JS Concepts

Event Loop, Call Stack, Callback Queue, this and that…….

🤔 AM I

🤔 AM I

I am
SHIHABUDHEEN US

🤔 AM I

I am
SHIHABUDHEEN US aka SHIHAB 👨‍✈️

🤔 AM I

I am
SHIHABUDHEEN US aka SHIHAB 👨‍✈️

👨‍💻

BOUNCE

QUEST
BORN TO ENGINEER

🤔 AM I

I am
SHIHABUDHEEN US aka SHIHAB 👨‍✈️

👨‍💻

BOUNCE

QUEST
BORN TO ENGINEER

shihabus

@type__error

shihabus

The 💡

JS is :

- Single threaded (one command at a time)
- Synchronously executed (each line is run order the code appears)

🌟 Hello World

```
const hello='Hello'

function printHello(){
  console.log(hello)
}

function printStr(inputStr){
  console.log(inputStr)
}

printHello()
printStr('World !')
```

What is output ❓

🌟 Hello World

```
const hello='Hello'

function printHello(){
  console.log(hello)
}

function printStr(inputStr){
  console.log(inputStr)
}

printHello()
printStr('World !')
```

Verify

```
// Hello
// World !
```

🌟 Hello World

```
const hello='Hello'

function printHello(){
  console.log(hello)
}

function printStr(inputStr){
  console.log(inputStr)
}

printHello()
printStr('World !')
```

Verify ✅

```
// Hello
// World !
```

🌈 Hello ⏰ World !

```
const hello='Hello'

function printHello(){
  console.log(hello)
}

function printStr(inputStr){
  console.log(inputStr)
}

printHello()

setTimeout(()=>printStr('World'),1000)

printStr('!')
```

`setTimeout(callback,[delay])`

Execute a specified block of code once after a specified time has elapsed.

🌈 Hello ⏰ World !

```
const hello='Hello'

function printHello(){
  console.log(hello)
}

function printStr(inputStr){
  console.log(inputStr)
}

printHello()

setTimeout(()=>printStr('World'),1000)

printStr('!')
```

Verify

```
// Hello
// .... some delay
// World
// !
```

🌈 Hello ⏰ World !

```
const hello='Hello'

function printHello(){
  console.log(hello)
}

function printStr(inputStr){
  console.log(inputStr)
}

printHello()

setTimeout(()=>printStr('World'),1000)

printStr('!')
```

Failed ❌

```
// Hello
// !
// .... some delay
// World
```

You promised me to be synchronous 😢

How 🤷

`setTimeout` **works**

Where does JS 🏃

Where does JS 🏃‍♀️

Most of the JS run inside your browser’s

JavaScript engine

A few weeks ago, I tweeted this interview question:

****** Answer the question in your head now before you proceed ******

About half the replies to the Tweet were wrong. The answer is **NOT** V8 (or other VMs)!! While famously known as "JavaScript Timers", functions like `setTimeout` and `setInterval` are not part of the ECMAScript specs or any JavaScript engine implementations. Timer functions are implemented by browsers and their implementations will be different among different browsers. Timers are also implemented natively by the Node.js runtime itself.

How is setTimeout implemented in node.js

Asked 7 years, 7 months ago Active 3 years, 2 months ago Viewed 7k times

26

15

I was wondering if anybody knows how setTimeout is implemented in node.js. I believe I have read somewhere that this is not part of V8. I quickly tried to find the implementation, but could not find it in the source(BIG).I for example found this timers.js file, which then for example links to timer_wrap.cc. But these file do not completely answer all of my questions.

- Does V8 have `setTimeout` implementation? I guess also from the source the answer is no.
- How is `setTimeout` implemented? javascript or native or combination of both? From timers.js I assume something along the line of both:

```
var Timer = process.binding('timer_wrap').Timer;`
```

It is part of the `window` object in the browser, not defined in ECMAScript. Therefore, other environments such as Node are not guaranteed to have it.

share edit follow

answered Apr 20 '16 at 20:56

Jacob See

730 • 6 • 17

add a comment

10

The `setTimeout()` function is actually exposed by the browser's `window` object as as such they aren't necessarily defined in the ECMAScript specification because they're not JavaScript features, they are features of the browser itself.

