i<10; i++) {
  html +=``;
}
document.write(html)
```

i	html
	""
0	
1	<...000.jpg><...001.jpg>
2	<...000.jpg><...001.jpg><...002.jpg>
3	<...000.jpg><...001.jpg><...002.jpg><...003.jpg>
...	...
9	<...000><...001><...002><...003>...<...008><...009.jpg>

11

日本語の識別子 (変数、関数、定数)

p. 110

- ◆ 識別子 (変数、定数、関数の名前) には日本語も使える

```
let ファイル名 = `pictures/picture00${i}.jpg`;
const 合計 = x + y + z;
const 平均 = 合計/3;
```

```
function おみくじを引く() {
  const y = Math.random();
  if (y < 0.3) {
    return "凶";
  }
  else if (y < 0.6) {
    return "小吉";
  }
  else {
    return "大吉";
  }
}
```

・プログラム部分で全角のスペースや記号などを入力しないように要注意!
"..." (...) '...'

・VS Codeでは ~~~ を表示してくれる

12

1. JSの基本構文

ch0407c.html

let html = ""; 変数宣言 代入 型の指定は不要 (cf. 他言語)

```
for (let i=0; i<16; i++) {
  let ファイル名 = `pictures/picture00${i}.jpg`;
  if (10<=i) { //数字2桁の場合
    ファイル名 = `pictures/picture0${i}.jpg`;
  }
  const 画像タグ = ``;
  html += 画像タグ; 文字列の連結
}
document.write(html); document領域にHTMLコードとして出力
```

出力

```


...



...

```

13

関数 (自作) —— 一連の処理をまとめる

ch0501.html

```
let x = Math.random();
if (x < 0.3) {
  x = >凶<;
}
else if (x < 0.6) {
  x = "小吉";
}
else { // それ以外なら
  x = "大吉";
}
x = `今日の運勢は${x}です`;
document.write(x);
```

→

```
const x = おみくじを引く();
document.write(`今日の運勢は${x}です`);
```

```
function おみくじを引く() {
  let y = Math.random();
  if (y < 0.3) {
    y = "凶";
  }
  else if (y < 0.6) {
    y = "小吉";
  }
  else { // それ以外なら
    y = "大吉";
  }
  return y; // 戻り値
}
```

p. 101 第5章

14

何回も呼び出す

ch0502.html

```
document.write(`西野さんの運勢: ${おみくじを引く()}<br>`);
document.write(`平原さんの運勢: ${おみくじを引く()}<br>`);
document.write(`家入さんの運勢: ${おみくじを引く()}<br>`);
```

```
function おみくじを引く() {
  const y = Math.random();
  if (y < 0.3) {
    return "凶";
  }
  else if (y < 0.6) {
    return "小吉";
  }
  else {
    return "大吉";
  }
}
```

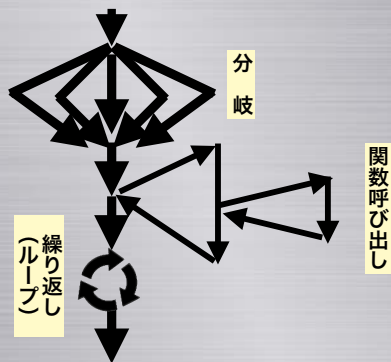
・...` の中で計算もできる
関数を呼び出すこともできる

・処理の「部品化」「概念化」
・関数はプログラムをわかりやすく、読みやすくするための重要な道具

15

制御構造 —— 処理の流れの制御 (アルゴリズム)

p. 111



16

データ構造 —— データの記憶手法

- 変数 let —— 記憶場所。varは旧式なので...
- 定数 const

p. 133 第7章

- 配列 —— データをまとめて記憶

```
const 漢数字 = ["零", "壹", "貳"];
```

ex0701.html

漢数字[0]	"零"
漢数字[1]	"壹"
漢数字[2]	"貳"

- 連想配列 (オブジェクトのプロパティ) p. 190 Note

