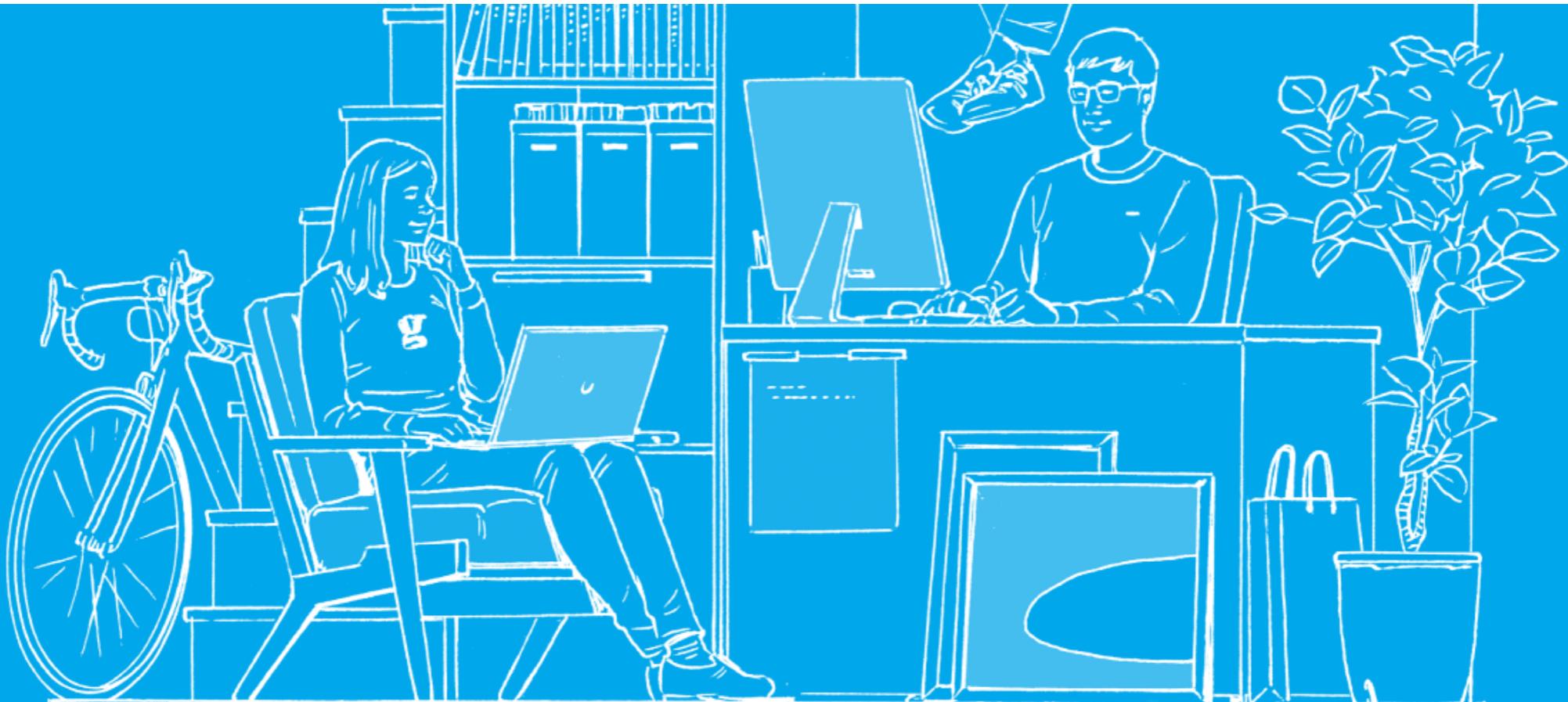




# JavaScript

## ~First training~



# G'sラーニングシステム sample

<https://github.com/GsCodeSample>

# 拡張機能

- Japanese Language Pack for Visual Studio Code
- jQuery Code Snippets
- EvilInspector
- Live Server
- Bracket Pair Colorize
- Auto Rename Tag
- vscode-icons
- PHP IntelliSense



<https://youtu.be/hizfrmSWXFs>

# 1回目アジェンダ

- JavaScriptできること（一部）
- JavaScript概要/基礎
- 変数/演算子と計算
- 条件分岐/IF分岐処理
- ランダム関数(Math)
- jQuery
- 演習：おみくじアプリ
- 課題：じゃんけんアプリ

