



# HTML5プロフェッショナル認定試験 レベル2 ポイント解説無料セミナー

2016年7月2日

株式会社富士通ラーニングメディア

拔山雄一



# 富士通ラーニングメディアのご紹介

## 1. 会社概要

|             |   |
|-------------|---|
| 設立          | 1977年6月30日  |
| 資本金         | 3億円（全額 富士通株式会社）   |
| 売上高         | 81億円（2014年度）  |
| 従業員         | 427名（2016年3月末現在）  |
| 事業内容        | 人材育成・研修サービス(公開コース <b>1,400</b> コース、年間 <b>86,000</b> 名受講)<br>個人のお客様向けパソコン教室（富士通オープンカレッジ） |
| 関係会社        | 株式会社富士通ラーニングメディア沖縄（研修サービス・研修サービスサポート）<br>株式会社富士通ラーニングメディア・スタッフ（人材派遣）                    |
| 出資会社        | 株式会社アクト・ブレーン・ベトナム（ソフトウェア開発など）   |
| 事業所<br>関連施設 | 東京・名古屋・大阪・沖縄に5拠点（約 <b>40</b> 教室、 <b>900</b> 名以上の定員数）                                    |



品川本社



品川LC



名古屋事業所/LC



関西事業所/LC



沖縄事業所/LC

## 2. 富士通ラーニングメディアが目指すこと

### ICT人材を支える人材育成のパートナーでありつづける

- 最高水準の『知』のサービスを提供することにより、お客様の真のパートナーを目指します。
- お客様の「成長のスパイラル」をサポートします。

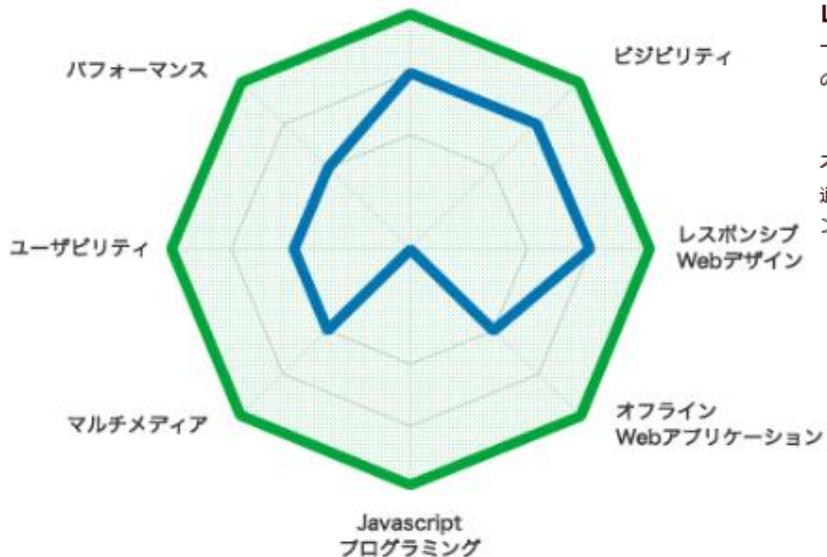


# HTML5プロフェッショナル認定資格 レベル2



Level.1   
Level.2 

HTML/HTML5  
マークアップ



## HTML/HTML5マークアップ

HTML5に関するタグの用途、構造の組み立て方に関する技術

## アクセシビリティ

JavascriptやCSS3などを用いて、デザイン仕様に沿った見やすい表示を行うための技術

## レスポンスWebデザイン

一つのソースで、スマートフォンなどの様々なデバイスの画面サイズに対応させるための技術

## オフラインWebアプリケーション

通信が常時接続状態ではない環境でも、効率的にWebコンテンツを動作させるための技術

## Javascriptプログラミング

Javascriptを使って、動的なWebコンテンツを作成する技術

## マルチメディア

3D・動画・音声ファイルなどのマルチメディアコンテンツの表示・再生に関する技術

## ユーザビリティ

ナビゲーション、地図表示など操作しやすいコンテンツを作成するための技術

## パフォーマンス

データベースや、並列処理を使ってコンテンツを効率良く高速に動作させるための技術



## 動的なコンテンツ作成のためのプログラミング言語

### ■ Ecma InternationalおよびW3Cによる標準化

JavaScriptの基本文法はEcma Internationalによって標準化されている（ECMA-262）。DOM APIやWeb APIはW3Cが標準化を担当。そのため、ブラウザ間の互換性がある程度保たれている。

### ■ 動的型付けのスクリプト言語

スクリプト言語であるため、コンパイルは不要。サーバサイド言語のPHPと同様に動的型付けである。

### ■ クライアントサイドでもサーバサイドでも実行可能

JavaScript はWebページを閲覧しているブラウザ上（クライアントサイド）で実行される。Node.jsなどの実行エンジンを使用すれば、Web サーバ上（サーバサイド）でも実行可能。



# JavaScriptのバージョン

## ■ ECMAScript 3

1999年12月公開。try-catchによる例外処理や正規表現などを追加。

## ■ ECMAScript 5

2009年12月公開（2011年には改訂版の5.1が公開）。strictモードやJSONオブジェクトの追加。

## ■ ECMAScript 6（2015）

2015年6月公開。クラスやモジュール、アロー演算子などの追加。本バージョン以降、バージョン名には公開時の西暦を採用し、1年ごとに新バージョンを公開する予定。



試験範囲はECMAScript 5.1で追加された文法までだよ!!  
ECMAScript2015を使う時は、ブラウザの対応状況に注意してね。  
2.2以降のAPIはW3Cが標準化を推進しているよ!!  
ECMAScript 2016も勧告されて開発スピードも上昇中!!



# 【参考】最新技術を使うために 1

Can I use  ? [Settings](#)

Detected your country as "Japan". Would you like to import usage data for that country?

[Import](#) [No thanks](#)

| CSS   | HTML5  | SVG  |
|---|--|--|
| <ul style="list-style-type: none"><li>▪ ::first-letter CSS pseudo-element selector</li><li>▪ ::placeholder CSS pseudo-element</li><li>▪ ::selection CSS pseudo-element</li><li>▪ :in-range and :out-of-range CSS pseudo-classes</li><li>▪ @font-face Web fonts</li><li>▪ Blending of HTML/SVG elements</li><li>▪ calc() as CSS unit value</li><li>▪ 2.1 selectors</li><li>▪ all property</li><li>▪ Animation</li><li>▪ Appearance</li><li>▪ background-attachment</li></ul> | <ul style="list-style-type: none"><li>▪ accept attribute for file input</li><li>▪ Audio element</li><li>▪ Audio Tracks</li><li>▪ Autofocus attribute</li><li>▪ Canvas (basic support)</li><li>▪ Canvas blend modes</li><li>▪ classList (DOMTokenList)</li><li>▪ Color input type</li><li>▪ contenteditable attribute (basic support)</li><li>▪ Custom Elements</li><li>▪ Custom protocol handling</li><li>▪ Datalist element</li></ul> | <ul style="list-style-type: none"><li>▪ Inline SVG in HTML5</li><li>▪ SVG (basic support)</li><li>▪ SVG effects for HTML</li><li>▪ SVG favicons</li><li>▪ SVG filters</li><li>▪ SVG fragment identifiers</li><li>▪ SVG in CSS backgrounds</li><li>▪ SVG in HTML img element</li><li>▪ SVG SMIL animation</li><li>▪ SVG fonts</li><li>▪ <b>All SVG features</b></li></ul> |

「Can I use」

<<<http://caniuse.com/>>>

ブラウザごとの対応状況を公開しているサイト



# 【参考】最新技術を使うために 2

「Mozilla Developer Network」  
<<<https://developer.mozilla.org/ja/>>>  
Web標準技術に関する情報が充実している。



## 【参考】最新技術を使うために 3

# TypeScript

learn

play

download

interact

TypeScript lets you write JavaScript the way you really want to.