```
const 電話帳 = {
  吉祥花子: "0422-22-4321",
  上杉謙信: "03-3311-3421",
  真田幸村: "0268-24-3311"
}
```

電話帳["吉祥花子"]	"0422-22-4321"
電話帳["上杉謙信"]	"03-3311-3421"
電話帳["真田幸村"]	"0268-24-3311"

```
document.write(電話帳["真田幸村"]);
```

17

演算子

- ◆ 算術演算子 (算術オペレータ) p. 59

- ・ + - * / % (余り)
- ・ ++ -- —— 1 足す、1 引く
- ・ x += y; x *= y; —— xにyを足す (掛ける)
(「+」は文字列の連結にも)

- ◆ 論理演算子 (ロジカルオペレータ) p. 58 p. 289

- ・ x<y x>y x<=y x>=y x===y x!==y && ||
- 0 == "" → 真 0 === "" → 偽 (型も考慮)
- 0 == ' ' → 真 0 === ' ' → 偽
- 0 == `` → 真 0 === `` → 偽
- ↑ これはひどい! ↑ 新しい規格で導入

18

タイマー（一定時間ごとの繰り返し）

課題 ロケット打ち上げ

p. 119 第6章

ロケットの画像を表示して、その下にカウントダウンする数字を表示して打ち上げの様子を真似よ



ch0601.html

28

カウントダウン

ch0601.html

```
let カウンタ = 10;
const タイマーID = setInterval(カウントダウン, 1000);
// 1秒 (1/1000*1000) ごとに繰り返す

function カウントダウン() {
  if (カウンタ >= 0) {
    // 数字
    // ロケットの画像
    画面書き換え(カウンタ, "pictures/rocket1.png");
    カウンタ--; // 自分を1減らす (算術演算子)
  }
  else {
    clearInterval(タイマーID); // タイマー停止
    画面書き換え("発射!", "pictures/rocket2.png");
  }
}

// setInterval()が返した値 (タイマーID) を
// 覚えておくと、これを使って終了できる
```

29

画面書き換え

ch0601.html

```
function 画面書き換え(テキスト, 画像ファイル) {
  document.open(); // クリア
  const html =
    <div style="text-align: center; font-size: 36pt;">
      ロケット発射<br>
      <br>
      ${テキスト}
    </div>; // テンプレートリテラル (可変部分付き文字列)
  document.write(html);
  document.close();
}
```

30

カウントダウン（名前付き関数）

```
let カウンタ = 10;
const タイマーID = setInterval(カウントダウン, 1000);

function カウントダウン() {
  if (カウンタ >= 0) {
    画面書き換え(カウンタ, "pictures/rocket1.png");
    カウンタ--; // 次に備える
  }
  else {
    clearInterval(タイマーID); // タイマー停止
    画面書き換え("発射!", "pictures/rocket2.png");
  }
}

function 画面書き換え(テキスト, 画像ファイル) {
  document.open(); // クリア
  const html =
    <div style="text-align: center; font-size: 36pt;">
      ロケット発射<br>
      <br>
      ${テキスト}
    </div>; // テンプレートリテラル (可変部分付き文字列)
  document.write(html);
  document.close();
}
```

ch0601.html

31

無名（匿名）関数

p. 125

ch0602.html

```
let カウンタ = 10;
const タイマーID = setInterval(
  function() { // 無名関数の定義の始まり 関数名なし
    if (カウンタ >= 0) {
      画面書き換え(カウンタ, "pictures/rocket1.png");
      カウンタ--; // 次に備える
    }
    else {
      clearInterval(タイマーID); // タイマー停止
      画面書き換え("発射!", "pictures/rocket2.png");
    }
  }, // ここまでがsetIntervalの第1引数の関数の定義
  1000); // setIntervalの第2引数

const タイマーID = setInterval(カウントダウン, 1000);
function カウントダウン() {
  ...
}
```

32

アロー関数

p. 126

ch0603.html

```
let カウンタ = 10;
const タイマーID = setInterval(
  () => { // アロー関数の定義の始まり
    if (カウンタ >= 0) {
      画面書き換え(カウンタ, "pictures/rocket1.png");
      カウンタ--; // 次に備える
    }
    else {
      clearInterval(タイマーID); // タイマー停止
      画面書き換え("発射!", "pictures/rocket2.png");
    }
  }, 1000);

