リスキリング

JavaScript 第２回（全３回） / 全７回

DOM(Document Object Model)

- HTML などの文書情報を表したモデル。
- モデルで表現されている文書の情報を取得、操作する枠組み

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！(DOM)

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

やってみよう！(BOM)

```
<html>
    <body>
        <div id="myid">Hello!</div>
    </body>
    <script type="text/javascript">
        console.log(location.href);
    </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る。
- 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する。

```
<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');
}

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

ネットワークから情報を取得する１

https://open-meteo.com/ を使用して、天気情報を取得してみましょう。

まずはブラウザで以下にアクセスしてみてください。

https://api.open-meteo.com/v1/forecast?latitude=35.6785&longitude=139.6823&hourly=temperature_2m&timezone=Asia%2FTokyo

文字列がいろいろ表示されると思います。この中には、東京の一週間分の気温の情報が含まれています。

これらの文字列は JSON(JavaScript Object Notation)形式 と呼ばれ、 JavaScript ではよく使われます。

次のページで、 JavaScript でこの情報を取得してみましょう。

ネットワークから情報を取得する２

```
const url = 'https://api.open-meteo.com/v1/forecast?latitude=35.6785&longitude=139.6823&hourly=temperature_2m&timezone=Asia%2FTokyo';

fetch(url)
  .then(data => data.json())
  .then(json => console.log(json))
```

これで、 JSON 形式の情報がコンソールに表示されます。

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) => {
  // JSON を文字列に治す。
  const json = JSON.stringify({ a: 1, b: 2, c: 3 }, null, '  ');
  // 保存する文字列の Blob オブジェクトを作成
  const blob = new Blob([json], { type: 'application/json' });
  // 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', 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
let csvArray = [];

// 改行ごとに配列化
let 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);
```

演習

1. ネットワークから CSV ファイルを取得しましょう。
2. 取得した CSV ファイルの情報をもとに、 HTML を生成して、画面に表示してみましょう。
3. 生成する HTML は `<ul><li>１つ目</li><li>２つ目</li></ul>` となるようにしましょう。

http://etp.xsrv.jp/reskilling/

例：

- 博士の愛した数式 小川 洋子
- 円周率πの不思議 堀場 芳数
- 超幾何関数入門 木村 弘信

オブジェクト

オブジェクトとは、
（乱暴に言うと）「データ型」とその「計算方法」の塊です。

データを表す構造とその計算ロジックを「クラス」という塊で表します。

例：分数を表すクラス

```
class Rational {
    var bunshi;
    var bunbo;
}
```

メソッド

オブジェクトの中で定義されている関数は「メソッド」と呼びます。

```
class Rational {
    time(rhs) {
        this.bunshi *= rhs.bunshi;
        this.bunbo *= rhs.bunbo;
    }
}
```

インスタンス

オブジェクトのデータが入っている本体を「インスタンス」と呼びます。

インスタンスの作成は「コンストラクタ」という特別なメソッドで行います。

コンストラクタを用いて、インスタンスを作成するためにいは、「new 演算子」を使います。

```
class Rational {
    constructor(bunshi, bunbo) {
        this.bunshi = bunshi;
        this.bunbo = bunbo;
    }
}

var bunsu = new Rational(1, 3);
```

静的メソッド

インスタンスと関係のないクラス内のメソッドを「静的メソッド」と呼びます。

```
class Rational {
    static time(lhs, rhs) {
        let bunshi = lhs.bunshi * rhs.bunshi;
        let bonbo = lhs.bunbo * rhs.bunbo;
        let instance = new Rational(bunshi, bunbo);
        return instance;
    }
}

var bunsu1 = new Rational(1, 3).time(new Rational(3, 2));
var bunsu2 = Rational.time(new Rational(1, 3), new Rational(3, 2));
```

インスタンス同士の比較

```
class Rational {
    equals(rhs) {
        if (this.bunshi !== rhs.bunshi) {
            return false;
        }
        if (this.bunbo !== rhs.bunbo) {
            return false;
        }
        return true;
    }
}

var bunsu1 = new Rational(1, 3).time(new Rational(3, 2));
var bunsu2 = Rational.time(new Rational(1, 3), new Rational(3, 2));
console.log(bunsu1.equals(bunsu2));
```

演習

- 上の分数クラスを利用して、分数同士の割り算メソッドを作成してください。