

🔗

🏠

🕒

📖

📄

🔍

✎

🔊

教材

🔍 検索

所属チーム ▼ 🔔¹ 👤

本文 目次 質問一覧 3件

ホーム 教材 JavaScriptの基礎を学ぼう 非同期処理を理解しよう

20章 非同期処理を理解しよう

非同期処理とは何かを解説します。

🕒 90分 🏆 - 📖 読了

20.1 本章の目標

本章では以下を目標にして学習します。

- 非同期処理とは何か、概要をつかむこと
- 同期処理と非同期処理の違いを知ること
- setTimeout 関数を使って非同期処理を作成すること

実は、JavaScriptで実行される処理には、**同期処理**と**非同期処理**の2種類があります。

これまでに書いてきたJavaScriptの処理はすべて、上から順番に実行される同期処理でした。同期処理の場合、1つの処理が終わるまで次の処理には進めません。

同期処理

処理A

完了！

→

処理B

進めない！

→

処理C

処理Bが完了しないと
処理Cへ進めない

しかし、処理を実行している間、同時に他の処理も実行したい場面もあります。

例えば、YouTubeに動画を投稿する場면을イメージしてみてください。

+ 質問する

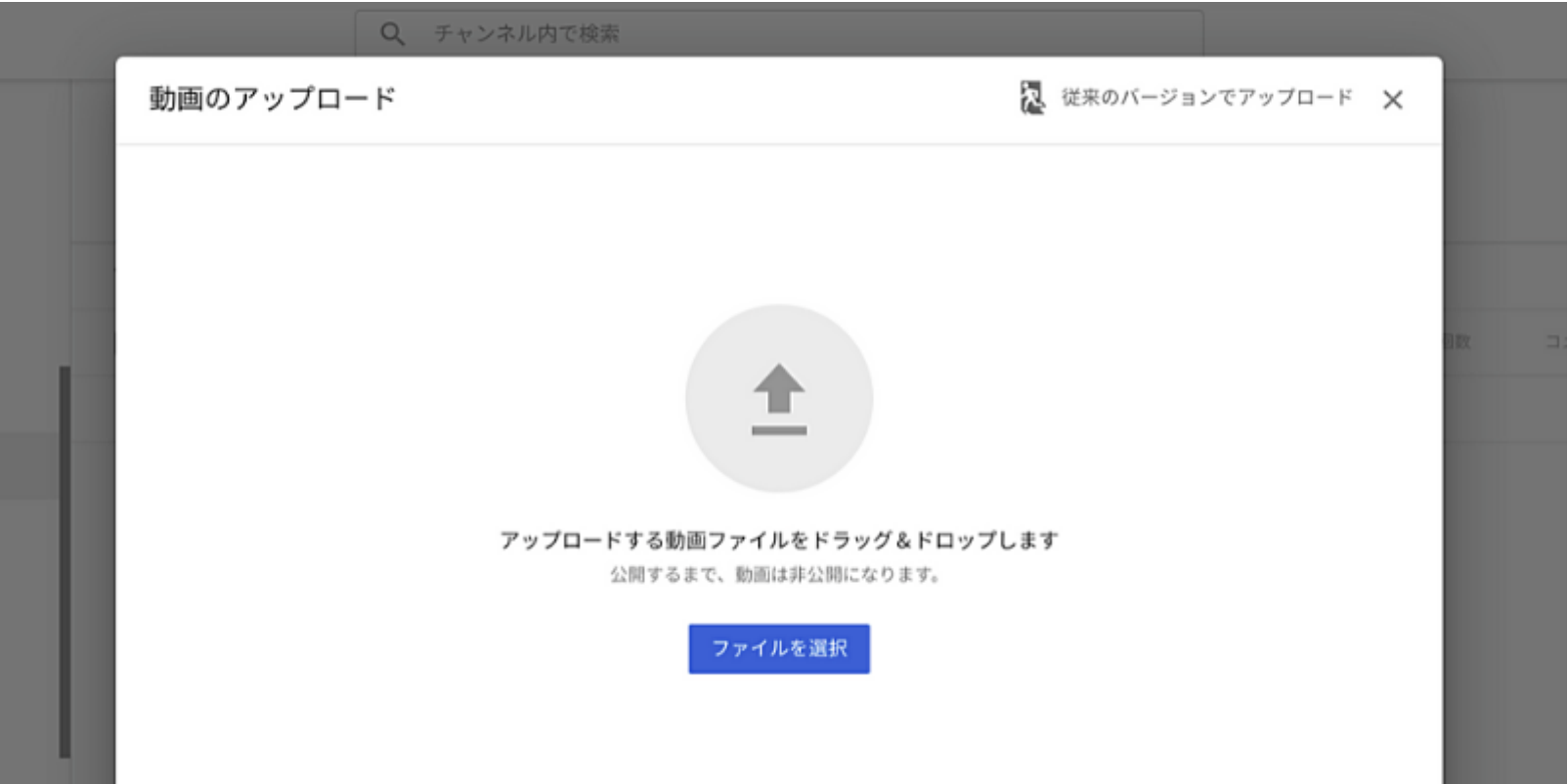
https://terakoya.sejuku.net/programs/60/chapters/686