// setIntervalの第2引数
```

下記ページに無名関数との違いの詳細
<https://qiita.com/suin/items/a44825d253d023e31e4d>

33

繰り返し —— インターバルを指定

```
let カウンタ = 10;
const タイマーID = setInterval(カウントダウン, 1000);
// 1秒 (1/1000*1000) ごとに繰り返す

function カウントダウン() {
  if (カウンタ >= 0) {
    画面書き換え(カウンタ, "pictures/rocket1.png");
    カウンタ--;
  }
  else {
    clearInterval(タイマーID); // タイマー停止
    画面書き換え("発射!", "pictures/rocket2.png");
  }
}

// setInterval()が返した値 (タイマーID) を
// 覚えておくと、これを使って終了できる
```

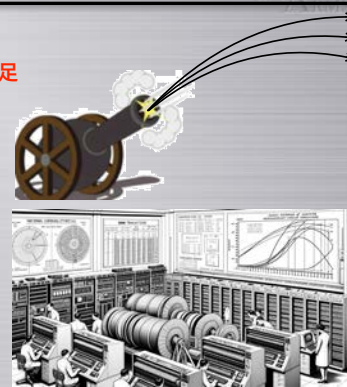
34

第9章 オブジェクト指向

p. 175 第9章

◆ 最初は数値計算が主
計算ができれば多くの人は満足

ほかの用途にも使われだした
もっと書きやすくなりたい！

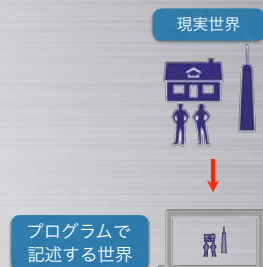


35

オブジェクト指向

- ◆ 世の中はもの (object) でできている
- ◆ ものを中心にしたプログラム
—— 自然なプログラム
- ・ 開発が容易
- ・ 保守（修正や改良）も容易

オブジェクト指向プログラミング



36

オブジェクト指向言語

- 1960年代から研究 Simula, Smalltalk
- 1990年頃から一般に広まる
- 最近の言語 — ほとんどオブジェクト指向
- JavaScriptもオブジェクト指向の機能をもつ
ただし従来型の手法も使える

プログラムを
わかりやすく
自然に
書けるようにするために生まれた考え方

37

関数も似た発想

```
document.write(今日の運勢: ${おみくじを引く()}<br>);  
document.write(明日の運勢: ${おみくじを引く()}<br>);  
document.write(明後日の運勢: ${おみくじを引く()}<br>);
```

```
function おみくじを引く() {  
  const y = Math.random();  
  if (y < 0.3) {  
    return "凶";  
  }  
  else if (y < 0.6) {  
    return "小吉";  
  }  
  else {  
    return "大吉";  
  }  
}
```

- 処理の「部品化」「概念化」「抽象化」
- 関数はプログラムを読みやすくする重要な道具
- オブジェクトも同様の発想

38

オブジェクト指向の考え方

◆プログラムで扱うもの（オブジェクト）を2つの側面から捉える

- どんな性質をもつか
- どんな動作をするか

◆たとえばムービーを扱うなら

- 性質
 - 横幅 `mv1.videoWidth`
 - 高さ `mv1.videoHeight`
 - 長さ（時間） `mv1.duration`
 - 状況（再生中か停止中か） `mv1.paused` 代入可
 - 現在の再生位置 `mv1.currentTime` 代入可
 - ボリューム `mv1.volume` 代入可
- 動作
 - 再生開始・停止 関数で表現→メソッド `mv1.play()` `mv1.pause()`

39

ブラウザのオブジェクト p. 193 第10章

みんなオブジェクト

アドレス (URL)

ウィンドウ

ドキュメント

イメージ

`document.images[0]`

ブラウザの各項目に対応するオブジェクトが用意されている
プログラマーは何もしなくても各オブジェクトのメソッドやプロパティが利用できる！

40

Images — 画像に関する情報

◆画像は複数あるので「配列」



`document.images[0]`
1番目の画像なので [0]
2番目の画像なら [1]

`document.images[0].src` → `pictures/picture000.jpg`
ソース（ファイル名）がわかる

代入もできる

`document.images[0].src = "pictures/picture002.jpg"`

画像が変えられる ←

41

スライドショー ch1001.html