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.

Any browser. Any host. Any OS. Open Source.

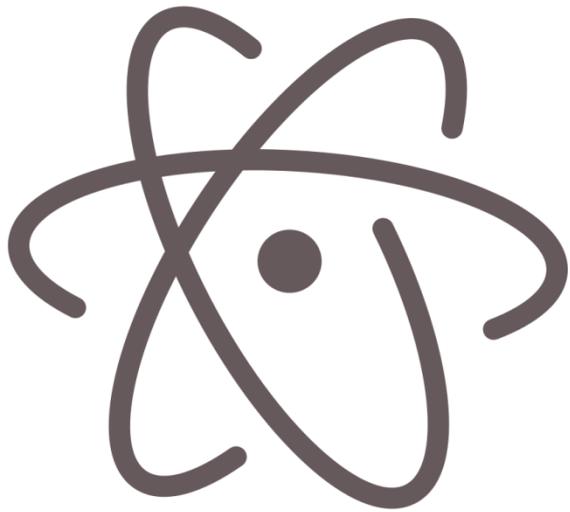
Get TypeScript Now



「TypeScript」

<<<http://www.typescriptlang.org/>>>

JavaScriptのシンタックスシュガー。ECMAScript2015で記述したソースコードをECMAScript 5などにコンパイルできる。そのほか、静的型付けなどの機能を提供してくれる。



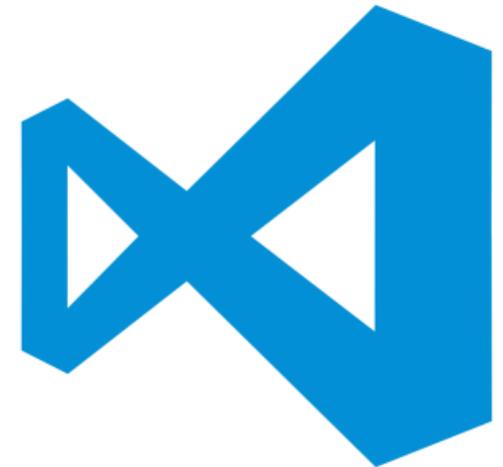
Atom

<<<https://atom.io/>>>



Brackets

<<<http://brackets.io/>>>



Visual Studio Code

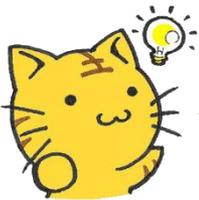
<<<https://code.visualstudio.com/>>>



# 開発ツール（ブラウザ編）

The screenshot shows a Firefox Developer Edition browser window displaying the HTML5 Professional Certification Level 2 exam outline page. The browser's developer tools are open, showing the HTML structure of the page. The HTML code includes a DOCTYPE declaration, conditional comments for Internet Explorer, and a main body structure with a container for the exam outline. The developer tools also show the CSS and JavaScript panels, with a JavaScript error message visible at the bottom: "TypeError: \$(...).find(...).css(...).transition is not a function".

各種ブラウザで「F12」キーを押すと、開発ツールが開くよ。  
JavaScriptのデバッグに役立つよ。



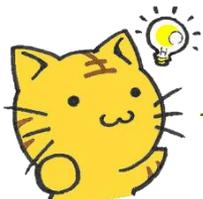


# 2.1.1 JavaScript文法

| 2.1.1 JavaScript文法 |   |
|--------------------|---|
| 重要度                | ★★★★★★★★★★ 10   |
| 出題種別               | <ul style="list-style-type: none"><li>• 知識問題</li><li>• コードリーディング問題</li><li>• 記述問題</li></ul>   |
| 説明（望まれるスキル）        | <ul style="list-style-type: none"><li>• JavaScriptの概要</li><li>• 演算子</li><li>• 配列</li><li>• 制御文</li><li>• 関数</li><li>• 型・オブジェクト</li><li>• プロパティ</li><li>• スコープ</li></ul> |

| 分類     | 型       | 値例             |
|--------|---------|----------------|
| プリミティブ | string  | 'LPI', "FLM"   |
|        | number  | 10, 20         |
|        | boolean | true, false    |
| オブジェクト | object  | new String("") |

1. `var str1 = "FLM";` // プリミティブ型
2. `var str2 = new String("LPI");` // オブジェクト型



代入した値によって、動的に型が決まるよ。

```
1. var str1 = "FLM";           // プリミティブ型
2. var str2 = new String("LPI"); // オブジェクト型
3.
4. console.log(str1);
5. console.log(str2);
6. console.log(str1.substring(0, 1));
7. console.log(str2.substring(0, 1));
```



プリミティブ型からオブジェクト型に暗黙的に変換されるから、  
プリミティブ型の変数でもメソッドを呼び出せるよ!

## ■ 関係演算子

| 演算子 | 説明                   |
|-----|----------------------|
| <   | 左辺は右辺より小さい           |
| <=  | 左辺は右辺以下である           |
| >   | 左辺は右辺より大きい           |
| >=  | 左辺は右辺以上である           |
| ==  | 左辺と右辺の値は等しい          |
| !=  | 左辺と右辺の値は等しくない        |
| === | 左辺と右辺はデータ型と値が等しい     |
| !== | 左辺と右辺はデータ型または値が等しくない |

```
1. var num1 = 10;
2. var num2 = "10";
3.
4. console.log(num1 == num2);    // true
5. console.log(num1 === num2);  // false
```

# サンプル問題 1

- 以下の変数、str1とstr2が宣言されています。選択肢のソースコードの実行結果として、trueと表示されるものを**すべて**選びなさい。

```
1. var str1 = "FLM";  
2. var str2 = new String("FLM");
```

- A) `console.log(str1 == str2);`
- B) `console.log(str1 === str2);`
- C) `console.log(typeof str1 == typeof str2);`
- D) `console.log(str1 instanceof String);`
- E) `console.log(str2 instanceof String);`

- 以下の変数、str1とstr2が宣言されています。選択肢のソースコードの実行結果として、trueと表示されるものを**すべて**選びなさい。

```
1. var str1 = "FLM";  
2. var str2 = new String("FLM");
```

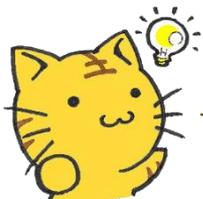
- A) console.log(str1 == str2);
- B) console.log(str1 === str2);
- C) console.log(typeof str1 == typeof str2);
- D) console.log(str1 instanceof String);
- E) console.log(str2 instanceof String);

typeof演算子は変数の型を返すよ。str1はstring、str2はobjectだよ。  
instanceof演算子はオブジェクトの型をbooleanで判定するよ。  
str1はネイティブなのでfalseが返されるんだ。



| 値         | 説明                             |
|-----------|--------------------------------|
| Infinity  | number型。無限を表す。-Infinityもあり。    |
| NaN       | number型。非数 (Not-A-Number) を表す。 |
| null      | プリミティブ型。空を表す。                  |
| undefined | プリミティブ型。未定義を表す。                |

```
1. var nan = NaN;  
2. console.log(typeof nan); // number  
3. console.log(nan + 10); // NaN  
4. console.log(nan.hoge); // undefined  
5. console.log(parseInt("a")); // NaN
```



