Node.js: File IO

`fs` Library

- Built into Node.js
- Allows easy interaction with files for read, write, existence validation, and permissios checking
- All methods are asynchronous, but many have synchronous implementations

Reading Files

- Say we need to read `./helloworld.txt` from disk:

```
js
//require the `fs` library
var fs = require('fs');
//call `fs.readFile`
//`utf-8` is an encoding to ensure we get text
fs.readFile('./helloworld.txt', 'utf-8', function(err, data){
//check for possible errors - more on this later
if (err) {
    //log that an error happened
    console.log(`an error occurred: ${err}`);
    //throw the error for handling by the caller
    throw err;
}
//otherwise, print the contents of the file
console.log(data);
});
```

- Like with `require`, the paths can be relative or absolute

Write Files Example

- Say we need to write to `./helloworld.txt` on disk:

```
js
//require the `fs` library
var fs = require('fs');
//make some content
var myContent = "hello world!\nHow are you today?";
//write that content to `helloworld.txt`
fs.writeFile('./helloworld.txt', myContent, function(err){
//check for possible errors
if (err) {
    //log that an error happened
    console.log(`there was a problem writing: ${err}`);
    //throw the error for handling by the caller
    throw err;
}
//otherwise, print a success message
console.log('content written.');
})
```

Removing a File

- Say we need to remove `./helloworld.txt`:

```
js
var fs = require('fs');
//`unlink` removes the file
fs.unlink('./helloworld.txt', function(err){
//check for possible errors
if(err){
    //log that an error happened
    console.log(`an error occurred: ${err}`);
    //throw the error for handling by the caller
    throw err;
}
});
```

Possible Errors with File Handling

- Trying to read a file that does not exist will cause an error
```
js
fs.readFile('/does/not/exist.txt', function(err, data){
//will print out an error object with the message
//"Error: ENOENT: no such file or directory, open '/does/not/exist.txt'"
//ENOENT is C shorthand for "Error No ENTry"
console.log(err);
});
```

- Trying to read or write to a file without proper permissions will also cause an error
```
js
//if `helloworld.txt` is set up to read-only (eg, `chmod 444 helloworld.txt`)
fs.writeFile('./helloworld.txt', 'overwriting hello world!', function(err){
//will print an error object with the message
//"Error: EACCES: permission denied, open './helloworld.txt'"
console.log(err);
});
```

Reading All Files in a Directory

- Say we want to know what files are in our current directory:

```
js
fs.readdir('.', function(err, files){
//`files` will now contain string file names in the
current directory
//note that the same existence and permissions rules from
read and write apply,
//meaning the directory must exist and you must have read
permissions on it
console.log(files);
});
```

Exercise: Self-duplication

- Create an application that reads itself as a file and creates a duplicate of itself.
- Hint: `__filename` contains the current file name

Exercise: Reading Multiple Files

- Combine `fs.readdir` and `fs.readFile`
- Print the name and contents of the read files
- BONUS: Use `Promises` to wrap `fs.readdir` and `fs.readFile`, using `Promise.all` to print out a helpful completion message with how many files were read