

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



1



資料

- ◆説明用のスライドのコピー
- ◆演習問題
- ◆サンプルデータ

●お送りしたメッセージをご確認の上、ご準備をお願いします
musha.com/sc/jsls (ジェイ・エス・エル・エス)



2

講座の構成

入門編

初心者向け



動画

www.musha.com/sc/jsut

実習編

基本概念

オブジェクト指向
とJavaScript
イベントの処理

今日の講座

3



今日の時間割

Part I 基本概念の簡単な紹介 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

4



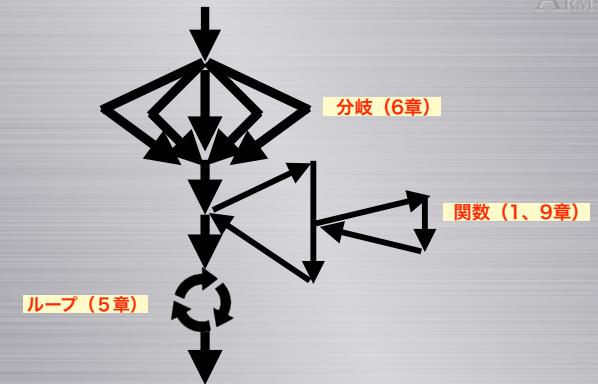
プログラミングの基礎概念

1. フロー制御 — 計算の手順 (アルゴリズム)
2. データ構造 — 情報の記憶手法
3. 演算子 — 加減乗除や文字の連結、大小、真偽の判断などの操作

5



1. フロー制御



6



2. データ構造

- ◆変数 (2章)、定数 (4章) — 値を記憶

x	0.314	let x = 0.314;
y	"大吉"	const y = "大吉";
ファイル名	"pict001.jpg"	ファイル名 = "pict001.jpg";

- ◆配列 (10章) — まとめて記憶

会員名簿	0	"東野ナカ"
	1	"家出オレ"
	2	"真田幸村"
	3	"武田信玄"

会員名簿 = ["東野ナカ", "家出オレ", "真田幸村", "武田信玄"];

- ◆オブジェクト (11, 15章) — 複雑な構造。後半で詳細

7



3. 演算子など

- ◆加減乗除と割算の余り (2章) — +, -, *, /, %
- ◆文字列の連結 (3章) — +
- ◆テンプレートリテラル (可変部分付き文字列、4章)
- ◆比較 (6章) — <, >, <=, >=
- ◆論理演算 (15章) — &&(AかつB)や|| (AあるいはB) ...

8



実習 (の準備)

- データとアプリの準備
- VS Codeを使える方、HTMLをよくご存の方は練習問題 (p. 15~) をから始めてください
「大丈夫な問題」は飛ばしても結構です。
ただし、説明は読んでください

9

データとアプリの準備 (p. 3)

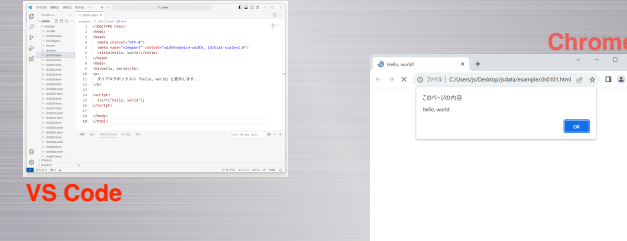
1. サンプル student.zip studentフォルダに展開
—— www.musha.com/sc/jsbes
2. ブラウザ (ウェブブラウザ) -- Google Chrome
—— www.musha.com/scjs?ap=chr
3. エディタ (テキストエディタ、コードエディタ)
-- VS Code (Visual Studio Code)
—— www.musha.com/scjs?ap=vsc

