

# JavaScript講座 第4回

2026/02/05

Tam

## 前回の復習

calendar 関数の基本となっているのはこの形です。

```
function calendar(year) {  
  let html = "<table><tr>";  
  for (let d = 1; d <= 31; d++) {  
    html += "<td>" + d + "</td>";  
  }  
  html += "</tr></table>";  
  return html;  
}
```

これで、以下の HTML が作られます。

```
<table><tr><td>1</td><td>2</td>...<td>31</td></tr></table>
```

## 前回の復習

```
function calendar(year) {  
  let html = "<table><tr>";  
  let wday = new Date(year, 0, 1).getDay(); // year年1月1日の曜日を取得  
  for (let d = 1; d <= 31; d++) {  
    if (wday === 0) { // もし日曜だったら、  
      html += "</tr><tr>"; // </tr><tr> タグで改行を入れます  
    }  
    html += "<td>" + d + "</td>";  
    wday = (wday + 1) % 7; // 曜日を進めます  
  }  
  html += "</tr>";  
  return html;  
}
```

# 改行

`</tr><tr>` タグで改行が入る仕組みは以下のような感じです。

```
<table>
  <tr>
    <td>1</td><td>2</td><td>3</td>
  </tr><tr>
    <td>4</td>...<td>10</td>
  </tr><tr>
    <td>11</td>...<td>17</td>
  </tr><tr>
    <td>18</td>...<td>24</td>
  </tr><tr>
    <td>25</td>...<td>31</td>
  </tr>
</table>
```

`<!-- ここで改行が入る -->`

`<!-- ここで改行が入る -->`

`<!-- ここで改行が入る -->`

`<!-- ここで改行が入る -->`

## 前回の復習

このままだと、以下のような感じで、最初の行が左に寄ってしまいます。

## 前回の復習

そこで、`1 2 3` の行の左側に曜日分だけ `<td></td>` で空白を入れます。

```
for (let i = 0; i < wday; i++) {  
  html += "<td></td>";  
}
```

# 前回の復習

これで完成する calendar 関数が以下ようになります。

```
function calendar(year) {
  let html = "<table><tr>";
  let wday = new Date(year, 0, 1).getDay();
  for (let i = 0; i < wday; i++) {
    html += "<td></td>";
  }
  for (let d = 1; d <= 31; d++) {
    if (wday === 0) {
      html += "</tr><tr>";
    }
    html += "<td>" + d + "</td>";
    wday = (wday + 1) % 7;
  }
  html += "</tr>";
  return html;
}
```

## 第4回／2月5日(木)

「JavaScriptとブラウザ」

- ブラウザからボタン操作でJavaScriptを実行させよう
- フロントとバックの違いを知ろう



# 祝日

祝日の情報は内閣府の

- 国民の祝日について <https://www8.cao.go.jp/chosei/shukujitsu/gaiyou.html>

というページで閲覧することが出来ます。

また、プログラムで扱いやすいように CSV 形式（後述）も同じページの以下のリンクで公開されています。

- 昭和30年（1955年）から令和8年（2026年）国民の祝日（csv形式：20KB）  
<https://www8.cao.go.jp/chosei/shukujitsu/syukujitsu.csv>

今日はこのデータを利用して、カレンダーで祝日表示をしたいと思います。

# CSV 形式

CSV(Camma Separate Variable)形式は、 記号でデータを区切って表現する形式です。

| 氏名   | 性別 | 生年月日      |
|------|----|-----------|
| 福山雅治 | 男  | 1969/2/6  |
| 藤原紀香 | 女  | 1971/6/28 |
| 北川恵子 | 女  | 1986/8/22 |

氏名, 性別, 生年月日  
福山雅治, 男, 1969/2/6  
藤原紀香, 女, 1971/6/28  
北川恵子, 女, 1986/8/22

# 祝日データ

祝日データをダウンロードして、メモ帳などで開くと以下のようにになっています。

```
国民の祝日・休日月日,国民の祝日・休日名称
1955/1/1,元日
1955/1/15,成人の日
1955/3/21,春分の日
:
2026/10/12,スポーツの日
2026/11/3,文化の日
2026/11/23,勤労感謝の日
```

