# JavaScript講座 第５回

2026/02/12

Tam

# 第5回／2月12日(木)

「インターネットを使った情報取得」

- APIを使って、インターネットからの情報を収集しよう
- 集めた情報から、Webサイトを更新してみよう

## 総おさらい

今日は、今までに学んだことを利用して、

復習しながら機能を追加していきます。

# 文字化けを直す

前回、デバッグツールのコンソール上で、祝日の名前が滅茶苦茶な記号で表示されていました。
（これを文字化けと呼びます。）

直すためには、正しい文字コードで利用する必要がありますが、今回は時間の関係で説明を省略します。

文字コードを直すために、 VSCode で UTF-8 で保存し直します。
※保存方法を画面で説明しますので、画面に注目をお願いします。

# dayHtml 関数の修正

機能追加をしやすいように dayHtml 関数を以下のように修正します。

```
function dayHtml(year, month, day, holidays) {
    const wday = new Date(year, month + 1, day).getDay(); // 曜日を取得
    let html = "";
    if (isHoliday(holidays, year, month, d) === true) {
        html = '<td style="color: red;">';
    } else if (wday === 0) {
        html = '<td style="color: red;">';                  // 日曜だけ赤色で表示
    } else {
        html = '<td>';
    }
    html += day + '</td>';
    return html;
}
```

※ `else if` は新しい書き方ですね。こういう使い方も出来るので、覚えておきましょう。

# calendar 関数の修正

```
function calendar(year, holidaysStr) {
    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>";
        }
        html += dayHtml(year, 1, d, holidays); // この行を追加
        wday = (wday + 1) % 7;
    }
    html += "</tr>";
    return html;
}
```

これで、祝日だけでなく、日曜も赤色で表示されます。

# 自分の予定の差し込み

自分の予定を書き込めるカレンダーを作っていきます。

今回は、祝日と同じ CSV 形式で、予定一覧を作成してきます。

ファイル名は `schedule.csv` としておきます。

```
2026/1/15,JS講座第１回
2026/1/22,JS講座第２回
2026/1/29,JS講座第３回
2026/2/5,JS講座第４回
2026/2/12,JS講座第５回
2026/2/19,JS講座第６回
```

※数字が一桁でも、前に 0 を入れないで作成してください。

# 自分の予定の読み込み

```
let holidaysStr = "";
let scheduleStr = "";
const url = "http://localhost:5500/syukujitsu.csv";
fetch(url).then(function (response) { // データのダウンロードを始める
    return response.text()            // ダウンロードしたデータをテキストに変換
}).then(function (result) {           //
    holidaysStr = result;             // 結果を代入
}).then(function () {                 //
    const url = "http://localhost:5500/schedule.csv";
    return fetch(url);                // データのダウンロードを始める
}).then(function (response) {         //
    return response.text()            // ダウンロードしたデータをテキストに変換
}).then(function (result) {           //
    scheduleStr = result;             // 結果を代入
}).then(function () {
    const table = calendar(2026, holidaysStr, scheduleStr); // カレンダーを呼び出し
    let element2 = document.getElementById("calendar");
    element2.innerHTML = table;
})
```

# calendar 関数の修正

calendar 関数の先頭に、以下の１行を追加します。

```
function calendar(year, holidaysStr) {
    let holidays = parse(holidaysStr);
    let schedule = parse(scheduleStr);                    // この行を追加
    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 += dayHtml(year, 1, d, holidays, schedule); // この行を修正
        wday = (wday + 1) % 7;
    }
    html += "</tr>";
    return html;
}
```

# getSchedule 関数の作成

以下の関数を作成してください。以前作った `isHoliday` 関数をコピペすると、入力が楽になります。

```
function getSchedule(schedule, year, month, day) {          //
    const dateStr = year + "/" + month + "/" + day + ","; // "2026/2/5," などと作成
    for (let i = 0; i < schedule.length; i++) {             // 繰り返し
        const s = schedule[i];                              // 情報を１つ読み込み
        if (s.startsWith(dateStr)) {                        // 一致していたら
            return s;                                       // 予定を返す
        }                                                   //
    }                                                       //
    return "";                                              // 一致しなかったら空文字
}
```

# dayHtml 関数の修正

まずは、予定がある日の背景色を変更していきましょう。

dayHtml 関数を修正していきます。

```
function dayHtml(year, month, day, holidays, schedule) {
    const wday = new Date(year, month - 1, day).getDay();
    let html = "";
    if (isHoliday(holidays, year, month, day) === true) {
        html = '<td style="color: red;">';
    } else if (wday === 0) {
        html = '<td style="color: red;">';
    } else {
        html = '<td>';
    }
    const s = getSchedule(schedule, year, month, day);       // ここから追加
    if (s !== "") {                                          //
        html += '<span style="background-color: yellow;">'; //
    } else {                                                 //
        html += '<span>';                                    //
    }                                                        // ここまで
    html += day + '</span></td>';                            // この行を修正
    return html;
}
```

## alert 関数

マウスでクリックしたら、予定を表示させるようにしましょう。

ここで `alert` 関数が便利に使えます。

```
alert("Hello");
```

とすると、ブラウザの上で "Hello" が表示されたダイアログが表示されます。

今回はこれを利用してきます。

# 予定の表示

```
function dayHtml(year, month, day, holidays, schedule) {
    const wday = new Date(year, month - 1, day).getDay();
    let html = "";
    if (isHoliday(holidays, year, month, day) === true) {
        html = '<td style="color: red;">';
    } else if (wday === 0) {
        html = '<td style="color: red;">';
    } else {
        html = '<td>';
    }
    const s = getSchedule(schedule, year, month, day);
    if (s !== "") {
        html += '<span style="background-color: yellow;"'; // この行と
        html += 'onclick="alert(\'' + s + '\')">';          // この行を修正
    } else {
        html += '<span>';
    }
    html += day + '</span></td>';
    return html;
}
```

# お疲れ様でした