まだの方は、テキスト (p. 3) をご覧の上、ダウンロード (インストール) してください

10

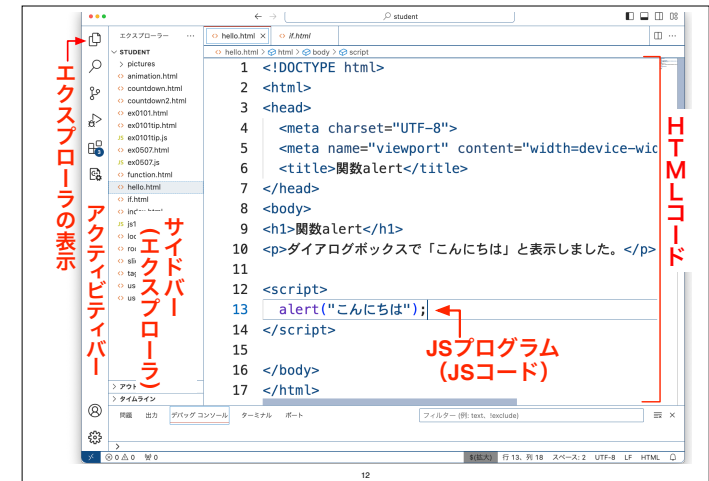
例題の実行 (p. 6)

1. エディタ (VS Code) で表示
・HTMLという規格に則った文字の並び (文字列)



2. ブラウザ (Chrome) で表示
・HTMLコードをルールに従って解釈した結果

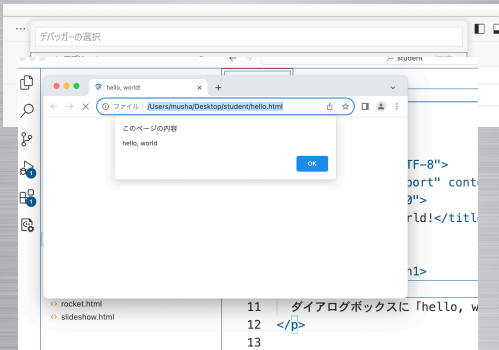
11



12

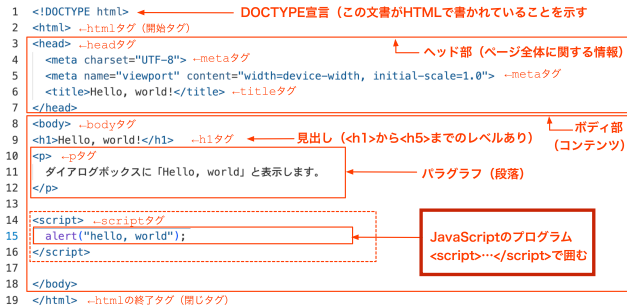
実行

- ・【実行】→【デバッグの開始】



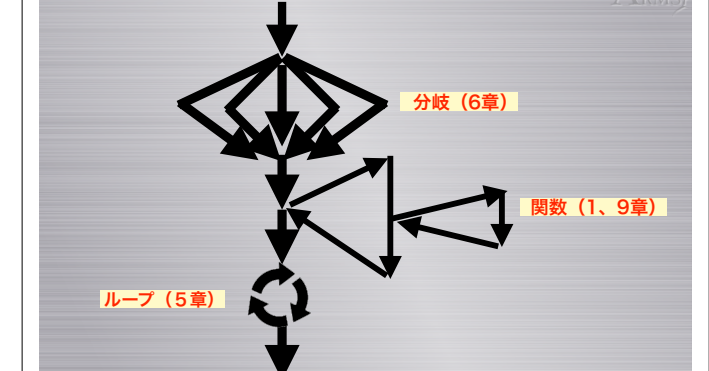
13

HTMLファイルの構成



14

1.フロー制御

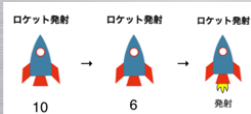


15

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

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



カウントダウン

10から0までカウントダウンせよ

10 9 8 7 6 5 4 3 2 1 0 発射!

