Introducción al Frontend + React

Aprende a desarrollar frontend utilizando las tecnologías más modernas

04: Javascript y el DOM

🦕

En clases pasadas

ES6+

Learn how to use
the most useful ES6+ features

const & let

Luis Resendiz
@luisrpw0

❌never use var

✔️

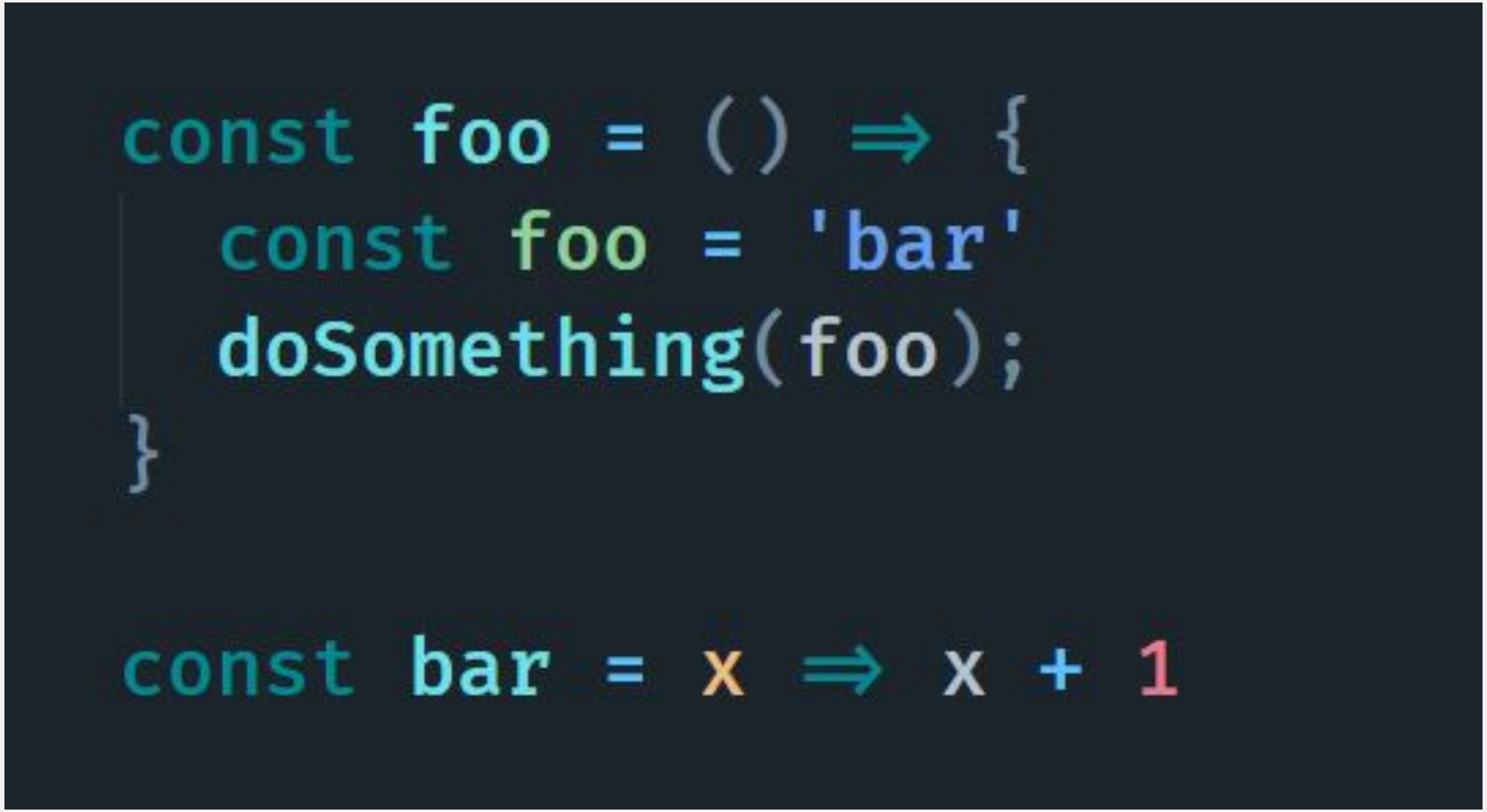
always use const
if mutate, then use let

ES6+

Learn how to use
the most useful ES6+ features

arrow functions

Luis Resendiz
@luisrpw0



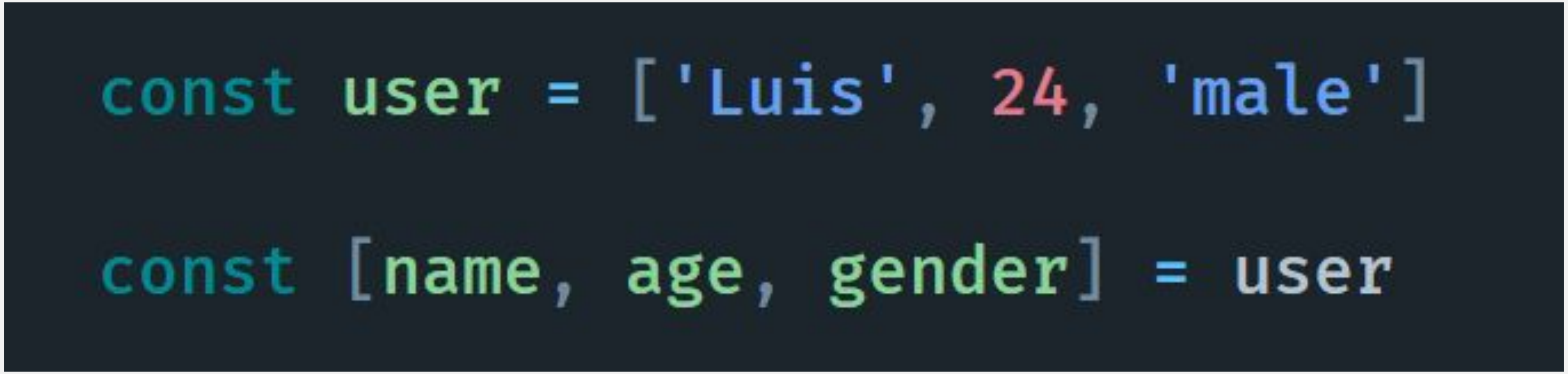
```javascript
const foo = () => {
  const foo = 'bar'
  doSomething(foo);
}

const bar = x => x + 1
```