```
let 画像番号 = 0;  
const タイマーID =  
  setInterval(画像を表示, 1000*2);  
  関数 時間 2秒ごとに  
  画像を表示() を繰り返す  
  
function 画像を表示() {  
  const ファイル名 = `pictures/picture00${画像番号}.jpg`;  
  document.images[0].src = ファイル名;  
  画像番号++;  
  if (5 <= 画像番号) {  
    clearInterval(タイマーID);  
  }  
}
```

この条件でタイマーを終了
`setInterval()`が返した値を覚えておく、これを使って終了できる

42

JSとオブジェクト指向

◆プログラムで扱うもの（オブジェクト）を2つの側面から捉え
たとえばブラウザのwindow

どんな性質を持つか

- 高さ (height)
- 幅 (width)
- 中身は？

`console.log("幅:${innerHeight}px 高さ:${innerWidth}px");`

どんな動作をするか

- ダイアログボックスを表示
- 入力を受け取る

変数で表現→プロパティ

`window.innerHeight`
`window.innerWidth`
`window.document`

関数で表現→メソッド

`window.alert()`
`window.prompt(x,y)`

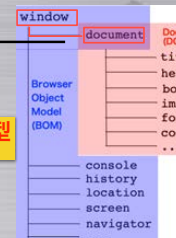
「window.」は省略可
変数は定義した変数と区別がつきにくいので省略しないほうがよいかな

43

document p. 207 第11章

◆メソッド

- `document.write()` HTMLコードを書き出す
`window.document.write()` と書いてもOK
- `document.getElementById("abc")` 典型
abcのIDをもつ部分のオブジェクトを得る
- `document.getElementsByClassName("xxx")`
"xxx"のクラスの要素を全部返す
- `document.getElementsByTagName("img")`
"img"のタグをもつ要素を全部返す
- `document.querySelector(セレクト)`
細かな条件（セレクト）を指定して、最初の要素を返す
詳細:
<https://developer.mozilla.org/ja/docs/Web/API/Document/querySelector>



44

document

◆メソッド

```
document.getElementById("abc") 典型  
abcのIDをもつ部分のオブジェクトを得る  
  
<div id="abc"> <!-- HTML -->  
  ♥♥♥  
</div>  
...  
  
// JSコード  
let x = document.getElementById("abc");  
x.innerHTML = "♥♥♥♥"; // 書き換え  
  
↓ 変数xを使わずに書けば  
document.getElementById("abc").innerHTML = "♥♥♥♥";  
同じ結果
```

ブラウザの中身を自由に書き換えられる

45

任意の部分の書き換え

ch1101j.html

```
let カウント = 10;
const タイマーID = setInterval(カウントダウン, 1000);