NaNやInfinityでtypeofを使うとnumberが返ってくるよ!  
見分けるにはisNaN関数などを使うしかないので注意だよ。

## サンプル問題 2

- 以下のソースコードの実行結果として、表示されるものを**すべて**選びなさい。

```
1. try {  
2.   console.log(5 / 0);  
3. } catch (e) {  
4.   console.log("catch");  
5. } finally {  
6.   console.log("finally");  
7. }
```

- A) エラーが発生して何も表示されない
- B) NaN
- C) Infinity
- D) catch
- E) finally

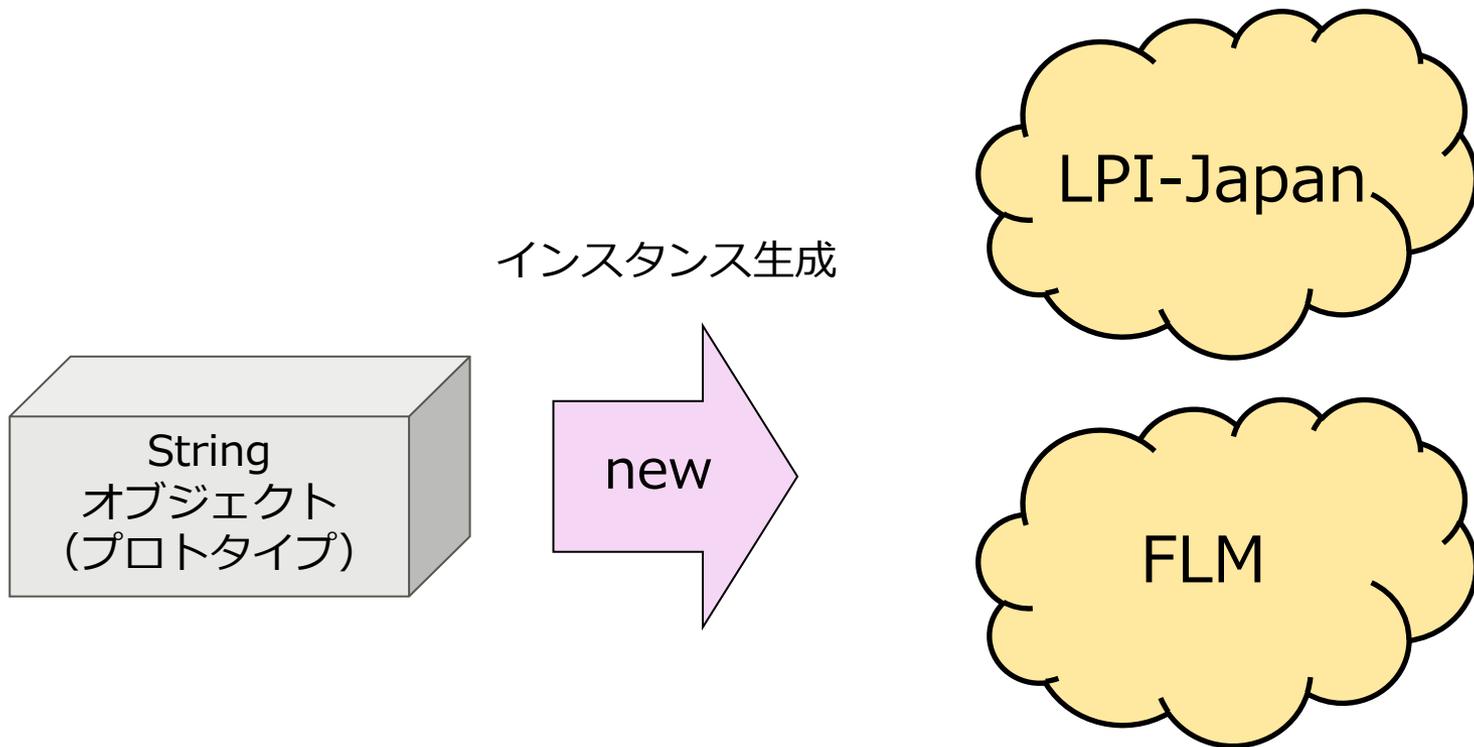
- 以下のソースコードの実行結果として、表示されるものを**すべて**選びなさい。

```
1. try {  
2.   console.log(5 / 0);  
3. } catch (e) {  
4.   console.log("catch");  
5. } finally {  
6.   console.log("finally");  
7. }
```

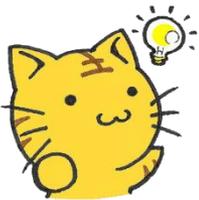
- A) エラーが発生して何も表示されない
- B) NaN
- C) Infinity
- D) catch
- E) finally

ゼロ除算をしてもエラーにならずにInfinityが返るよ。  
特殊数値はisNaNやisInfinite関数で確認できる!!



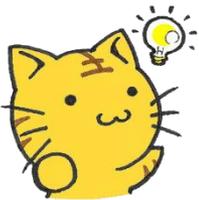


クラスではなく、オブジェクトからインスタンス生成するんだ。  
クラスと違ってオブジェクトは拡張可能だよ。  
インスタンスはオブジェクト (プロトタイプ) のプロパティ (データ)  
とメソッド (処理) を引き継ぐよ。



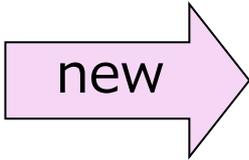
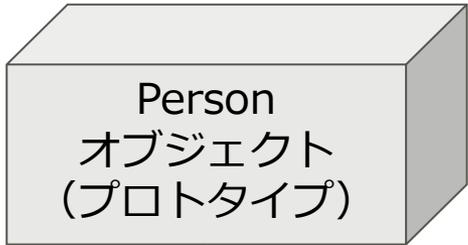
```
1. Date.prototype.getJaFullYear = function () {  
2.   return this.getFullYear() + "年";  
3. }  
4. var now = new Date();  
5. console.log(now.getJaFullYear());
```

ビルトインオブジェクトでも拡張可能だよ。でも、仕様が分かりにくくなるからビルトインオブジェクトを直接拡張することはお勧めしないよ。



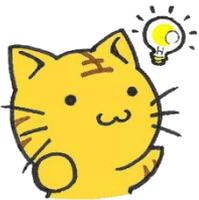
インスタンスのプロトタイプに  
greetを追加している。

インスタンス生成

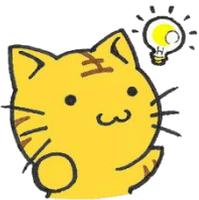


greetを使用する場合、  
プロトタイプに問い合わせる。

プロトタイプに追加しないと、  
インスタンスごとにgreetがコピーされちゃうよ。



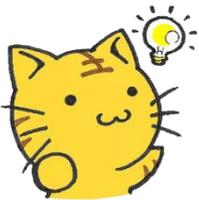
```
1. function Person(x, y) {  
2.   this.name = x;  
3.   this.age = y  
4. }  
5. Person.prototype.greet = function() {  
6.   console.log("I'm " + this.name + this.age + " years old");  
7. }  
8.  
9. var me = new Person("Yuichi", 10);  
10. me.greet();
```



コンストラクタは関数として定義するよ。  
普通の関数と見分けるために大文字を使うことが多いよ。

# 【参考】 ECMAScript2015のclassで作成

```
1. class Person2 {  
2.   constructor(x, y) {  
3.     this.name = x;  
4.     this.age = y;  
5.   }  
6.   greet() {  
7.     console.log("I'm " + this.name + this.age + " years old.");  
8.   }  
9. }
```



コードの意味としては、functionオブジェクトを使用したオブジェクト定義と同じだよ。でも、OO言語経験者にはこっちの方が分かりやすい!!