You can see from the specification section in the previously linked documentation that it uses the **WHATWG HTML Living Standard** :

Specifications

2 Answers

Active Oldest Votes

22

You've done most of the work already. V8 doesn't provides an implementation for `setTimeout` because it's not part of ECMAScript. The function you use is implemented in timers.js, which creates an instance of a `Timeout` object which is a wrapper around a C class.

There is a comment in the source describing how they are managing the timers.

```
// Because often many sockets will have the same idle timeout we will not
// use one timeout watcher per item. It is too much overhead.  Instead
// we'll use a single watcher for all sockets with the same timeout value
// and a linked list. This technique is described in the libev manual:
// http://pod.tst.eu/http://cvs.schmorp.de/libev/ev.pod#Be_smart_about_timeouts
```

Is setTimeout a part of JavaScript it self or it is just an api that the browser provides?

Asked 4 years, 2 months ago Active 4 years, 2 months ago Viewed 2k times

Is setTimeout a part of JavaScript it self or it is just an api that the browser provides ?

The Overflow Blog

Parts of JS Engine

- Thread of Execution
- Call Stack
- Memory/variable environment

So where is 🤔

- Timer
- console
- Debugging tools
- Network request handler
- Sockets
- Storages: localStorage, sessionStorage, indexDB

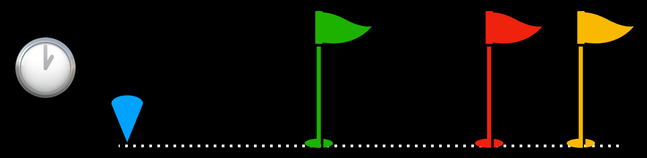
JSE is not alone

Javascript Runtime Engine ⚙️

The Great Partition

console

Hello



Callbacks		
Durations		

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Global Memory

hello:'Hello'

printHello: f

printStr: f

Call Stack

global()

The Great Partition

console

Hello

0 ms 1000 ms

Callbacks	printStr('World')
Durations	1000ms

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Global Memory

5 ms

hello:'Hello'

printHello: f

printStr: f

Call Stack

global()

The Great Partition

console

```
Hello
!
```

0 ms 1000 ms

Callbacks	printStr('World')
Durations	1000ms

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Global Memory

5 ms

6 ms

hello:'Hello'

printHello: f

printStr: f

Call Stack

printStr()

global()

The Great Partition

console

```
Hello
!
```

0 ms 1000 ms

Callbacks	printStr('World')
Durations	1000ms

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Global Memory

hello:'Hello'

printHello: f

printStr: f

5 ms

6 ms

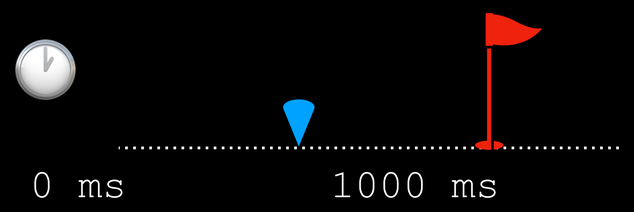
Call Stack

global()

The Great Partition

console

```
Hello
!
```



Callbacks	printStr('World')
Durations	1000ms

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Global Memory

hello:'Hello'

printHello: f

printStr: f

5 ms

6 ms

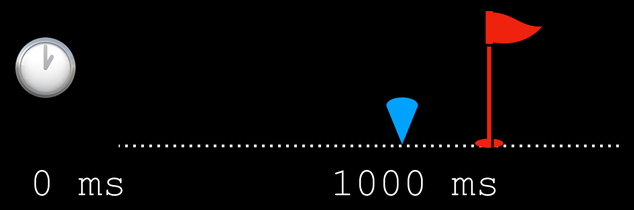
Call Stack

global()

The Great Partition

console

```
Hello
!
```



Callbacks	printStr('World')
Durations	1000ms

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Global Memory

hello:'Hello'

printHello: f

printStr: f

5 ms

6 ms

Call Stack

global()

The Great Partition

console

Hello

!