# 授業ルール 円滑に授業をすすめるために

- CODE打ち間違いが多い人は文字を大きくすること
- CODE打ったのに「動かない」！  
ブラウザ右クリック→検証→「console」エラー確認！！
- 常に保存！「Ctrl + S」キーで保存！  
自動保存機能をONにしておけばOK！
- 疑問があっても授業内で立ち止まるな！  
できるだけ1度授業を通してやろう！  
授業は小説と同じ、最後で全てが繋がることが多々ある。

見るだけでOK マークがあるよ！

# 授業用 Sample

<https://github.com/yamazakidaisuke/GsCodeSample>

# Javascript概要

# Javascript概要

## 【各言語の役割】

HTML	構造・文章・意味
CSS	デザイン（装飾） 、レイアウト
Javascript	ユーザー操作・イベント発生からの インタラクティブな動き

注意) JavaScript と Java は似ている部分もありますが、基本的には異なっています

見るだけでOK

# Javascript基礎

# HTMLのscriptを記載する場所②

※サンプル：sample01.html を作成

```
<!doctype html>
<html lang="ja">
<head>...</head>
<body>
  <h1>sample01:コメント確認</h1>
  ...
  ...
  ...
  ...

<script>
</script>
</body>
</html>
```

＜重要＞  
ブラウザは上から下へ順番に処理をします

「body閉じタグ」の直前に書く

# javascriptの基礎

---

## 【サンプル動作確認】

### ◇alertを使用

```
<script>
  alert("Hello, world"); //ダブルクオートで文字列を括っている
</script>
```

### ◇console.logを使用

[表示方法：ブラウザ → 画面右クリック → “検証”を選択 ]

```
<script>
  console.log("Hello, world"); //ダブルクオートで文字列を括っている
</script>
```

# Javascriptの基礎

---

## 【 javascriptの書き方 】

◇Javascriptコメント記述箇所

```
<head>
<script>
    // scripタグ内にJavaScriptを記述！
</script>
</head>
```

## 【 コメント 】

◇一行コメント

```
//一行コメント
```

◇複数行コメント

```
/*
JavaScriptコード
複数行コメント
*/
```

# 变数

## 【 変数 (*let* [ちょっと前まで *var* が主流] ) 】

数値や文字を入れる箱のことです。

javascriptは他言語と違い、変数の型を使用しません。

( 自動で認識し、型変換がおこなわれます )

```
<script>
```

```
let name1 = 'hello'; //文字列(string) ※文字列は「'」「"」どちらでもOK  
let name2 = "world"; //文字列(string) ※文字列は「'」「"」どちらでもOK  
let age1 = 100; //数値(number) //数値は「'」「"」を使用しなくてもOK  
let age2 = -10; //数値(number) //数値は「'」「"」を使用しなくてもOK  
let num = null; //何もない
```

```
</script>
```

### 入力エラー

- ・ 初めに「数値」が書かれている場合
- ・ 「-」などが書かれている場合

### エスケープ処理

エスケープ文字は「\」（バックスラッシュ）  
特殊文字の改行やタブは「\n」「\t」  
※ 「 option + \ 」でバックスラッシュ

# Javascript

---

## 【定数】

変数は上書き可能ですが「`const`」を使えば値を上書き不可にすることが出来ます。

```
const teisu = 'constで宣言すると上書き不可です'; //上書き不可能  
const num = 100; //上書き不可能
```

## 【スコープ】

let	変数	ブロックスコープ
const	定数	ブロックスコープ

# javascriptの基礎

## 【変数 (*const*)】

数値や文字を入れることで  
javascriptは自動で認識し、  
(自動で認識し、全

これはまずい！！！

```
<script>  
  
const name1 = 'hello'; //文字列(string) ※文字列は「'」「"」どちらでもOK  
  
const 1age = 100; //数値(number) //数値は「'」「"」を使用しなくてもOK  
  
</script>
```

3つ、どこかおかしいよ！！！

# 演算子と計算

# javascriptの基礎

## 【演算子 (operator) 】

「+」や「-」などの計算を行うためのものです

+	足し算、または文字の結合
-	引き算
*	かけ算
/	割り算
%	割った余り

```
<script>
```

```
//計算の場合
```

```
let sum1 = 1 + 9;  
let sum2 = 1 - 5;  
let sum3 = 2 * 4;  
let sum4 = 10 / 2;  
let sum5 = 10 % 3;  
</script>
```

