Getting started with Async/Await

11 APRIL 2017

Project Files

> *Click here to download the project files.*

TL;DR

The async/await function is part of the ES2017 specification. The purpose of async/await functions is to simplify the behavior of using promises synchronously. You can create an async function by marking a function `async`:

```
async function myFn() {
  return 5;
}
```

When an async function is called it returns a `Promise`. If an async function returns a value, the `Promise` will be resolved with that value:

```
myFn().then(d => console.log(d)); // -> 5
```

You can use the `await` operator inside an `async` function to pause the function until the `Promise` that is marked with `await` is resolved:

```
async function getUserMessages() {
  const userInfo = await userService.getUserInfo(); // functio
  const userMessages = messageService.getMessagesFor(userInfo.
  return userMessages; // userMessages is promise that will re
}

getUserMessages().then(messages => console.log(messages));
```

Dawn of Doom

I'm sure you are familiar with a variation of this kind of code containing a lot of nested callbacks:

```
first(function (d1) {
  second(function (d2) {
    third(function(d3) {
      forth(function (d4) {
        console.log(d1 + d2 + d3 + d4);
      });
    });
  });
});
```

This happens beause each operation relies on the result of the previous one. Now, if each function returns a promise, you can re-write the above:

```
function all() {
  var r1;
  var r2;
  var r3;
  first()
  .then(function(d1) {
    r1 = d1;
    return second();
  })
  .then(function(d2) {
    r2 = d2;
    return third();
  })
  .then(function(d3) {
    r3 = d3;
    console.log(r1, r2, r3);
  });
}
all();
```

Looking at the amount of time that each call takes, you can tell how long it will take in total to see the log result:

- `first()`: 1 second
- `second()`: 2 seconds
- `third()`: 3 seconds
- total time: 1 + 2 + 3 = 6 seconds

The list above is to emphasize that each promise has to be resolved until we move on to the next one. That's why in total it will take 6 seconds to print the final message to the