ES6+

Learn how to use
the most useful ES6+ features

array destructuring

Luis Resendiz
@luisrpw0



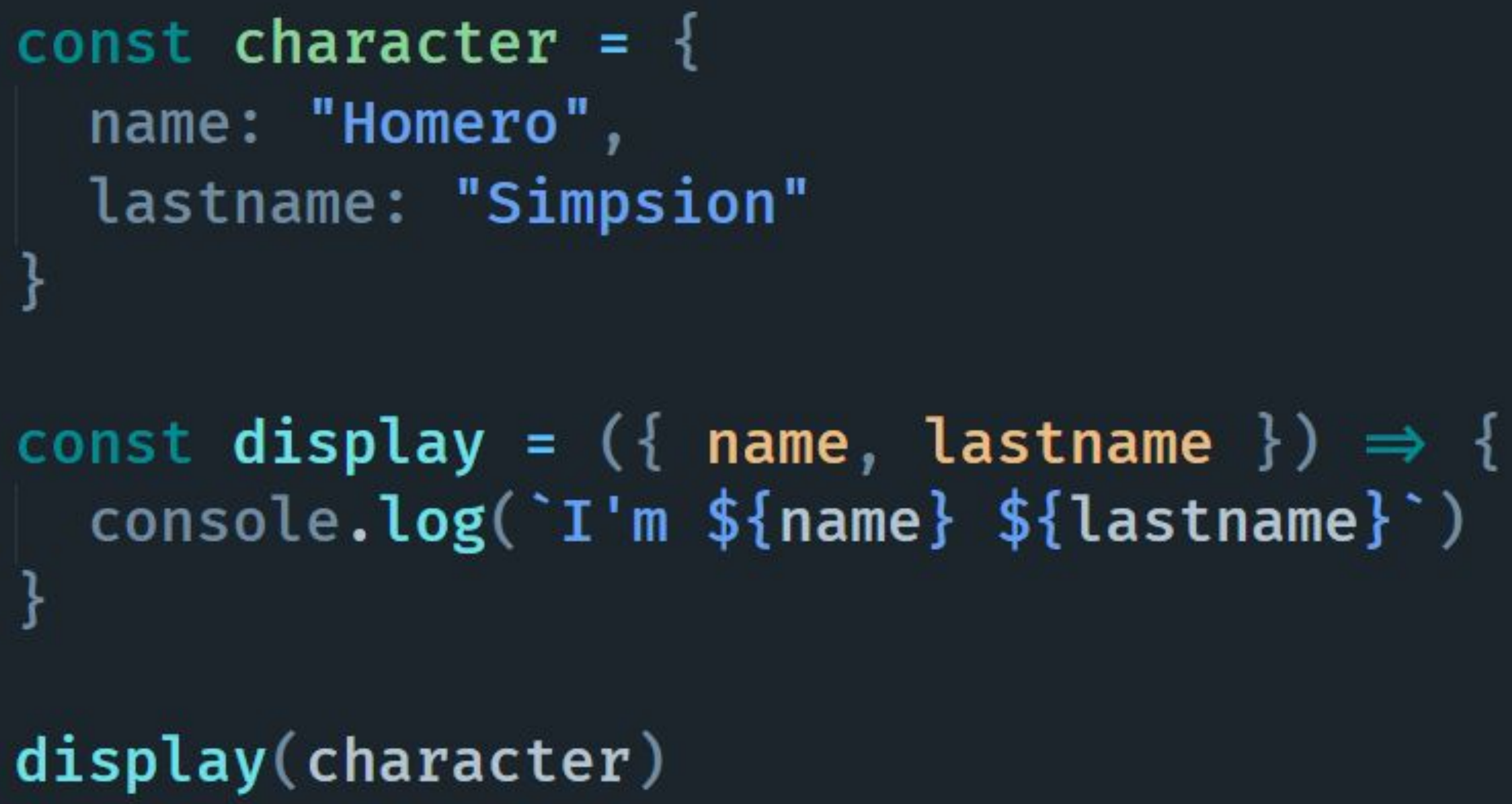
```javascript
const user = ['Luis', 24, 'male']

const [name, age, gender] = user
```

ES6+

Learn how to use
the most useful ES6+ features

object destructuring

Luis Resendiz
@luisrpw0



```javascript
const character = {
  name: "Homero",
  lastname: "Simpsion"
}

const display = ({ name, lastname }) => {
  console.log(`I'm ${name} ${lastname}`)
}

display(character)
```

expressions & statements

closures

```
function greeting(msg) {
    return function who(name) {
        console.log(`${ msg }, ${ name }!`);
    };
}

var hello = greeting("Hello");
var howdy = greeting("Howdy");

hello("Kyle");
// Hello, Kyle!

hello("Sarah");
// Hello, Sarah!

howdy("Grant");
// Howdy, Grant!
```

🐹

DOM

document.
getElementById( )

document.
createElement( )

document.
querySelector( )

addEventListener

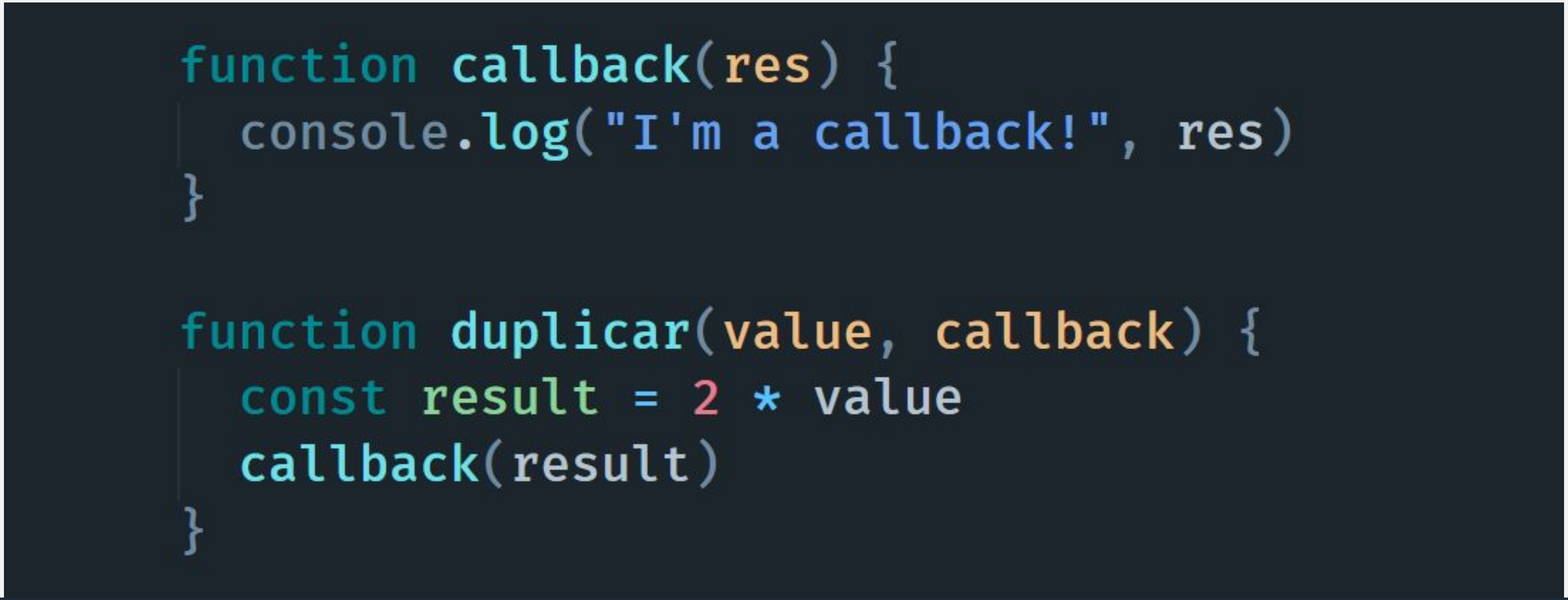
Preguntas?