1/6





>



動画は容量が大きいいため、アップロードには時間がかかります。そのため、同期処理ではアップロードが完了するまで何もできなくなってしまうです。

つまりアップロードのキャンセルや詳細情報の入力など一切の操作ができず、ただひたすらアップロードの完了を待つしかない状態となってしまうです。

しかし実際には以下のように、アップロードが始まるとすぐに詳細情報を入力できる状態になります。



このように、処理を実行している間、同時に他の処理も実行できる仕組みが**非同期処理**です。

非同期処理について学び、ユーザーにとってより利便性の高いWebサイトやアプリケーションを制作できるようになりましょう。

20.2 非同期処理とは

非同期処理とは、**処理を実行している間、同時に他の処理も実行できる仕組み**のことです。同期処理と非同期処理の違いは以下のとおりです。



🏠

🕒

📖

📄

🔍

✎

🔊

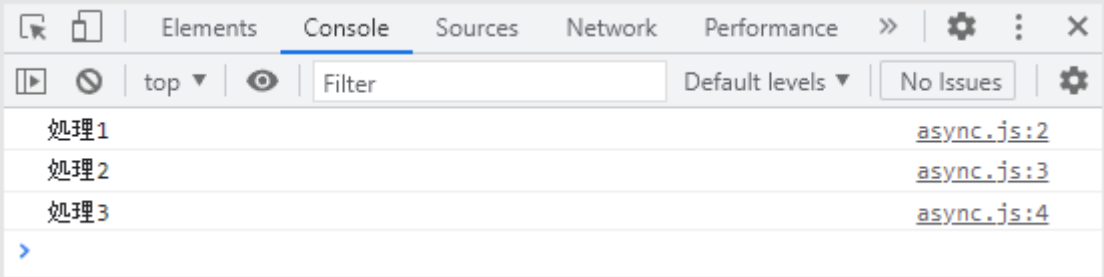
```
1 + <!DOCTYPE html>
2 + <html lang="ja">
3 +
4 + <head>
5 +   <meta charset="UTF-8">
6 +   <title>非同期処理</title>
7 + </head>
8 +
9 + <body>
10 +   <script src="js/async.js"></script>
11 + </body>
12 +
13 + </html>
14
```

続いて、js フォルダ内に新しく async.js というファイルを作成してください。 async.js を作成したら、以下のように編集しましょう。

```
async.js

1 + // 同期処理を実行する
2 + console.log('処理1');
3 + console.log('処理2');
4 + console.log('処理3');
5
```

では async.html をブラウザで開き、デベロッパーツールのコンソールを確認してみてください。以下のように、処理が上から順番に実行されていればOKです。



2. setTimeout関数で非同期処理を作成する

次は、 setTimeout 関数を使って非同期処理を作成してみましょう。

setTimeout 関数は「3秒後に画像を表示する」など、一定時間待ったあとに処理を実行する非同期処理です。

setTimeout 関数の書き方は以下のとおりです。なお、待ち時間はミリ秒（1000分の1秒）で指定します。


JSファイル（見本）


```
1 setTimeout(() => {
2   処理
3 }, 待ち時間);
4
```


とても複雑なコードに見えますが、以下のように2つの引数を渡しているだけです。「関数」の部分にアロー関数を記述して見やすいように改行すると、上記のコードになります。


