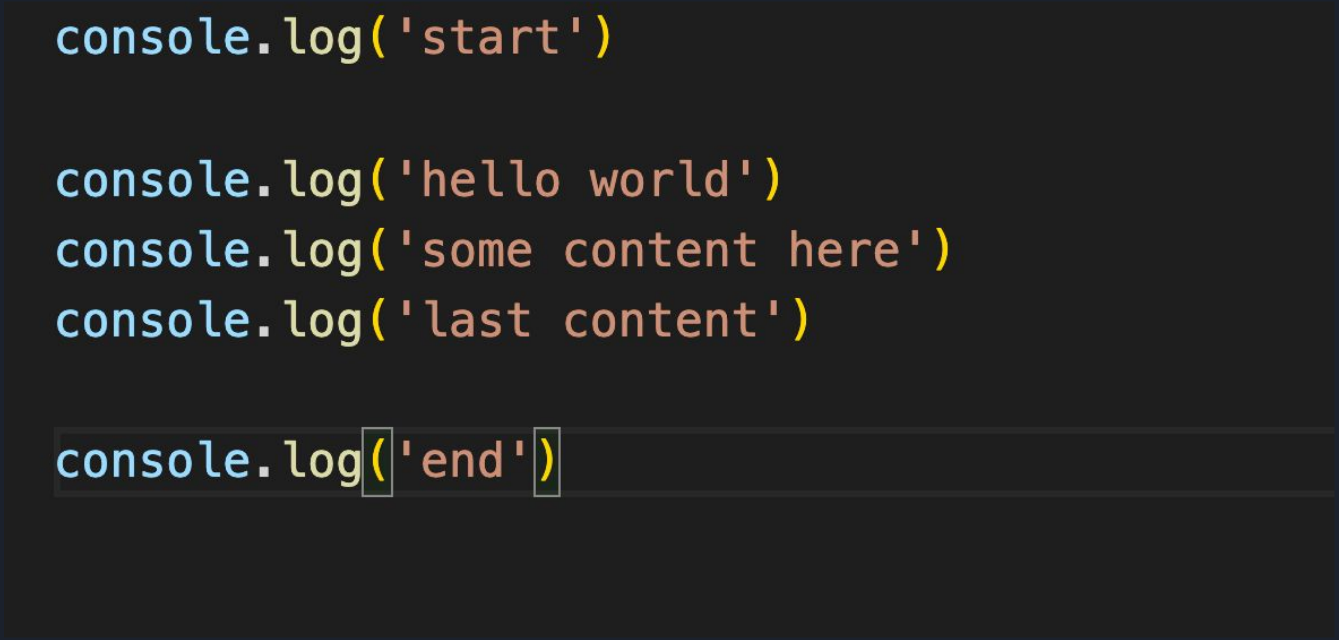
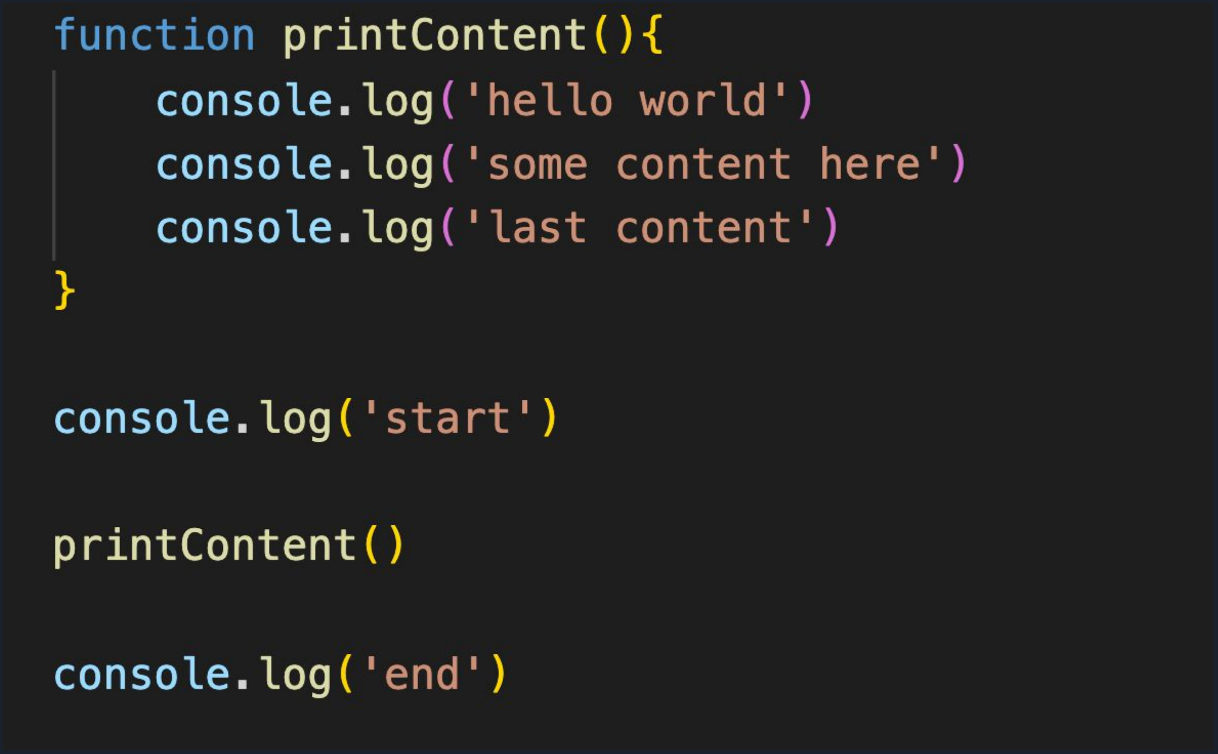
Asynchronous
Promise
async/await