Todos

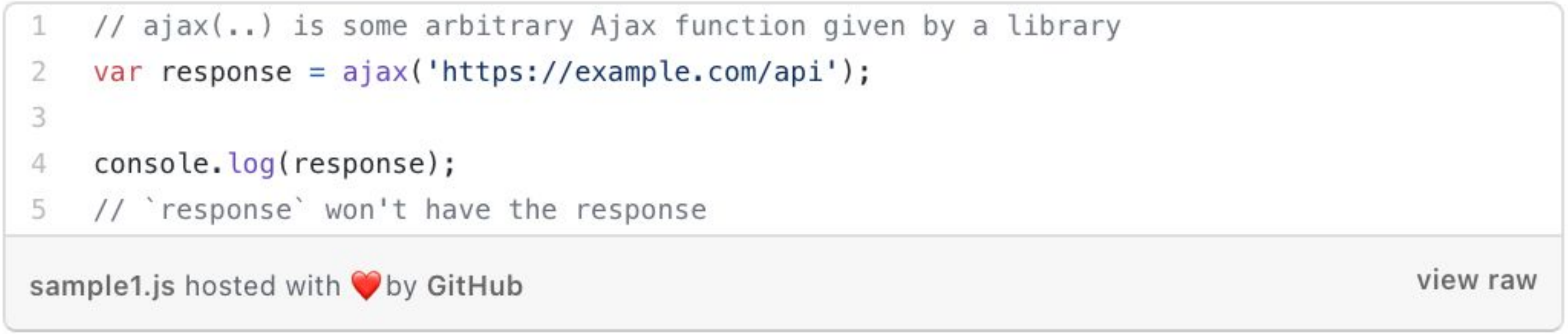
👽

callbacks

A callback is a plain Javascript function passed to some method as an argument or option

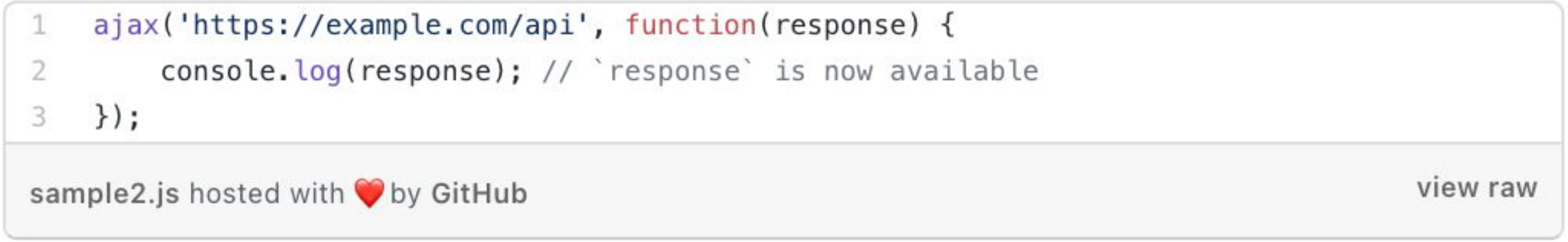


```
function callback(res) {
  console.log("I'm a callback!", res)
}

function duplicar(value, callback) {
  const result = 2 * value
  callback(result)
}
```



```javascript
// ajax(..) is some arbitrary Ajax function given by a library
var response = ajax('https://example.com/api');

console.log(response);
// `response` won't have the response
```

sample1.js hosted with ❤ by GitHub view raw



```javascript
ajax('https://example.com/api', function(response) {
    console.log(response); // `response` is now available
});
```

sample2.js hosted with ❤ by GitHub view raw `alert()`

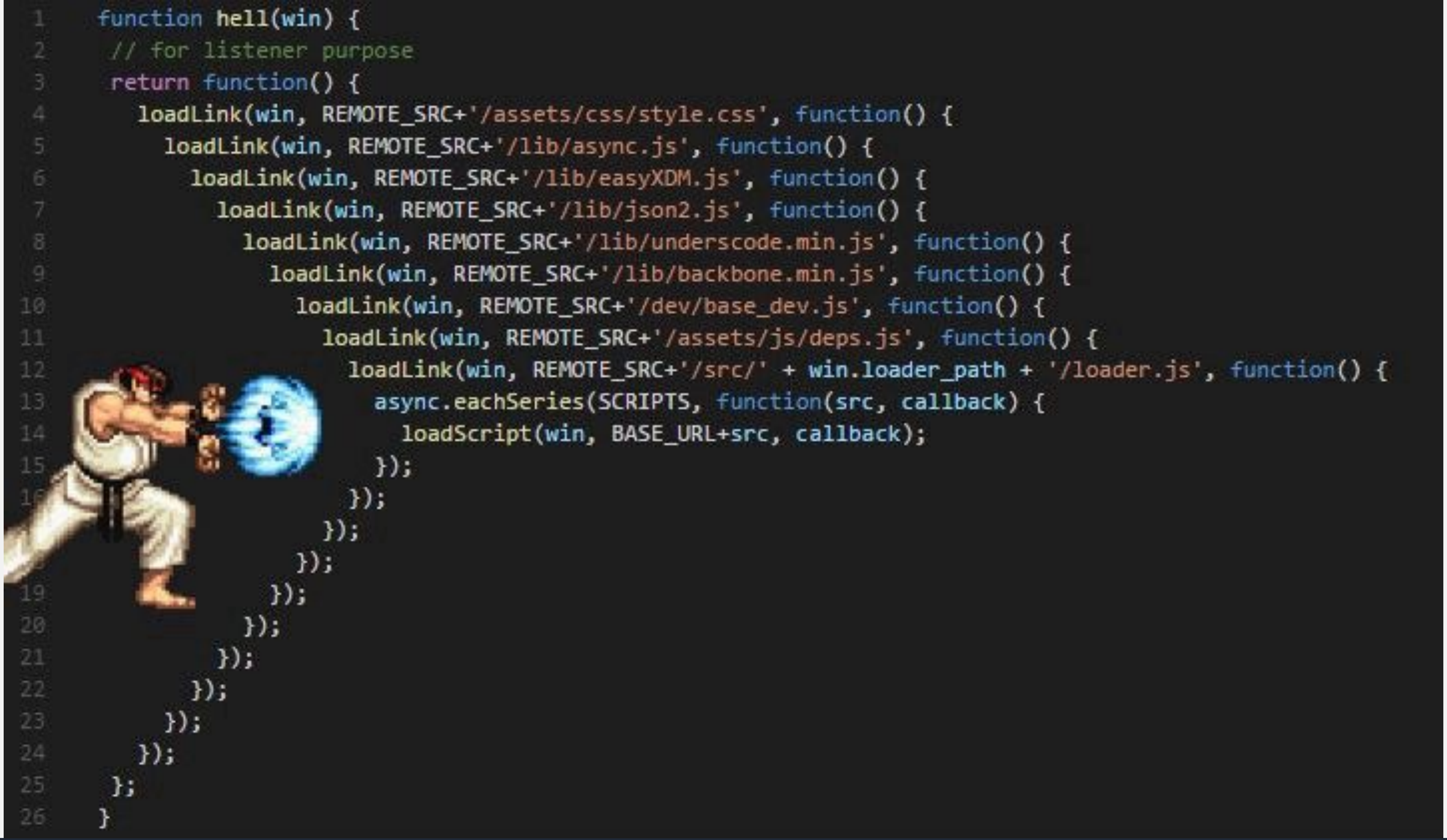
setTimeOut()

setInterval()