# javascriptの基礎

## 【演算子 (operator) 】

演算子で計算

```
<script>  
  let n = 1;  
    n = n + 9;          //n+9をnに代入  
  alert(n);                //n=10
```

```
let n = 1;  
  n += 9;            //n+9をnに代入  
  alert(n);              //n=10
```

// 「 \*= 」, 「 /= 」も同様に記述できます。

```
</script>
```

# javascriptの基礎

## 【演算子 (operator) 】

演算子は文字と変数の結合にも使います。

<script>

```
let s = "ABC";           //文字と文字（変数）のケース1
s = s + "DEF";      // "ABC"+ "DEF"
console.log(s);         // "ABCDEF"
```

---

```
let s = "ABC";           //文字と文字（変数）のケース2
s += "DEF";          // "ABC"+ "DEF"
console.log(s);         // "ABCDEF"
```

<重要> 文字("ABC")と数値(10)で + 演算子を使った場合 → "ABC10"となる  
</script>

## 【変数として使用できない文字】

他の構文や将来的に使われる言語を変数として使用することができない

### 予約語とキーワード

“MDN javascript 予約語”で検索！！

# 条件分歧

# javascriptの基礎

## 【 IF分岐 処理 】

if文を使用することで、条件によって違う処理をおこなうことができる

```
if(条件式){ //条件に合えばtrueでここ内の…}内を処理  
    条件式の結果がtrueの場合実行されるスクリプト  
}
```

```
if(条件式){ //条件に合えばtrueでここ内の…}内を処理  
    条件式の結果がtrueの場合実行されるスクリプト  
}else{  
    条件式の結果がfalseの場合実行されるスクリプト  
}
```

```
if(条件式1){          //条件に合えばtrueでここ内の…}内を処理  
    条件式の結果がtrueの場合実行されるスクリプト  
}else if(条件式2){ //条件に合えばtrueでここ内の…}内を処理  
    条件式の結果がtrueの場合実行されるスクリプト  
}
```

# javascriptの基礎

//複数条件の分岐処理も可能！

**if(条件式1){**

    条件式の結果がtrueの場合実行されるスクリプト

**}else if(条件式2){**

    条件式の結果がtrueの場合実行されるスクリプト

**}else if(条件式3){**

    条件式の結果がtrueの場合実行されるスクリプト

**}**

//複数条件の分岐処理と「条件以外」の指定も可能！

**if(条件式1){**

    条件式の結果がtrueの場合実行されるスクリプト

**}else if(条件式2){**

    条件式の結果がtrueの場合実行されるスクリプト

**}else if(条件式3){**

    条件式の結果がtrueの場合実行されるスクリプト

**}else{**

    条件式の結果がfalseの場合実行されるスクリプト

**}**

# javascriptの基礎

## 【 IF分岐 : 比較演算子】 if文の条件式には「比較演算子」を使用します

```
const value = 10;  
if(value == 10){ //Q.条件は合います?  
    条件式の結果がtrueの場合実行されるスクリプト  
}  
  
if(value > 0) { //Q.条件は合います?  
    条件式の結果がtrueの場合実行されるスクリプト  
}else{  
    条件式の結果がfalseの場合実行されるスクリプト  
}
```

## 【 比較演算子 (relation) 】 2つの値を比べる際に使用します。

==	等しければ true
!=	等しくなければ true
>	左側の方が大きければ true
<	左側の方が小さければ true
>=	左側の方が大きい、または同じなら true
<=	左側の方が小さい、または同じなら true

# javascriptの基礎

## 【 IF分岐 処理：問題 】

```
const a = 1;  
const b = "山";
```

```
if( 条件式 ){ //aが1以上、代入されている場合、true処理を実行  
    alert("1以上だよ");  
}
```

```
if( 条件式 ){ //bに"山"が代入されていない場合、true処理を実行  
    alert("山じゃないよ！");  
}else{  
    alert("山だよ");  
}
```

