



**G's ACADEMY**  
**TOKYO**

# JavaScript

## ~First training~



# 拡張機能

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



<https://youtu.be/hizfrmSWXFs>

# 1 回目アジェンダ

- JavaScriptでできること
- JavaScript概要/基礎
- 変数/演算子と計算
- 関数(Math)
- jQuery（表示を簡単にする：ライブラリ）
- 条件分岐/IF分岐処理
- 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！

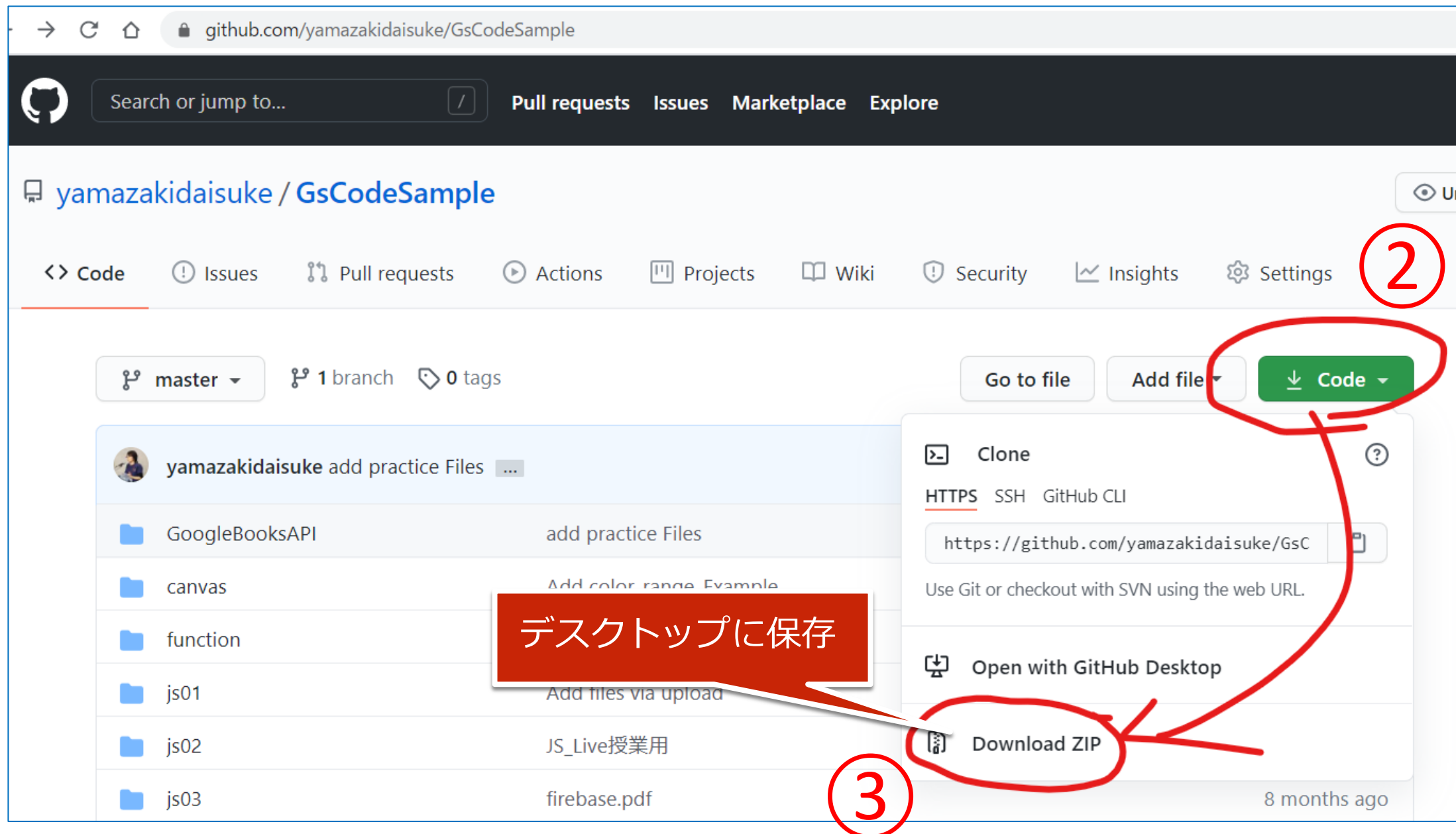
できるだけ1度授業を通してやろう！

授業は小説と同じ、最後で全てが繋がることが多々ある。

# トレーニング準備

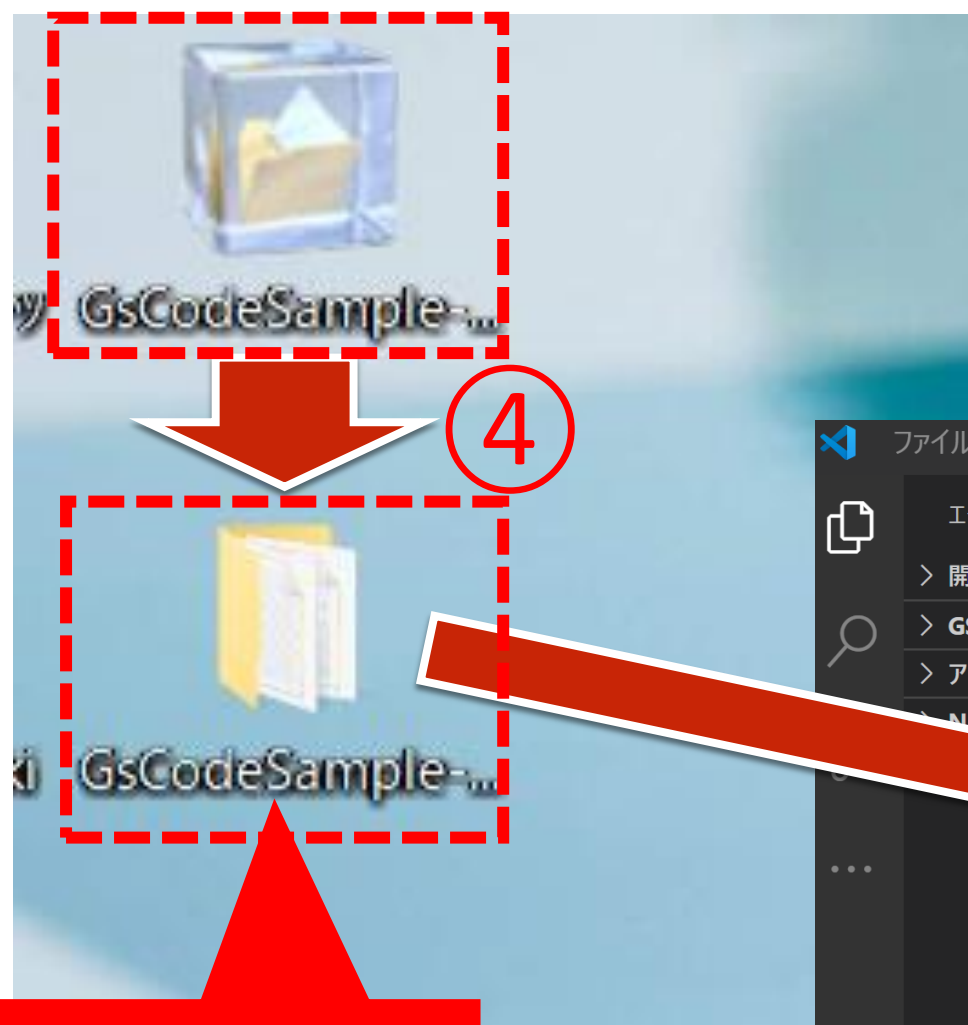
# トレーニング Sample (資料別)

