JavaScript Class 08

Cefalo School
Instructor: Beroza Paul

Asynchronous JavaScript

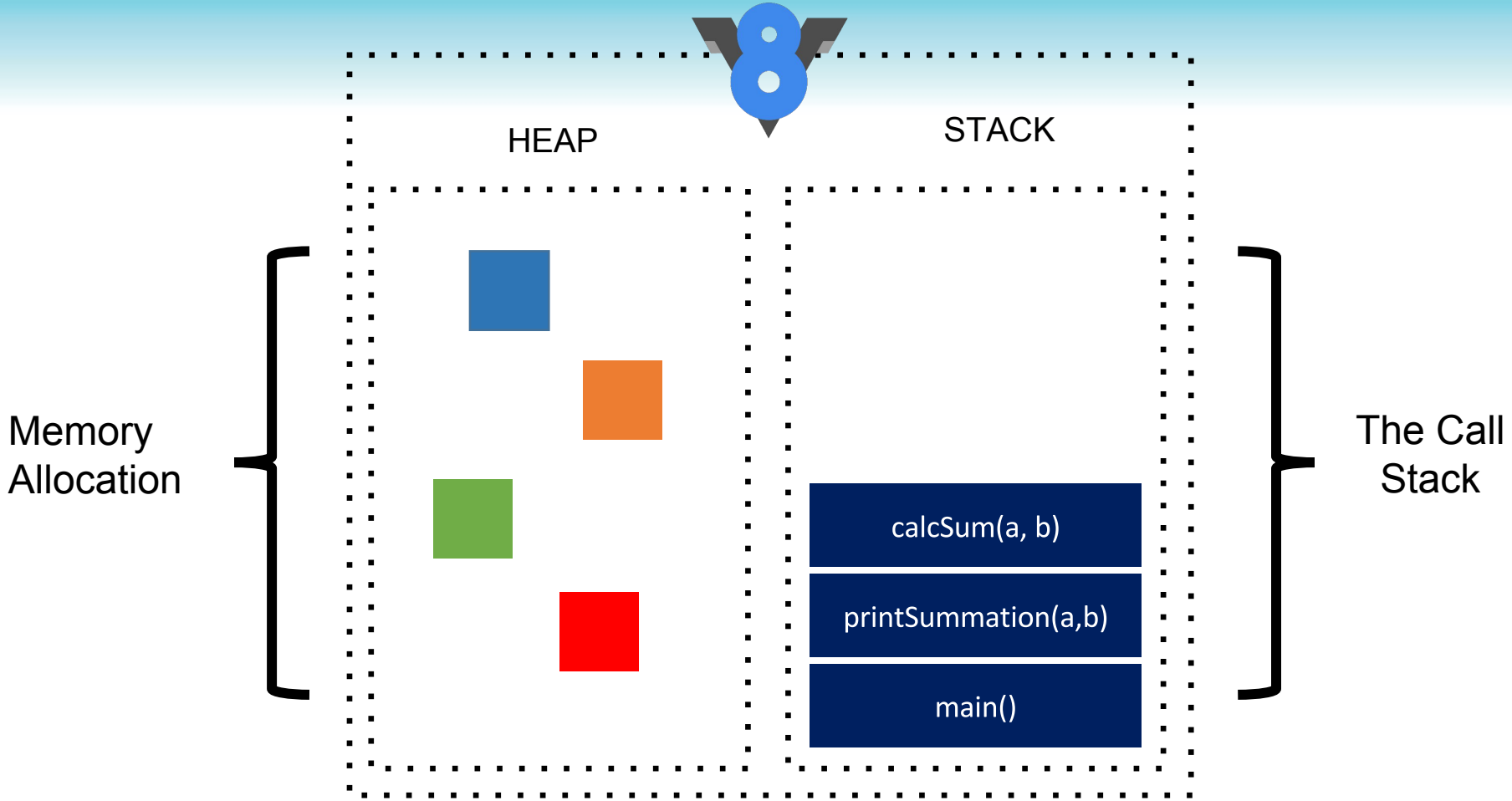
How? Let's visualize!