# javascriptの基礎

## 【論理演算子（Logical operator）】

2つの真偽値をさらに比較する際に使用します

&&	AND処理：両方とも同じなら true
	OR処理：どちらかが trueなら true
!	NOT演算：trueならfalseへ falseならtrueへ

```
const value = 10;  
if(value >= 5 && value <= 15){ //valueは5～15の数値であればtrue
```

条件式の結果がtrueの場合実行されるスクリプト

```
}
```

```
if(value == 5 || value == 10 ) { //valueは5か10の数値であればtrue
```

条件式の結果がtrueの場合実行されるスクリプト

```
}else{
```

条件式の結果がfalseの場合実行されるスクリプト

```
}
```

## 関数 (function)

### 【変数のスコープ】

変数スコープ範囲にも2つの種類があります

- ・グローバル変数：全ての領域から参照が可能
- ・ブロック変数：関数内だけや特定の場所からしか参照ができない

```
<script>
const a = 1;          // グローバル変数
if(a==0){
    const b = 1;    // ifブロック内の変数
    alert(a);
}else{
    const b = 2;    // ifブロック内の変数
    alert(a);
}
alert( b );
</script>
```

※ {...} ブロック内で let or const を使って変数宣言しないと外部の変数を参照します。



見るだけでOK

# Mathオブジェクト

# Math(乱数)

## 【Mathオブジェクトとは?】

Mathオブジェクトの中には沢山の関数が入っていますが、よく使われるのは、乱数生成する random関数です。以下のサンプルを打ってみましょう。

## 【Math.random】

Mathオブジェクトの中のよく使われる関数として、乱数生成する random関数があります。

```
const num = Math.random();
alert( num );
```

結果例： 0.7101355947191998 (小数点が表示される)

## 【Math.floor】

Mathオブジェクトの中には、端数切り下げる「floor」や、四捨五入する「round」のような関数があります。今回は端数繰り上げ「ceil」を使ってみましょう。

```
const num = Math.ceil( Math.random() * 5 );
alert( num );
```

結果例： 1, 2, 3, 4, 5 のどれかが表示される

## ランダム数値を作成しよう

---

### 【 ランダム数値を作成しよう】

random関数はJavaScriptにあらかじめ用意されてるMathオブジェクト内の1つの関数です。MathオブジェクトのMath.ceil（切り上げ）関数と組み合わせて使います。結果として●●の箇所に数字を入れると「1～●●」の間の数が取得できます

```
const num = Math.ceil( Math.random() * ●● );
```

### 【 例 】

```
const num = Math.ceil( Math.random() * 5 );
```

```
console.log( num );
```

ポイントは、  
サンプルの使い方をそのまま使っていくことです。  
「ランダムな数字を作りたいなー」と思ったら、  
上記のサンプルをそのまま使いましょう！！

そして、調べる時間を短縮し、作成することに時間をかけましょう！

# 演習

# 【演習】 おみくじアプリ作成

## ◆仕様：

- ・ ページ読み込み時に「大吉・中吉・小吉・吉・凶」の5種類の文字をランダムでalert関数で表示する
- ・ ヒント：  
「大吉・中吉・小吉・吉・凶」のalert関数は IF文の中に記述します。

# DOM

見るだけでOK

# HTMLファイル

=

# document

見るだけでOK

ウィキペディア  
フリー百科事典メインページ  
コミュニティ・ポータル  
最近の出来事

ページ ノート

閲覧 編集

# Document Object Model

Document Object Model (DOM) は W3C から勧告されて

書や XML から構築される Level XML を読み取る仕組み

Document をツリー構造に構築し、  
プログラム側から簡単にアクセスで  
きる仕組み  
(ブラウザが勝手におこなう)

```
<!DOCTYPE html>
<html lang="ja">...</html>
▶ <head>...
▼ <body class="mediawiki ltr sitedir-ltr ns-0 ns-subject page-Document_Object_Model skin-vector action-view">
  <div id="mw-page-base" class="noprint"></div>
  <div id="mw-head-base" class="noprint"></div>
  ▼ <div id="content" class="mw-body" role="main">
    <a id="top"></a>
    ▶ <div id="siteNotice">...</div>
    <div class="mw-indicators">
    </div>
    <h1 id="firstHeading" class="firstHeading" lang="ja">Document Object Model</h1>
    ▼ <div id="bodyContent" class="mw-body-content">
      <div id="siteSub">出典: フリー百科事典『ウィキペディア (Wikipedia)』</div>
      <div id="contentSub"></div>
      ▶ <div id="jump-to-nav" class="mw-jump">...</div>
      ▼ <div id="mw-content-text" lang="ja" dir="ltr" class="mw-content-ltr">
        ▶ <table class="vertical-navbox nowraplinks" style="float:right;clear:right;">
```