① <https://github.com/yamazakidaisuke/GsCodeSample>

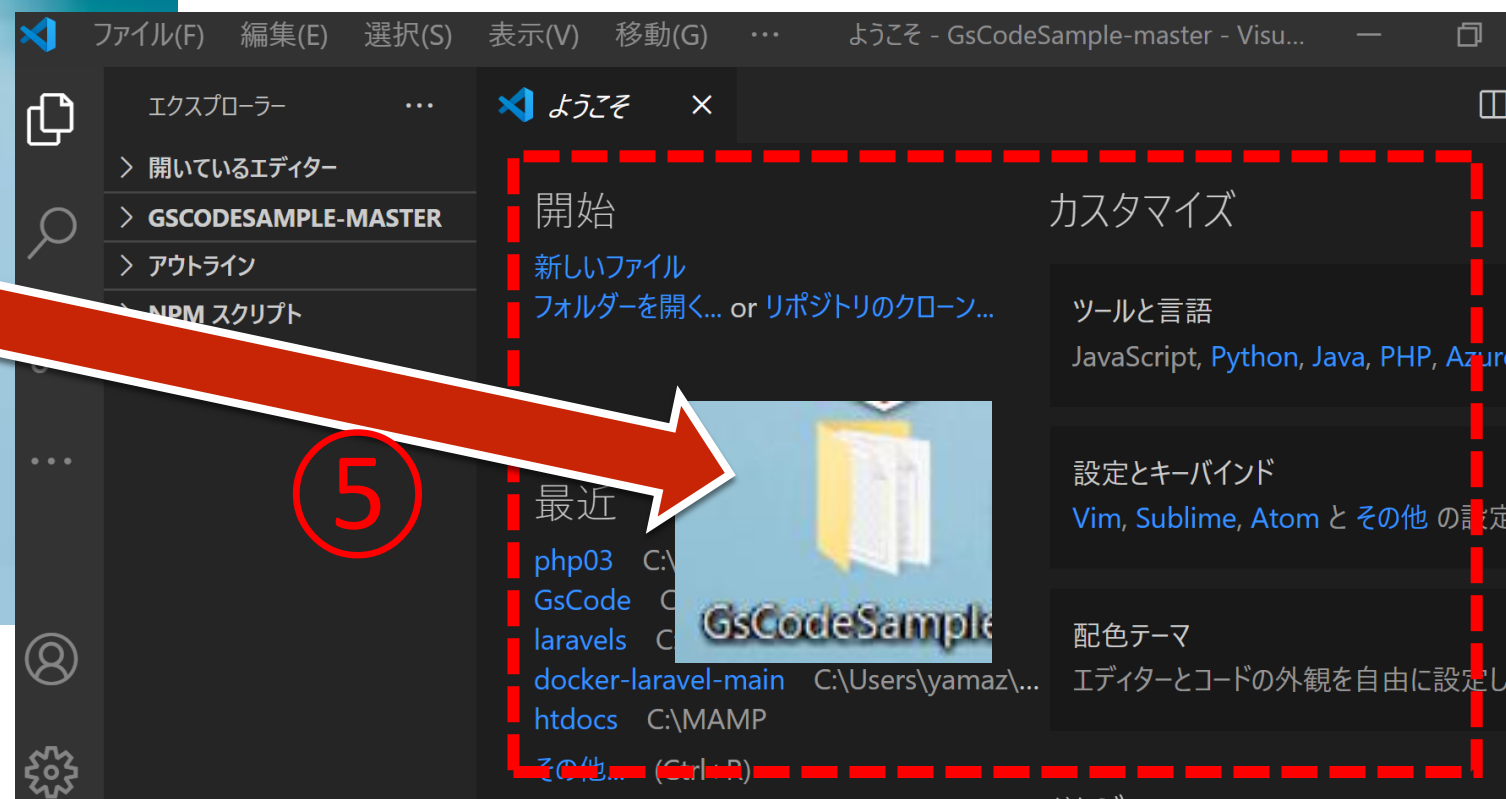




- ④ ZIPサンプルファイルを展開（解凍）
- ⑤ [GsCodeSample...]フォルダをドラッグ&ドロップ

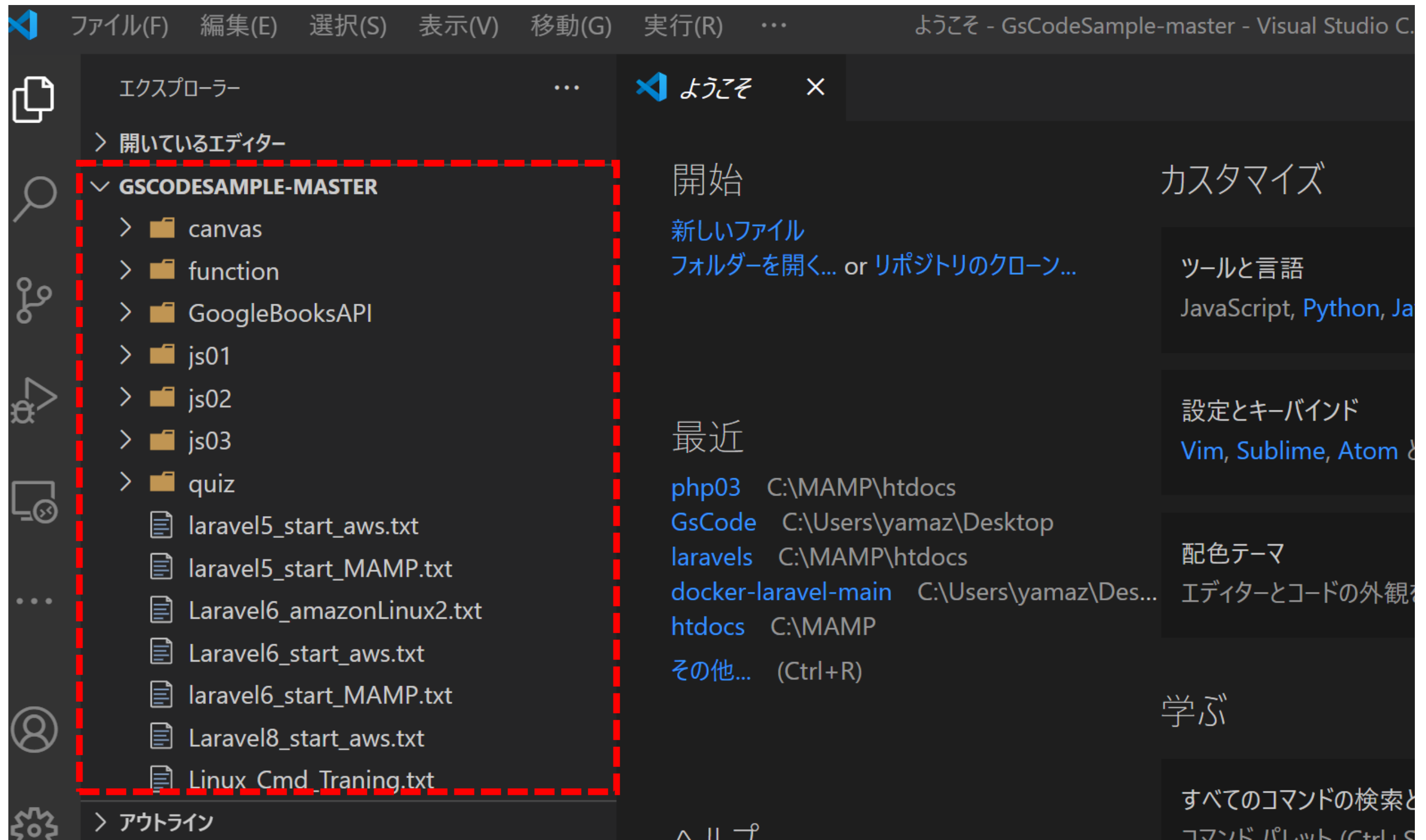


作業フォルダ



# ⑥ こうなっていればOK

## 毎回このフォルダで作業





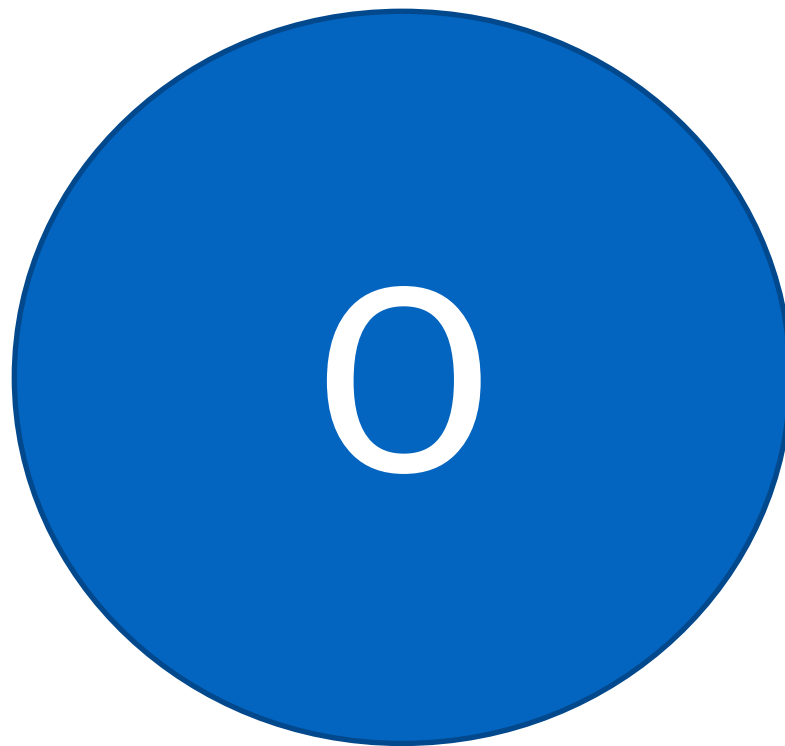
始める前に

新しい学びには必ず  
「違和感・ストレス」  
が付きものです

成長したい？  