Blocking and Non Blocking Code / Asynchronous



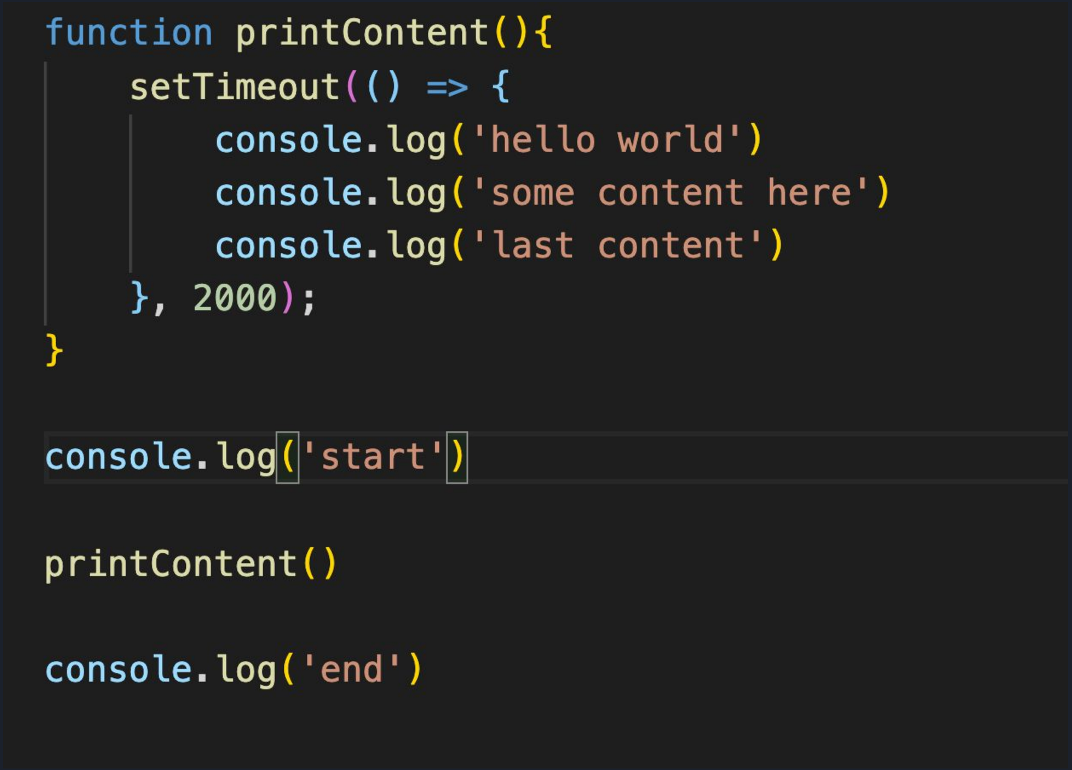
```
console.log('start')

console.log('hello world')
console.log('some content here')
console.log('last content')

console.log('end')
```