16

サブ課題 カウントダウン

```
let カウンタ = 10;
let 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) {
    document.write(カウンタ + " "); // 連結
    カウンタ--; // 自分を1減らす (算術演算子)
  } else {
    document.write("発射!");
    clearInterval(タイマーID); // タイマー停止
  }
}
```

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

17

サブ課題 カウントダウン

```
let カウンタ = 10;
let 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) {
    document.write(カウンタ + " ");
    カウンタ--; // 自分を1減らす
  } else {
    document.write("発射!");
    clearInterval(タイマーID);
  }
}
```

表示	カウンタ
	10
10_	9
10_9_	8
10_9_8_	7
10_9_8_7_	6
10_9_8_7_...	...
10_9_...2_	1
10_9_...2_1_	0
10_9_...2_1_0_	-1
10_9_...1_0_発射!	

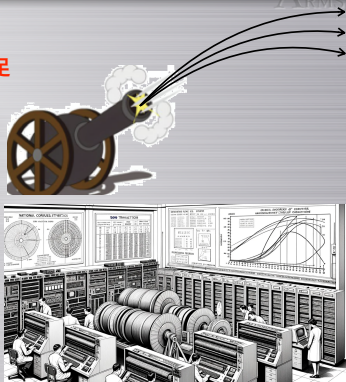
18

3. オブジェクト指向

◆ 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
もっと書きやすくしたい!



19

3. オブジェクト指向

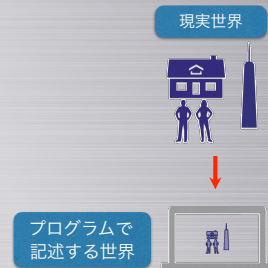
◆ 世の中はもの (object) でできている

◆ 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



20

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

21

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

1. どんな性質をもつか
2. 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
- ・ 高さ
- ・ 長さ (時間)
- ・ 状況 (再生中か停止中か)
- ・ 現在の再生位置
- ・ ボリューム

変数で表現 → プロパティ

mv1.videoWidth

mv1.videoHeight

mv1.duration

mv1.paused 代入可

mv1.currentTime 代入可

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()

22

JSのオブジェクトの定義

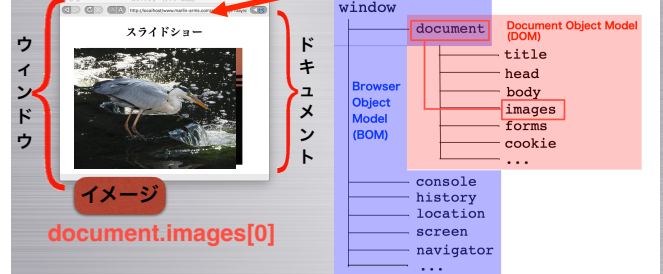
```
let mv1 = { 幅: 640, //プロパティの集まり
            高さ: 480,
            長さ: 30,
            音量: 15
          };
```

23

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)



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!

24

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
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?

変数で表現 → プロパティ

window.innerHeight

window.innerWidth

window.document

・ どんな動作をするか

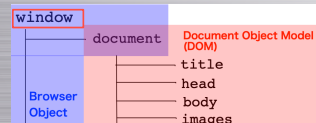
- ・ ダイアログボックスを表示
- ・ 入力を受け取る

関数で表現 → メソッド

window.alert()

window.prompt(x,y)

「window.」は省略可



25

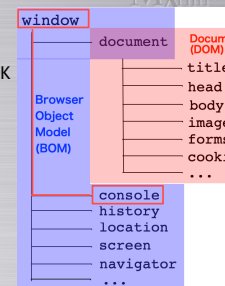
window.console

◆ メソッド

- ・ console.log("abc")
window.console.log("abc") と書いてもOK
- ・ console.clear()
- ・ console.time("timer1")
- ・ console.timeEnd("timer1")
- ・ console.table(object)
- ・ ...

