


[ASP.NET](#) | [CLOUD](#) | [CSS](#) ▾ | [DEVOPS](#) | [HTML5](#) | [JAVASCRIPT](#) ▾ | [NOSQL](#) | [PHP](#) | [PYTHON](#) | [RUBY](#) | [WEB DEV](#) | [WEB SERVERS](#) | [WORDPRESS](#) | [A](#)
[Home](#) » [JavaScript](#) » Demystifying the concepts of 'Asynchronous JavaScript'

ABOUT SOUMYAJIT BASU



Soumyajit is a QA/DevOps engineer by profession and a technology enthusiast by passion. He loves communicating about technology and is an author at his own blog platform as well as in Dzone and Web Code Geeks.



Demystifying the concepts of 'Asynchronous JavaScript'

Posted by: [Soumyajit Basu](#) in [JavaScript](#) on November 6th, 2017

Javascript has been one of the vital constituents for application development nowadays in the world of information technology. Every application that is developed now is web-based, and when it comes to accessing the application through the web browser, you got to depend on javascript.

So, this blog is basically about breaking down the concepts of asynchronous requests

in javascript which plays a vital role in how javascript functions. As we all know that javascript is single threaded, so definitely if we are to use synchronous

calls for each and every action in the browser, then the browser must halt the rendering of the page in order to complete the execution of the javascript code. When the browser encounters synchronous

javascript tag, it blocks the page from rendering until the code execution completes. Modern websites have moved from this method because it presents a direct risk to delaying page load time. The downside associated with this method is that the entire site is blocked from loading until the tag fully loads.

This can considerably slow response time associated with vendors, slow internet traffic etc. Now for the seamless experience of processing requests and rendering the page content in a web view, you have to depend upon asynchronous calls. This would not only separate javascript tags but also load them independently from each other. The asynchronous

request minimises the impact of loading external javascript files on the page rendering process.

Functioning of Call Stack in Javascript

Javascript is a single threaded programming language, which means it has a single call stack

and it can serve only one request at a time. Let us consider the following codebase

```
01 function multiply(a,b){
02   return a*b;
03 }
04
05 function square(n){
06   return multiply(n,n);
07 }
08
```



JavaScript Interview Questions

- Succeed in any JavaScript job interview
- Uplift your career
- Claim a higher salary

[Download for Free](#)

NEWSLETTER

39,892 insiders are already receiving weekly updates and complimentary whitepapers!

Join them now to gain **ACCESS** to the latest news in the developers world, as well as in HTML5, CSS, JavaScript, WordPress related technologies.

Email address:

Sign up

JOIN US



have a blog with unique and interesting content

With **1,500** unique visitors, we have placed a resource site among the top 1000 sites in the world. You can join our team and become a part of our success story.

multiply

two numbers. The second function calls the

multiply

function and the third function calls the

square

function followed by a

console.log

Quick Fact: So, the

call stack

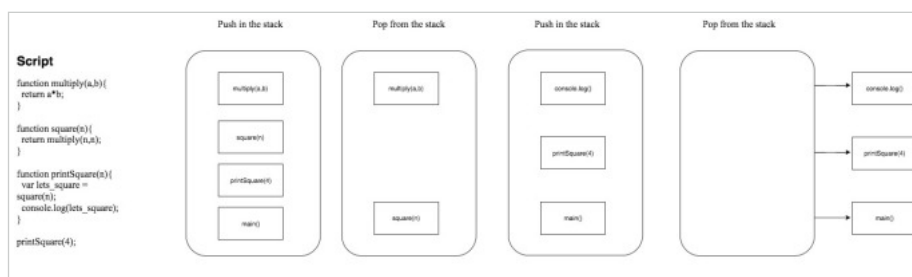
is basically the data structure which records where in the program we are. When we step into a function we push something onto the stack, when we return something from the function we pop up from the top of the stack. The objects that gets pushed in the call stack are referred to as

stack frames

. So each of the function within the call stack occupies a single

stack frame

So, if we follow the above codebase, diagrammatically it can be represented as follows. When a function is called, it is pushed on to the stack. When the function returns a value, it is popped out from the stack.



