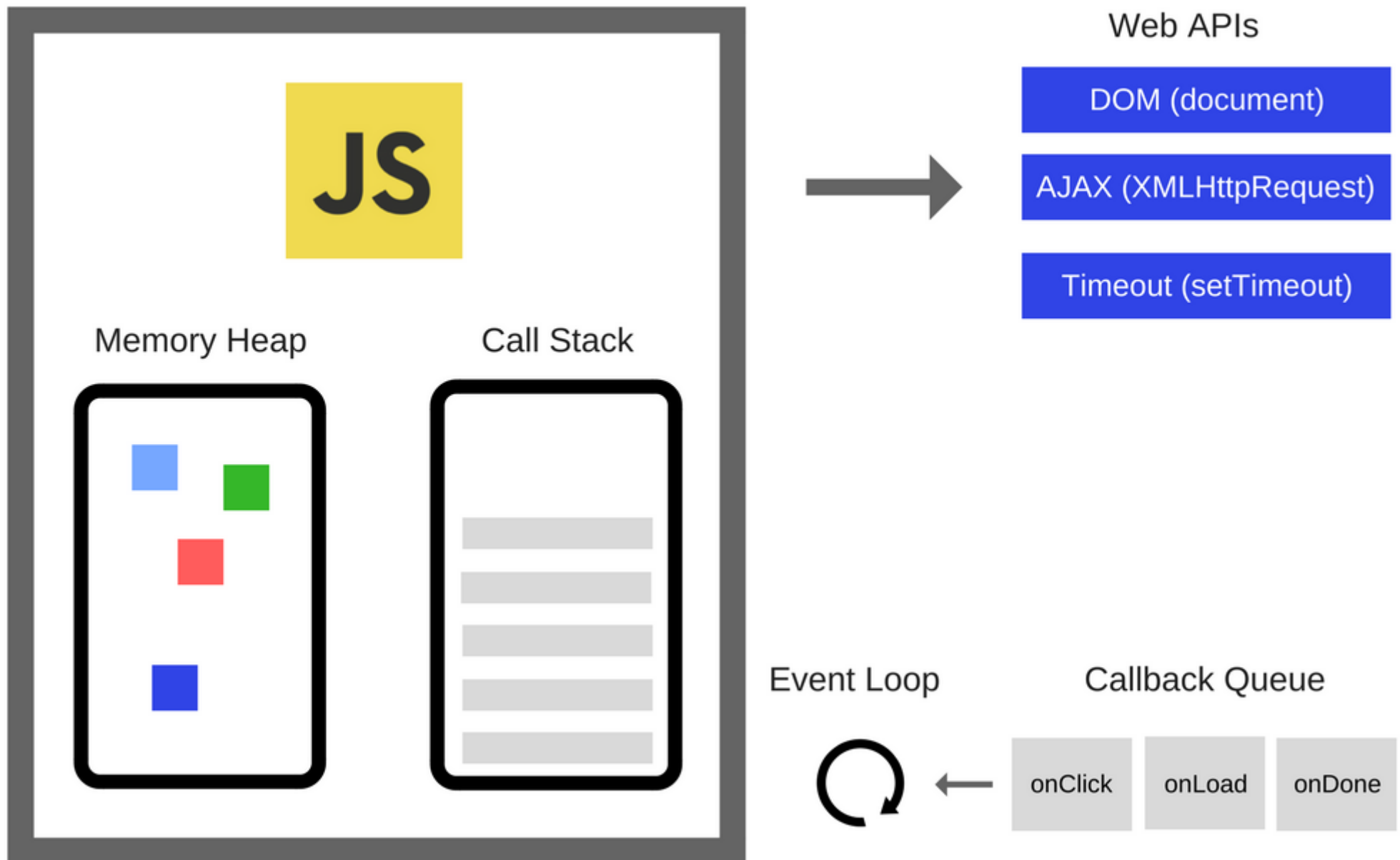


How JavaScript work in Browser

What the heck is the event loop
anyway - Philip Roberts

What you are JavaScript?