JavaScript Engines

- **V8** - developed by Google for Google Chrome
- **JavaScriptCore** - developed by Apple for Safari
- **SpiderMonkey** - by Netscape Navigator for Firefox
- etc

Chrome V8

I have a CALL STACK and a HEAP and that's all!
But how do you (v8) look?


HEAP
STACK
Memory Allocation
The Call Stack
calcSum(a, b)
printSummation(a,b)
main()

The call stack

- Added to a stack when it needs to be executed
- Cleared from the stack when the it is done

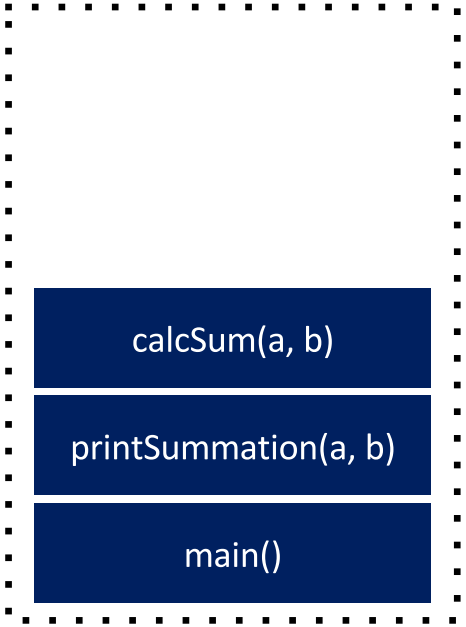
One thread

||

One call stack