Blocking and Non Blocking Code / Asynchronous



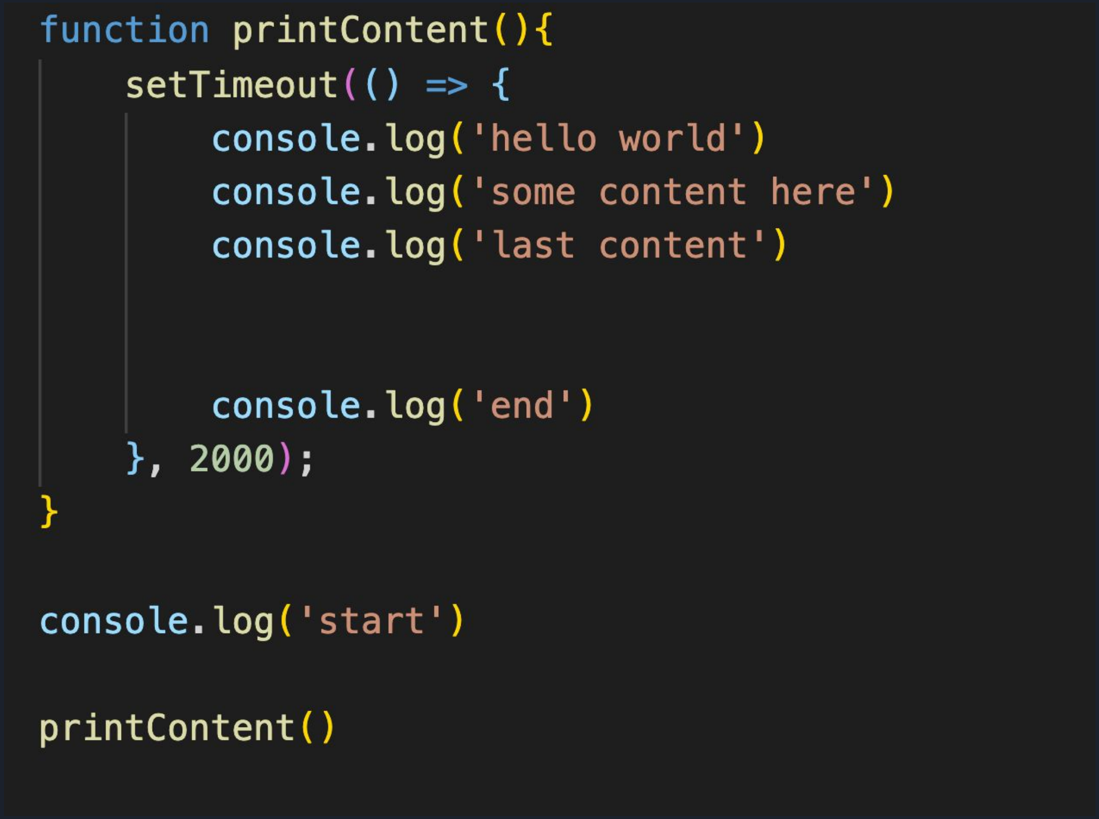
```
function printContent(){
    console.log('hello world')
    console.log('some content here')
    console.log('last content')
}

console.log('start')

printContent()

console.log('end')
```