So, the above diagram is a visual representation of how does javascript store elements within the

call stack

How does asynchronous javascript function work?

Now let us understand how is asynchronous javascript functioning. Let us look at a diagram first, which will clarify the interaction between the js runtime engine

web APIs

task queue

and the

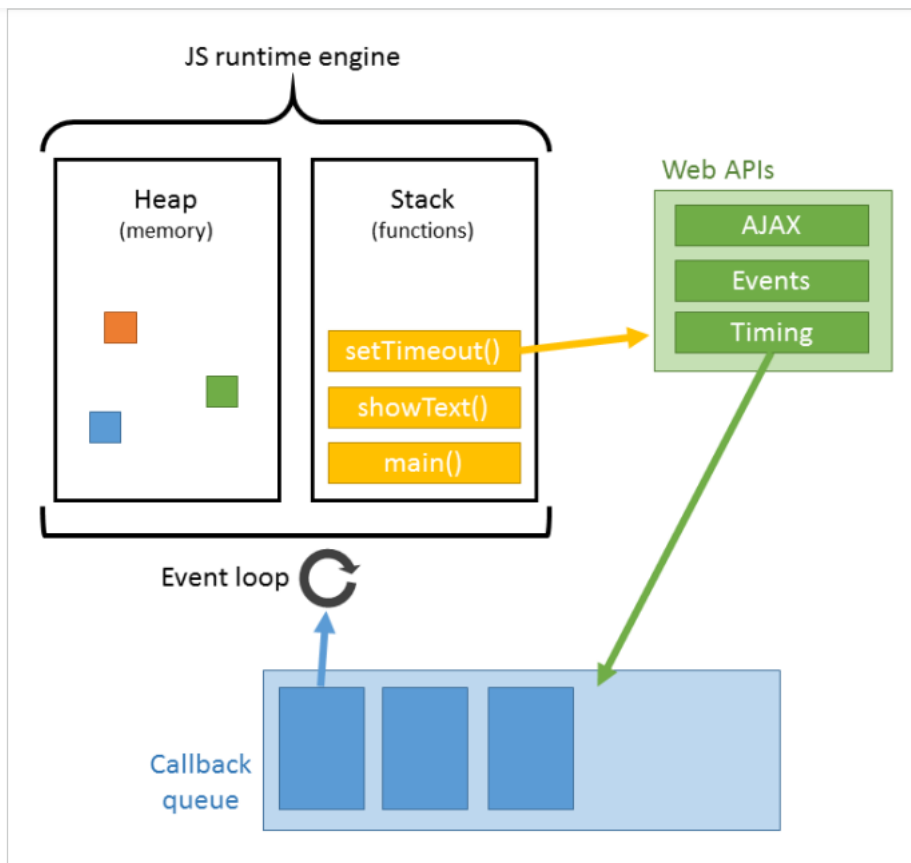
event loop



JavaScript Interview Questions

- Succeed in any JavaScript job interview
- Uplift your career
- Claim a higher salary

[Download for Free](#)



The js runtime engine consists of the

1. The

Heap

which basically takes care of the memory allocation.

2. The

call Stack

which consists of all the javascript function.

3. The

Web APIs

are threads provided by browsers that can only be requested for performing specific asynchronous operations.

4. The

Task Queue

or the

callback Queue

which consists of the event callbacks or tasks and

5. The

Event Loop

which is responsible in transferring the tasks to be executed from the

Task Queue

to the

call Stack

when the stack is empty.