- 以下のソースコードを実行した際に、**eオブジェクト自体に定義されているプロパティやメソッドをすべて**選択しなさい。

```
1.function Employee() {  
2.  this.name = "foge";  
3.  this.dept = "Development";  
4.  this.work = function() {  
5.      console.log("do test");  
6.  }  
7.}  
8.Employee.prototype.manage = function() {  
9.  console.log("manage");  
10.}  
11.var e = new Employee();
```

- A) name            C) work
- B) dept            D) manage

- 以下のソースコードを実行した際に、**eオブジェクト自体に定義されているプロパティやメソッドをすべて**選択しなさい。

```
1.function Employee() {  
2.  this.name = "foge";  
3.  this.dept = "Development";  
4.  this.work = function() {  
5.      console.log("do test");  
6.  }  
7.}  
8.Employee.prototype.manage = function() {  
9.  console.log("manage");  
10.}  
11.var e = new Employee();
```

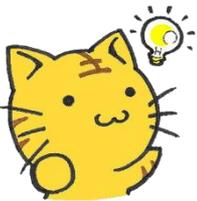
manageはeオブジェクトの  
プロトタイプに定義されているよ。  
Object.getOwnPropertyNames()  
で確認もできる。

- (A) dept
- (B) name
- (C) work
- D) manage



```
1. var nums = [10, 20, 30];  
2. for (var num in nums) {  
3.     console.log(num);    // 0、1、2と出力  
4. }
```

for/inを配列で使用すると、配列の要素に対して処理を繰り返すよ。  
上述の例では配列の要素番号が出力されてるんだ!!  
挙動がわかりにくいので、使用しない方がいいよ。



## サンプル問題 4

- 以下のソースコードを実行した結果、表示されるものとして正しいものを選びなさい。

```
1. var nums = [10, 20, 30];  
2. nums.foge = "hoge";  
3. for (var num in nums) {  
4.   console.log(num);  
5. }
```

- A) 10 20 30
- B) 10 20 30 fgefoge
- C) 0 1 2
- D) 0 1 2 foge

- 以下のソースコードを実行した結果、表示されるものとして正しいものを選びなさい。

```
1. var nums = [10, 20, 30];  
2. nums.foge = "hoge";  
3. for (var num in nums) {  
4.   console.log(num);  
5. }
```

- A) 10 20 30
- B) 10 20 30 fgefoge
- C) 0 1 2
- D) 0 1 2 foge

自作のプロパティが実行されるよ。  
ECMAScript2015で追加された、for/ofを使うと  
選択肢Aと同じ結果になるよ。





## 2.2.1 イベント

| 2.2.1 イベント  |   |
|-------------|---|
| 重要度         | ★★★★★★★ 8   |
| 出題種別        | <ul style="list-style-type: none"><li>• 知識問題</li><li>• コードリーディング問題</li><li>• 記述問題</li></ul>                                     |
| 説明（望まれるスキル） | <ul style="list-style-type: none"><li>• JavaScriptのページ読み込みや、ユーザ操作によって発生するイベントの発生タイミングを理解しており、イベント処理を行うコードを記述することができる。</li></ul> |



# イベントとは

① マウスオーバーでパネルを表示

- 資格取得のメリット
- 今後取得したい資格No.1
- Web資格の中で最も求められている理由
- 合格者の声

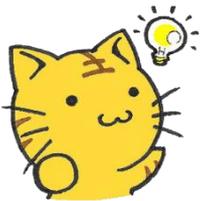
② マウスアウトでパネルを非表示

HTML5 Professional Certification

HTML5 レベル1  
Markup Professional

HTML5 OSS-DB  
受験チケット×認定教材  
キャンペーン  
受験バウチャーを購入すると

ユーザーの動作（イベント）に合わせて処理を実行できるよ。



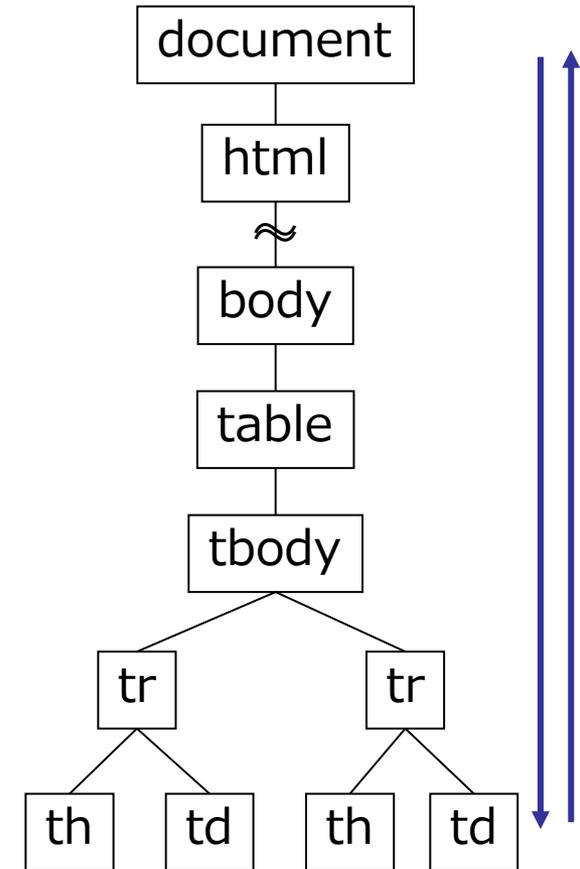
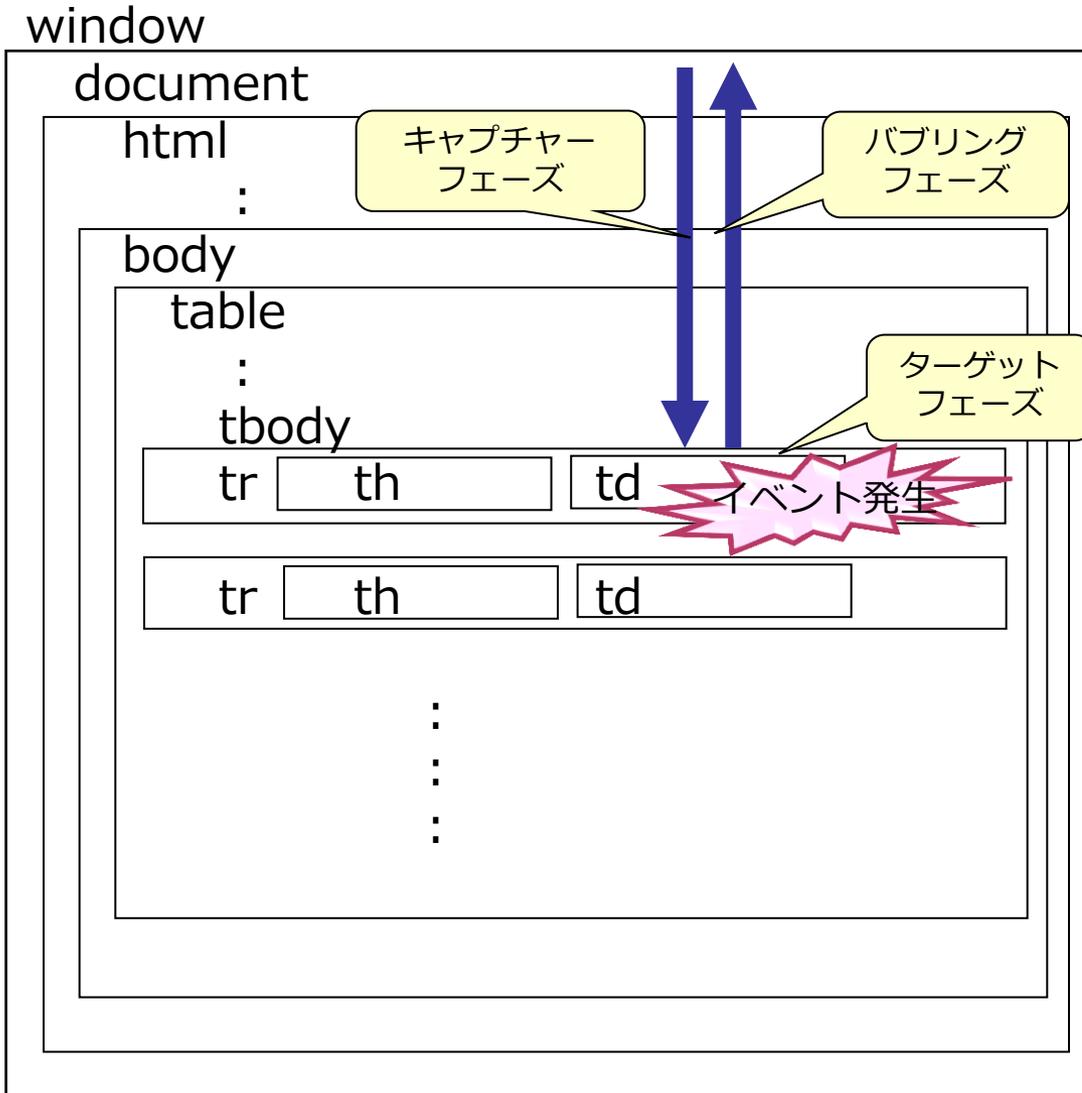
```
1. <button id="btn" onclick="alert('FLM')"></button>  
2.  
3. var btn = document.getElementById("btn");  
4. btn.onclick = function(){};  
5. btn.addEventListener("click", function() {}, false);
```

