Create and Respond to Asynchronous Logic with the Async/Await Pattern

John Papa
DEVELOPER ADVOCATE

@john_papa www.johnpapa.net

Async/Await with TypeScript

- Overview
- How to think about async/await
- Creating async functions
- Throwing errors
- Using try/catch blocks
- Promise.all
- for await of
- Callbacks, promises and async/await

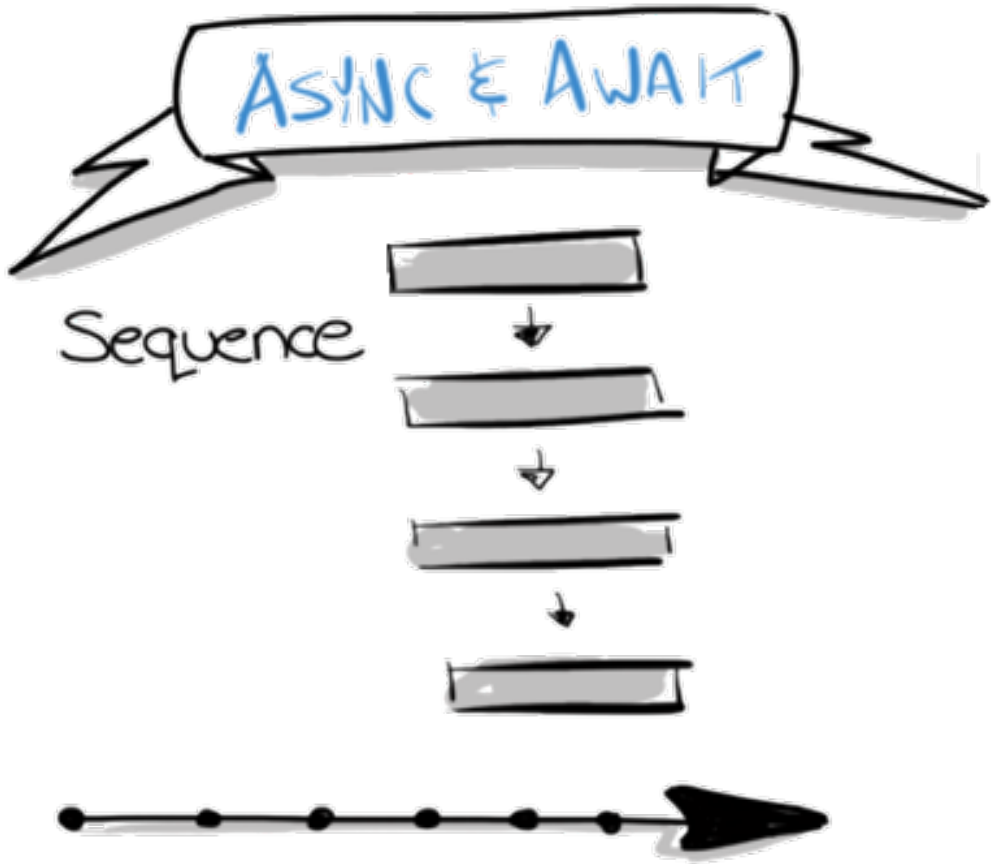
How to Think About Async/Await

Async functions make it possible to treat functions returning Promises as if they were synchronous.

Credit: caniuse.com

https://caniuse.com/#search=await

The Async / Await Pattern



Async/Await Overview

1. Makes asynchronous code behave like synchronous code
2. Can be used in place of or with promises
3. Simplifies the sequence of code that runs
4. Wide support in browsers (https://caniuse.com/#search=await)

Analyzing Async/Await

await

- Wait for the async function to fulfill
- Must be in scope of async function

```
async function example() {
  showProgressbar(true);
  let hero: Hero;
  hero = await getHero(email);
  hero.orders = await getOrders(hero);
}
```

Analyzing Async/Await

await

- Wait for the async function to fulfill
- Must be in scope of async function

sync calls

- Subsequent code “awaits” the async calls to complete

```
async function example() {
  showProgressbar(true);
  let hero: Hero;
  hero = await getHero(email);
  hero.orders = await getOrders(hero);
  showHero(hero);
  showProgressbar(false);
}
```

Analyzing Async/Await

await

- Wait for the async function to fulfill
- Must be in scope of async function

sync calls

- Subsequent code “awaits” the async calls to complete

catch

- Catch all errors from the try

```
async function example() {
  showProgressbar(true);
  let hero: Hero;
  try {
    hero = await getHero(email);
    hero.orders = await getOrders(hero);
    showHero(hero);
    showProgressbar(false);
  } catch (error) {
    showMessage(error);
  }
}
```