Let us consider a small code snippet to understand how is the asynchronous request is processed in the back end.

```
1 console.log("Hi");
2
```



JavaScript Interview Questions

- Succeed in any JavaScript job interview
- Uplift your career
- Claim a higher salary

[Download for Free](#)

Script

```
console.log("Hi");
setTimeout(function cb(){
  console.log("There");
}, 5000);
console.log("Christmas is just
after one month");
```

Console

Hi

Stack

log('Hi')

main()

Web APIs



Task Queue

The first function to be called in the stack is

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

. We get 'Hi' in the console.

Script

```
console.log("Hi");
setTimeout(function cb(){
  console.log("There");
}, 5000);
console.log("Christmas is just
after one month");
```

Console

Hi

Stack

setTimeout(cb)

main()

Web APIs



Task Queue

Next, the

```
setTimeout(cb)
```

is called to the stack. We pass the callback function and the timer to the setTimeout function. Now, remember the

```
setTimeout
```

is an API provided by the browser. Now, the browser is going to handle the timer specified in the code. The function call for

```
setTimeout
```

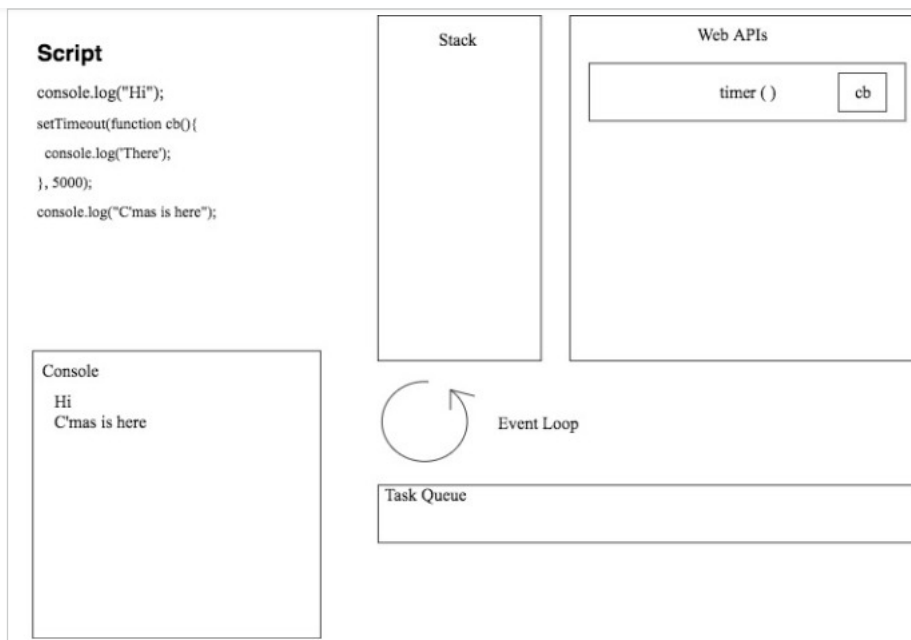
is complete so we can pop it out of the stack.



JavaScript Interview Questions

- Succeed in any JavaScript job interview
- Uplift your career
- Claim a higher salary

[Download for Free](#)



