

# ストップウォッチ作成体験会

presented by Lumos

# ツールのインストール

- Google Chrome  
(<https://www.google.com/intl/ja/chrome/>)
- Visual Studio Code  
(<https://code.visualstudio.com/download>)

## 作業フォルダの作成

自分の使いやすい場所に、作業用のフォルダを作ってください。  
名前はstop-watchにしましょう。

# HTMLファイルの作成

- VScodeで新しいファイルを作成
- cmd+S/ctr+Sで、ファイル名をindex.htmlに。
- !入力後、tabキーを押してテンプレートを作成

すると...



```
1  <!--ビックリマークを入力して、タブを押すとテンプレートができる。-->
2  <!DOCTYPE html>
3  <html lang="en">
4    <head>
5      <meta charset="UTF-8" />
6      <meta http-equiv="X-UA-Compatible" content="IE=edge" />
7      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
8      <title>Document</title>
9    </head>
10   <body></body>
11 </html>
12
```

こんな感じに。

# htmlを書いてみよう

以下のように、htmlファイルはタグを使って文章を記述していきます。実際に書いてブラウザに表示されるか確認しましょう。

- `<p>こんにちは。</p>`
- `<button>押してね</button>`
- `<h1>大見出しだよ</h1>`

# ボタンとディスプレイを設置しよう

ストップウォッチに必要な要素を設置していきましょう。

- タイトル
- 秒数が映るディスプレイ
- スタート/ストップ/リセットボタン

## タグにidをつけよう

- pタグにはid='display'
- スタートボタンにはid='start'
- ストップボタンにはid='stop'
- リセットボタンにはid='reset'



# 完成形



```
1  <body>
2      <h1>ストップウォッチ</h1>
3      <p id="display">0.00</p>
4      <div class="button">
5          <button id="start">スタート</button>
6          <button id="stop">ストップ</button>
7          <button id="reset">リセット</button>
8      </div>
9  </body>
```

# JavaScriptを使ってみよう

- `now = new Date();`
- `console.log([出力したいもの])`
- 関数の書き方  
    `function 関数名(){処理}`  
    例 : `function plus(a,b) {console.log(a+b);}`
- `setInterval([関数],[ミリ秒数])`

## やりたいことを考えてみよう

- スタートボタンを押したらカウントが始まる。
- ストップボタンを押したらカウントが止まる。
- リセットボタンを押したら0.00になる。

## ## 下準備をしよう

- `window.onload()` // ページが読み込まれた時に、()内の処理を実行する。

これから書くコードは全てこの()内に記述します。

- 引数、定数の定義をしよう



```
1  const startButton = document.getElementById('start');  
2  const display = document.getElementById('display');  
3  startButton.onclick = start;
```

# カウンターを作ろう

1. `let startTime = new Date();` //開始時刻を記録
2. `counter = setInterval(function(){  
 const now = new Date();  
 display.innerHTML = ((now - startTime)/1000).toFixed  
, 20);` //現在時刻と開始時刻の差をdisplayに反映

# start()関数を作ろう。

スタート関数はスタートボタンが押された時に起動し、カウントの開始とスタートボタンの無効化を担う。



```
1 function start() {  
2     startButton.onclick = null;  
3     let startTime = new Date();  
4     counter = setInterval(function(){  
5         const now = new Date();  
6         display.innerHTML = ((now - startTime)/1000).toFixed(2);  
7     })  
8 }
```

# スタートボタンを押してみよう

うまくいけばこのようになります。

## ストップウォッチ

4.70

スタート

ストップ

リセット

# ストップボタンを作ろう

- ストップボタンを取得する  
`const stopButton = document.getElementById('stop')`
- ストップボタンを押すと呼び出される関数を設定  
`stopButton.onclick = stop;`
- setInterval関数を止めるときは、clearInterval関数を使用。  

```
function stop(){  
  clearInterval(counter);  
}
```



# ストップボタンを作ろう

定義はこのような感じ。



```
1  function stop() {  
2  clearInterval(counter);  
3  }
```

# ストップボタンを作ろう

stop関数はこのような感じ



```
1  const startButton = document.getElementById('start');  
2  const stopButton = document.getElementById('stop');  
3  const display = document.getElementById('display');  
4  startButton.onclick = start;  
5  stopButton.onclick = stop;
```

# リセットボタンを作ろう

- リセットボタンを取得

```
const resetButton = document.getElementById("reset");
```

- リセットボタンを押すと呼び出される関数を設定

```
resetButton.onclick = reset;
```

- displayの表示をゼロに戻そう

```
display.innerHTML = "0.00"
```

# 仮完成！！

おめでとうございます！！

これで、ストップウォッチの最低限の機能は作れました。

バグや機能は不足がたくさんあるので、是非ご自身で勉強して完成させてみてください！！

Lumosに仮入会してくださる方は、一緒に最後まで作ってみませんか？