||

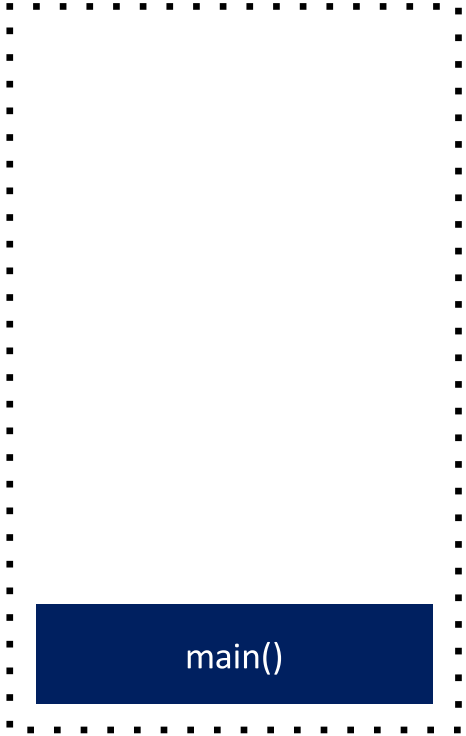
One thing at a time



```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```

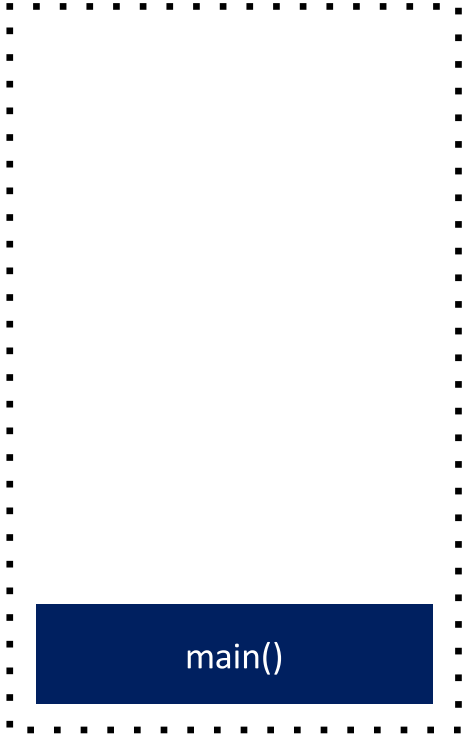
THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```



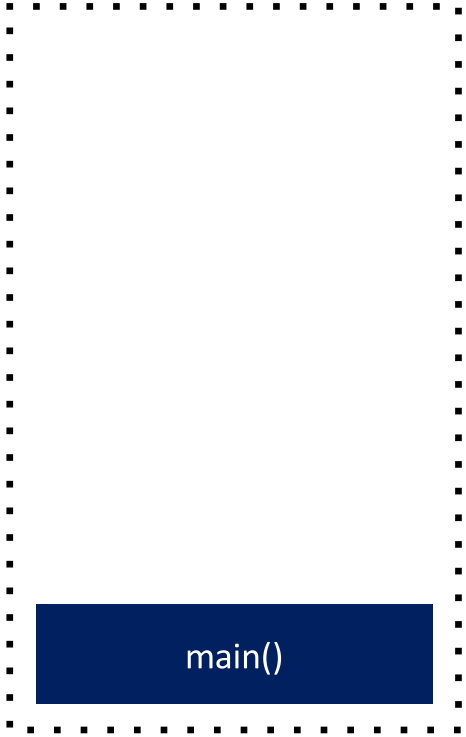
THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```



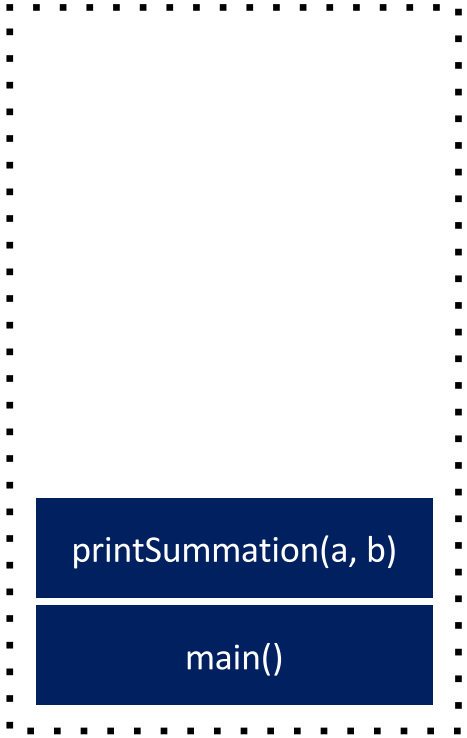
THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```



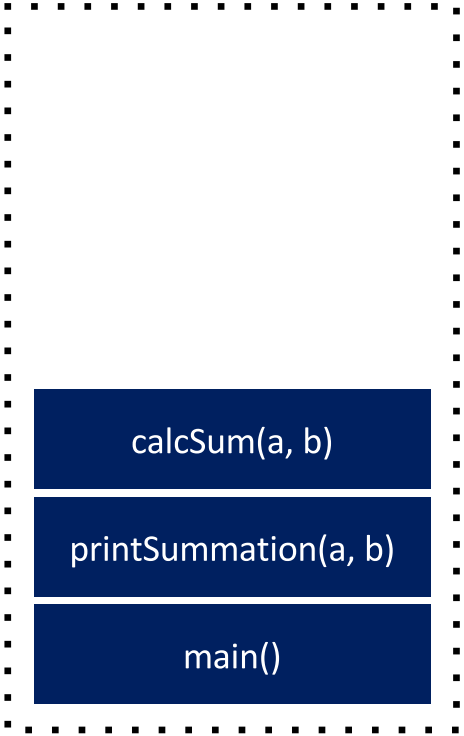
THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```