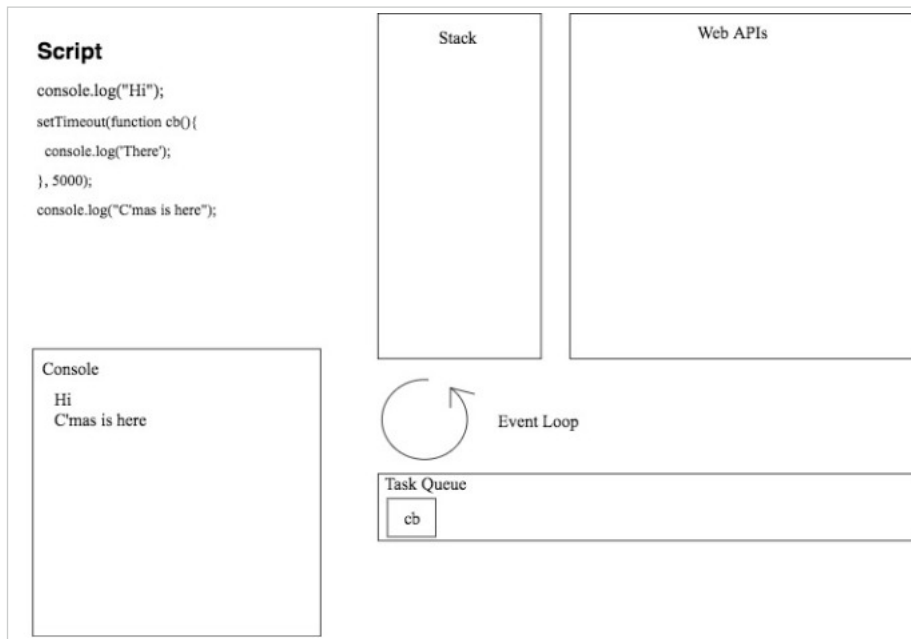
So, we have the timer in the Web API, which is going to complete after 5 seconds. Now the Web API cannot just start to modify the code otherwise, it would start appearing randomly. That is when the

Task Queue

or the

Callback Queue

kicks in.



Once the Web APIs are done, the callback gets pushed to the task queue, and finally we have the

Event Loop

Event Loop

is the simplest little piece in this whole equation and it has only one very simple job. It checks out the

Call Stack

and the

Task Queue



JavaScript Interview Questions

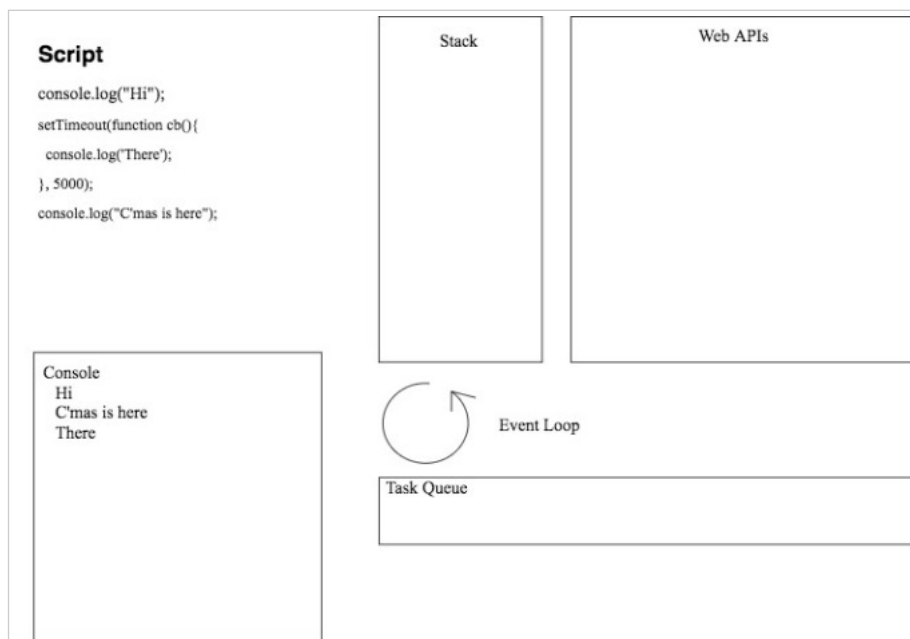
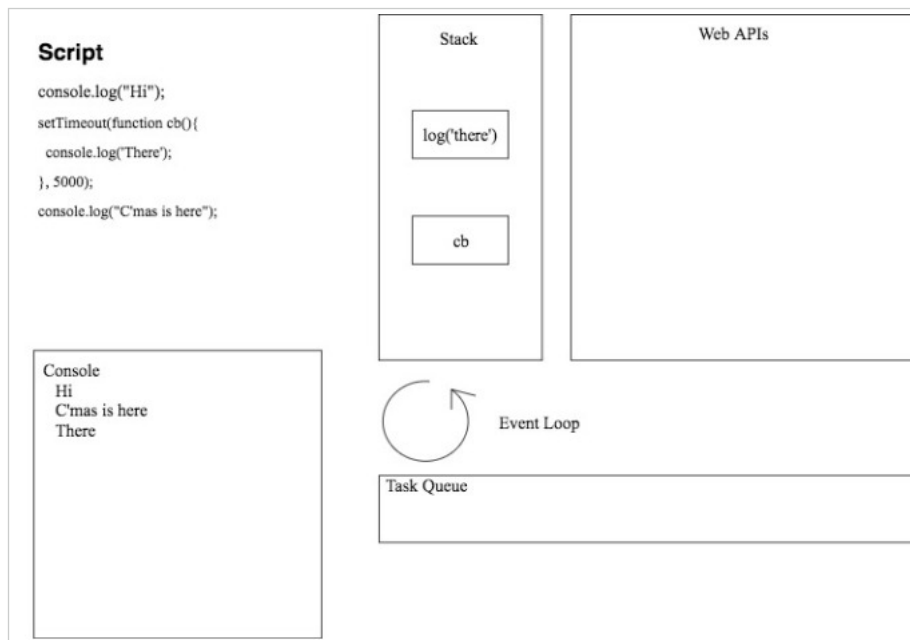
- Succeed in any JavaScript job interview
- Uplift your career
- Claim a higher salary