Create a Counter

🔥

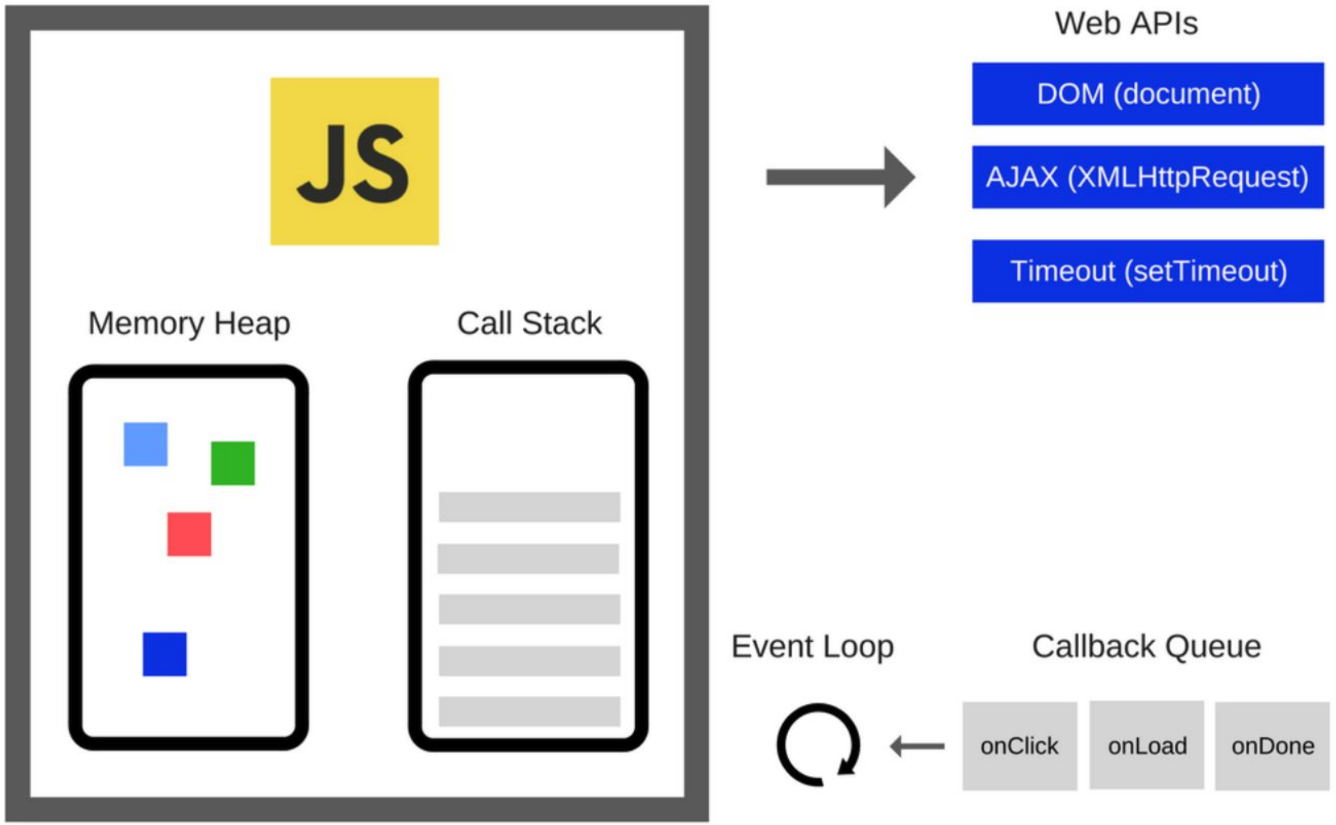
callback hell

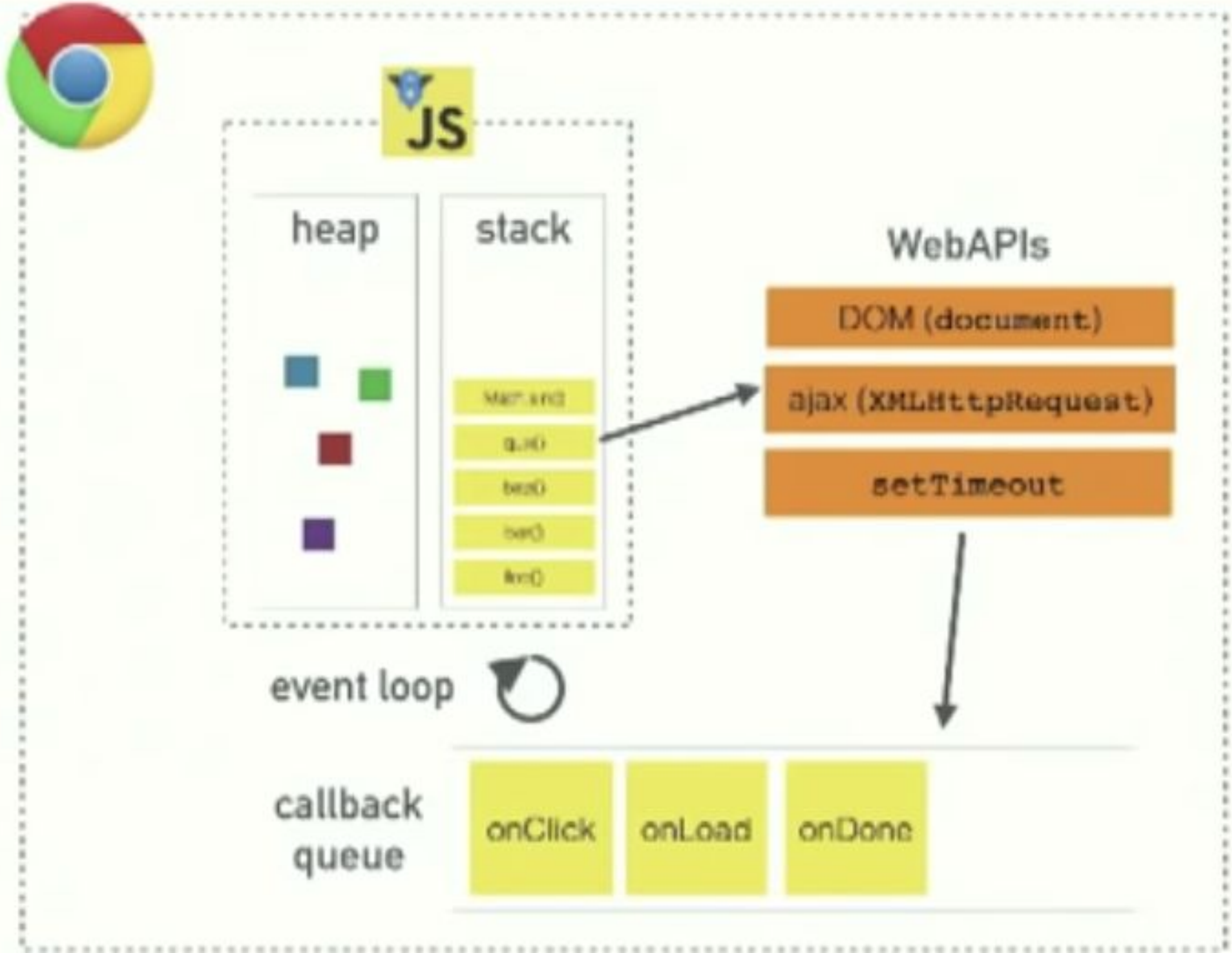


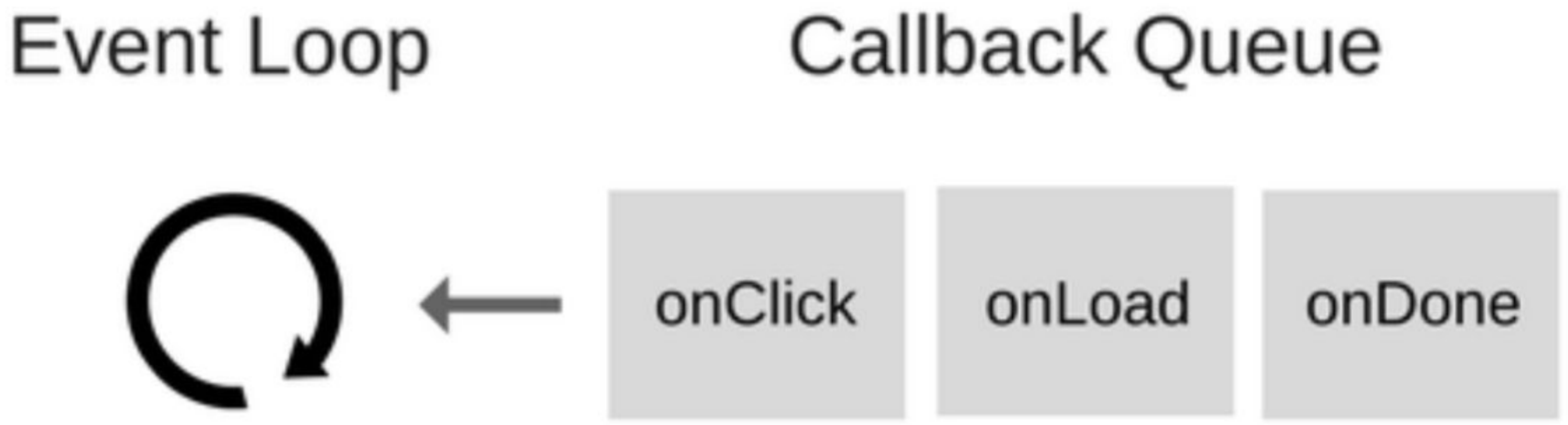
```
function hell(win) {
 // for listener purpose
 return function() {
  loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
   loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
    loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
     loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
      loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
        loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
         loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
          loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
           async.eachSeries(SCRIPTS, function(src, callback) {
            loadScript(win, BASE_URL+src, callback);
           });
          });
         });
        });
       });
      });
     });
    });
   });
  });
 };
}
```