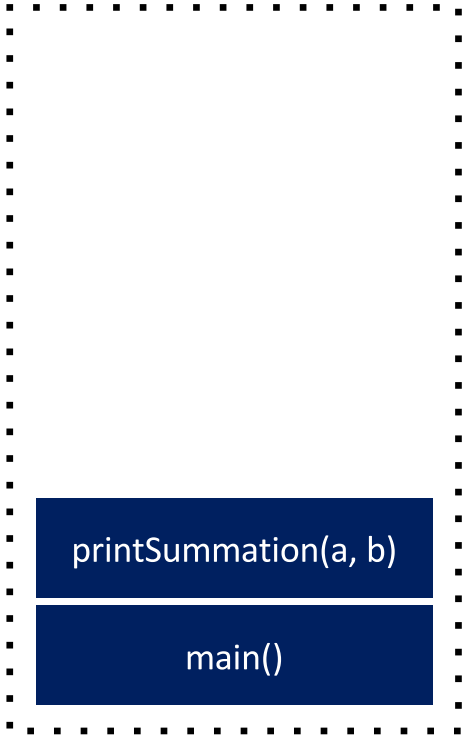
THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```



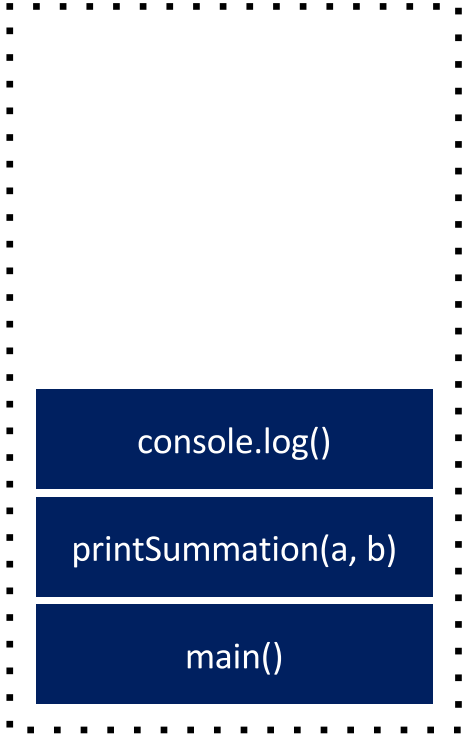
THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```



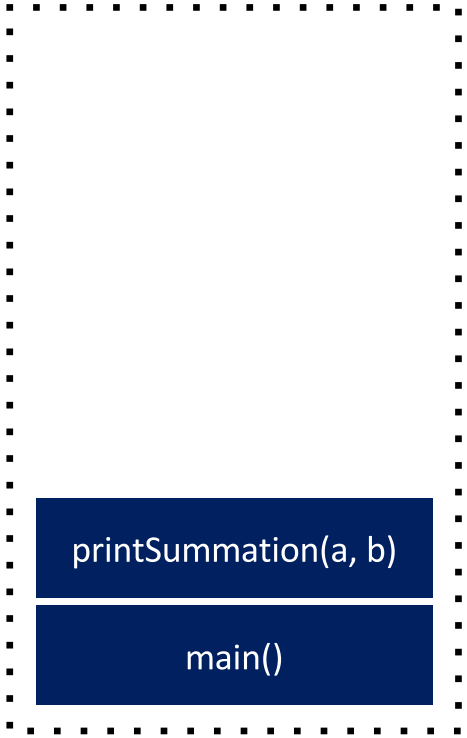
THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```



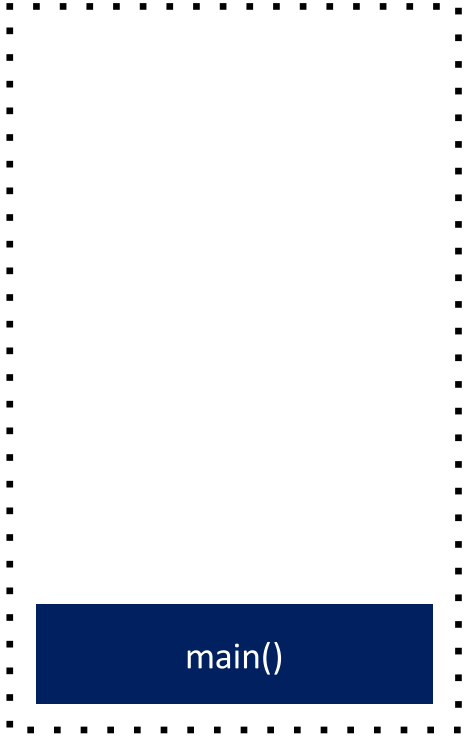
THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```



THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```



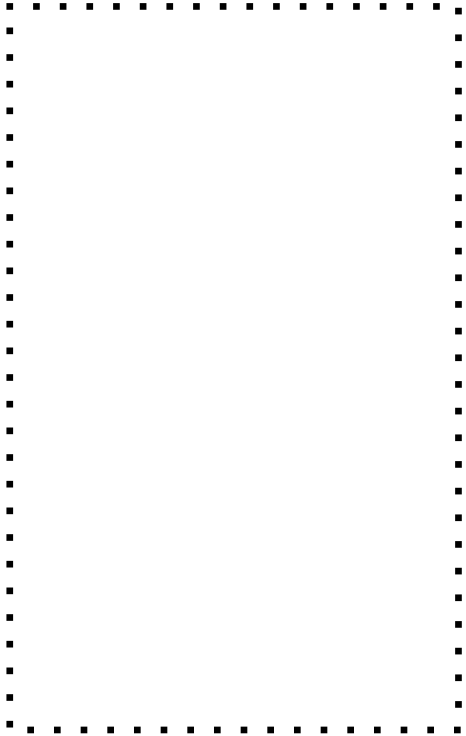
THE CALL STACK

