Callbacks

Callbacks as we know them

- When things happen asynchronously, we need to provide code that runs when it's ready
- Typically we provide a callback function for when the operation is done, that may receive some data
- Most asynchronous operations so far have been either user events (click, resize) or after some time (setTimeout)

Callbacks in Node

- Because we want our server code to run fast, most of Node is run in this asynchronous style
- This allows for us to not be blocking the main thread from handling other things
- These behave the same as other callbacks, but often come in a more conventional and uniform format

Node Callback Convention

- Also known as "error-first", "errback", "Node-style callbacks"
- Callback functions accept an error argument of type `Error` as the first parameter
- If the operation has additional data, they'll be additional parameters after error
- The callback only fires once, so you won't get both an error and a result at different times
- This convention allows for us to write reusable code that can rely on the function signature

Example of a Node callback

```
const fs = require("fs");
fs.readFile("./myfile.txt", "utf-8", function(err, data) {
  // Handle errors where myfile.txt isn't there
  if (err) {
    console.error("Unable to read myfile.txt!");
    console.error(err);
    process.exit(1); // Exit the program
  }

  // Print out the contents of myfile.txt
  console.log(data);
});
```

Implementing our own Node-style callback

```
function getFileLength(path, cb) {
  // Immediately return an error if they didn't give us a path
  // Return so that callbacks in readFile don't trigger
  if (!path) {
    cb(new Error("No path provided!"));
    return;
  }

  // Attempt to read the file
  fs.readFile(path, "utf-8", function(err, data) {
    // If we had an issue, return the error
    if (err) {
      cb(err);
    }
    // Otherwise return the file length
    else {
      cb(null, data.length);
    }
  });
}
```

Exercise: Drink Refill

- Write a module the exports a function that takes three parameters: `drinkType`, `hasIce`, and `callback`
- `drinkType` is mandatory. If it is empty, null, or undefined, call back with an `Error` stating that it is
- If the drink is `"iced tea"` and does not have ice, call back with an `Error` stating that drink requires ice
- Otherwise, refill the drink, simulating the time for refill with a `setTimeout` of one second. Call back with a message specifying that the drink has been refilled
- Write a callback function that accepts the `error` and `message`
- Test out your new function by importing the module and calling it a few times!