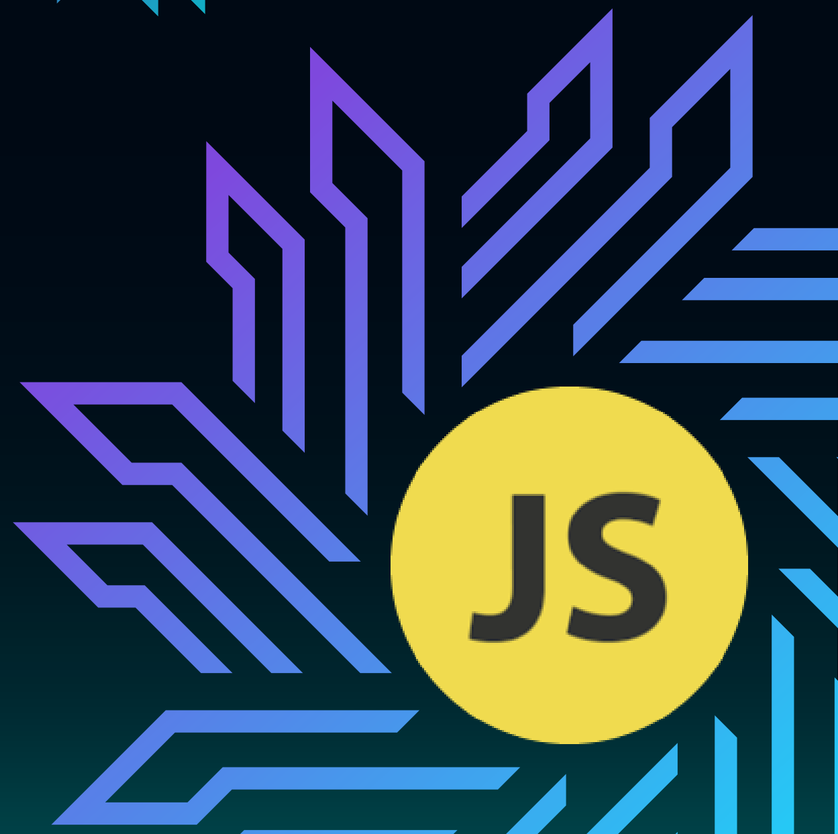
MNNIT CC

JAVASCRIPT

WEBSTER - 3rd Class

Created By : CC MNNIT

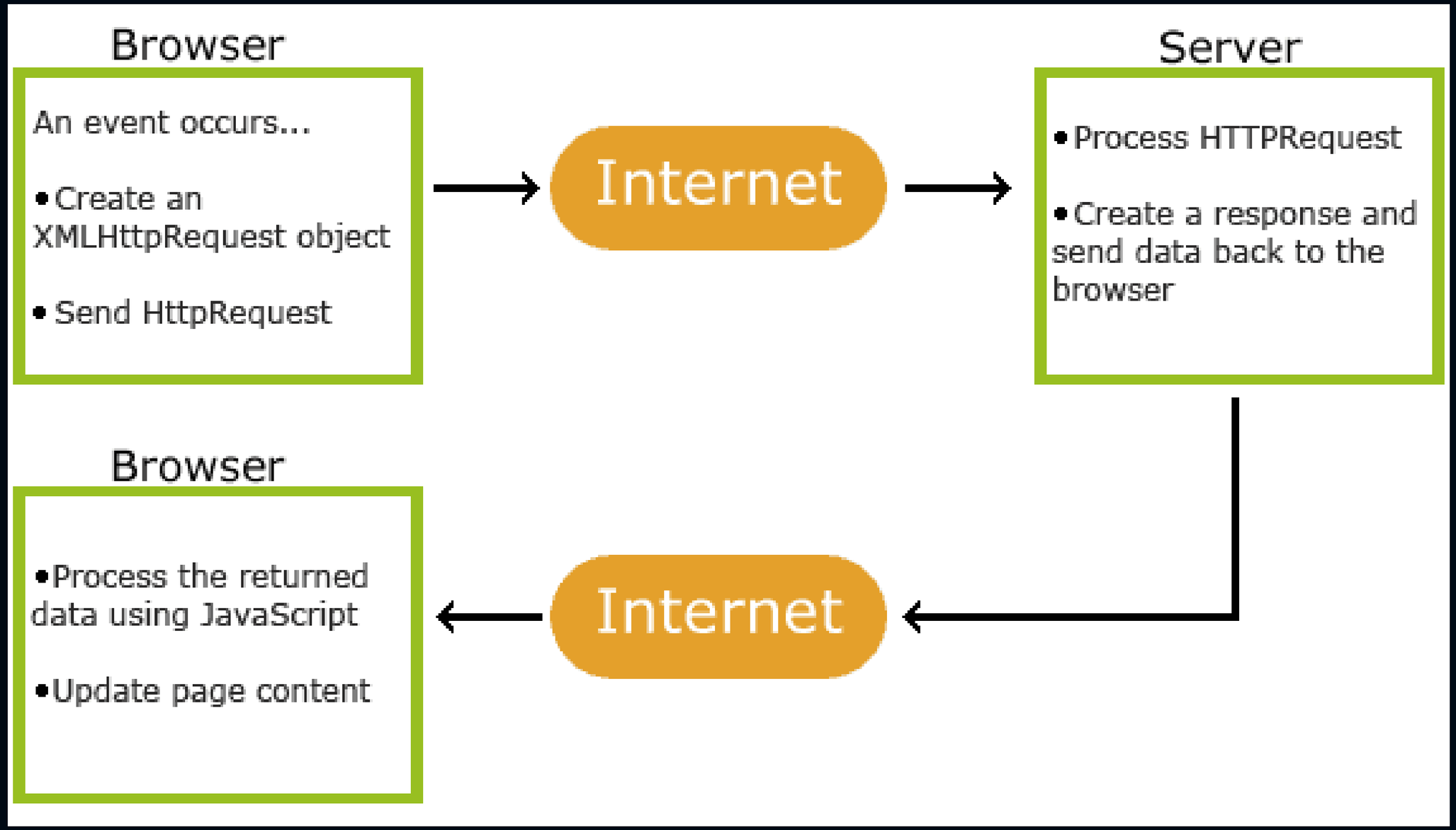


AJAX

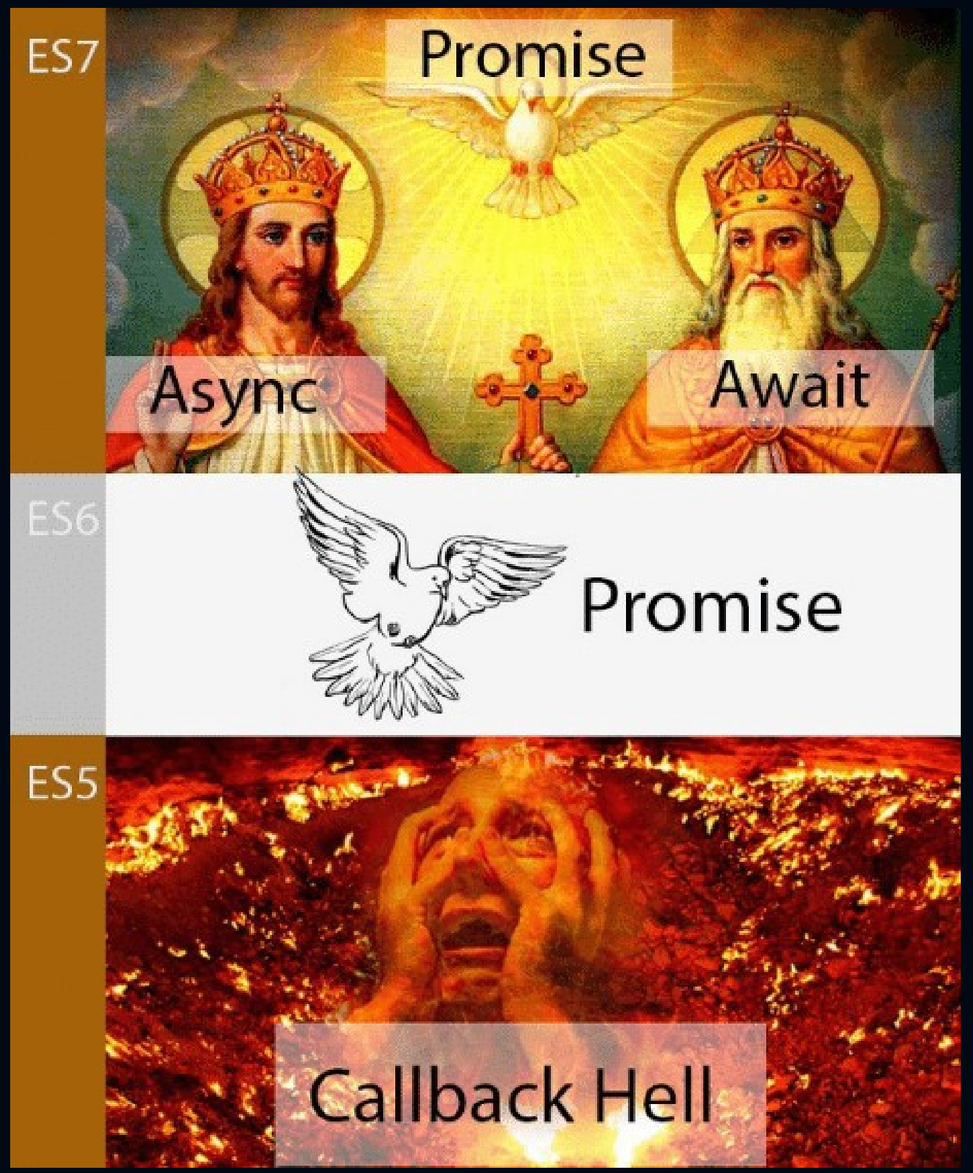
AJAX is a developer's dream, because you can:

- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

eg: 2_ajax\ex1.html


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Internet
Browser
• Process the returned data using JavaScript
• Update page content

Async Javascript



Asynchronous programming is a technique that enables your program to start a potentially long-running task and still be able to be responsive to other events while that task runs, rather than having to wait until that task has finished. Once that task has finished, your program is presented with the result.