見るだけでOK

# Selectors

## DOM (要素にアクセスして属性を取得する)

id属性を指定して属性の値を取得するとき。

- ① `const mybtn = document.getElementById('mybtn');` //昔からの記述方法
- ② `const mybtn = document.querySelector('#mybtn');` //新しい記述方法

mybtn変数 の中は

```

```

指定方法

`mybtn.id`

`mybtn.src`

`mybtn.className`

コンソールで取得できるか確認

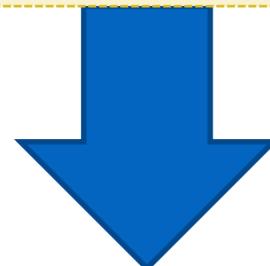
```
console.log( mybtn.id );           //id名を取得  
console.log( mybtn.src );         //src属性値を取得  
console.log( mybtn.className );    //class属性値を取得※classのみ特殊
```

HTML)

```
<div id="a1">id属性を持つdiv要素</div>
```

idを指定することで「参照や変更」が可能

```
const a1 = document.querySelector("#a1");
```



変数からスタイルシートを指定することも可能

```
a1.style.color      ="#f00";  
a1.style.fontSize  ="40px";  
a1.innerHTML       ="やっぱり勉強会はいい！";
```

【要素を指定する方法：CSSのセレクタと同様でOK】

id が main\_content の要素を取得

**document.querySelector("#main\_content");**

div 直下の p を取得

**document.querySelector("div>p");**

input 要素 type 属性 button の要素を取得

**document.querySelector("input[type='button']);**

class が left-column の最初の要素を取得

**document.querySelector(".left-column");**

# EVENT

見るだけでOK

## 【イベント】

イベントは、ページの読み込み、マウスボタンをクリック、マウスオーバーなどの「何かしらの動作（イベント）が起こった時」に発生します。

入力...

ボタン

## 【イベントハンドラ】

**onload**

ページや画像の読み込みが完了した時に発生

**onclick**

要素やリンクをクリックした時に発生

**onchange**

フォーム要素の選択、入力内容が変更された時に発生

### ◆クリックイベントの記述例

```
document.querySelector("#mybtn").onclick = function( e ){
    alert('Hello world');
};
```

※上記の「... function( e ){...}」のeは省略可能。eにはclickした場所のx,y座標などが格納されている。

見るだけでOK

## 【 イベントハンドラ : 一覧 】

<b>onblur</b>	ページやフォーム要素からフォーカスが外れた時に発生
<b>onfocus</b>	ページやフォーム要素にフォーカスが当たった時に発生
<b>onchange</b>	フォーム要素の選択、入力内容が変更された時に発生
<b>onselect</b>	テキストが選択された時に発生
<b>onsubmit</b>	フォームを送信しようとした時に発生
<b>onreset</b>	フォームがリセットされた時に発生
<b>onabort</b>	画像の読み込みを中断した時に発生
<b>onerror</b>	画像の読み込み中にエラーが発生した時に発生
<b>onload</b>	ページや画像の読み込みが完了した時に発生
<b>onunload</b>	ウィンドウを閉じた時、他のページに切り替えた時に発生
<b>onclick</b>	要素やリンクをクリックした時に発生
<b>ondblClick</b>	要素をダブルクリックした時に発生
<b>onkeyup</b>	押していたキーをあげた時に発生
<b>onkeydown</b>	キーを押した時に発生
<b>onkeypress</b>	キーを押してる時に発生
<b>onmouseout</b>	マウスが離れたした時に発生
<b>onmouseover</b>	マウス乗った時に発生
<b>onmouseup</b>	クリックしたマウスを上げた時に発生
<b>onmousedown</b>	マウスでクリックした時に発生
<b>onmousemove</b>	マウスを動かしている時に発生