詳細はこちら

<https://developer.mozilla.org/ja/docs/Web/API/console>



26

document

◆ 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")
abcのIDをもつ部分のオブジェクトを得る

```
let x = document.getElementById("abc");
x.innerHTML —— HTMLコード (●●●) がわかる
```

```
<div id="abc"> <!-- HTML -->
```

```
</div>
```

```
x.innerHTML = "▲▲▲"; // 書き換え
```

```
// document.getElementById("abc").innerHTML = ... と同じ
```

超強力 -- ブラウザの中身を自由に書き換えられる

27

document.getElementById cntdwn.js

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

setInterval()が返した値を使って終了



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

cntdwn.html

28

タイマー — 一定時間ごとの繰り返し cntdwn.js

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

setInterval()が返した値を使って終了

cntdwn.js



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

cntdwn.html

29

無名 (匿名) 関数 cntdwn2.js

1回しか使わないなら名前はいらない

```
let カウンタ = 10;
const タイマーID = setInterval(
  function() {
    document.getElementById("cntr").innerHTML = カウンタ;
    カウンタ--;
    if (カウンタ < 0) {
      document.getElementById("cntr").innerHTML = "発射";
      clearInterval(タイマーID); // タイマー停止
    }
  }, 1000);
// 第1引数
```

第2引数

```
const timerID = setInterval(カウントダウン, 1000);
function カウントダウン() {
  ...
}
```

「展開」してしまう

30

アロー関数 (ES2015) cntdwn3.js

```
let カウンタ = 10;
const タイマーID = setInterval(
  () => {
    document.getElementById("cntr").innerHTML = カウンタ;
    カウンタ--;
    if (カウンタ < 0) {
      document.getElementById("cntr").innerHTML = "発射";
      clearInterval(タイマーID); // タイマー停止
    }
  }, 1000);
// 第1引数
```

第2引数

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

31

JSのオブジェクトの定義

```
let mv1 = {
  幅: 640, //プロパティの集まり
  高さ: 480,
  長さ: 30,
  音量: 15,
  // 音量をセット: function (n) {
  //   this.音量 = n;
  // } // メソッドもプロパティの一種
};

...
if (mv1.音量 > 最大音量) {
  mv1.音量をセット(10);
  ...
}
```

32

document.getElementsByClassName

◆ Documentのメソッド

- document.getElementsByClassName("xxx")
"xxx"のクラスの要素を全部返す (「配列様のオブジェクト」)

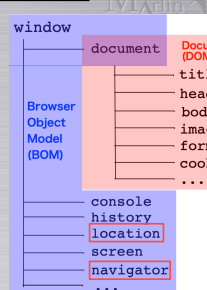
```
<div class="xxx">...</div>
...
<div class="xxx">...</div>
...
<div class="xxx">...</div>
```

33

Navigator や Location

◆ 言語や機種によるページの切り替え

```
let 言語 = navigator.language;
let 機種 = navigator.platform;
if (言語 === "ja") {
  if (機種 === "iPhone") {
    location.href = "/jp/smart.html";
  } // 現在のURL (代入可)
  else {
    location.href = "/jp/index.html";
  }
}
else {
  if (機種 === "iPhone") {
    location.href = "/en/smart.html";
  }
  else {
    location.href = "/en/index.html";
  }
}
```



34

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"
画像がえられる ←

35

スライドショー (タイマーで画像更新)

Windowのメソッド

```
let 画像番号 = 0;
const タイマーID = setInterval(
  画像を表示, 1000*2);
// 2秒ごとに画像を表示し、関数 時間を繰り返す

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と、これを使って終了できる

36

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)

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！

37

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



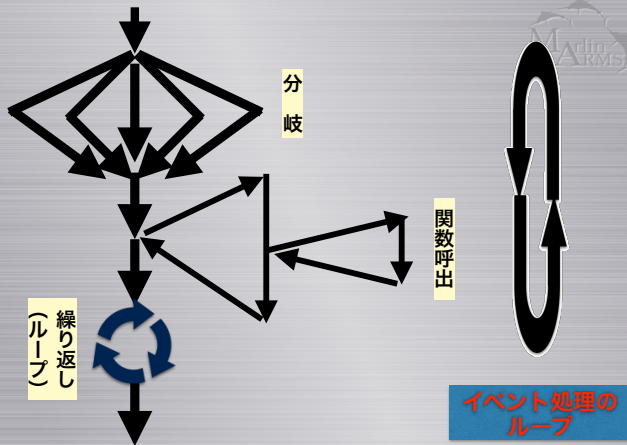
38

4. イベントの処理

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

39



40

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

41

Photo Gallery

- ◆ 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

42

JSからスタイルの変更

