JavaScript 入門講座

JavaScript 第４回 / 全６回

JavaScript から HTML の書き換え（準備）

- HTML の書き換えたいタグに `id="hello"` を書き込みます。

```
<html>
<body>
    <h1 id="hello">Hello World</h1>
</body>
    <script type="text/javascript" src="sample0401.js"></script>
</html>
```

※第３回の HTML と、 `<script>` タグの位置を変えてあります。

JavaScript から HTML の書き換え

以下の JavaScript により、 "Hello World" の文字が "Good Morning" に書き換わります。

```
const element = document.getElementById('hello');
element.innerText = "Good Morning";
```

練習

1. setTimeout 関数を利用して、 5秒後に文字が書き換わるようにしてみましょう。
2. 余力があれば、 5秒ごとに "Hello World" と "Good Morning" の表示が入れ替わるようにしてみましょう。

setTimeout 関数の使い方：

ほぼ setInterval と同じです。ただし、関数は１回しか呼ばれません。

```
function display() {
    const element = document.getElementById('hello');
    element.innerText = "Good Morning";
}

setTimeout(display, 1000);
```

HTML の挿入

以下のようにすると、 HTML タグも差し込むことが出来ます。

```
const element = document.getElementById('hello');
element.innerHTML = '<FONT COLOR="#FF0000">Good Morning</FONT>';
```

※タグを文字列で書き換えるよりも効率の良い方法があります。

木構造

- 根から葉へと木が先端に行くまでの経路と似ているので、以下のようなデータ構造を「木構造」と言います。
- HTML は木構造。

```
html
├ head
└ body
    └ h1
        └ "Hello World"
```

DOM(Document Of Model)

HTML が木構造で表されているという性質を使い、 JavaScript から HTML を操作できる DOM という便利な概念があります。

- document.body で HTML の BODY タグが取れます。
- このタグに innerHTML でタグを書き込めば、何でも出来ます。

```
document.body.innerHTML = '<h1>Good Evening</h1>';
```

- 書き込みにも DOM を使うと、より便利に HTML 構造が書けます。

DOM サンプル

以下の JavaScript で、 IMG タグの src 属性のみを書き換えることが出来ます。

```
<html>
<body>
    <img id="dog" src="https://x.gd/9Uow0"></img>
</body>
<script type="text/javascript" src="sample0403.js"></script>
</html>
```

```
const element = document.getElementById('dog');
element.setAttribute('src', 'https://x.gd/LLx5A');
```

やってみよう！(DOM)

```
<html>
    <body>
        <div id="myid">Hello!</div>
    </body>
    <script type="text/javascript">
        const element = document.getElementById("myid");
        console.log(element.innerText);
        element.innerHTML = '<span style="color:#ff0000;">Good Evening!</span>';
    </script>
</html>
```

BOM(Browser Object Model)

- ブラウザの情報に関するオブジェクトモデル
- ブラウザの情報を取得、操作する枠組み

やってみよう１！(BOM)

```
<html>
    <body>
        <div>Hello!</div>
    </body>
    <script type="text/javascript">
        console.log(location.href);
    </script>
</html>
```

やってみよう２！(BOM)

```
<html>
    <body>
        <div>Hello!</div>
    </body>
    <script type="text/javascript">
        location.href = "http://www.google.co.jp/";
    </script>
</html>
```

イベント１

- あるイベントが発生したときに、 JavaScript プログラムを起動させることが出来ます。
- これをイベントの登録などと表現し、プログラムが起動することを発火などと表現します。

```
<input type="button" value="button" id="myid2" onclick="buttonClick()">

<script type="text/javascript">
    function buttonClick(){
        alert('Click');
    }
</script>
```

イベント２

HTML と JavaScript ファイルは、別ファイルに分けておきたい。

```
function buttonClick(){
    alert('Click');
}

const button = document.getElementById('myid2');
button.addEventListener('click', buttonClick);
```

ファイルへの保存

```
<a href="#" id="weather" download="weather.json">JSON ダウンロード</a>
```

```
document.getElementById('weather').addEventListener('click', (event) => {
  // 保存する文字列の Blob オブジェクトを作成
  const blob = new Blob(["<html><body><h1>hello</h1></body></html>"],
    {type: 'text/html'});
  // a 要素の href 属性に Object URL を セット
  event.currentTarget.href = window.URL.createObjectURL(blob);
});
```

ファイルの読み込み

今書き込んだ JSON フィアルを読み出してみましょう。
ファイルの読み書きには通常 File API を使用します。

```
<input id="myfile" type="file">
```

```
const f = document.getElementById('myfile');
f.addEventListener('change', function(evt) {
    const input = evt.target;
    const file = input.files[0];
    const reader = new FileReader();
    reader.onload = () => {
        // 読み出し結果の表示
        console.log(reader.result);
    };
    reader.readAsText(file); // 読み込み開始
});
```

CSV ファイル

- Excel のような表計算のような構造をもったデータ
- １行で一つのデータの塊を表し、各データは記号「`,`」で区切ります。

タイトル	著者	発行年
博士の愛した数式	小川 洋子	2003
円周率πの不思議	堀場 芳数	1989
超幾何関数入門	木村 弘信	2007

```
タイトル,著者,発行年
博士の愛した数式,小川　洋子,2003
円周率πの不思議,堀場　芳数,1989
超幾何関数入門,木村　弘信,2007
```

CSV ファイルの読み込み

```
// 配列を定義
const csvArray = [];

// 改行ごとに配列化
const lines = body.split(/\r\n|\n/);

// 1行ごとに処理
for (let i = 0; i < lines.length; ++i) {
    let cells = lines[i].split(",");
    csvArray.push(cells);
}

console.log(csvArray);
```

※画面表示などで使用する制御文字エスケープシーケンスについては、 JavaScript 本格入門 P73 を参照