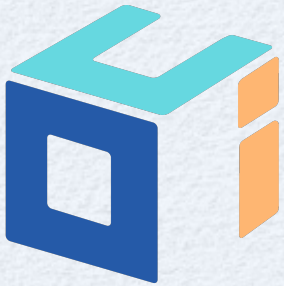
`async` and `await`

Mark Volkmann
Object Computing, Inc.



OCI | TRAINING

Overview

- **`async`** and **`await`** are two new keywords being added to JavaScript in ES2017
- Make it much easier to work with functions that return promises
- Can be used today in browsers by utilizing Babel
- Enabled by default in Node v7.6 and above

Promises

- Represent a value that may be available in the future
- In general, a promise is
  resolved when things go right and
  rejected when things go wrong

Promise Example

- Code on next slide uses promises directly by chaining calls to **`then`** and **`catch`**
  - uses Fetch API
    - commonly used to make REST calls from client side web app code
    - has methods that return promises
  - same technique would apply to any functions that return promises

```
const fetch = require('node-fetch');

function demo() {
  const urlPrefix = 'http://localhost:3000';
  const username = 'mvolkmann';
  const storeName = 'Taco Bell';
  let url = `${urlPrefix}/people/${username}/zip`;
  let zip;
  fetch(url)
    .then(res => res.text())
    .then(zipCode => {
      zip = zipCode;
      console.log('zip =', zip);
      url = `${urlPrefix}/stores/locations?zip=${zip}&name=${storeName}`;
      return fetch(url);
    })
    .then(res => {
      if (res.status === 404) {
        throw new Error(`There are no ${storeName} stores in ${zip}.`);
      }
      return res.json();
    })
    .then(locations => {
      console.log(`${storeName} locations are:`);
      for (const location of locations) {
        console.log(location);
      }
    })
    .catch(e => console.error(e.message));
}

demo();
```

async/await Example

- Code on next slide uses **`async`** and **`await`** keywords
- Note how all calls to functions that return promises are inside a **`try`** block preceded by the **`await`** keyword, and errors from them are handled in the **`catch`** block
- **`await`** keyword can be applied to any function call, even ones that do not return promises
  - allows functions that previous returned a promise to be changed to return an ordinary value without breaking existing callers
- Note how this makes writing asynchronous code look similar to writing synchronous code

```
const fetch = require('node-fetch');

async function demo() {
  const urlPrefix = 'http://localhost:3000';
  const username = 'mvolkmann';
  const storeName = 'Taco Bell';

  try {
    let url = `${urlPrefix}/people/${username}/zip`;
    let res = await fetch(url);
    const zip = await res.text();
    console.log('zip =', zip);

    url = `${urlPrefix}/stores/locations?zip=${zip}&name=${storeName}`;
    res = await fetch(url);
    if (res.status === 404) {
      throw new Error(`There are no ${storeName} stores in ${zip}.`);
    }

    const locations = await res.json();
    console.log(`${storeName} locations are:`);
    for (const location of locations) {
      console.log(location);
    }
  } catch (e) {
    console.error(e.message);
  }
}

demo();
```

Questions

- What happens if **`await`** is used inside a function that is not marked as **`async`**?
  - you'll get a **`SyntaxError`**
- What happens if you call a function marked as **`async`** that returns a promise without using **`await`**?
  - it just returns the promise object without waiting for it to resolve or reject
- What happens if you call a function using **`await`**, but the function is not marked as **`async`**?
  - it returns its value immediately
- Which kinds of functions can be marked as **`async`**?
  - this works with regular functions and arrow functions, but methods cannot be marked as **`async`**

Recommendation

- Search your code for calls to **`then`** and **`catch`** and replace all of them with use of **`async`** and **`await`**
- It will make your code much easier to read!

- These slides and the code in them can be found at https://github.com/mvolkmann/async-await-screencast