Blocking and Non Blocking Code / Asynchronous



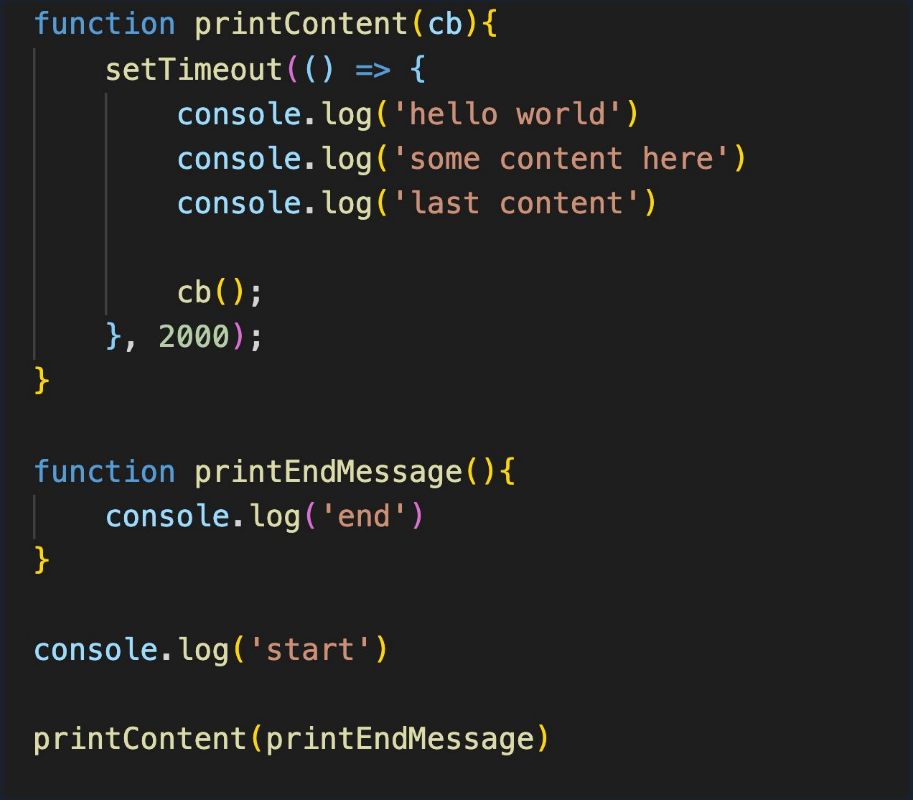
```
function printContent(){
    setTimeout(() => {
        console.log('hello world')
        console.log('some content here')
        console.log('last content')
    }, 2000);
}

console.log('start')

printContent()

console.log('end')
```