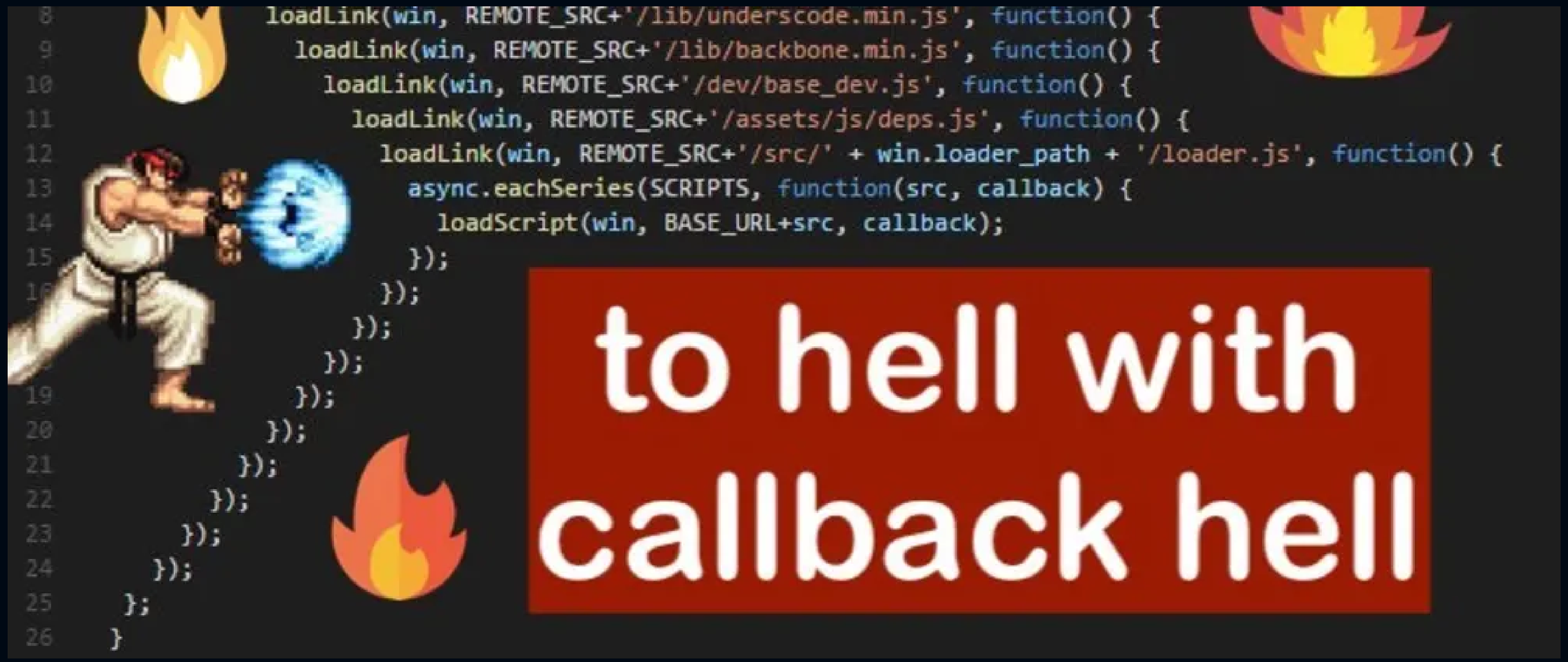
Event Loop

JavaScript is single-threaded: only one task can run at a time. Usually that's no big deal, but now imagine you're running a task which takes 30 seconds. During that task we're waiting for 30 seconds before anything else can happen

Callback

"I will call back later!"
A callback is a function passed as an argument to another function
This technique allows a function to call another function
A callback function can run after another function has finished

Callback Hell



This code hurts the eyes and writing this code hurt the brain. Do you see how the code gradually extends to the right? This is what ppl who code affectionately call the pyramid of doom aka the defining characteristic of callback hell .

Promises

"I Promise a Result!"

"Producing code" is code that can take some time

"Consuming code" is code that must wait for the result

A Promise is a JavaScript object that links producing code and consuming code

Promise Object Properties

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.