function カウントダウン() {
  document.getElementById("counterArea").innerHTML = カウント;
  カウント--;
  if (カウント < 0) {
    document.getElementById("counterArea").innerHTML = "発射";
    clearInterval(タイマーID); // タイマー停止
  }
}
```



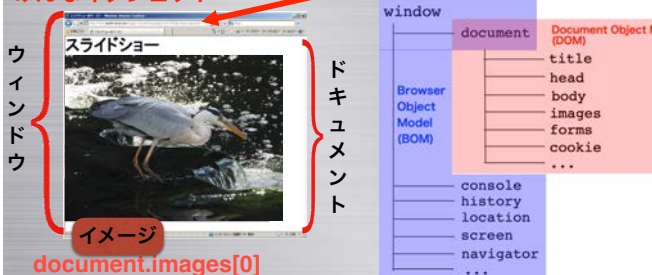
```
<h3>ロケット発射</h3>
<div></div>
<div id="counterArea"></div>
```

46

ブラウザのオブジェクト

みんなオブジェクト

アドレス (URL)



document.images[0]

ブラウザの各項目に対応するオブジェクトが用意されている
プログラマーは何もしなくても各オブジェクトのメソッドや
プロパティが利用できる！

47

ブラウザ内はキャンパス

- ◆ ブラウザ内はプログラマーにとっての「キャンパス」
- ◆ アイディア次第で何を書く（描く）こともできる
- ◆ スタイル (CSS) の操作も可能 → あとで例題
- ◆ 動かすこともできる
- ◆ ブラウザオブジェクトの関数（メソッド）や変数（プロパティ）を利用



48

第12章 イベント処理

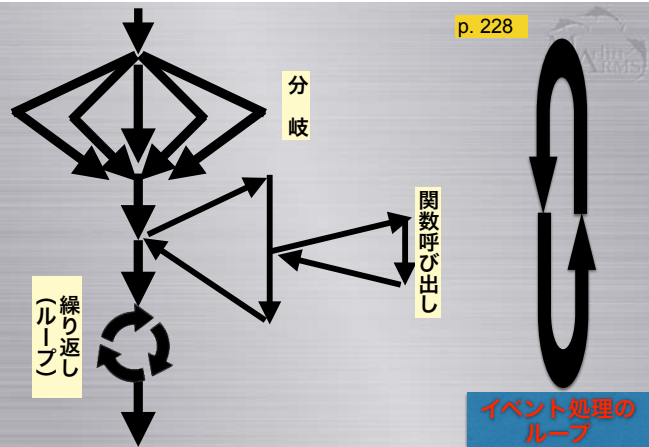
p. 225 第12章

プログラムの実行順序

- ◆ 上から下に順番に実行される（大原則）
- ◆ 関数 —— 別の部分が実行されてまた戻る
- ◆ 分岐 —— 一部だけが実行される
- ◆ ループ —— 繰り返される
- ◆ イベント —— 何かが起こると突然行われる（あらかじめ「罠」を仕掛けておく）

- ・ 読み込み時に実行できるものは実行し、イベント処理は罠を仕掛ける
- ・ 罠が仕掛けられたイベントが起こるのを待つ
- ・ 該当するイベントが起こる → イベント処理

49



50

イベント

- ◆ 何かが起こると突然行われる → 罠を仕掛ける

```
obj.addEventListener("mouseover", () => {
  // マウスが上に来た時の処理
});
obj.addEventListener("mouseout", () => {
  // マウスが外に出た時の処理
});
obj.addEventListener("click", () => {
  // クリックした時の処理
});
...
```

51

Photo Gallery

ch1201j2.html



◆ JSからスタイルも変更できる

- ・ display: none → 画像を表示しない
- ・ display: inline → 画像を普通に表示

```
document.images[0].style.display = "inline"; //表示
document.images[0].style.display = "none"; //消去
```

HTML + CSS + JavaScript

52

JSからスタイルの変更

ch1201j2.html

```




document.images[i].addEventListener("mouseover", () => { //罠
  document.images[i+画像の枚数].style.display = "inline";
});

document.images[i].addEventListener("mouseout", () => {
  document.images[i+画像の枚数].style.display = "none";
});
```

53

ムービー版 (JSからスタイルの変更)

問題12-16
prob1216.html



54

JSからスタイルの変更

問題12-16 prob1216.html

```

```

```
<video class="bigMovie" src="movies/no0${i}.mp4" ...  
  playsinline muted loop style="display: block;">
```

```
smallPicts[i].addEventListener("mouseover", () => {  
  bigMovies[i].style.display = "block";  
  bigMovies[i].play();  
});
```

```
smallPicts[i].addEventListener("mouseout", () => {  
  bigMovies[i].pause();  
  bigMovies[i].style.display = "none";  
});
```

55

今日の時間割

午前 基本概念 (の復習)
JavaScriptは (では) どう書く
実習
お昼休み
午後 タイマー (無名関数, アロー関数)
オブジェクト指向とJS
イベント処理
実習

56

実習 バグを少なくするコツ

◆ コピペ (copy & paste) をする

- 入力すると間違える
- 他のファイルから使える部分はコピペ
- 他の部分から使える部分はコピペ