JavaScriptのイベント登録の方法は3つ。  
一般的には、addEventListenerが使われるよ。なぜなら、

- イベントに1つ以上のハンドラを追加できる!
- キャプチャリングとバブリングを制御できる!

ちなみに、IE6-8は対応していなかったので  
MS独自拡張のattachEventも使われていたよ。





DOM ツリー

- 以下のソースコードがあるWebページ上で、「click me」ボタンをクリックした際に表示される文字列として正しいものを1つ選びなさい。なお、「→」は表示順を表します。

```
1. <div onclick="alert('div click')">
2.     <button id="btn">click me</button>
3. </div>
4. <script>
5.     var btn = document.getElementById("btn");
6.     btn.addEventListener("click", function(){
7.         alert("button click");
8.     });
9. </script>
```

- A) button click → div click
- B) div click → button click
- C) button click
- D) div click

- 以下のソースコードがあるWebページ上で、「click me」ボタンをクリックした際に表示される文字列として正しいものを1つ選びなさい。なお、「→」は表示順を表します。

```
1. <div onclick="alert('div click')">
2.     <button id="btn">click me</button>
3. </div>
4. <script>
5.     var btn = document.getElementById("btn");
6.     btn.addEventListener("click", function(){
7.         alert("button click");
8.     });
9. </script>
```

- A) button click → div click
- B) div click → button click
- C) button click
- D) div click

button要素のクリックイベントが伝播して、バブリングフェーズでdiv要素のclickイベントが発生するよ。



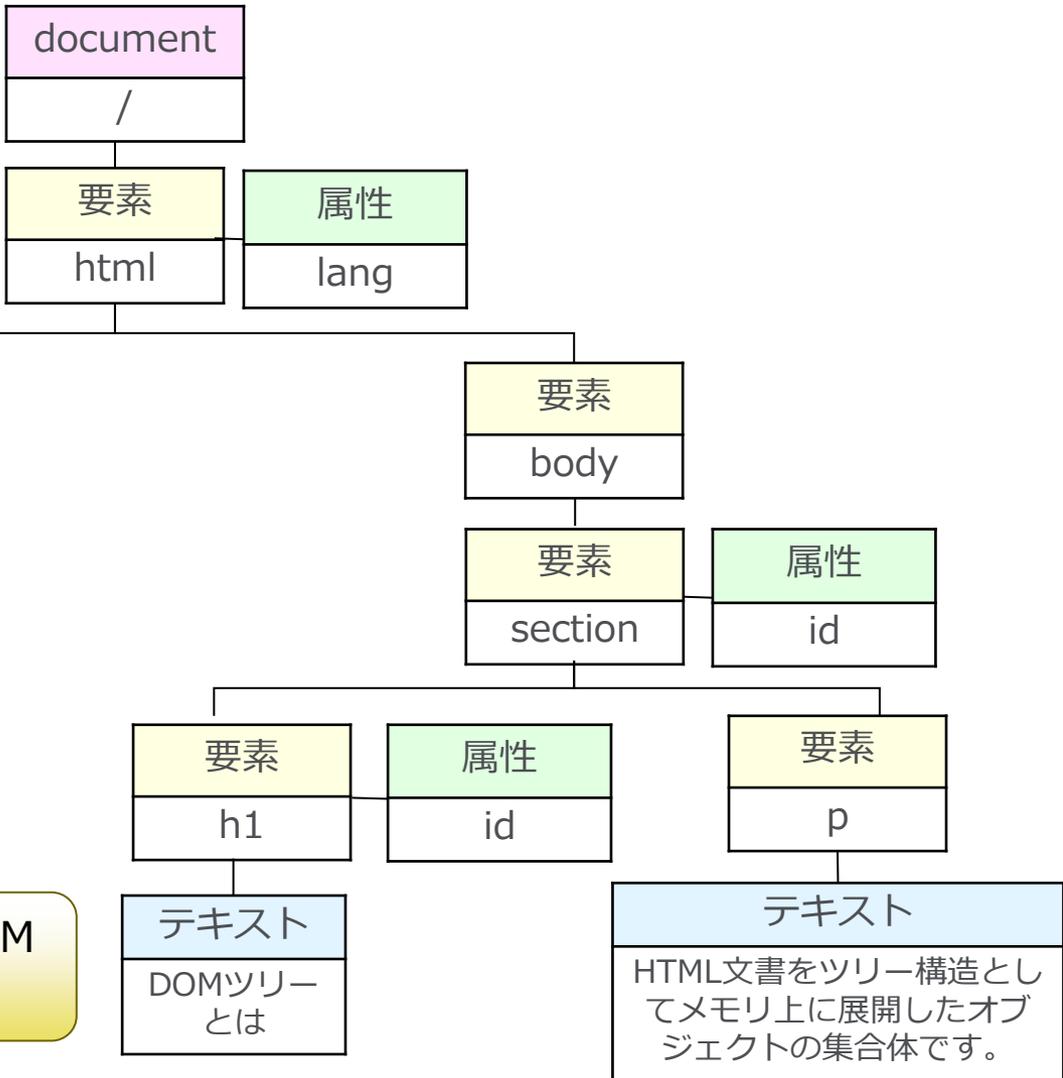


## 2.2.2 ドキュメントオブジェクト／DOM

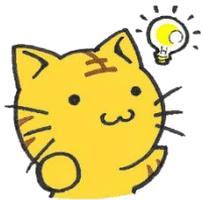
| 2.2.2 ドキュメントオブジェクト／DOM |   |
|------------------------|---|
| 重要度                    | ★★★★★★ 6  |
| 出題種別                   | <ul style="list-style-type: none"><li>• 知識問題</li><li>• コードリーディング問題</li></ul>                                    |
| 説明（望まれるスキル）            | <ul style="list-style-type: none"><li>• イベント発生時などにDOMを利用して、HTMLの内容を読み込む、書き換える処理を行うコードの記述方法について理解している。</li></ul> |