今後の人生は選びたい？

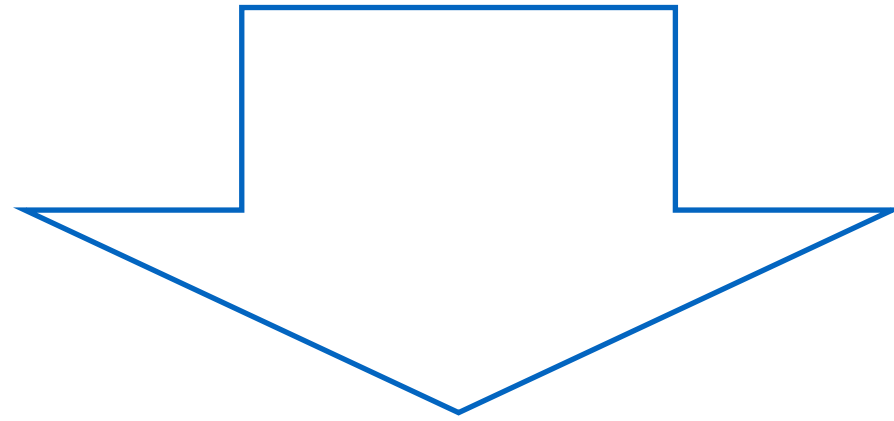
# 成長の妨げ

- 素直になれない心
- 変わることへのストレス



そこから逃げたら、成長しません

# プログラミングで 「行動力」 UP!!



ただのプログラミング研修ではない！  
今までの自分を変えることができます！

# 皆さんが学ぶのは

- プログラミング
- 行動力（自走力）
- やり切り力

新しい学びへの挑戦  
スタート！！



# 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

# 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">
<bhead>...</bhead>
<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

# プログラム基礎

# 4つの柱だけ覚えればOK

---

1. 関数

2. 変数・定数

3. 分岐処理

4. 繰り返し処理

# Javascript基礎



## 【 サンプル動作確認 】

### ◇alertを使用

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

#### <入門必須知識>

1. 文字列は `"` か `'` でくくること!!
2. 行末に `;` セミコロンを必ず記述する!!