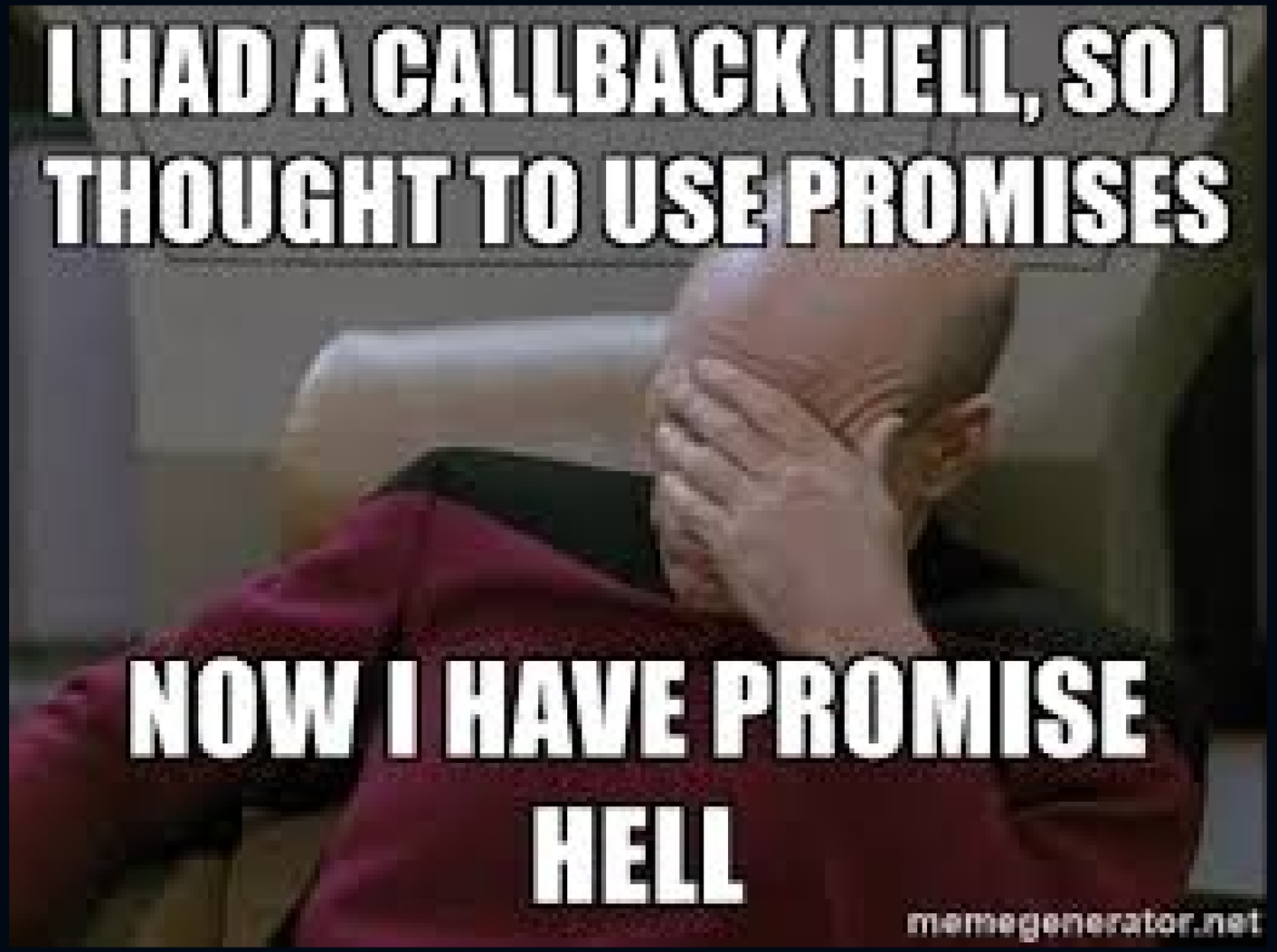
Syntax

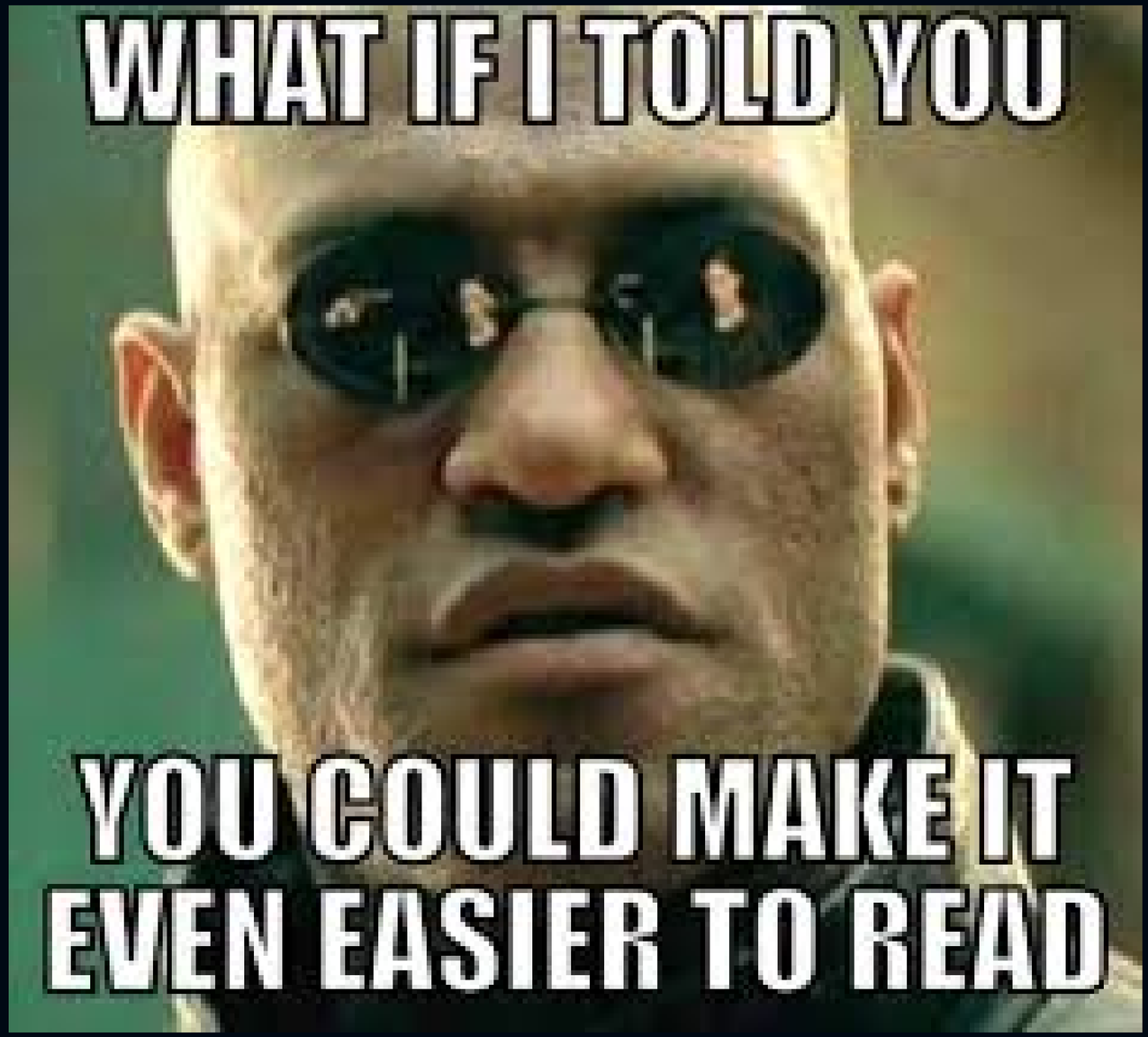
```
let myPromise = new Promise(function(myResolve, myReject) {
// "Producing Code" (May take some time)

  myResolve(); // when successful
  myReject();  // when error
});

// "Consuming Code" (Must wait for a fulfilled Promise)
myPromise.then(
  function(value) { /* code if successful */ },
  function(error) { /* code if some error */ }
);
```

```
const myPromise = new Promise((resolve, reject) =>{setTimeout(() =>
{resolve("foo");}, 300);});


myPromise
  .then((value) => `${value} and bar`)
  .then((value) => `${value} and bar again`)
  .then((value) => `${value} and again`)
  .then((value) => `${value} and again`)
  .then((value) => {
    console.log(value);
  })
  .catch((err) => {
    console.error(err);
  });
```

Promise Hell

```
connectDatabase()
  .then((database) => {
    return findAllBooks(database)
      .then((books) => {
        return getCurrentUser(database)
          .then((user) => {
            return pickTopRecommendation(books, user);
          });
      });
  });
```


I HAD A CALLBACK HELL, SO I THOUGHT TO USE PROMISES
NOW I HAVE PROMISE HELL
memegenerator.net


WHAT IF I TOLD YOU
YOU COULD MAKE IT
EVEN EASIER TO READ

Async-Await

"async and await make promises easier to write"

async makes a function return a Promise
await makes a function wait for a Promise

```
function myFunction() {
  return Promise.resolve("Hello");
}
```

```
async function myFunction() {
  return "Hello";
}
```

```
myFunction().then(
  function(value) {myDisplayer(value);}
);
```

```
const val = await  myFunction();
```