# DOM (Document Object Model) とは

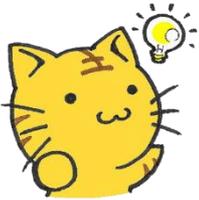


HTML要素はツリー構造のDOMとして解析されるよ。



```
1. <div id="message">
2. </div>
3. <script>
4. var mess = document.getElementById("message");
5. mess.innerHTML = "<p>Fujitsu Learning Media.</p>";
6. </script>
```

DOM (Document Object Model) でHTMLを操作できる!!  
業務ではjQueryなどのライブラリで操作する方が一般的だよ。



- JavaScriptを使用して、「実行前」のHTMLを「実行後」のように書き換えたい。正しく変更できるものを1つ選びなさい。なお、「target」にはdiv要素が、「yesButton」にはbutton要素が参照されているものとする。

- 実行前

```
1. <div id="target">
2.   <button id="yes">yes</button>
3. </div>
```

- 実行後

```
1. <div id="target">
2.   <button id="yes">yes</button><button>no</button>
3. </div>
```

- A) `target.innerHTML = "<button>no</button>";`
- B) `var noButton = document.createElement("button");`  
`noButton.innerHTML = "no";`  
`target.insertBefore(noButton, yesButton);`
- C) `target.hasAttribute("<button>no</button>");`
- D) `var noButton = document.createElement("button");`  
`noButton.innerHTML = "no";`  
`target.appendChild(noButton);`

- JavaScriptを使用して、「実行前」のHTMLを「実行後」のように書き換えたい。正しく変更できるものを1つ選びなさい。なお、「target」にはdiv要素が、「yesButton」にはbutton要素が参照されているものとする。

- 実行前

```
1. <div id="target">
2.   <button id="yes">yes</button>
3. </div>
```

- 実行後

```
1. <div id="target">
2.   <button id="yes">yes</button><button>no</button>
3. </div>
```

- A) target.innerHTML = "<button>no</button>";
- B) var noButton = document.createElement("button");  
noButton.innerHTML = "no";  
target.insertBefore(noButton, yesButton);
- C) target.hasAttribute("<button>no</button>");
- D) var noButton = document.createElement("button");  
noButton.innerHTML = "no";  
target.appendChild(noButton);**





## 2.8.1 Web Storage

| 2.8.1 Web Storage |   |
|-------------------|---|
| 重要度               | ★★★★ 4  |
| 出題種別              | <ul style="list-style-type: none"><li>• 知識問題</li><li>• コードリーディング問題</li></ul>                        |
| 説明（望まれるスキル）       | <ul style="list-style-type: none"><li>• Web Storageの特徴や仕様を理解し、読み込み・書き込みを行うコードを理解することができる。</li></ul> |



# Web Storageの特徴

- ブラウザへのデータの保存や取得ができる  
ブラウザにデータを保存するための機能です。ブラウザを閉じるまで、またはブラウザを閉じた後もデータを残せます。
- Key-Value配列の形式で複数のデータを保存できる  
データはKey-Value配列で保存されます。そのため、キーを指定して値を格納できます。
- サイズの大きなテキストデータを保存できる  
データの保存容量がオリジンごとに5MB程度あるため、データ件数が多いJSON など、大きいサイズのテキストデータでも格納できます。ただし、ブラウザごとに保存容量は異なります。
- クロスオリジンのアクセスが制限されている  
同一オリジンのWebページが保存したデータのみ取得できます。そのため、誤って異なるオリジンのWebページが保存したデータを変更したり、参照されたりする危険が低いです。

|       | Cookie                  | Web Storage             | Indexed Database               | WebSQL                |
|-------|-------------------------|-------------------------|--------------------------------|-----------------------|
| 保存容量  | 4KB                     | 5MB-10MB                | 5MB-10MB                       | 5MB-10MB              |
| 保存期間  | 有期限                     | 無期限/セッション               | 無期限                            | 無期限                   |
| 送受信   | リクエストごと                 | JSで操作時のみ                | JSで操作時のみ                       | JSで操作時のみ              |
| データ形式 | 文字列                     | 文字列                     | ネイティブ/<br>オブジェクト               | ネイティブ/<br>オブジェクト      |
| 非同期   | ×                       | ×                       | ○                              | ○                     |
| 特徴    | 設定によっては、HTTP通信でも送受信される。 | シンプルなAPIで、大容量データを保存できる。 | 複雑なデータを扱える。APIが複雑なため、実装が少々難しい。 | SQL文ライクの文法でデータを操作できる。 |
| 備考    | 最も実装が進んでいる。             |                         | DBとは操作方法が異なるので注意。              | 仕様策定が中止されている。         |

```
1. sessionStorage.setItem("Item", "Book");  
2. // sessionStorage.Iem = "Book"; でも同じ動作  
3. var storage = sessionStorage.getItem("Item");  
4. console.log(storage);  
5. sessionStorage.removeItem("Item");
```

sessionStorageとlocalStorageの違いは保存期間だけ!!  
sessionStorageはセッションが終了すると削除、  
localStorageは永続保存させるよ。





# サンプル問題 7

■ Web Storageの説明として正しいものを1つ選びなさい。

- A) key-value形式でデータを保存できる。
- B) stringやBlobを格納できる。
- C) 非同期で動作するため、パフォーマンスが優れている。
- D) cookieよりも容量が小さい。

■ Web Storageの説明として正しいものを1つ選びなさい。

- A) key-value形式でデータを保存できる。
- B) stringやBlobを格納できる。
- C) 非同期で動作するため、パフォーマンスが優れている。
- D) cookieよりも容量が小さい。

Web StorageはシンプルなAPIで使いやすいけど、同期処理でパフォーマンスはよくないよ。また、文字列しか格納できない仕様になっているんだ。





## 2.9.1 WebSocket

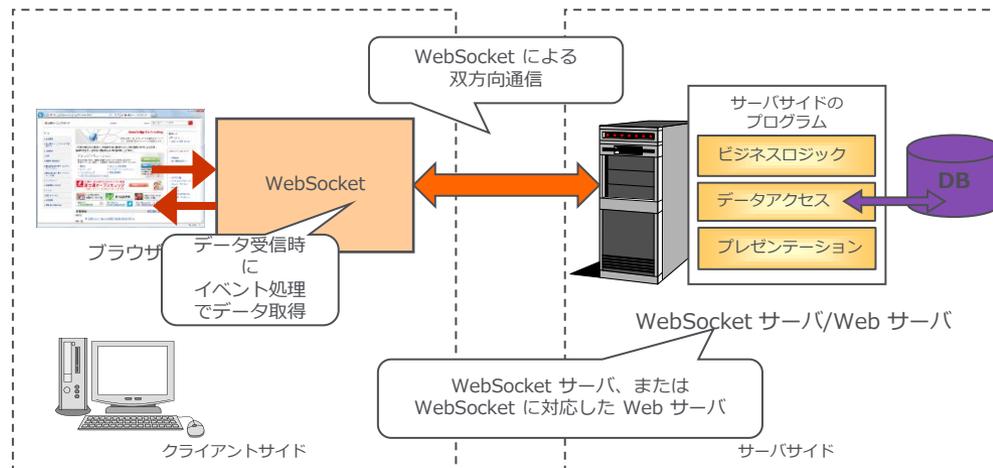
| 2.9.1 WebSocket |   |
|-----------------|---|
| 重要度             | ★★ 2  |
| 出題種別            | <ul style="list-style-type: none"><li>知識問題</li></ul>  |
| 説明（望まれるスキル）     | <ul style="list-style-type: none"><li>WebSocketの特徴を理解し、通信を行うにあたって必要な知識について理解している。</li></ul> |