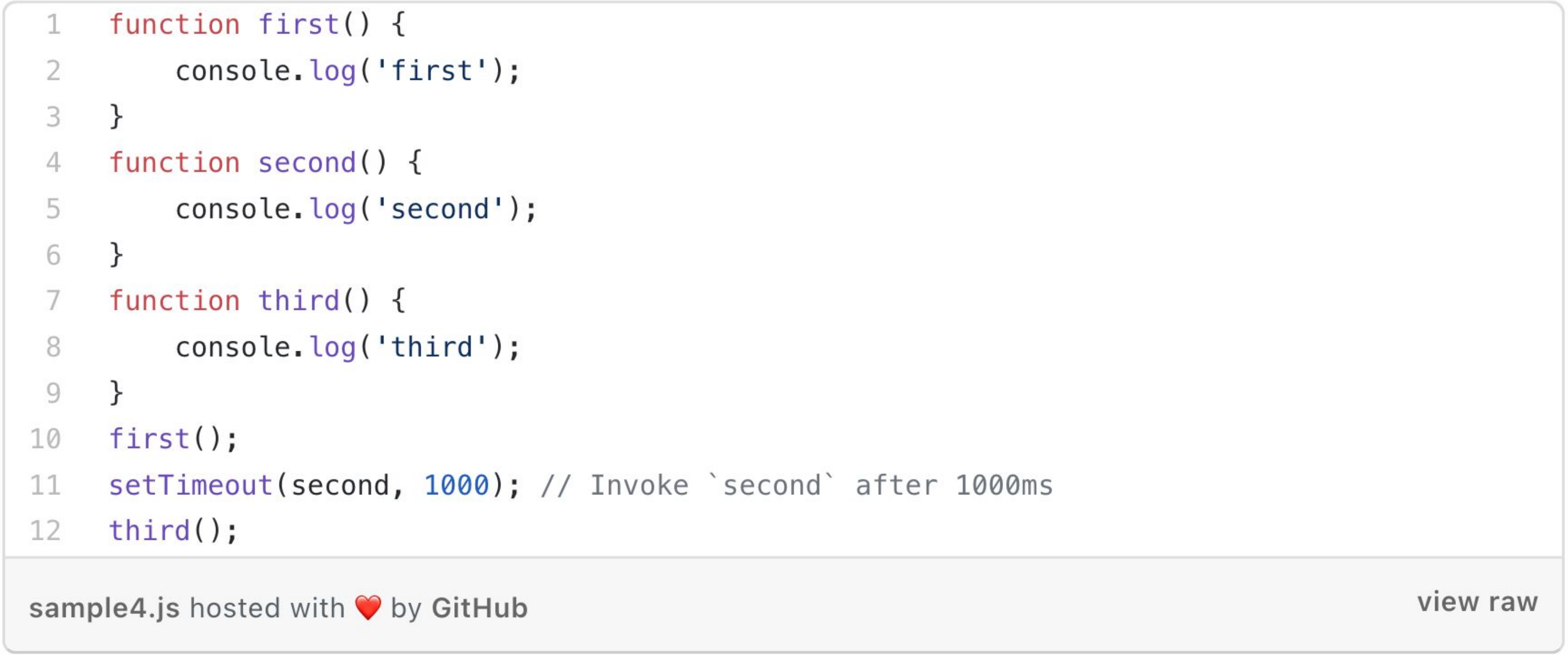
Preguntas?

Async Javascript


JS
Memory Heap
Call Stack
Web APIs
DOM (document)
AJAX (XMLHttpRequest)
Timeout (setTimeout)
Event Loop
Callback Queue
onClick
onLoad
onDone

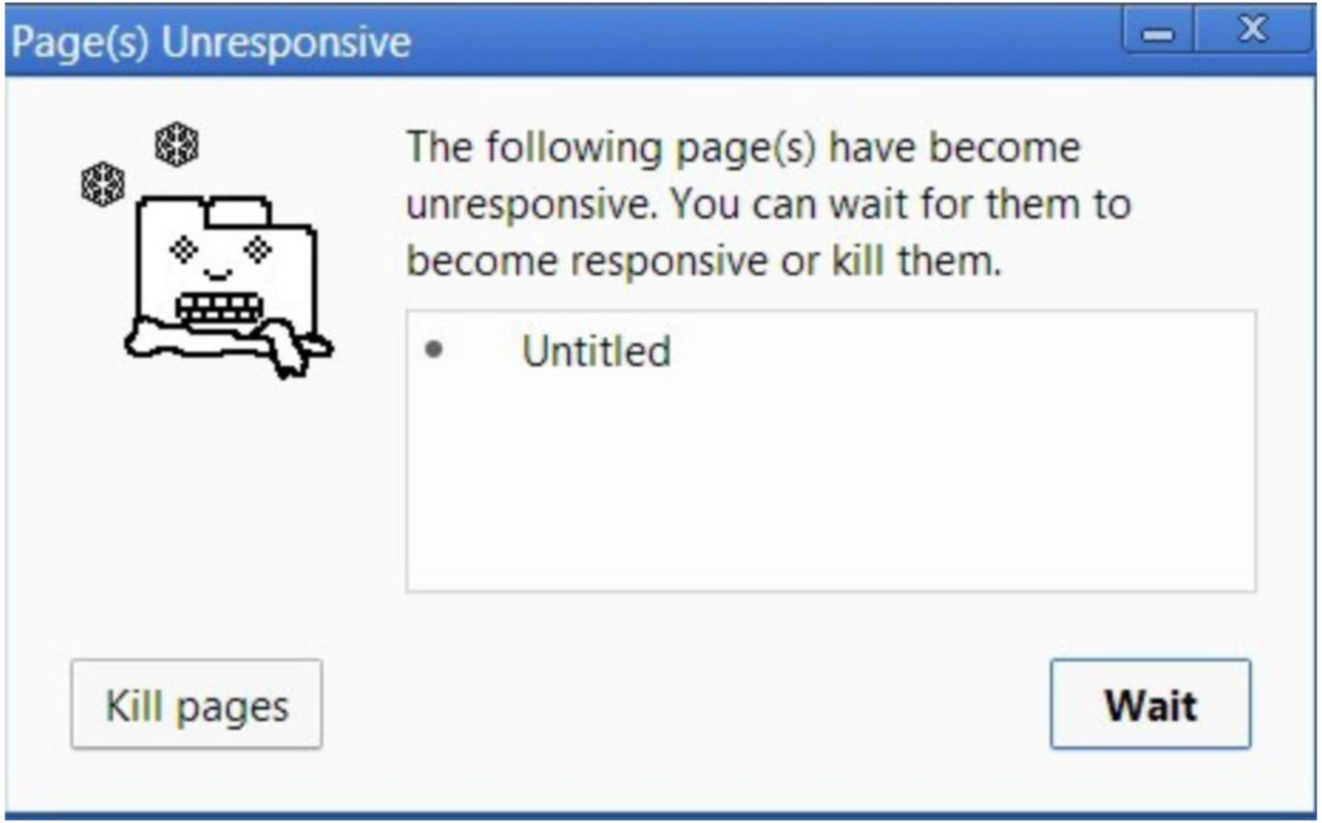

JS
heap
stack
[illegible]
qux()
baz()
bar()
foo()
WebAPIs
DOM (document)
ajax (XMLHttpRequest)
setTimeout
event loop
callback queue
onClick
onLoad
onDone