```
function calcSum(a, b){
    return a + b;
}
function printSummation(a, b){
    var result = calcSum(a, b);
    console.log(result);
}
printSummation(10, 5);
```

THE CALL STACK

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

getCourseName();
```



THE CALL STACK

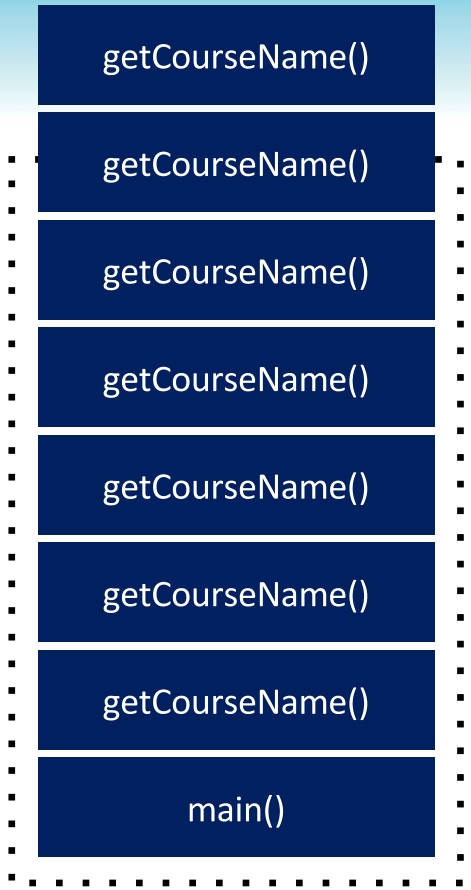
```
function getCourseName(){
  return getCourseName();
}

getCourseName();
```

Elements Console Sources Network Timeline Profiles Application Security Audits

top ▼ Preserve log

```
▶Uncaught RangeError: Maximum call stack size exceeded
    at getCourseName ((index):45)
    at getCourseName ((index):46)
    at getCourseName ((index):46)
    at getCourseName ((index):46)
    at getCourseName ((index):46)
    at getCourseName ((index):46)
    at getCourseName ((index):46)
    at getCourseName ((index):46)
    at getCourseName ((index):46)
    at getCourseName ((index):46)
```

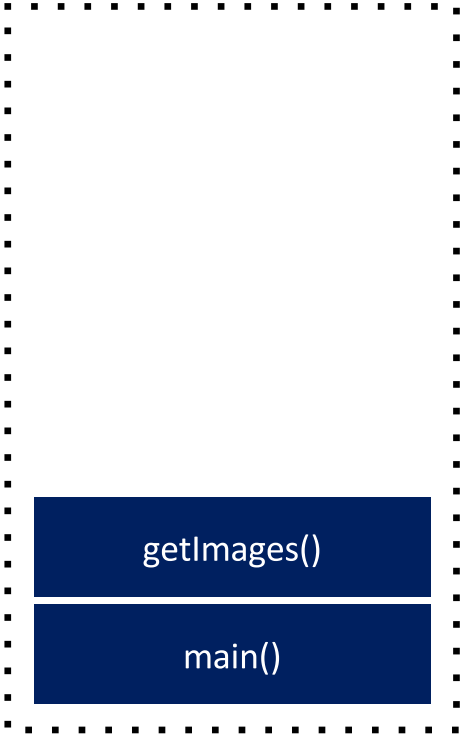


THE CALL STACK

What is blocking?

```
var a = $.getImages();
var b = $.sendEmails();


console.log(a);
console.log(b);
```