```
if (...) {  
  x = "凶";  
} else if (...) {  
  x = "小吉";  
} else {  
  x = "大吉";  
}
```

57

バグを少なくするコツ

◆ コピペ (copy & paste) をする

- 入力すると間違える
- 他のファイルから使える部分はコピペ
- 他の部分から使える部分はコピペ

```
if (...) {  
  x = "凶";  
} else if (...) {  
  x = "小吉";  
} else if (...) {  
  x = "中吉";  
} else {  
  x = "大吉";  
}
```

58

バグを少なくするコツ

◆ コピペ (copy & paste) をする

- 入力すると間違える
- 他のファイルから使える部分はコピペ
- 他の部分から使える部分はコピペ

◆ 全角・半角に注意!

- 記号など (約物) はすべて半角

```
if (...) {  
  x = "凶";  
} else if (...) {  
  x = "小吉";  
} else if (...) {  
  x = "中吉";  
} else {  
  x = "大吉";  
}
```

59

「約物」のまとめ — みんな半角 p. 156

(...)

- 関数の引数 `alert(x); function xxx(x) { ... }`
- if文やfor文などの条件 `if (x < 0.3) { ... }`
- 計算の優先順位 `a*(b+c)`

{...} — 複数の文をまとめる

- 関数の定義全体 `function xx(x) { ... }`
- if文やfor文などの実行部分 `if (x < 0.3) { ... }`

"..."と'...'と`...` — 単純な文字列
ダブルクォート (二重引用符)、シングルクォート (一重引用符)、バッククォート

`...\${...}...` — 可変部分付き文字列 (テンプレートリテラル)
``

その他

```

; , // /*...*/ === !== && ||
< > <= >=

```

60

バグを少なくするコツ

◆ コピペ (copy & paste) をする

- 入力すると間違える
- 他のファイルから使える部分はコピペ
- 他の部分から使える部分はコピペ

◆ 全角・半角に注意!

- 記号など (約物) はすべて半角
- プログラム部分で全角のスペースや記号などを入力しないように注意!
- VS Codeでは `~~~` を表示 → 修正

```
if (...) {  
  x = "凶";  
} else if (...) {  
  x = "小吉";  
} else if (...) {  
  x = "中吉";  
} else {  
  x = "大吉";  
}
```

61

実習 初回の方におすすめ

◆ アニメーションの問題 (タイマー)

- 問題6-1 ~ 問題6-4 — タイマーの使い方
- 問題11-4 ~ 問題11-9 — アニメーション example/ch1104.html をベースに

● 初めて実習編にご出席の方

- 午前の実習は、基本的には最初からやってみてください。他のプログラミング言語をよくご存じの方は「これは大丈夫」と思う問題は飛ばしていただいても結構です
- 午後の実習は次の順番でやってみてください。簡単なアニメーションが作れます
 - 問題6-1 ~ 6-4 — タイマーの使い方
 - 問題11-4 ~ 11-9 — タイマーを使ったアニメーション。example/ch1104.html をベースにしてください
 - 問題11-9まで終わったら、午前実習の続きをやってください

● 実習編が2回目の方 (あるいは実習編のあとで入門編をお受けになっている方)

— 最初の問題から始めて、「前回やったので大丈夫」と思うものは飛ばして進んでください

62

教科書・参考資料

◆ 『実践JavaScript!』 www.musha.com/sc/js

◆ 入門編のビデオをご覧ください www.musha.com/sc/jsut

◆ 入門編を含め繰り返し受講いただけます (説明の時間も問題を解いていただいてもOK)
www.musha.com/sc/jsjdsc
↑ 教科書をお持ちの方用の割引リンク

◆ 講座に関するご質問はいつでもどうぞ。メッセージ機能あるいはメールなどで。

◆ 月額「もくもく会」もあり (1回分で月何回でも参加可)
www.musha.com/sc/jsmk

63