Event Loop
Callback Queue
onClick
onLoad
onDone



```javascript
function first() {
    console.log('first');
}
function second() {
    console.log('second');
}
function third() {
    console.log('third');
}
first();
setTimeout(second, 1000); // Invoke `second` after 1000ms
third();
```

sample4.js hosted with ❤️ by **GitHub** view raw

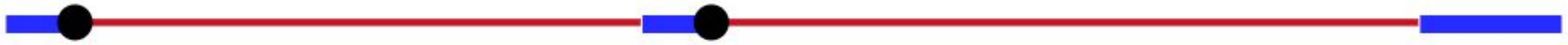


Page(s) Unresponsive

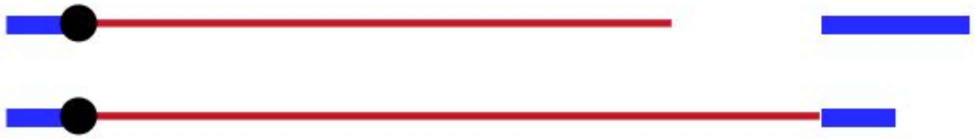
The following page(s) have become unresponsive. You can wait for them to become responsive or kill them.

- Untitled

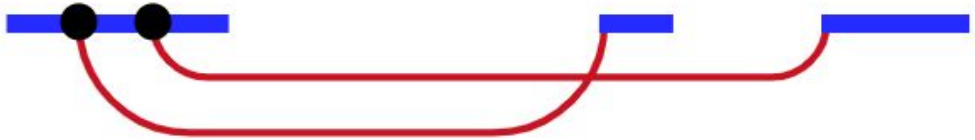
synchronous, single thread of control



synchronous, two threads of control



asynchronous



Single threaded and asynchronous

a

single-threaded

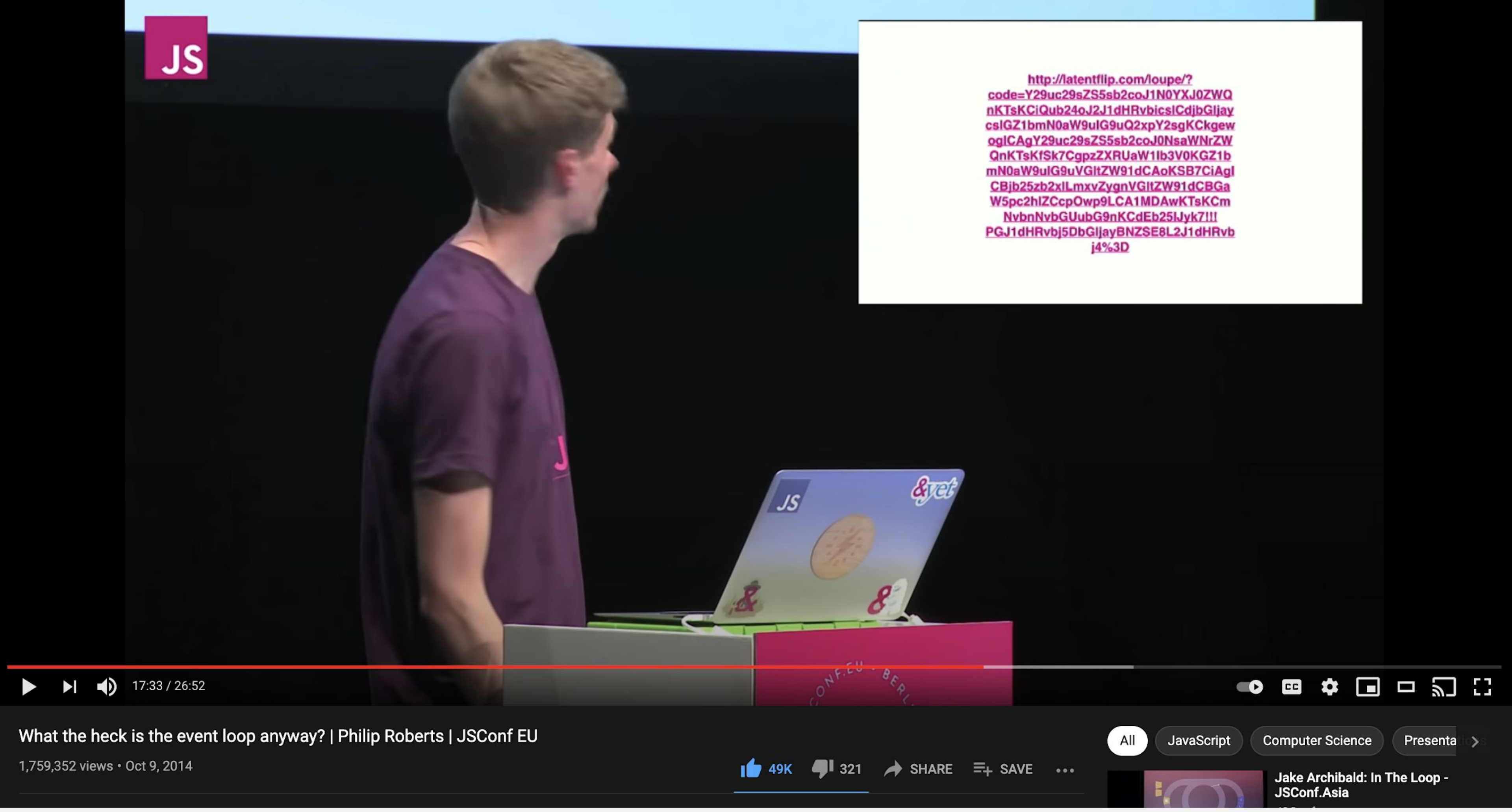
non-blocking

asynchronous

concurrent

language

latentflip.com/
loupe



http://latentflip.com/loupe/?
code=Y29uc29sZS5sb2coJ1N0YXJ0ZWQ
nKTsKCiQub24oJ2J1dHRvbicsICdjbGljay
csIGZ1bmN0aW9uIG9uQ2xpY2sgKCkgew
ogICAgY29uc29sZS5sb2coJ0NsaWNrZW
QnKTsKfSk7CgpzZXRUaW1lb3V0KGZ1b
mN0aW9uIG9uVGltZW91dCAoKSB7CiAgI
CBjb25zb2xlLmxvZygnVGltZW91dCBGa
W5pc2hlZCcpOwp9LCA1MDAwKTsKCm
NvbnNvbGUubG9nKCdEb25lJyk7!!!
PGJ1dHRvbj5DbGljayBNZSE8L2J1dHRvb
j4%3D