## ■ 双方向通信を実装できる

クライアントとWebSocketサーバ間で接続を維持した双方向通信を実装できます。XMLHttpRequestやServer-Sent Eventsなど、一回の通信ごとにリクエストとレスポンスを行う HTTP ベースの通信と比較して、常時接続して双方向通信を行うWebSocketは、送受信処理の効率が高いです。

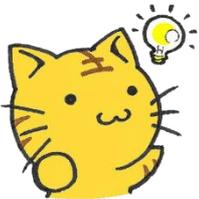
## ■ WebSocketサーバを使用する

通信方式として HTTP ではなくWebSocket Protocolを使用します。そのため、サーバ側はWebSocket Protocolに対応したWebSocketサーバを使用する必要があり、WebSocketに対応していない従来のWebサーバは使用できません。



|        | Server-Sent Events | WebSocket          | XMLHttpRequest           |
|--------|--------------------|--------------------|--------------------------|
| プロトコル  | HTTP               | WebSocket Protocol | HTTP                     |
| 非同期通信  | ○                  | ○                  | ○                        |
| 双方向通信  | ×                  | ○                  | ○                        |
| プッシュ通知 | ○                  | ○                  | ×                        |
| 通信効率   | △                  | WebSocket サーバ      | △                        |
| 特徴     | Webサーバからの一方向通信に特化。 | 双方向のリアルタイム通信。      | ロングポーリングによってリアルタイム通信を実現。 |
| 備考     | IE/Edgeが未実装。       | チャットなどで広く使用されている。  | ブラウザ実装状況が最も良い。           |

```
1. var socket = new WebSocket("ws://localhost:8888/");
2. socket.send("Hello, World");
3. socket.onmessage = function (e) {
4.     console.log(e.data);
5. }
```



WebSocketプロトコルを使用するので、  
URLのスキーマはwsかwssになるよ。  
サーバに情報を送信するにはsendを、  
受信するにはmessageイベントを使うよ。

- WebSocketの説明として正しいものを1つ選びなさい。
- A) リアルタイム通信を実現する場合、1ラウンドトリップが終了するごとに再接続が必要になる。
- B) サーバからクライアントに対して、情報送信をできるが、クライアントからサーバにデータ送信はできない。
- C) サーバを介さずにクライアント同士のP2P（Peer to Peer）通信を実現する。
- D) クライアントとサーバを常時接続し、効率的なデータの送受信を実現する。

- WebSocketの説明として正しいものを1つ選びなさい。
- A) リアルタイム通信を実現する場合、1ラウンドトリップが終了するごとに再接続が必要になる。
- B) サーバからクライアントに対して、情報送信をできるが、クライアントからサーバにデータ送信はできない。
- C) サーバを介さずにクライアント同士のP2P（Peer to Peer）通信を実現する。
- D) クライアントとサーバを常時接続し、効率的なデータの送受信を実現する。**

A)はXMLHttpRequest、B)はServer-Sent Events、C)はWebRTCの説明だよ。



| 2.10.1 Geolocation APIの基本と位置情報の取得 |   |
|-----------------------------------|---|
| 重要度                               | ★★ 2  |
| 出題種別                              | <ul style="list-style-type: none"><li>知識問題</li></ul>                                  |
| 説明（望まれるスキル）                       | <ul style="list-style-type: none"><li>Geolocation APIの概要と利用時の注意点について理解している。</li></ul> |

## ■ GPS を使用した位置情報取得

モバイル端末などに搭載されている GPS 受信機にブラウザ経由でアクセスし、正確な位置情報を取得できます。GPS 受信機が使用できない場合でも、ネットワーク装置と連携し、大まかな位置情報を取得できます。

## ■ ユーザーによる使用可否の設定

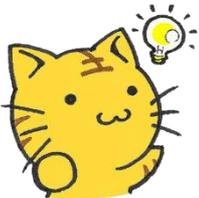
位置情報はユーザーのプライバシーにかかわる情報です。そのため、ユーザーが位置情報の使用可否を設定できます。つまり、必ずしも位置情報を取得できるわけではありません。



## ■ 地図サービスとの連携

既存の地図サービスの中には、Geolocation APIを使用したサービス連携が可能なものもあります。

```
1. navigator.geolocation.getCurrentPosition(  
2.   function (position) {  
3.     //Positionオブジェクトによる取得  
4.     var latitude = position.coords.latitude; //緯度  
5.     var longitude = position.coords.longitude; //経度  
6.     var accuracy = position.coords.accuracy; //誤差  
7.     var altitude = position.coords.altitude; //高度  
8.   },  
9.   function (err) {  
10.    console.error(err.message);  
11.});
```



getCurrentPositionメソッドの代わりにwatchPositionメソッドを使うと位置情報が変更するたびに処理を呼び出せるよ。

- Geolocation APIの説明として正しいものを1つ選びなさい。
- A) GPS受信機が存在しない端末でWebページを閲覧した場合、位置情報を取得することができない。
- B) ユーザーが情報提供を拒否した場合でも、特権設定をしておけば位置情報を取得できる。
- C) タイムアウトなどのエラーが発生した場合、`getCurrentPosition`メソッドの第1引数に指定した関数が実行される。
- D) 取得できる位置情報として、緯度や経度、高度などがある。

- Geolocation APIの説明として正しいものを1つ選びなさい。
- A) GPS受信機が存在しない端末でWebページを閲覧した場合、位置情報を取得することができない。
- B) ユーザーが情報提供を拒否した場合でも、特権設定をしておけば位置情報を取得できる。
- C) タイムアウトなどのエラーが発生した場合、getCurrentPositionメソッドの第1引数に指定した関数が実行される。
- D) 取得できる位置情報として、緯度や経度、高度などがある。**