0 ms 1000 ms

Callbacks	
Durations	

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Global Memory

5 ms

hello:'Hello'

6 ms

printHello: f

printStr: f

Call Stack 1005 ms

global()

The Great Partition

console

Hello

!

0 ms 1000 ms

Callbacks	
Durations	

	printStr('World')

Callback Queue

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Event Loop

Global Memory

5 ms

hello:'Hello'

6 ms

printHello: f

printStr: f

Call Stack

1005 ms

global()

The Great Partition

console

Hello
!
World

0 ms 1000 ms

Callbacks	
Durations	

Callback Queue

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Event Loop

Global Memory

5 ms

hello:'Hello'

6 ms

printHello: f

printStr: f

Call Stack 1005 ms

> 1005 ms

printStr()

global()

The Great Partition

console

Hello
!
World

0 ms 1000 ms

Callbacks	
Durations	

Callback Queue

Browser

JS

```
...

function printStr(inputStr){
  console.log(inputStr)
}

....

setTimeout(()=>{
  printStr('World')
},1000)

printStr('!')
```

Event Loop

Global Memory

5 ms

hello:'Hello'

6 ms

printHello: f

printStr: f

Call Stack 1005 ms

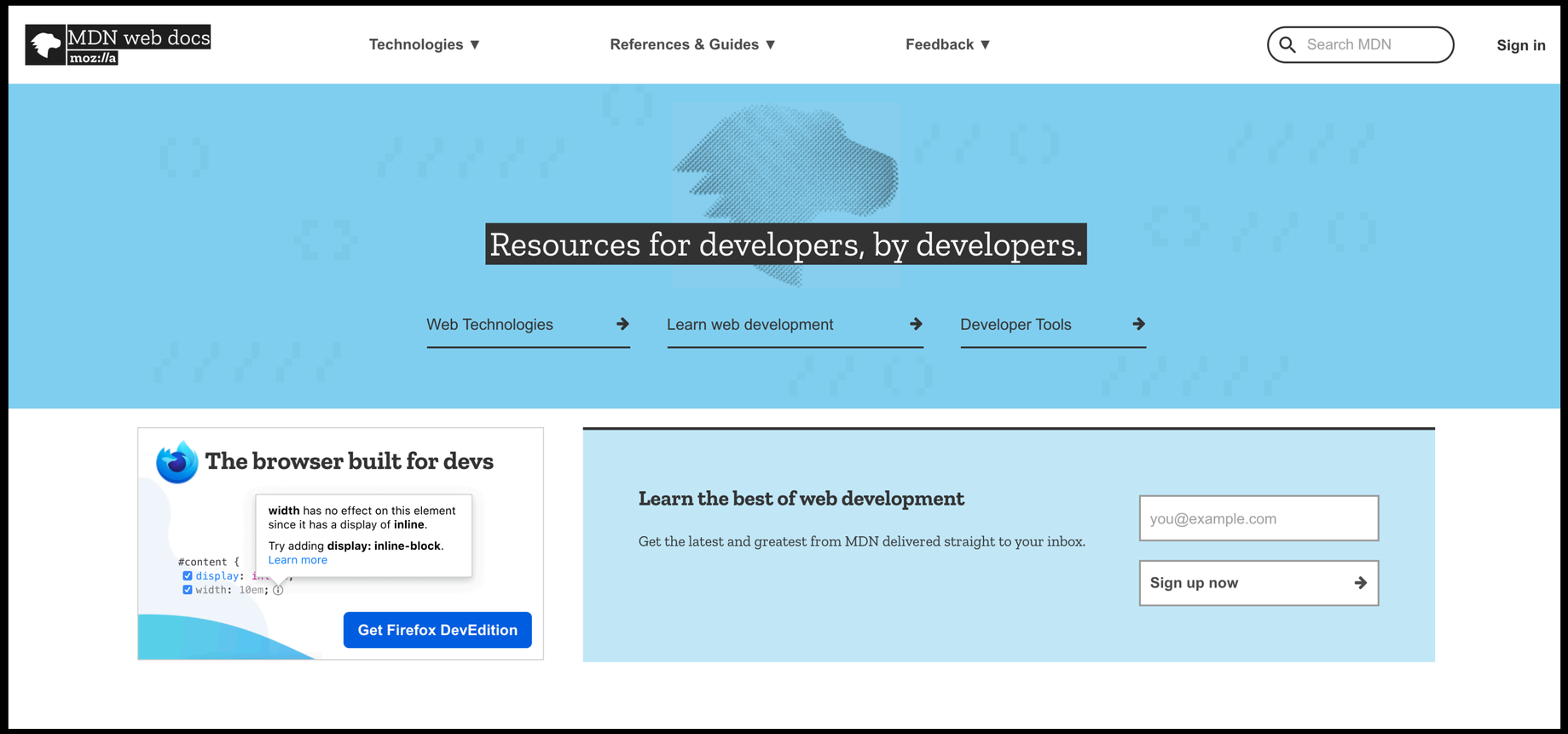
> 1005 ms

global()

Glossary

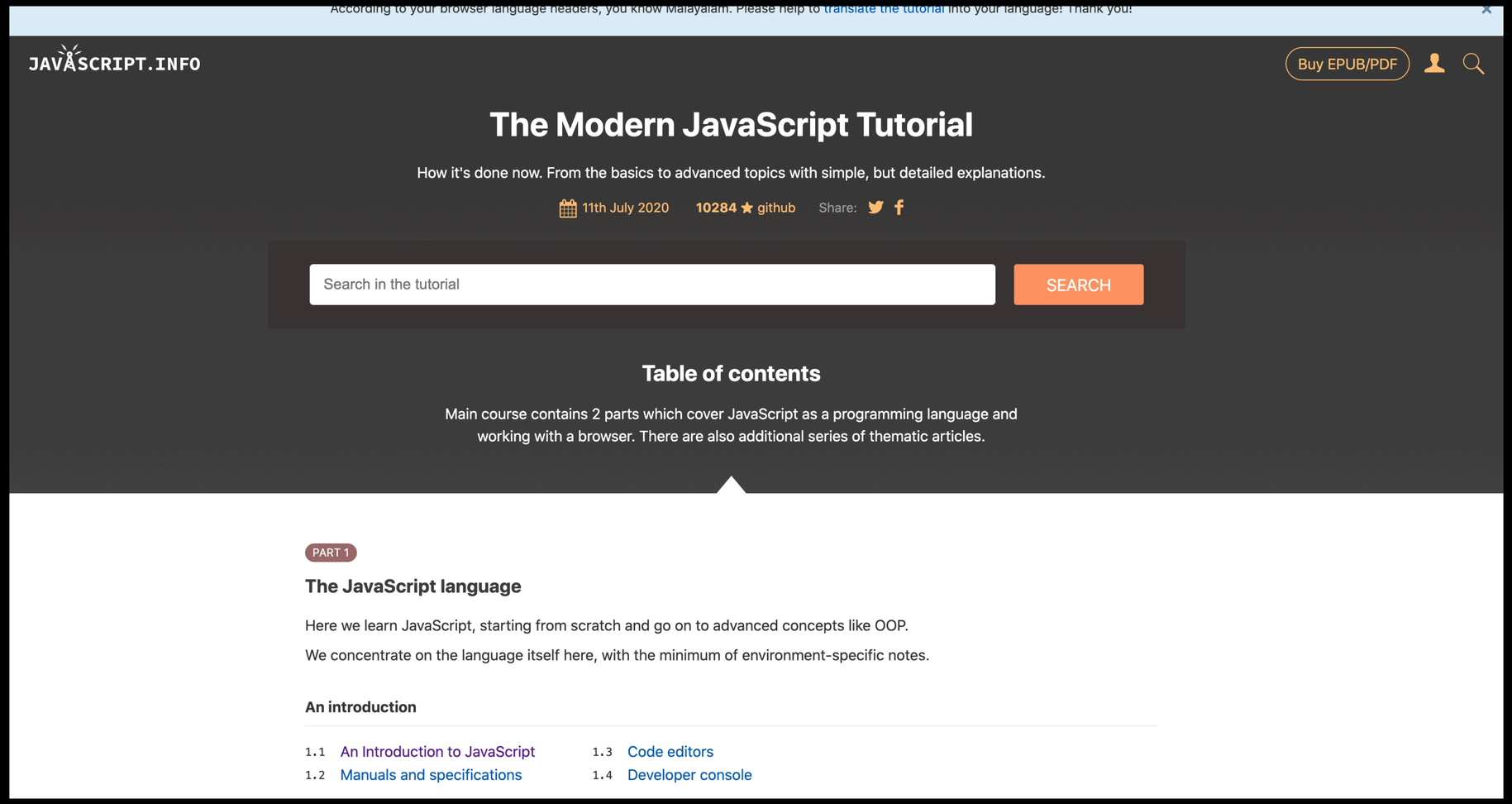
- Event Loop ✅
- Call Stack ✅
- Callback Queue ✅
- this and that….. ✅