Blocking and Non Blocking Code / Asynchronous



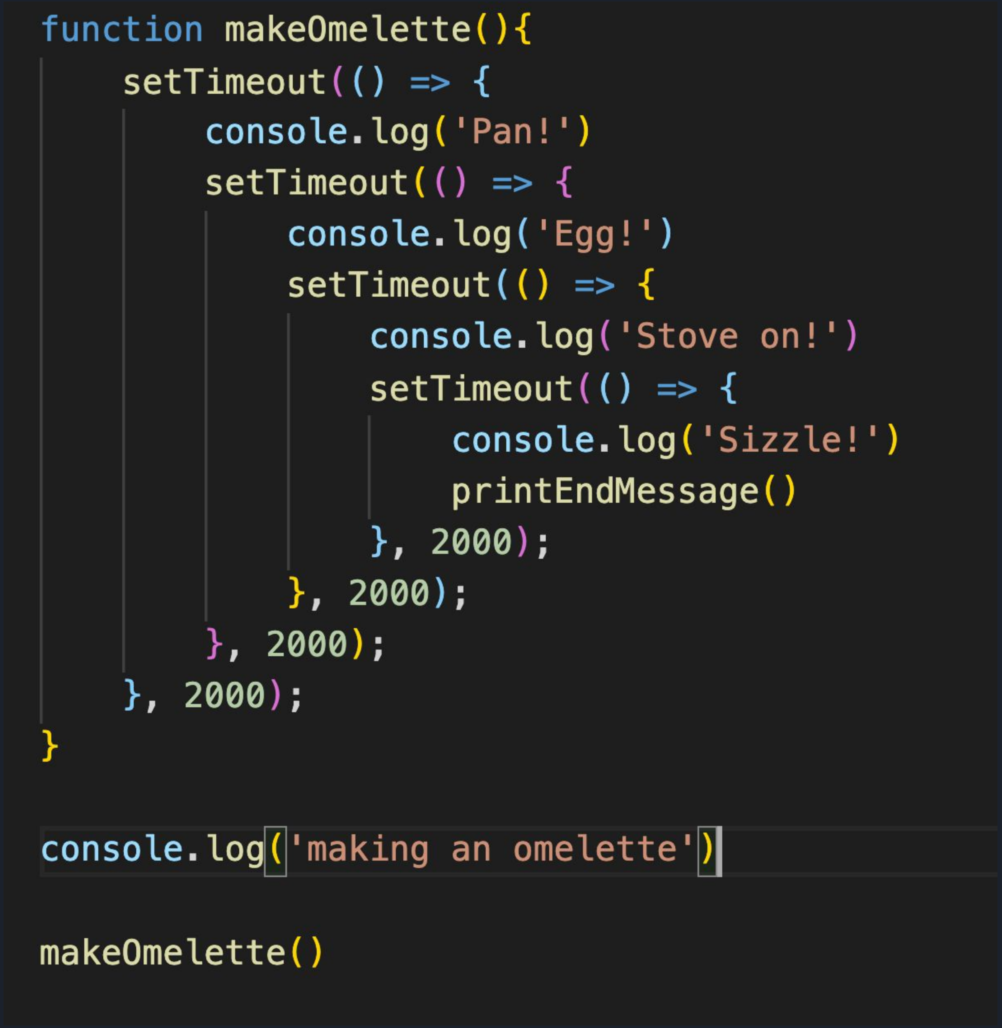
```javascript
function printContent(){
    setTimeout(() => {
        console.log('hello world')
        console.log('some content here')
        console.log('last content')


        console.log('end')
    }, 2000);
}

console.log('start')

printContent()
```

Callback pattern



```
function printContent(cb){
    setTimeout(() => {
        console.log('hello world')
        console.log('some content here')
        console.log('last content')

        cb();
    }, 2000);
}

function printEndMessage(){
    console.log('end')
}

console.log('start')

printContent(printEndMessage)
```