Analyzing Async/Await

await

- Wait for the async function to fulfill
- Must be in scope of async function

sync calls

- Subsequent code “awaits” the async calls to complete

catch

- Catch all errors from the **try**

finally

- Always called
- Great for cleaning up

```
async function example() {
  showProgressbar(true);
  let hero: Hero;
  try {
    hero = await getHero(email);
    hero.orders = await getOrders(hero);
    showHero(hero);
  } catch (error) {
    showMessage(error);
  } finally {
    showProgressbar(false);
  }
}
```

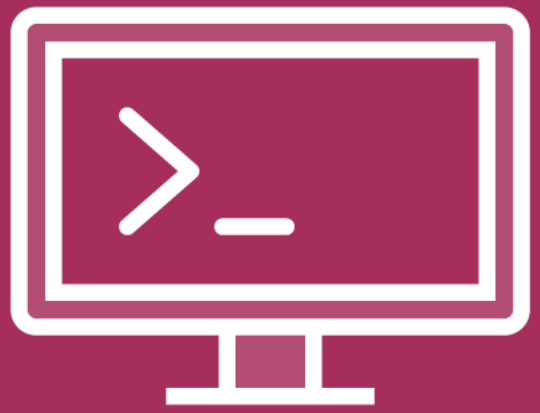
Analyzing Async/Await

try/catch/finally

- Important for error handling
- Consider if the error is intended for a developer or a user
- Clean up

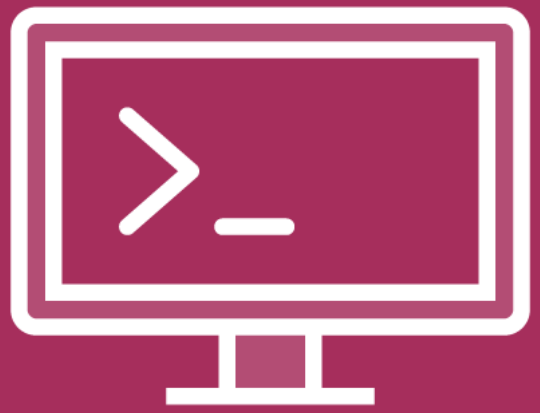
```
async function example() {
  showProgressbar(true);
  let hero: Hero;
  try {
    hero = await getHero(email);
    hero.orders = await getOrders(hero);
    showHero(hero);
  } catch (error) {
    showMessage(error);
  } finally {
    showProgressbar(false);
  }
}
```

Demo



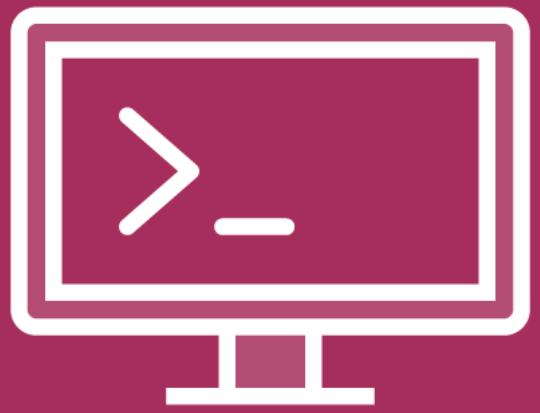
Creating Async Functions

Demo



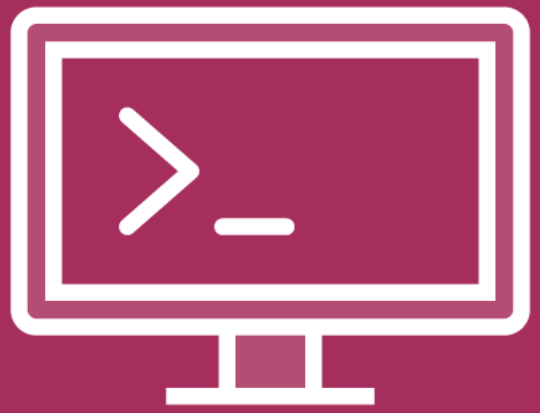
Throwing Errors

Demo



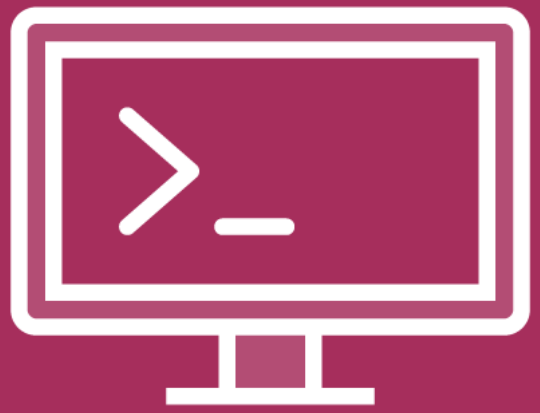
Getting Heroes with Promise.all

Demo



Promise.all and for await of

Demo



Callbacks, Promises, Async/Await

Summary

It’s async, but behaves like sync

Creating async functions

The impact of throwing errors

Use try/catch/finally blocks

Promise.all

for await of

Callbacks, promises and async/await