What the heck is the event loop anyway? | Philip Roberts | JSConf EU

1,759,352 views • Oct 9, 2014

Jake Archibald: In The Loop - JSConf.Asia

😺

🏍️

promises

Promesa

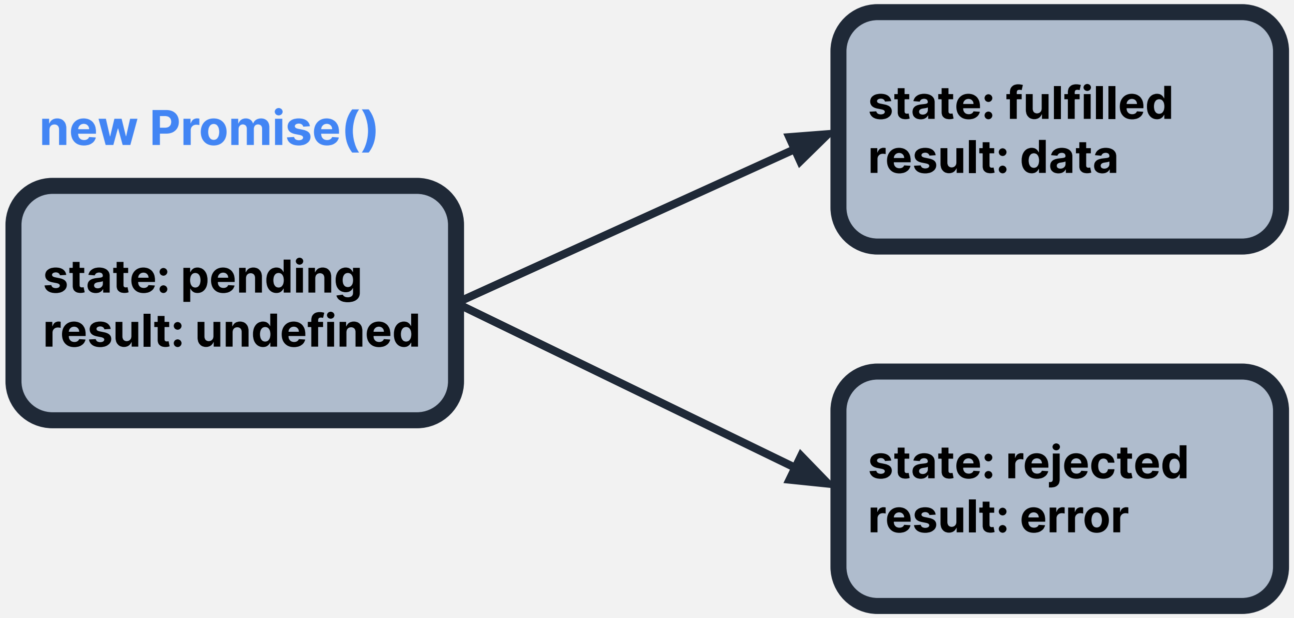
es un objeto que representa la terminación o el fracaso de una operación asíncrona

- 🕒**Pending:** initial state or execution in progress

- ✔️**Fulfilled:** execution is completed with success result

- ❌**Rejected:** execution is completed with failure result

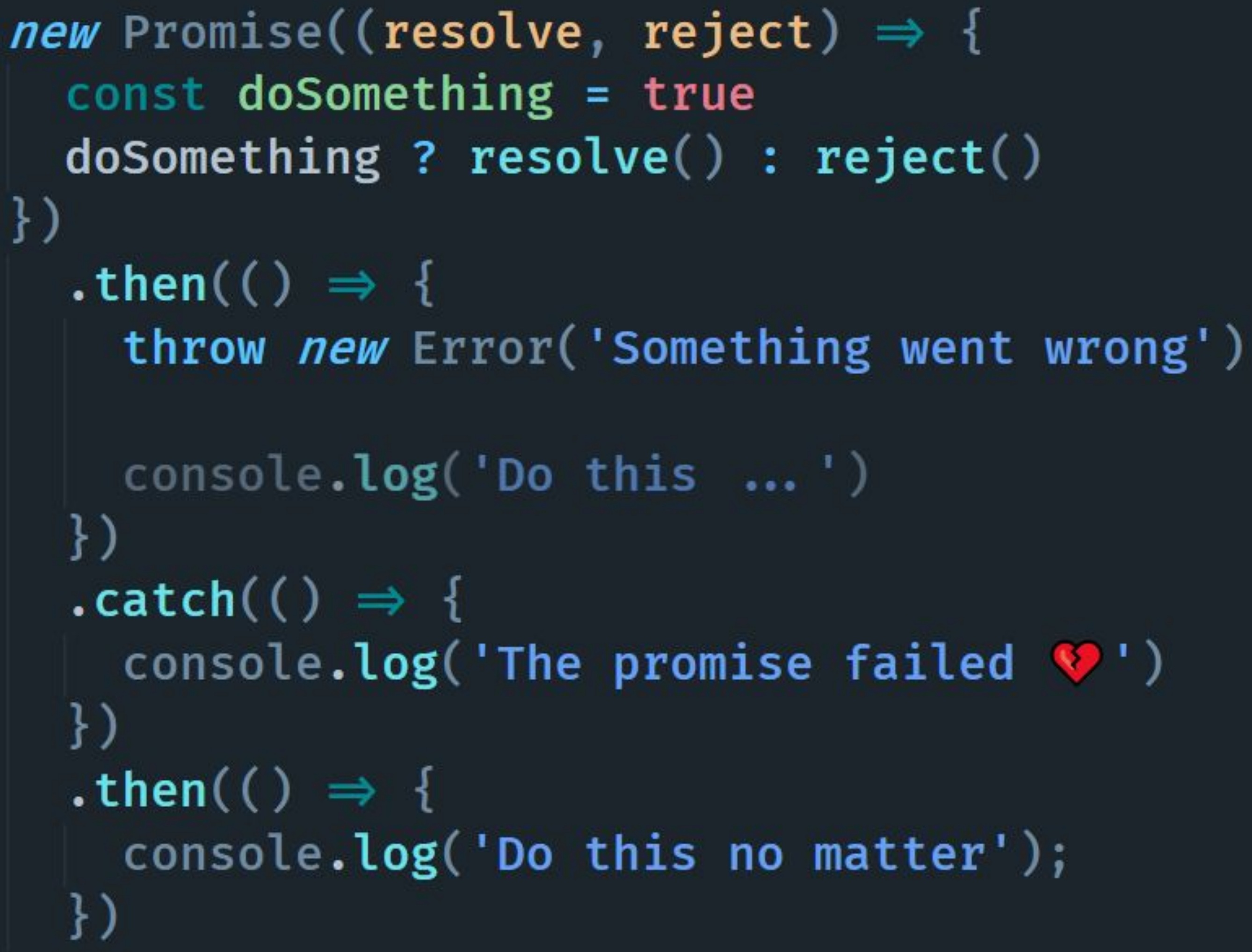
Promise flow



Preguntas?

🐯

.then



```javascript
new Promise((resolve, reject) => {
  const doSomething = true
  doSomething ? resolve() : reject()
})
  .then(() => {
    throw new Error('Something went wrong')

    console.log('Do this ... ')
  })
  .catch(() => {
    console.log('The promise failed 💔')
  })
  .then(() => {
    console.log('Do this no matter');
  })
```