カラムが2列しかありませんが、これも CSV形式となっています。

## データの配置

今回はこの祝日データを JavaScript に読み込んで使っていきましょう。

本来なら、毎回この祝日データをサーバからダウンロードするのですが、ブラウザでの JavaScript のセキュリティにより、同じサイトからのダウンロードしか許可されていないため、今回はローカルに保存します。

祝日データをダウンロードして、デスクトップの JS4 フォルダに  
ファイル名 `syukujitsu.csv` で保存してください。

また、ローカルPC で擬似的にサーバ配信機能を実現するため、  
VSCode の拡張機能 Live Server をインストールします。

# データの取得

以下のようにすると、ネットワーク経由でデータを取得することが出来ます。

`.then` の引数は関数になります。

```
const url = "http://localhost:5500/syukujitsu.csv"
fetch(url).then(function (response) {
  return response.text()
}).then(function (result){
  const holidaysStr = result;
  console.log(holidaysStr);
  const table = calendar(2026, holidaysStr);
  let element2 = document.getElementById("calendar");
  element2.innerHTML = table;
});
```

// データのダウンロードを始める  
// ダウンロードしたデータをテキストに変換  
//  
// 結果を代入  
// 表示  
// カレンダーを呼び出し

※取得したデータが文字化けしますが、今回は使わないので、無視します。

# データの解析

必要なのは祝日の年月日だけです。祝日の名前は使用しませんので、年月日だけを使えるようにしましょう。

`split` 関数が便利に使えます。

使い方：

以下のようすることで、変数 `str` 内の文字列を `,` 記号で分割し、`array` 配列に代入します。

```
const str = "apple,orange,strawberry";  
let array = str.split(',');  
console.log(array);
```

```
["apple", "orange", "strawberry"]
```

# 改行で分割

CSV形式を、まずは改行で1行ごとに分割して、配列に格納しましょう。

本当に分割できているのかを確認するために、実験で行番号を付けて出力してみます。

```
function parse(holidaysStr) {           // 関数の定義
  let lines = holidaysStr.split("\n");    // 文字列を改行で分割
  for (let i=0; i < lines.length; i++) {  //
    const line = lines[i];               // 1行読み込み
    console.log(i, line);                 // 行番号とともに1行表示
  }                                       //
  return lines;                           // データを返却
};                                         //
```

※ `\n` で改行コードになります。

## 呼び出し

calendar 関数の冒頭で先の `parse` 関数を呼び出して、正しく実行できるか試してみましょう。

```
function calendar(year, holidaysStr) { // 引数に holidaysStr を追加
    let holidays = parse(holidaysStr); // この行を追加
    : // 以降の行は書き換えない
}
```

ただしく表示されれば、これで holidays 変数に祝日情報が代入されています。



# 祝日判定

祝日かどうかを判定する関数を作成していきましょう。

```
function isHoliday(holidays, year, month, day) {  
    const dateStr = year + "/" + month + "/" + day + ","; // "2026/2/5," などと作成  
    for (let i=0; i < holidays.length; i++) { // 繰り返し  
        const holiday = holidays[i]; // 祝日情報を1つ読み込み  
        if (holiday.startsWith(dateStr)) { // 祝日情報と一致していたら  
            return true; // true を返す  
        } //  
    } //  
    return false; // 一致しなかったら false を返す  
}
```

# 祝日表示

いよいよカレンダーに祝日表示してみましょう。

```
function calendar(year) {
  let holidays = parse(holidaysStr);
  let html = "<table><tr>";
  let wday = new Date(year, 0, 1).getDay();
  for (let i = 0; i < wday; i++) {
    html += "<td></td>";
  }
  for (let d = 1; d <= 31; d++) {
    if (wday === 0) {
      html += "</tr><tr>";
    }
    if (isHoliday(holidays, year, 1, d) === true) { // ここから追加
      html += '<td style="color: red;">' + d + '</td>'; //
    } else { //
      html += "<td>" + d + "</td>"; //
    } // ここまで
    wday = (wday + 1) % 7;
  }
  html += "</tr>";
  return html;
}
```

**お疲れ様でした**