実習編
JavaScriptで学ぶ プログラミング入門 丸１日コース

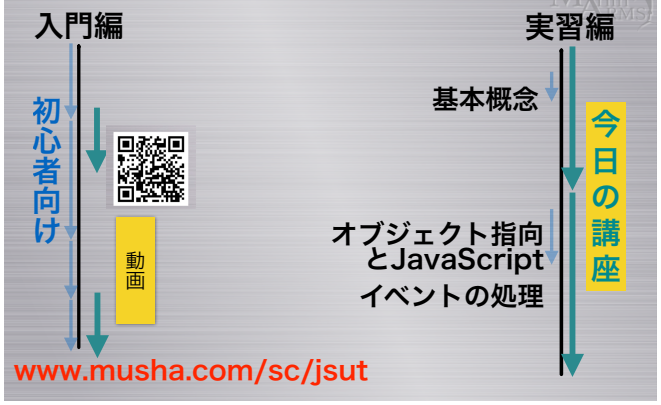
マーリンアームズ株式会社
武舎（むしゃ）広幸

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）

講座の構成



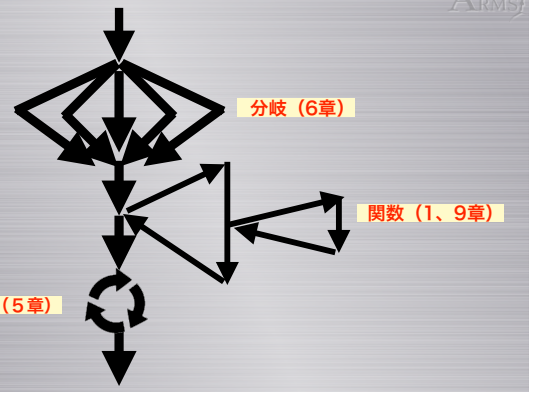
今日の時間割

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

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

1.フロー制御



2. データ構造

- 変数（2章）、定数（4章） —— 値を記憶

x	0.314
y	"大吉"
ファイル名	"pict001.jpg"

```
let x = 0.314;
const y = "大吉";
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

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II(AあるいはB) ...

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

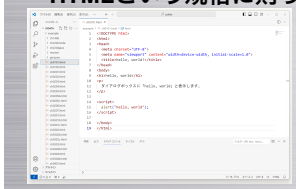
データとアプリの準備（p. 3）

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

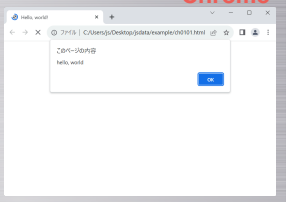
例題の実行（p. 6）

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）

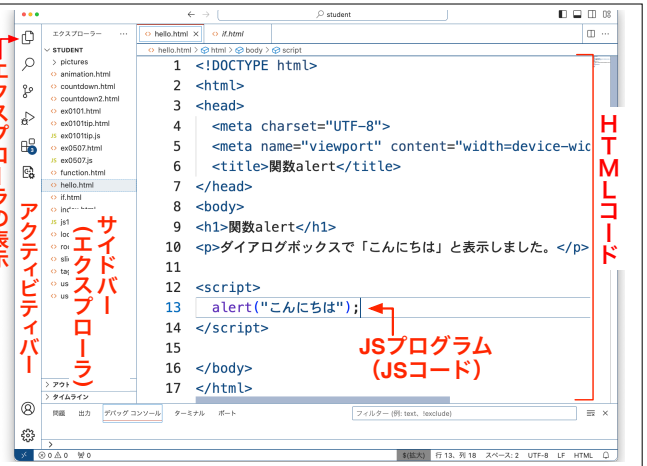


VS Code

Chrome

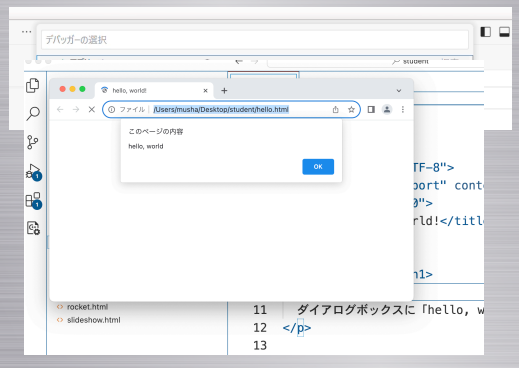


2. ブラウザ（Chrome）で表示
   ・HTMLコードをルールに従って解釈した結果



実行

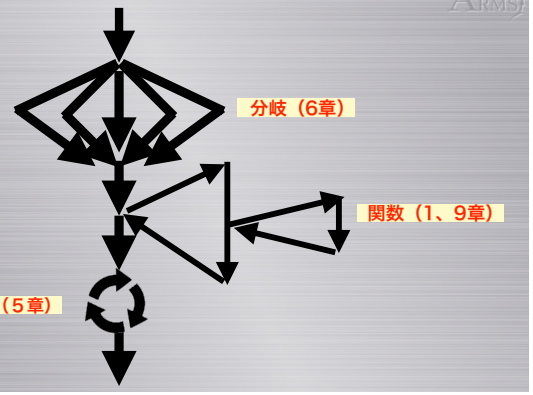
・［実行］→［デバッグの開始］



HTMLファイルの構成