[Download for Free](#)

and pushes it onto the

Call Stack

which effectively runs it.



Therefore, asynchronous javascript plays a very important role in the proper processing of the events and the callbacks. Without it, the rendering of contents within the pages become quite difficult and takes quite a toll on the performance and the response time for an application.

Published on Web Code Geeks with permission by Soumyajit Basu, partner at our WCG program. See the original article here: [Demystifying the concepts of 'Asynchronous JavaScript'](#)

Opinions expressed by Web Code Geeks contributors are their own.

Do you want to know how to develop your skillset to become a **Web Rockstar**?

Subscribe to our newsletter to start Rocking [right now!](#)
To get you started we give you our best selling eBooks for **FREE!**

1. Building web apps with Node.js

JavaScript Interview Questions

- Succeed in any JavaScript job interview
- Uplift your career
- Claim a higher salary

[Download for Free](#)



Email address:

[Sign up](#)

Leave a Reply

Be the First to Comment!

Notify of



Start the discussion

KNOWLEDGE BASE

[Courses](#)

[Minibooks](#)

[Resources](#)

THE CODE GEEKS NETWORK

[.NET Code Geeks](#)

[Java Code Geeks](#)

[System Code Geeks](#)

[Web Code Geeks](#)

HALL OF FAME

[HTML5 Drag and Drop Example](#)

[Javascript copy to clipboard example](#)

[JavaScript Document Ready Example](#)

[JavaScript For Loop Example](#)

[JavaScript Sort Array Example](#)

[JavaScript String Compare example](#)

[JavaScript Submit Form Example](#)

[jQuery Autocomplete Widget Example](#)

[Single Page Apps](#)

[Twitter Bootstrap Modal Example](#)

ABOUT WEB CODE GEEKS

WCGs (Web Code Geeks) is an independent online community focused ultimate Web developers resource center; targeted at the technical arct team lead (senior developer), project manager and junior developers al the Web designer, Web developer and Agile communities with daily nev domain experts, articles, tutorials, reviews, announcements, code snipp source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Web Code Geel property of their respective owners.

Web Code Geeks and all content copyright © 2010-2017, Exelixis Media P.C. | [Terms of Use](#) | [Privacy Policy](#) | [Contact](#)



JavaScript Interview Questions

- Succeed in any JavaScript job interview
- Uplift your career
- Claim a higher salary

[Download for Free](#)