見るだけでOK

# jQueryライブラリ

見るだけでOK

# jQueryとは

- CSSセレクタと同じ記述で対象個所を指定できる
- Vanilla.JS(素のJSのこと)よりコード記述量を抑えられる  
※短く記述できる → 初心者の的、打ち間違いが防げる！
- 拡張ライブラリが豊富 (カルーセル...以下リンクを参考)

「jquery」はJSを短縮して書けるライブラリです。  
Webサイトのちょっとした機能向上に良く使われます！

※React,Vue,angularとは分野が違うことを知っておきましょう。

参考に <https://furien.jp/columns/140/>

見るだけでOK

# jQueryとJavaScriptの記述例

セレクタ

→

メソッド

何に対して

→

関数を実行する

```
$("#email").html("HTMLや文字列");
```

```
document.querySelector("#email").innerHTML="文字列";
```

同じ意味だけどjQueryは超短い！

見るだけでOK

# jQuery準備

# 「jquery CDN」で検索

A screenshot of a Google search results page. The search query 'jquery CDN' is entered in the search bar. Below the search bar are navigation links for 'すべて' (All), 'ショッピング' (Shopping), '書籍' (Books), '動画' (Videos), '画像' (Images), 'もっと見る' (More), '設定' (Settings), and 'ツール' (Tools). A message indicates approximately 17,400,000 results found in 0.35 seconds. The first result is highlighted with a red box and labeled 'jQuery CDN'. The snippet for this result includes the URL 'code.jquery.com' and a brief description: 'jQuery CDN - Latest Stable Versions. Powered by. jQuery Core. Showing the latest stable release in each major branch. See all versions of jQuery Core.' Below the snippet are links to 'jQuery Core', 'jQuery UI', 'jQuery Mobile', and 'jQuery Color'. A sidebar titled '他の人はこちらも検索' (Other people also searched) lists related queries: 'jquery cdn google', 'jquery cdn cloudflare', 'jquery migrate cdn', 'jquery slim 違い', 'jquery cdn 403', and 'code jquery com cookie'. The entire screenshot is framed by a light gray border.



code.jquery.com

Plugins Contribute E

jQuery write less, do more.

Your d  
deve

jQuery Core jQuery UI jQuery Mobile jQuery Color QUnit PEP

## jQuery CDN - Latest Stable Versions

Powered by  StackPath

**jQuery Core**

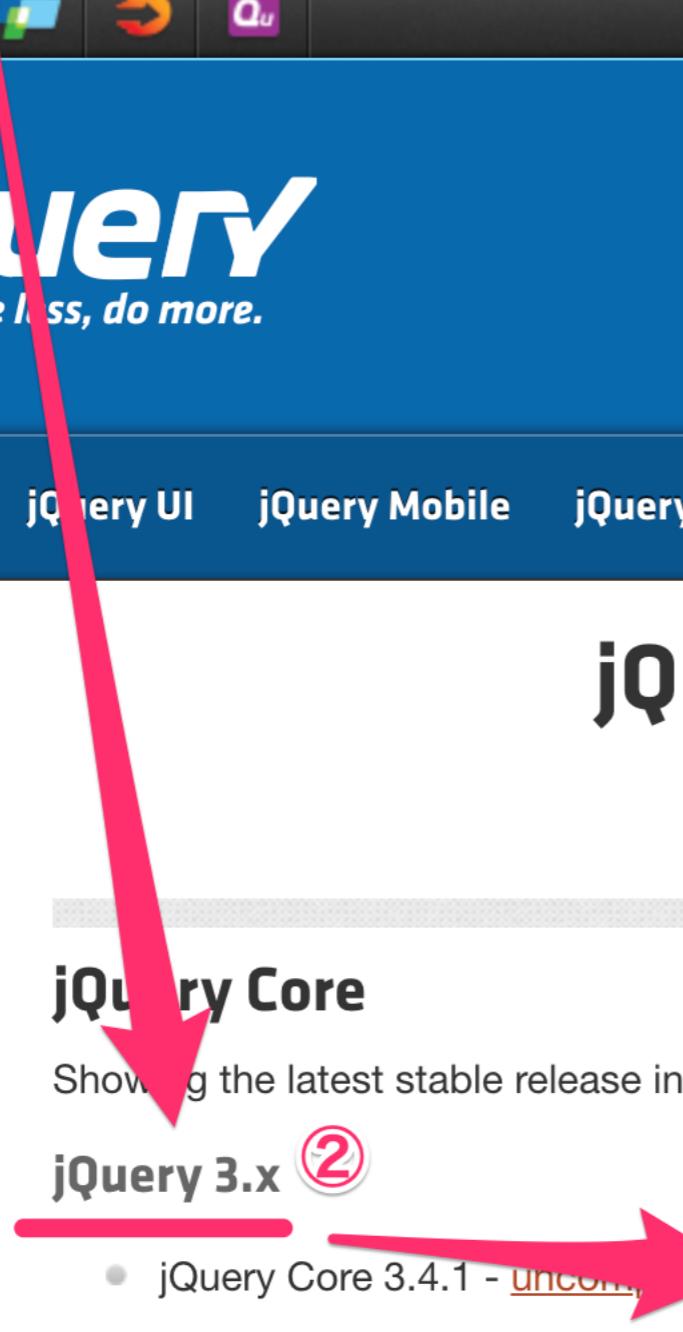
Showing the latest stable release in each major branch. [See all versions of jQuery Core.](#)