https://terakoya.sejuku.net/programs/60/chapters/686


4/6























JSファイル（見本）

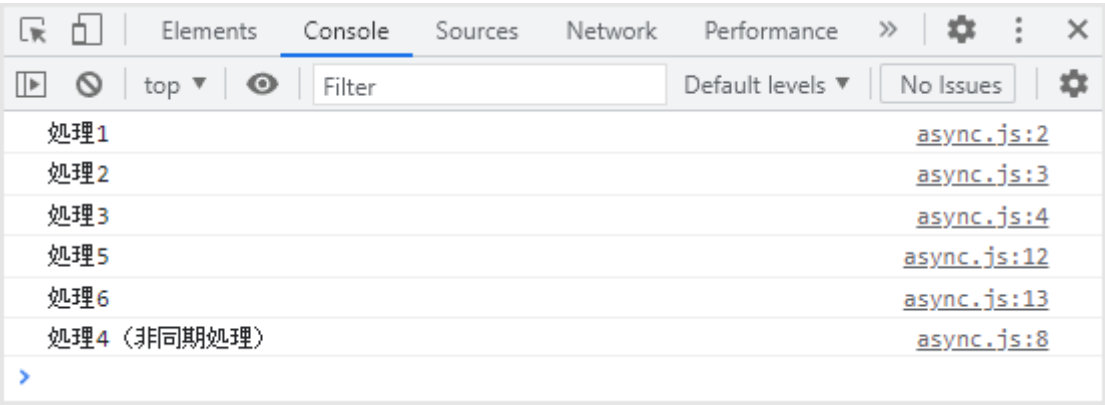
```
1  setTimeout(関数, 待ち時間);
2
```

では実際にやってみましょう。 `async.js` を以下のように編集してください。以下のコードでは `setTimeout` 関数を使い、4つ目の `console.log()` が2秒後に実行されるようにしています。

`async.js`

```
1  // 同期処理を実行する
2  console.log('処理1');
3  console.log('処理2');
4  console.log('処理3');
5
6 + // 2秒（2000ミリ秒）の待ち時間を設定し、非同期処理を実行する
7 + setTimeout(() => {
8 +   console.log('処理4（非同期処理）');
9 + }, 2000);
10 +
11 + // 同期処理を実行する
12 + console.log('処理5');
13 + console.log('処理6');
14
```

では `async.html` をブラウザで開き、デベロッパーツールのコンソールを確認してみてください。約2秒待って、以下のように表示されればOKです。



処理4だけは2秒後に実行される非同期処理なので、処理の完了を待たずに処理5、6が先に実行されているのがわかります。

まとめ

本章では以下の内容を学習しました。

- JavaScriptで実行される処理には、**同期処理**と**非同期処理**の2種類がある
- 同期処理は上から順番に実行され、1つの処理が終わるまで次の処理には進めない
- 非同期処理とは、**処理を実行している間、同時に他の処理も実行できる仕組み**のことである
- `setTimeout` 関数を使うことで、一定時間待ったあとに処理を実行する非同期処理を作成できる



```
1 // setTimeout関数の書き方
2 setTimeout(() => {
3   処理
4 }, 待ち時間);
5
```

長きにわたるJavaScript基礎編の学習も、これでついに終了です。本当にお疲れさまでした。

特にHTML/CSSから来た方にとっては初のプログラミング言語となり、内容も難しかったと思います。しかし、大きく成長できたはず
です。

本教材で身につけた基礎知識を生かせば、動きのあるワンランク上のWebサイトだけでなく、アプリケーションを制作することもでき
ます。応用編では実際にタイピングゲームを制作するので、ぜひ挑戦してみてください。

理解度を選択して次に進みましょう

ボタンを押していただくと次の章に進むことができます

～50%

50～80%

80～100%

最後に確認テストを行いましょう

下のボタンを押すとテストが始まります。

教材をみなおす

テストをはじめる

前に戻る

25 / 26 ページ

次に進む

く 一覧に戻る

改善点のご指摘、誤字脱字、その他ご要望はこちらからご連絡ください。