WDD330

Ribeka Nanjo

W0 Reading (L7)

Further Functions

Immediately Invoked Function Expressions. (IIFE)

- It is invoked as soon as it's defined.

e.g.
```
(function(){
const temp = 'World';
console.log(`Hello ${temp}`);
})();
<< 'Hello World'
```

- **Safe use of strict mode**
  - One of the problems with simply placing 'use strict' at the beginning of a file is that it will enforce strict mode on all the JavaScript in the file.
  - To solve the problem, "use strict " is placed inside an IIFE

e.g.
```
(function() {
'use strict';
// All your code would go inside this function
})();
```

Promises

- It represents the future result of an asynchronous operation.
- They help simplify the process, and avoid the convoluted code that can result from using multiple callbacks.

Pending: not yet fulfilled or rejected.
Fulfilled: the result is a value (complete).
Rejected: the result is an error (failed).

Async Function

- Using await operator

- Async makes a function return a Promise. await makes a function wait for a Promise

E.g.

async function:

```
async function loadGame(userName) {
try {
const user = await login(userName);
const info = await getPlayerInfo (user.id);
// load the game using the returned info
}
catch (error){
throw error;
}
}
```

Currying

- Currying is a transformation of functions that translates a function from callable as f(a, b, c) into callable as f(a)(b)(c)
- It doesn't call a function, just transforms it.

E.g.

```
function curry(func,...oldArgs) {
return function(...newArgs) {
const allArgs = [...oldArgs,...newArgs];
return func(...allArgs);
}}
const divider = (x,y) => x/y;
divider(10,5);
<< 2
```