```




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

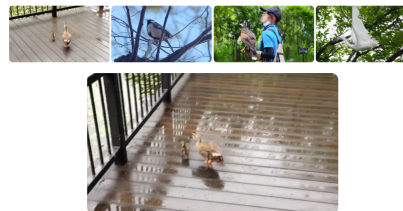
document.images[i].addEventListener("mouseout", () => {
    document.images[i+画像の枚数].style.display = "none";
});
```

43

JSからスタイルの変更（ムービー版）

周囲の鳥たち

クリックすると再生開始します。



44

デバッグ

- ◆ 一発で動くことは滅多にない
- ◆ プログラムを修正しては、再実行（ブラウザに読み込んで確認）を繰り返す。忍耐が必要
- ◆ デバッグ（debug） - プログラムがうまく動かないときに、動くようにする作業
 - デバッグの由来 de（取る）bug（虫）
 - <https://gigazine.net/news/20120910-first-computer-bug/>
 - <https://ja.wikipedia.org/wiki/継電器>
- ◆ 慣れないと自分で見つけるのは難しい。
少し考えてわからなかったら講師に聞いてください

45

「use strict;」の指定

- ◆冒頭に指定しておくと、未定義の変数がエラーになる

```
"use strict";
let x = 3;
let y = x*12;
X = y+10; // ←エラー → スペルミスがわかる
alert(x);
Uncaught ReferenceError: X is not defined
```

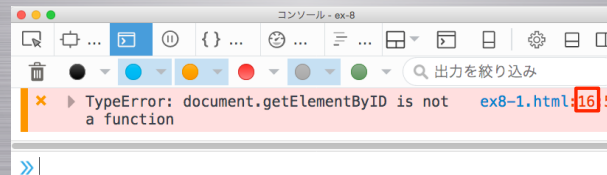
- ◆指定しておかないと、変数定義しなくても使えてしまう

```
let x = 3;
let y = x*12;
X = y+10; // ←エラーにならない。→ おかしな動作
alert(x); // 3が表示される
```

46

コンソールのエラー表示

- ◆構文におかしい場合コンソールにエラーを表示
 - メッセージを読んで理解を試みましょう
(そのうち意味がわかってくる)
- ◆行番号→どこにエラーがあるかが(だいたい)わかる



47

デバッグの道具console.log()

- ◆途中経過の表示 (どんな言語・開発環境でも使える手法)

```
"use strict";
for (let i=0; i<10; i++) {
  const ファイル名 = `pictures/picture00${i}.jpg`;
  console.log(`ファイル名=${ファイル名}`);
  const 画像タグ = ``;
  console.log(`画像タグ=${画像タグ}`);
  document.write(画像タグ);
}
```

●VS Codeの「デバッグコンソール」あるいはChromeのConsole

・Chromeの場合

右クリック (control+クリック) → [検証] → [Console]
or Ctrl+Shift+J (※+option+J)
or [その他のツール] → [デベロッパーツール] → [Console]
([表示] → [開発/管理] → [JavaScript コンソール])

48

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!
- エディタで等幅フォントを指定しておくとうりやすい
- VS Codeではエラーを示してくれる)

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

49

「約物」のまとめ - みんな半角

```
(...)
```

- 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
```

ダブルクオート (二重引用符)、シングルクオート (一重引用符)、バッククオート

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

```
``
```

その他

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

50

参考書 & 練習問題の解答

- ◆今日の練習問題解答ページ
(↓スライドのコピーにも記載)

www.musha.com/sc/jsja
ID jsbook
PW kichijoji875 (吉祥寺はなこ)

- ◆入門編のビデオをご覧いただけます

www.musha.com/sc/jsut



51

参考書 & 練習問題の解答

- ◆この講座をベースにした書籍『実践JavaScript!』

- 講座の内容をより詳しく解説

www.musha.com/sc/js



- ◆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)
- ◆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。
- ◆「もくもく会」もご検討ください

www.musha.com/sc/jsmk

52