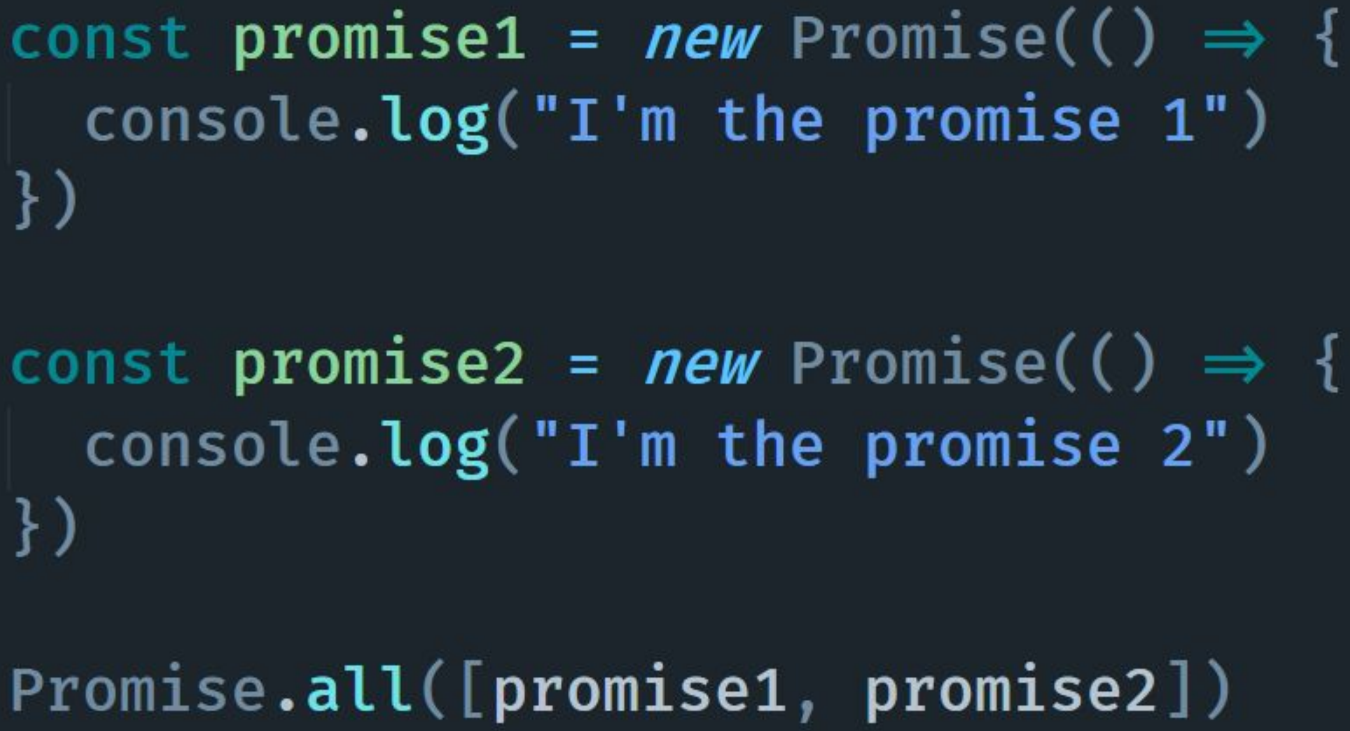
```
doSomething()
  .then(result => doSomethingElse(result))
  .then(otherResult => doTheThirdThing(otherResult))
  .then(finalResult => console.log(finalResult))
  .catch(errorHandler)
```

Preguntas?

🐔

Static methods

Promise.all()



```javascript
const promise1 = new Promise(() => {
  console.log("I'm the promise 1")
})

const promise2 = new Promise(() => {
  console.log("I'm the promise 2")
})

Promise.all([promise1, promise2])
```

Promise.allSettled() ES2020

```
const promise1 = new Promise(() => {
  console.log("I'm the promise 1")
})

const promise2 = new Promise(() => {
  console.log("I'm the promise 2")
})

Promise.allSettled([promise1, promise2])
```

Promise.race()

```javascript
const promise1 = new Promise(() => {
  console.log("I'm the promise 1")
})

const promise2 = new Promise(() => {
  console.log("I'm the promise 2")
})

Promise.race([promise1, promise2])
```

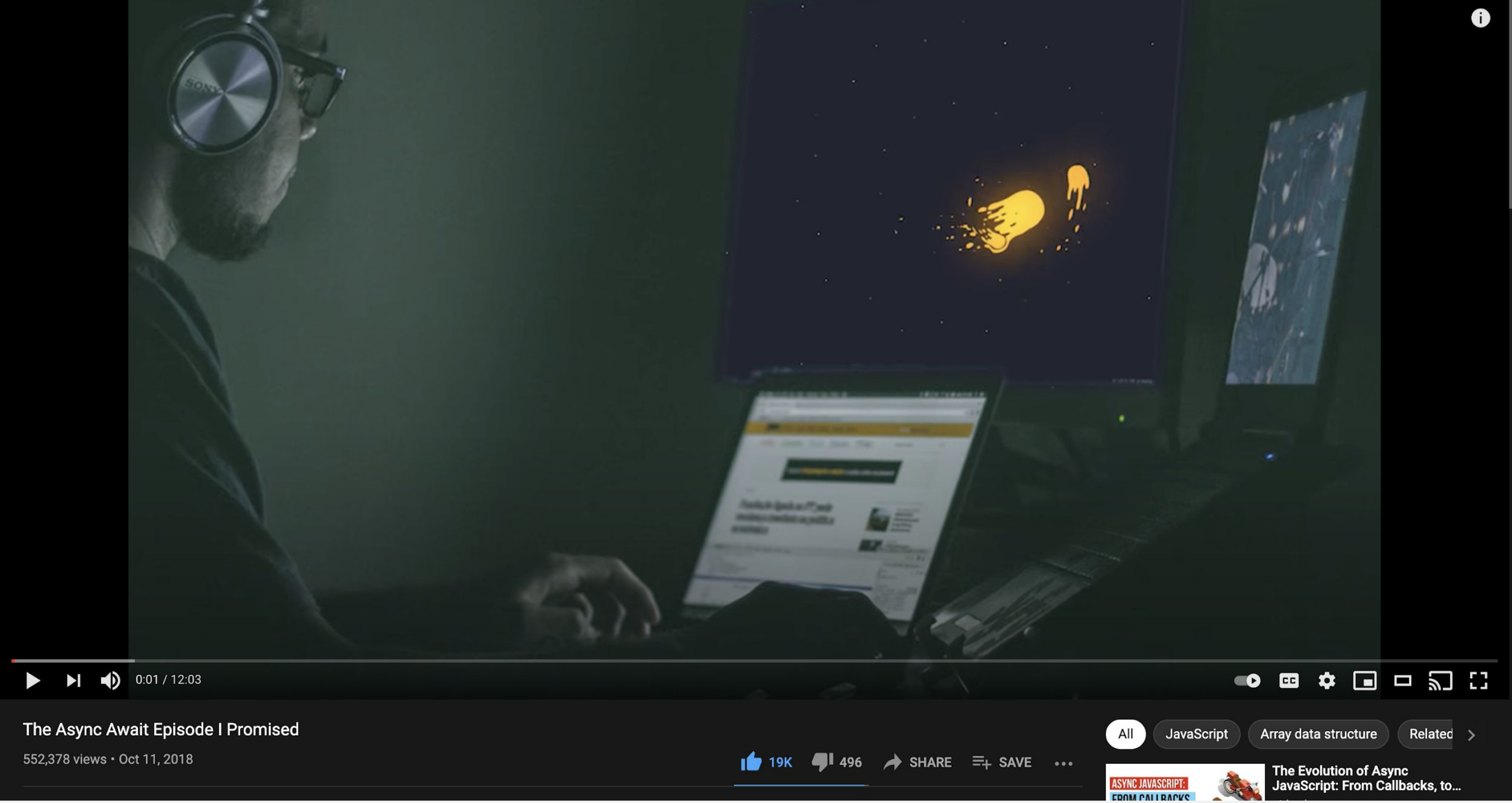
Preguntas?

🐼

async / await

```
const foo = async () => {
  const res = await largeProcess()
  const finish = await otherLargeProcess()
}
```

```javascript
const foo = async () => {
  try {
    const res = await largeProcess()
    const finish = await otherLargeProcess()
  } catch (error) {
    console.log("Some process failed! :c")
  }
}
```



The Async Await Episode I Promised

552,378 views • Oct 11, 2018

The Evolution of Async JavaScript: From Callbacks, to...

- **Promises:** shorturl.at/beoLS
- **Async/await:** shorturl.at/oKX48
- **Fetch:** shorturl.at/joplZ

Preguntas?