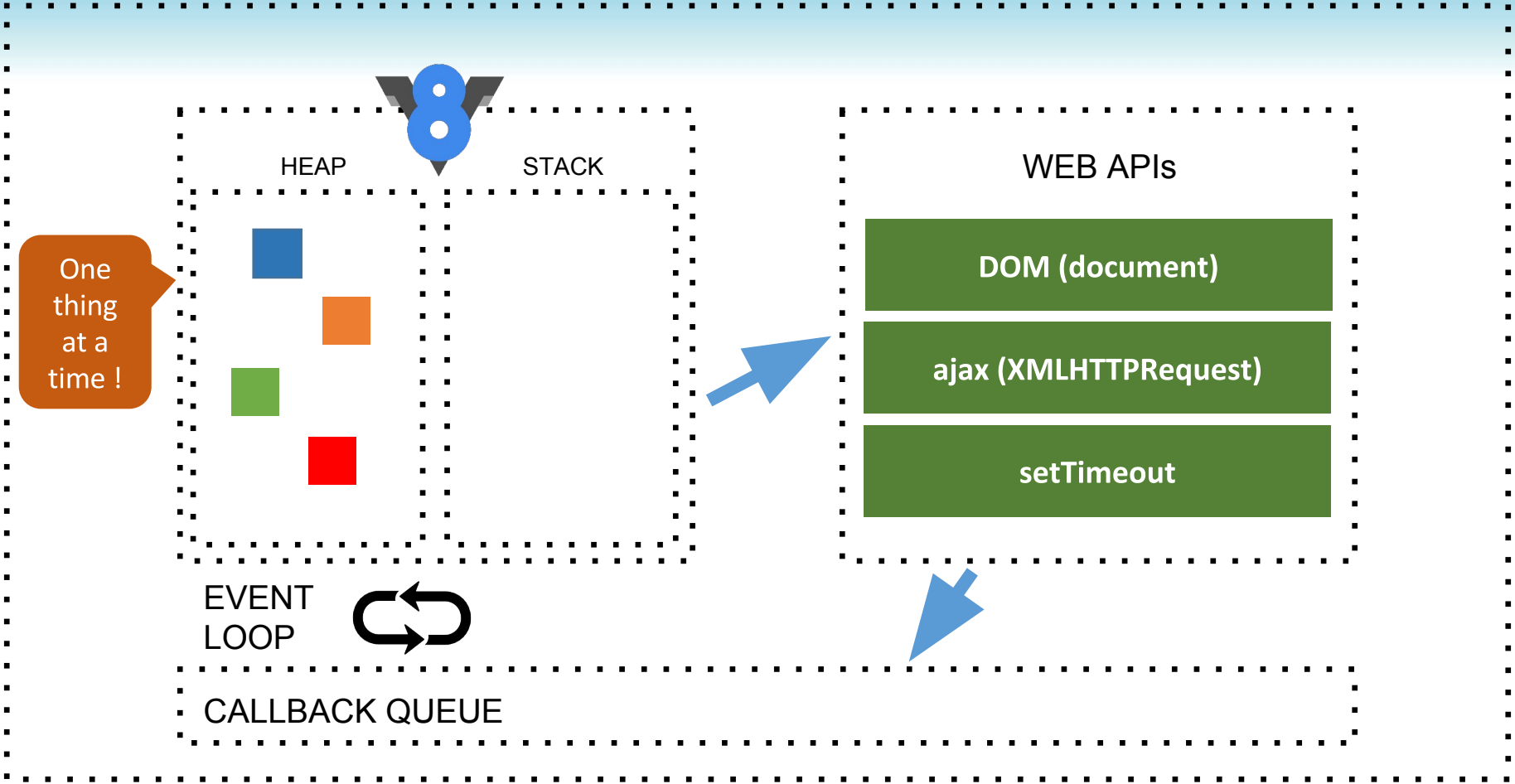
THE CALL STACK

How do we solve this?

Asynchronous callbacks.

What is event loop?