```
<!DOCTYPE html>  ← DOCTYPE宣言（この文書がHTMLで書かれていることを示す）
<html>  ←htmlタグ（開始タグ）
<head>  ←headタグ
  <meta charset="UTF-8">  ←metaタグ
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  ←metaタグ
  <title>Hello, world!</title>  ←titleタグ
</head>                          ← ヘッド部（ページ全体に関する情報）
<body>  ←bodyタグ
<h1>Hello, world!</h1>  ←h1タグ  ← 見出し（<h1>から<h5>までのレベルあり）
<p>  ←pタグ
  ダイアログボックスに「Hello, world」と表示します。  ← パラグラフ（段落）
</p>
                                 ← ボディ部（コンテンツ）
<script>  ←scriptタグ
  alert("hello, world");  ← JavaScriptのプログラム <script>…</script>で囲む
</script>
</body>
</html>  ←htmlの終了タグ（閉じタグ）
```

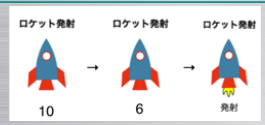

1.フロー制御



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！

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
  }
}
```

第1引数 関数　第2引数 時間間隔

1秒（1/1000*1000）ごとに繰り返す

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

サブ課題　カウントダウン

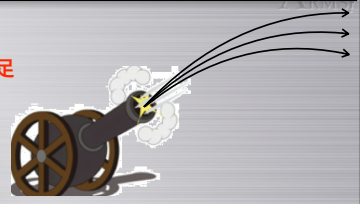
```
let カウンタ = 10;
let タイマーID =
   setInterval(カウントダウン,1000);

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);
  }    // タイマー停止
}
```

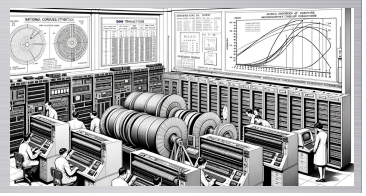
表示	カウンタ
	10
10_	9
10_9_	8
10_9_8_	7
10_9_8_7_	6
10_9_8_7_…	…
10_9_…2_	1
10_9_…2_1_	0
10_9_…2_1_0_	-1
10_9_…1_0_発射！	

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！

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

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

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　　変数で表現→プロパティ
    - 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　　関数で表現→メソッド
    - 再生開始・停止　　mv1.play()　mv1.pause()

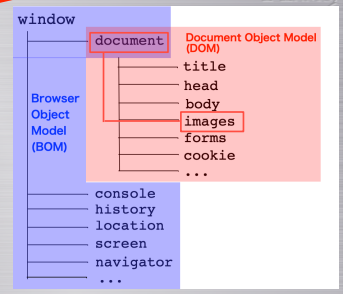
JSのオブジェクトの定義

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


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]

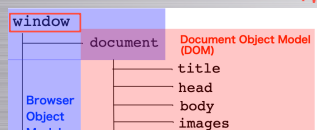


ブラウザの各項目に対応するオブジェクトが用意されている
プログラマーは何もしなくても各オブジェクトのメソッドやプロパティが利用できる！

JSとオブジェクト指向

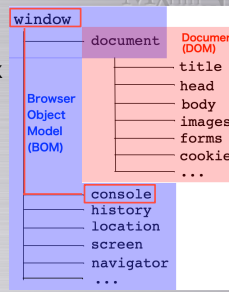
- プログラムで扱うもの（オブジェクト）を２つの側面から捉え
  たとえばブラウザのwindow
  - どんな性質を持つか　　変数で表現→プロパティ
    - 高さ（height）　　window.innerHeight
    - 幅（width）　　window.innerWidth
    - 中身は？　　window.document
  - どんな動作をするか　　関数で表現→メソッド
    - ダイアログボックスを表示　　window.alert()
    - 入力を受け取る　　window.prompt(x,y)

「window.」は省略可



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)
  - ... ...



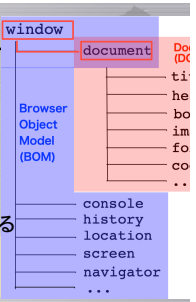
詳細はこちら
https://developer.mozilla.org/ja/docs/Web/API/console

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

document.getElementById

cntdwn.js

```
let カウント = 10;                          繰返す関数      間隔
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.html

```
<h3>ロケット発射</h3>
<div><img src="rocket.png"></div>
<div id="count"></div>
```


タイマー ―― 一定時間ごとの繰り返し

```
let カウント = 10;                          繰返す関数      間隔
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

cntdwn.html

```
<h3>ロケット発射</h3>
<div><img src="rocket.png"></div>
<div id="count"></div>
```


無名（匿名）関数

cntdwn2.js

1回しか使わないなら名前は入らない

```
let カウンタ = 10;
const タイマーID = setInterval(                    第１引数
  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);
```