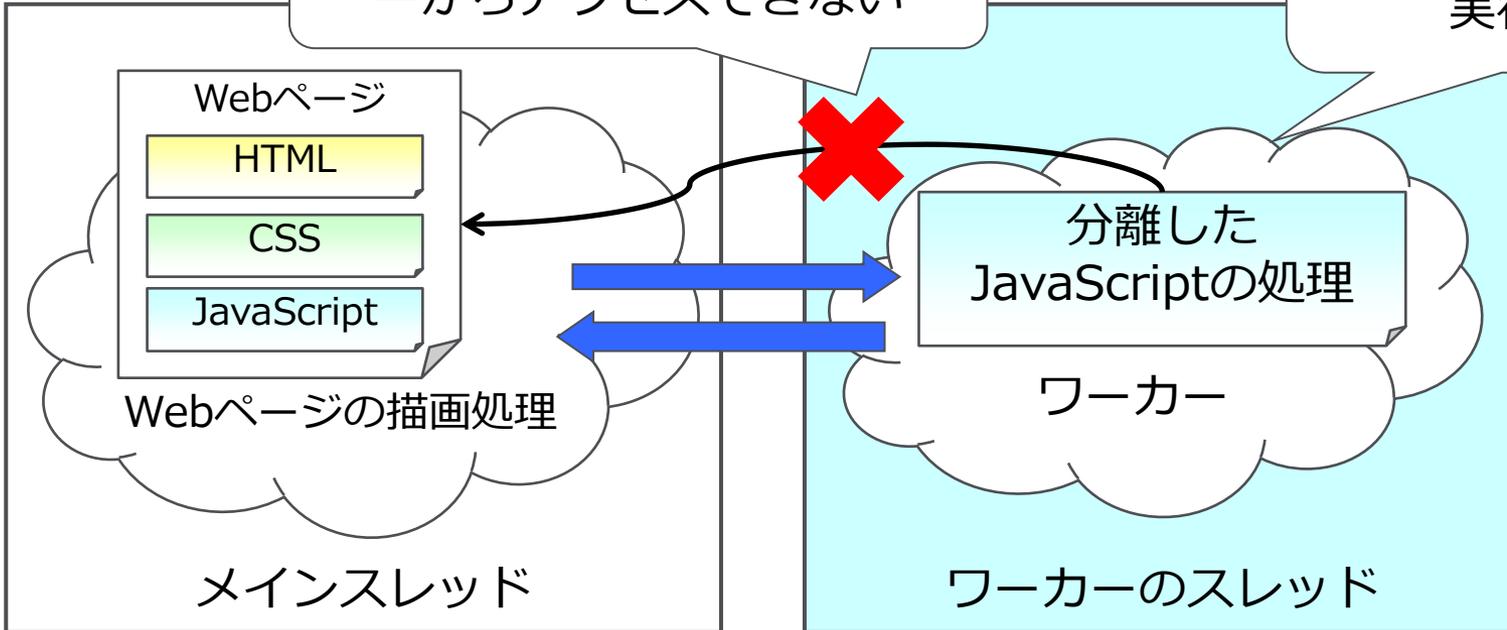
Geolocation APIでは、特権設定など、セキュリティリスクになる設定は使用できないよ。



| 2.11.1 並列処理の記述 |   |
|----------------|---|
| 重要度            | ★★★★ 4  |
| 出題種別           | <ul style="list-style-type: none"><li>• 知識問題</li><li>• コードリーディング問題</li></ul>                                  |
| 説明（望まれるスキル）    | <ul style="list-style-type: none"><li>• Web Workersの特徴を理解し、並列処理やエラーの検出を行うコードを読みどのように動作するか理解する事ができる。</li></ul> |

Webページ上の処理にワーカーからアクセスできない

JavaScriptの処理をバックグラウンドで実行する



ブラウザの実行プロセス

クライアント



# Web Workersの特徴

- ワーカーを分離することで、JavaScript の処理を並列に処理できる  
JavaScriptの処理単位であるワーカーは、メインスレッドの Web ページの処理から分離されます。複数のワーカーを実行することで JavaScript の同時実行制御が行えます。

- ワーカーの処理をJSファイルに記述する

並列処理するプログラムは、呼び出し元とは別のJSファイルに記述する必要があります。Web Workersが書かれたファイルは、インスタンス生成したタイミングで読み込まれます。

- メッセージを使用してワーカー間のデータを交換する

メインスレッドとワーカー間は別スレッドのため、データを共有していません。そのため、DOMへのアクセスやwindowオブジェクト（XHRなどの一部オブジェクトはアクセス可能）に直接アクセスできません。データを送受信するためには、postMessageメソッドを使用します。



# Web Workers

## ■ index.html

```
1. var work = new Worker("worker.js");
2. work.postMessage(100);
3. work.onmessage = function(e) {
4.     console.log(e.data);
5. }
```

## ■ worker.js

```
1. self.onmessage = function(e) {
2.     var data = e.data;
3.     for (var i = 0; i < 10000; i++) {
4.         data += i;
5.     }
6.     self.postMessage(data);
7. }
```

- 以下のソースコードを実行して、エラーになる行を1つ選びなさい。

- index.html

```
1. var work = new Worker("worker.js");
2. work.postMessage(100);
3. work.onmessage = function(e) {
4.     console.log(e.data);
5. }
```

- worker.js

```
I. self.onmessage = function(e) {
II.     var data = e.data;
III.    document.getElementById("message").innerHTML = data;
IV.    postMessage(true);
V. }
```

- A) 1行目
- B) 2行目
- C) Ⅲ行目
- D) Ⅳ行目

- 以下のソースコードを実行して、エラーになる行を1つ選びなさい。

- index.html

```
1. var work = new Worker("worker.js");
2. work.postMessage(100);
3. work.onmessage = function(e) {
4.     console.log(e.data);
5. }
```

- worker.js

```
I. self.onmessage = function(e) {
II.     var data = e.data;
III.    document.getElementById("message").innerHTML = data;
IV.    postMessage(true);
V. }
```

- A) 1行目
- B) 2行目
- C) Ⅲ行目
- D) Ⅳ行目

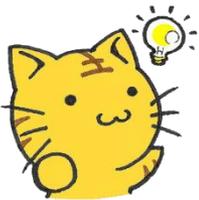
Web Workersからdocumentオブジェクトにはアクセスできないよ。postMessageメソッドはグローバル関数だから、selfがなくても大丈夫!



# 【参考】 ECMAScript2015の非同期処理 (1/2)

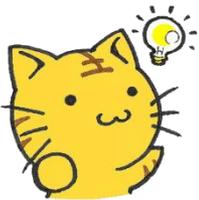
```
1. var data = 10;
2. var p1 = new Promise(function (resolve, reject) {
3.     for (var i = 0; i < 10000; i++) {
4.         data += i;
5.     }
6.     if(data >= 0){
7.         resolve(data);
8.     }
9.     reject(Error("data is negative."))
10. });
```

PromiseはECMAScript2016で追加された非同期処理だよ!!  
成功時に呼び出す値をresolveに、失敗時に呼び出す値はrejectに登録するんだ。Battery Status APIのような比較的新し目のAPIはPromiseを採用しているから、文法を知っておくとAPIも覚えやすくなるよ。



```
1.p1.then(function (data) {  
2.  alert(data);  
3.})  
4. .catch(function (err) {  
5.   alert(err.message);  
6. });
```

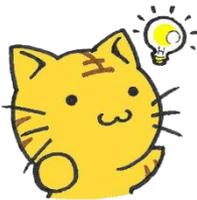
Promiseの処理が成功するとthenメソッドが実行されるよ。  
失敗するとcatchメソッドの方が実行されるよ。  
非同期処理の失敗/成功時の処理を分かりやすく書けるんだ。



# 【参考】 ECMAScript2015のArrow関数

```
1. var p1 = new Promise((resolve, reject) => { });  
2. p1.then(data => { })  
3. .catch(err => { });
```

Arrow関数を使うと、関数定義を「(引数) => {}」の書式で書けるよ。  
引数が1つの場合は、「( )」も省略できるんだ。  
JavaScriptはたくさん関数を書くから、ちょっとだけ書きやすくなるよ。



# LPI-JAPAN HTML5 Professional Certification

Open the Future with **HTML5**.



**FUJITSU**

shaping tomorrow with you