- It watches the Call Stack and the Callback Queue
- If the Stack is empty, it takes the first element in the Callback Queue, and pushes it into the Stack


HEAP
STACK
One thing at a time !
WEB APIs
DOM (document)
ajax (XMLHTTPRequest)
setTimeout
EVENT LOOP
CALLBACK QUEUE

CODE

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

setTimeout(function (){
    console.log("there");
}, 5000);

console.log("Howdy?");
```

CONSOLE

STACK

WEB APIs

EVENT LOOP

CALLBACK QUEUE

CODE

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

setTimeout(function (){
    console.log("there");
}, 5000);

console.log("Howdy?");
```

CONSOLE

```
Hi
```

STACK

console.log("Hi")

main()

WEB APIs

EVENT LOOP

CALLBACK QUEUE

CODE

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

setTimeout(function (){
    console.log("there");
}, 5000);

console.log("Howdy?");
```

CONSOLE

```
Hi
```

STACK

setTimeout(callback)

main()

WEB APIs

timer

EVENT LOOP

CALLBACK QUEUE

CODE

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

setTimeout(function (){
    console.log("there");
}, 5000);

console.log("Howdy?");
```

CONSOLE

```
Hi
Howdy?
```

STACK

console.log("Howdy?")

main()

WEB APIs

setTimeout(callback)

timer

EVENT LOOP

CALLBACK QUEUE

CODE

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

setTimeout(function (){
    console.log("there");
}, 5000);

console.log("Howdy?");
```

CONSOLE

```
Hi
Howdy?
```

STACK

main()

WEB APIs

EVENT LOOP

CALLBACK QUEUE

callback

CODE

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

setTimeout(function (){
    console.log("there");
}, 5000);

console.log("Howdy?");
```

CONSOLE

```
Hi
Howdy?
```

STACK

WEB APIs

EVENT LOOP

CALLBACK QUEUE

callback

CODE

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

setTimeout(function (){
    console.log("there");
}, 5000);

console.log("Howdy?");
```

CONSOLE

```
Hi
Howdy?
there
```

STACK

console.log("there")

callback

WEB APIs

EVENT LOOP

CALLBACK QUEUE

CODE

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

setTimeout(function (){
    console.log("there");
}, 5000);

console.log("Howdy?");
```

CONSOLE

```
Hi
Howdy?
there
```

STACK

WEB APIs

EVENT LOOP

CALLBACK QUEUE

Remember this?

```
for(var i = 0; i < 3 ;i++) {
    setTimeout(function(){
      console.log(i);
    })
}
```

The solution!

```
for(var i = 0; i < 3 ;i++) {
    setTimeout(function(i){
      console.log(i)
    }(i))
}
```

HTML5 JavaScript APIs

- Web Sockets and Messaging, WebRTC
- Canvas, SVG, WebGL
- File API, File System API, Indexed DB, Offline, **Web Storage**
- Browser, Shadow DOM, Typed Arrays, Web Workers
- Animation Timing, Media, Pointer Lock, Web Audio
- etc

What is Web Storage?

Web storage allows users to securely store data in the form of key/value pairs in much easier way than cookies.

Two mechanisms of web storage

1. sessionStorage
2. localStorage

sessionStorage

sessionStorage maintains a separate storage for each given origin that gets cleared when the page session ends.

sessionStorage Facts

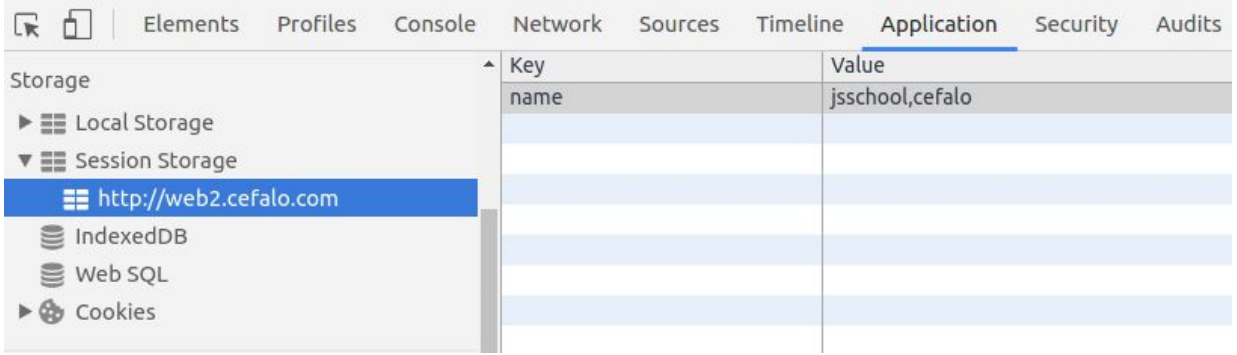
- It is tied to the protocol, hostname, and port of the page
- It is unique to a particular window or tab
- Value automatically converts into string before being stored