**jQuery 3.x** 

- jQuery Core 3.4.1 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)

**jQuery 2.x**

- jQuery Core 2.2.4 - [uncompressed](#), [minified](#)



C ⌂ 🔒 code.jquery.com

Plugins Contribute Events Support

Your donations help fund development and growth.

SUPPORT THE PROJECT

jQuery write less, do more.

jQuery Core jQuery UI jQuery Mobile jQuery Color QUnit PEP

## jQuery CDN – Latest Stable Versions

**COPY**

**Code Integration**

```
<script
  src="https://code.jquery.com/jquery-3.4.1.min.js"
  integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
  crossorigin="anonymous"></script>
```

The `integrity` and `crossorigin` attributes are used for [Subresource integrity \(SRI\) checking](#). This allows browsers to ensure that resources hosted on third-party servers have not been tampered with. Use of SRI is recommended as a best-practice, whenever libraries are loaded from a third-party source. Read more at [srihash.org](#)

**jQuery**

- Showing 1–10 of 10 results

**jQuery 3.x**

- jQuery 3.4.1 - [uncompressed, minified](#)

**jQuery 2.x**

- jQuery Core 2.2.4 - [uncompressed, minified](#)

**jQuery 1.x**

- jQuery 1.12.4 - [uncompressed, minified](#)

以下のようにHTML内に「ペースト」して使用します。

```
98 ... </nav>
99 ...
100 </footer>
101 <p id="scroll-top"><a href="#">^</a></p>
102 <script
103   src="https://code.jquery.com/jquery-3.4.1.min.js"
104   integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
105   crossorigin="anonymous"></script>
106 <script src="assets/sticky/sticky.min.js"></script>
107 <script src="assets/js/common.js"></script>
108 </body>
109 </html>
```

# jQuery 基本

# jQuery文法

---

セレクタ → メソッド → イベント

何に対して → どのような事を → どのタイミングで

例)

```
$("#email").on("click",function(){
  $("#email").html("HTMLや文字列");
});
```

# セレクタは覚える！

---

**セレクタ → メソッド → イベント**

何に対して → どのような事を → どのタイミングで

# jQuery [セレクタ]

- CSSセレクタ互換の記述が可能

#id → \$("#id名")

.class → \$(".class名")

Element → \$("element名")

[name=a] → \$("[name=a]")

サンプルファイルで練習：\$(セレクタ).html("練習");

# jQuery [セレクタ] (参考)

---

セレクタ (CSSと同様)

複数セレクタを指定    `$("div, p, a")`

要素の親子関係                         `$("div > p")`

先祖子孫                                 `$("div p")`

最初の要素                                 `$("li:first")`

奇数の要素                                 `$("td:even")`

偶数の要素                                 `$("td:odd")`

などなど…



見るだけでOK

# メソッドは覚える！

---

セレクタ → メソッド → イベント

何に対して → どのような事を → どのタイミングで

# jQuery[メソッド=命令]

---

例)

```
$("セレクタ").メソッド();
```

```
var elem = '<a href="#">次ページ</a>';  
$("#id").html(elem);           //html処理されて表示  
$("#id").text(elem);          //htmlを文字として表示  
$("#id").css("color","#ff0");  
$("#id").show(4000);           //hide  
$("#id").prepend(elem);        //要素の先頭にHTML要素を追加  
$("#id").append(elem);         //要素の最後にHTML要素を追加  
$("#id").empty();              //子要素を全て削除  
$("#id").remove();             //要素を全て削除  
$("#id").trigger("click");     //対象をクリックする
```