I have a call stack, an event loop,
a callback queue and other apis
and stuff



Call Stack

```
function multiply(a, b) {  
    return a * b;  
}  
  
function square(n) {  
    return multiply(n, n);  
}  
  
function printSquare(n) {  
    var squared = square(n);  
    console.log(squared);  
}  
  
printSquare(4);
```



stack

Call Stack

```
function multiply(a, b) {  
    return a * b;  
}
```

```
function square(n) {  
    return multiply(n, n);  
}
```

```
function printSquare(n) {  
    var squared = square(n);  
    console.log(squared);  
}
```

```
printSquare(4);
```

stack

multiply(n, n)

square(n)

printSquare(4)

main()

JS

Call Stack

```
function foo() {  
  throw new Error('Oops!');  
}
```

```
function bar() {  
  foo();  
}
```

```
function baz() {  
  bar();  
}
```

```
baz();
```

The screenshot shows a web browser's developer console with the 'Network' tab selected. A red 'x' icon indicates an error. The error message is 'Uncaught Error: Oops!'. The call stack is visible, showing the sequence of function calls that led to the error: 'foo' at 'oops.js:2', 'bar' at 'oops.js:7', 'baz' at 'oops.js:11', and '(anonymous function)' at 'oops.js:14'. The console also shows a search bar, a filter icon, and a close button.

Frame	File	Line
Uncaught Error: Oops!	oops.js	2
foo	oops.js	2
bar	oops.js	7
baz	oops.js	11
(anonymous function)	oops.js	14



Call Stack

```
function foo () {  
    return foo();  
}
```

```
foo( );
```

() 100

foo()

foo()

foo()

foo()

foo()

foo()

foo()

foo()

```
main( )
```

blocking

What happens when things are slow?

the solution?
asynchronous callbacks

Here's a function,
Call me maybe?

Concurrency & the Event Loop

One thing at a time,
except not really.

- Javascript run in single thread but browser have Web API that can do other these, uses threads (in node we have c++ api)

```
JS console.log('Hi');
```

```
setTimeout(function cb() {  
  console.log('there');  
}, 5000);
```

```
console.log('JSConfEU');
```

Console

Hi

stack

setTimeout cb

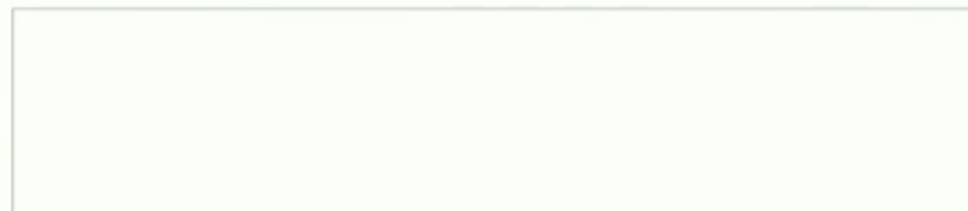
main()

webapis

event loop



task
queue



JS

```
console.log('Hi');
```


```
setTimeout(function cb() {  
  console.log('there');  
}, 5000);
```

```
console.log('JSConfEU');
```

Console

Hi

JSConfEU

stack**webapis**timer()

cb

event loop

task
queue

When webapis done there job they push call back function into the task queue, **event loop** have simple job once stack is empty, push that call back on the top of stack

```
JS console.log('Hi');
```

```
setTimeout(function cb() {  
  console.log('there');  
}, 5000);
```

```
console.log('JSConfEU');
```

Console

Hi

JSConfEU

stack

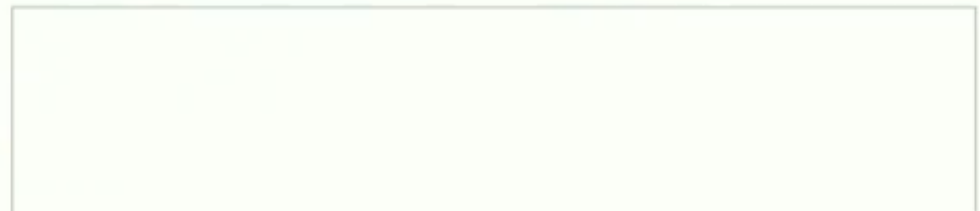
cb

webapis

event loop



task
queue



```
JS console.log('Hi');
```

```
$.get('url', function cb(data) {  
    console.log(data);  
});
```

```
console.log('JSConfEU');
```

Console

Hi

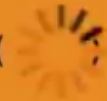
stack

\$.get('u', cb)

main()

webapis

XHR(



cb

event loop



task
queue