sessionStorage code snippet

```
sessionStorage.setItem("name", new Array('jsschool', 'cefalo'));
console.log(sessionStorage.getItem('name'));
console.log(typeof sessionStorage.name);
```

Console

```
Jsschool,cefalo
string
```



localStorage

Similar to sessionStorage except data will be available even after the window/tab is closed

localStorage Facts

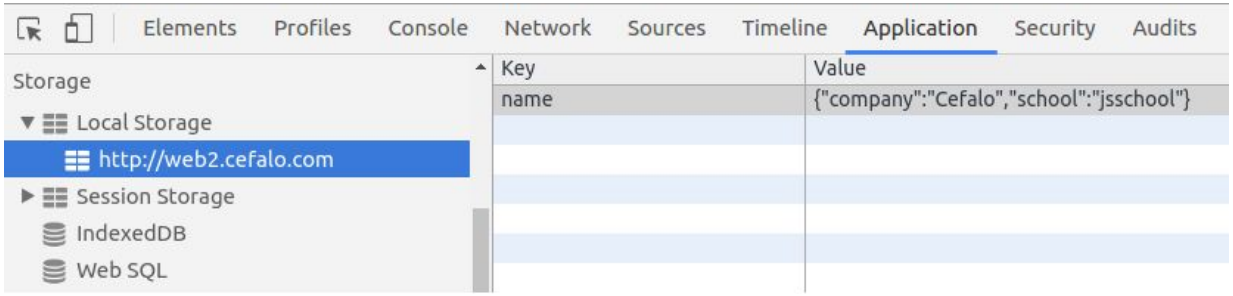
- No expiration date for localStorage
- Data stored in http://cefalo.com is not accessable from https://cefalo.com
- Data stored in incognito mode is not accessible from normal mode
- No way to expand localStorage default (5MB) quotas

localStorage code snippet

```
localStorage.setItem("name", JSON.stringify({'company' : 'Cefalo',
'school' : 'jsschool'}));
console.log(localStorage.getItem('name'));
console.log(typeof localStorage.name);
```

Console

```
{"company":"Cefalo"
,"school":"jsschool
"}
string
```



Cookie

A cookie is a small text file that's stored in browser.

- A name-value pair containing the actual data
- An expiry date after which it is no longer valid
- The domain and path of the server it should be sent to

```
document.cookie =
  'ppkcookie1=testcookie; expires=Thu, 2 Aug 2017 20:47:11 UTC; path=/'
```

localStorage vs cookie

localStorage	Cookie
Data is not sent to the server with every http request	Data is sent to the server with every http request
It stores data with no expiration date	Cookie has cookie expiry
It has 5MB space per domain (depends on browsers and versions)	It has limited 4KB space
It can only be read in client-side	Cookies are primarily for reading server-side

JSONP

- JSON with padding
- Work around the same-origin policy via script element injection
- Limited to **GET** method
- Security issues and recommended is **CORS**

CORS

- Content ownership protection
- Mechanism gives web servers cross-domain access controls
- Compatible with modern browsers

Thank you!

Reference

1. https://developer.mozilla.org/en/docs/Web/JavaScript/EventLoop
2. http://www.quirksmode.org/js/cookies.html
3. https://en.wikipedia.org/wiki/Web_storage
4. http://html5index.org/
5. https://en.wikipedia.org/wiki/JSONP
6. https://www.html5rocks.com/en/tutorials/cors/
7. https://www.youtube.com/watch?v=8aGhZQkoFbQ
8. https://www.html5rocks.com/en/tutorials/offline/quota-research/