Reference



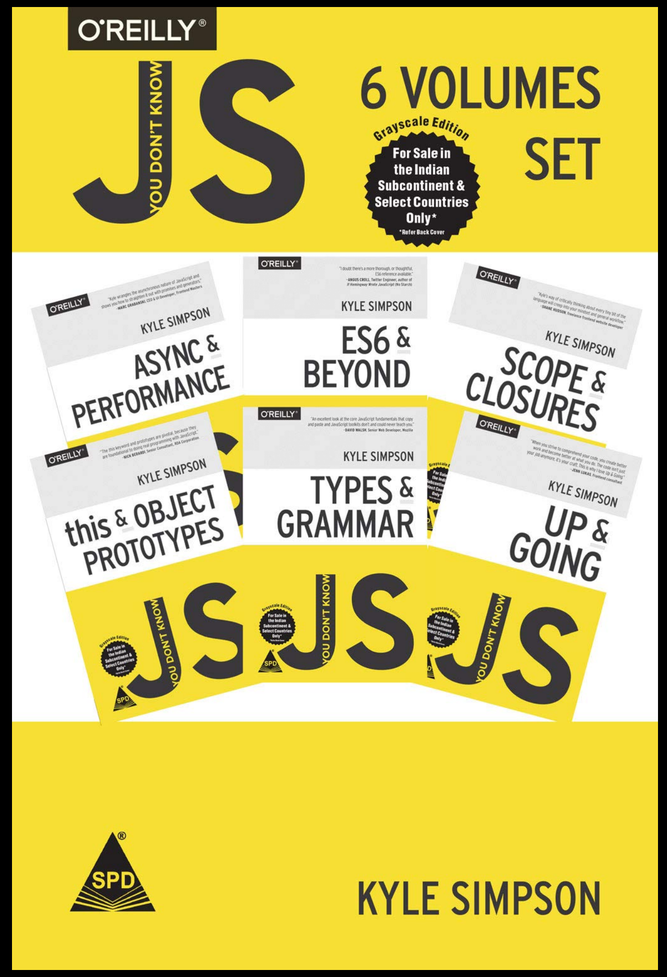
https://developer.mozilla.org/en-US/

Reference



https://javascript.info/

Reference



⛔ piracy please

📧 🙋‍♂️

shihabus

@type__error

shihabus

?

What if

```
console.log('Hello')

setTimeout(()=> {
  console.log('World')
}, 0)

console.log('!')
```

STAY

curious
updated
home