「展開」してしまう

第２引数

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


アロー関数（ES2015）

cntdwn3.js

```
let カウンタ = 10;
const タイマーID = setInterval(                    第１引数
  () => {      functionを省略
    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２引数

下記ページに普通の関数との違いの詳細

https://qiita.com/suin/items/a44825d253d023e31e4d

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);
  ...
}
```


document.getElementsByClassName

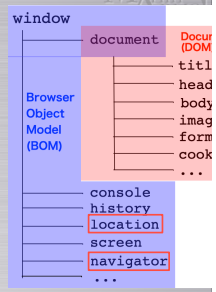
- **Documentのメソッド**
  - document.getElementsByClassName("xxx")
    "xxx"のクラスの要素を全部返す（「配列様のオブジェクト」）

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


Navigator や Location

- **言語や機種によるページの切り替え**

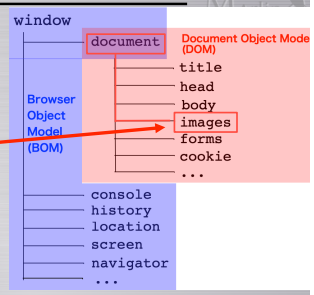
```
let 言語 = navigator.language;
let 機種 = navigator.platform;
if (言語 === "ja" ) {
  if (機種 === "iPhone") {
    location.href = "./jp/smart.html";
  }  // 現在のURL（代入も可）
  else {
    location.href = "./jp/index.html";
  }
}
else {
  if (機種 === "iPhone") {
    location.href = "./en/smart.html";
  }
  else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



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 ←

スライドショー（タイマーで画像更新）

```
let 画像番号 = 0;
const タイマーID =
        setInterval(画像を表示, 1000*2);
```

Windowのメソッド

引数に関数を渡す

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);
  }
}
```

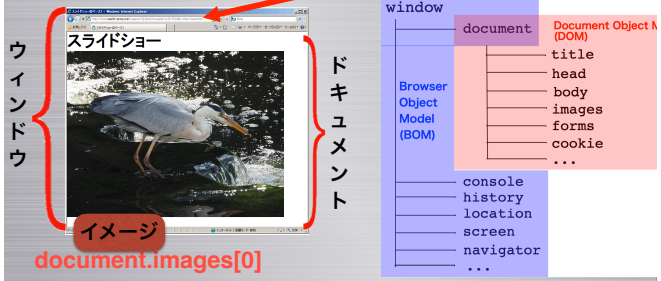
この条件でタイマーを終了

setInterval()が返した値を覚えておくと、これを使って終了できる

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！

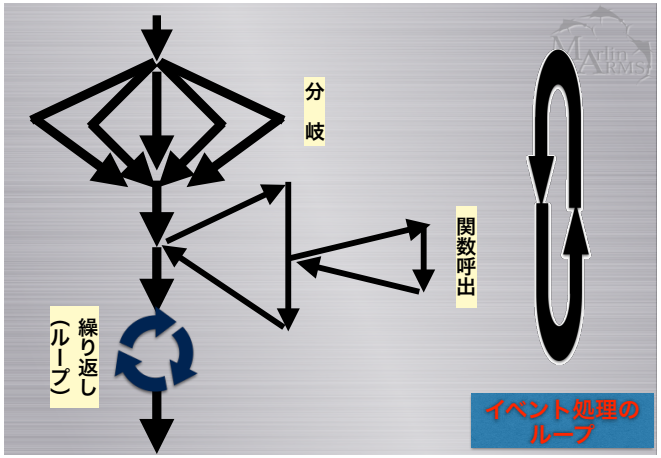
ブラウザ内はキャンバス

- ブラウザ内はプログラマーにとっての「キャンバス」
- 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

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



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
});
...
```


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

JSからスタイルの変更

```
<img src="picture00${i}.jpg" style="width: 120px;">
```

```
<img src="pictures/picture00${i}.jpg"
style="width: 480px; display: none;" >
```

inline
none

```
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";
});
```


JSからスタイルの変更（ムービー版）

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

「"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
```

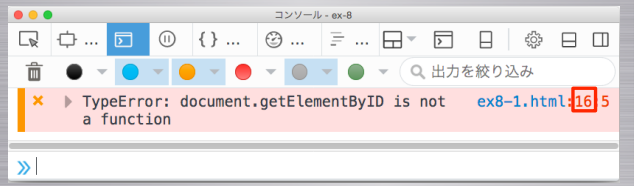
✦指定しておかないと、変数定義してなくても使えてしまう

```
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
```


コンソールのエラー表示

✦構文的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



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 = `<img src="${ファイル名}">`;
  console.log(`画像タグ=${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］）

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わかりやすい
- 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";
}
```


「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

`...${...}...`―― 可変部分付き文字列(テンプレートリテラル)
`<img src="${ファイル名}">`

その他
- ;　.　,　//　/*...*/　===　!==　&&　||
  <　>　<=　>=

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

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/scjs

✦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