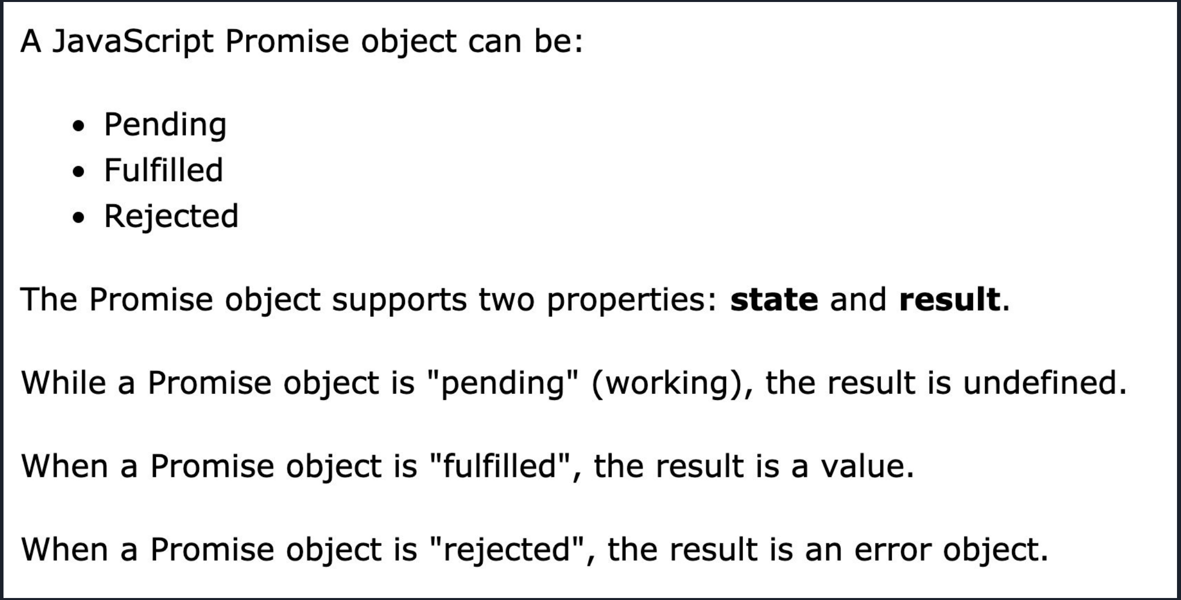
Callback hell



```
function makeOmelette(){
    setTimeout(() => {
        console.log('Pan!')
        setTimeout(() => {
            console.log('Egg!')
            setTimeout(() => {
                console.log('Stove on!')
                setTimeout(() => {
                    console.log('Sizzle!')
                    printEndMessage()
                }, 2000);
            }, 2000);
        }, 2000);
    }, 2000);
}

console.log('making an omelette')

makeOmelette()
```

Promise object

https://www.w3schools.com/js/js_promise.asp



A JavaScript Promise object can be:

- Pending
- Fulfilled
- Rejected

The Promise object supports two properties: **state** and **result**.

While a Promise object is "pending" (working), the result is undefined.

When a Promise object is "fulfilled", the result is a value.

When a Promise object is "rejected", the result is an error object.

Promise object - Example

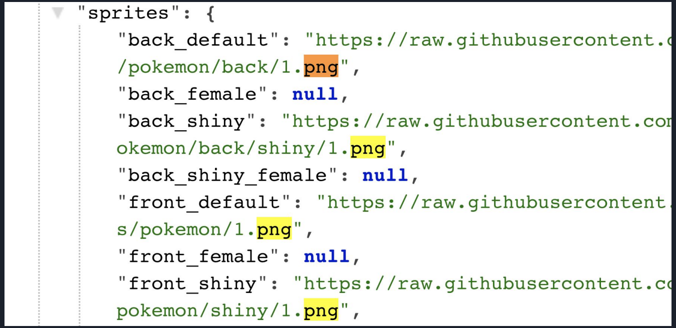
async / await syntax

Recommended reading material

Chapter 11 https://eloquentjavascript.net/11_async.html

Practice exercise

1. Go through the following documentation to learn about Fetch API in Javascript
2. Make an API call to the following end point
   https://pokeapi.co/api/v2/pokemon?offset=0&limit=1000
3. Upon receival of the API response, remember to parse JSON
4. Once you have managed to format the response in JSON (Javascript object format), display a anchor tag element with the name of each pokemon
5. When user clicks on each of the pokemon name, make another API call to retrieve that pokemon information and show it on the screen, use the front_default value under sprites of each pokemon as the display image



```
▼ "sprites": {
    "back_default": "https://raw.githubusercontent.c
    /pokemon/back/1.png",
    "back_female": null,
    "back_shiny": "https://raw.githubusercontent.co
    okemon/back/shiny/1.png",
    "back_shiny_female": null,
    "front_default": "https://raw.githubusercontent.
    s/pokemon/1.png",
    "front_female": null,
    "front_shiny": "https://raw.githubusercontent.c
    pokemon/shiny/1.png",
```