### ◇console.logを使用

[結果の表示方法： ブラウザ → 画面右クリック → 検証 ]

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

# 変数

# Javascript

## 【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
```

```
</script>
```

## 【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
```

### 変数名の注意！！

- ・ 変数名の初めに「数値」が書かれている場合
- ・ 変数名に「-」が書かれている場合

# 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  
*/
```

## 【変数 (const)】

数値や文字を入れることができることです

javascriptは他言語と違い

(自動で認識し、変数

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 name2 = "world" //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、どこがおかしいよ！！！！

# 演算子と計算



## 【 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>
```

# 関数

# 関数とは

## 関数は2種類ある

1. 組み込み関数（ビルトイン・ネイティブ）  
プログラミング言語の仕様で、最初から標準で用意されている関数のこと
2. ユーザー定義関数  
ユーザー（プログラマ）がコード上で定義、実装した関数のこと

今日はこっちをやります！

# 関数の見分け方

変数と違い( )がある

- `alert("ピコーン");`
- `console.log("コンソールに表示");`
- `Math.random();`
- `Math.ceil();`
- `if(){...};`
- `for(){...};`

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

結果例 : 0.7101355947191998 (小数点が表示される)

## 【Math.foor】

Mathオブジェクトの中には、端数切り下げする「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 );  
console.log( 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】

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！



# jQuery基本

# jQuery文法

---

セレクトタ → 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！

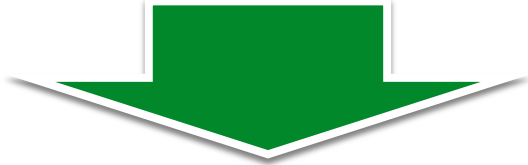
---

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]

---

- 例えばHTMLに以下 1 行がある場合  
`<div id="view"> 表示</div>`
- jQueryからHTML内 `id="view"`を指定  
`$("#view")`  
  
`$("#view").html("表示を上書き");`

# メソッドは覚える！

---

セレクト → **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)

```
$("#セレクトタ").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").css("color","ff0000");
```

```
$("#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
```

# イベントは覚える！

---

セレクト → 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(){  
    $("セレクト").css("color", "#ff0");  
    $("セレクト ").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！

---

セレクト → 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！！！！

# 休憩

# 条件分岐

## 【 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
}
```

//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

```
}
```

## 【 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

## 【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

<b>&amp;&amp;</b>	AND処理：両方とも同じなら true
<b>  </b>	OR処理：どちらかが trueなら true
<b>!</b>	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  
}
```



### 【 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。

# 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](http://kadai/janken_tpl.html)

## ◇ジャンケンアプリの処理概要（次ページにイメージ図あり）

- 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  
「負け」「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。

+α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

# できあがりは こんなイメージ



# 過去のじゃんけん

- [http://tealimpala23.sakura.ne.jp/jyanken NIWA 24/opning.html](http://tealimpala23.sakura.ne.jp/jyanken_NIWA_24/opning.html)
- <http://kou1-141.sakura.ne.jp/yugato/>
- [http://shirafu.sakura.ne.jp/janken yanagi 30/jankenman.html](http://shirafu.sakura.ne.jp/janken_yanagi_30/jankenman.html)
- [http://karate-artist.sakura.ne.jp/html izumi 03/janken/index.html](http://karate-artist.sakura.ne.jp/html_izumi_03/janken/index.html)
- [http://oga-chan.jp/janken ogawa 08/player-select/](http://oga-chan.jp/janken_ogawa_08/player-select/)
- <https://cheeseac-lab1020.sakura.ne.jp/Janken/>
- [http://tomoya07oga.sakura.ne.jp/js01 haifu/kadai/janken\\_tpl.html](http://tomoya07oga.sakura.ne.jp/js01_haifu/kadai/janken_tpl.html)
- [https://ai-doctor.sakura.ne.jp/JS Janken hariu 25/](https://ai-doctor.sakura.ne.jp/JS_Janken_hariu_25/)