# イベントは覚える！

---

セレクタ → メソッド → イベント

何に対して → どのような事を → どのタイミングで

# jQuery[イベント]

例)

```
$("#id").on("click", function(){
    $("セレクタ").css("color","#ff0");
    $("セレクタ ").append("<p>123</p>");
});
```

"click"," dblclick","mouseout" , "mousedown" , "mouseup", "mousemove"  
"mouseenter", "mouseleave" , "change", "select", "focus", "submit", "resize",  
"scroll", "ready", "keydown", "keypress", "keyup"…

onメソッドはページをロードした後に追加された要素に対してもイベント取得が可能！

# jQuery[イベント]

---

◆クリックイベントを覚える手順

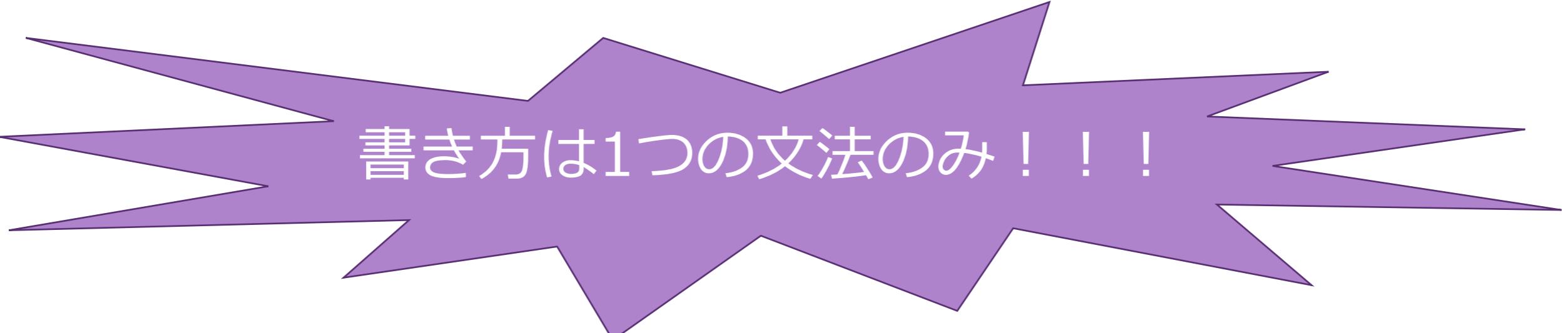
1. **`$(()).on();`**
2. **`$("#セレクタ").on();`**
3. **`$("#セレクタ").on("click");`**
4. **`$("#セレクタ").on("click", );`**
5. **`$("#セレクタ").on("click", function());`**
6. **`$("#セレクタ").on("click", function(){});`**
7. **`$("#セレクタ").on("click", function(){ //改行  
});`**

# 最初はこれ覚えておけば！

---

**セレクタ → メソッド → イベント**

何に対して → どのような事を → どのタイミングで



書き方は1つの文法のみ！！！

# 演習

# 【演習】 おみくじアプリ作成

◇仕様：

- ・ サンプル" omikuji\_tpl.html "を使用して作成
- ・ “おみくじボタン”クリック後に「大吉・中吉・小吉・吉・凶」の5種類をランダムでHTMLに表示する  
**表示記述例)**  
**\$("#id名").html("大吉");**
- ・ 「大吉・中吉・小吉・吉・凶」はIF文の中に記述します

# 課題

# 【課題】じゃんけんアプリ作成

kadai/janken\_tpl.html

## ◇ジャンケンアプリの処理概要

- ①. 「グー、チョキ、パー」のどれかをクリック
- ②. "コンピュータの出した手は?"の文字列が置き換わります。  
「コンピュータ: グー」「コンピュータ: パー」...
- ③. 「勝ち」の箇所に

「あなたの負け」「あなたの勝ち」を表示するようにします。

これを土台に「ジャンケンアプリ」が次回への課題  
IF文・イベントを覚えるのに最適なプラクティスです。

+aでスキルアップも目指してください。

上記が最低ラインとして制作