



Passing in Functions for Later Execution and JSON

By: Henry Britton



Javascript and Single Thread

“JavaScript is a single-threaded programming language. This means that JavaScript can do only one thing at a single point in time.”

- Starts execution at the top of the page, works down
- If it is working on something, you can't interact with the web-browser
- So if a function takes a long time, then theoretically no interaction with the browser would be possible



Definition: Blocking Function

A blocking function is any function that takes a long time to execute.

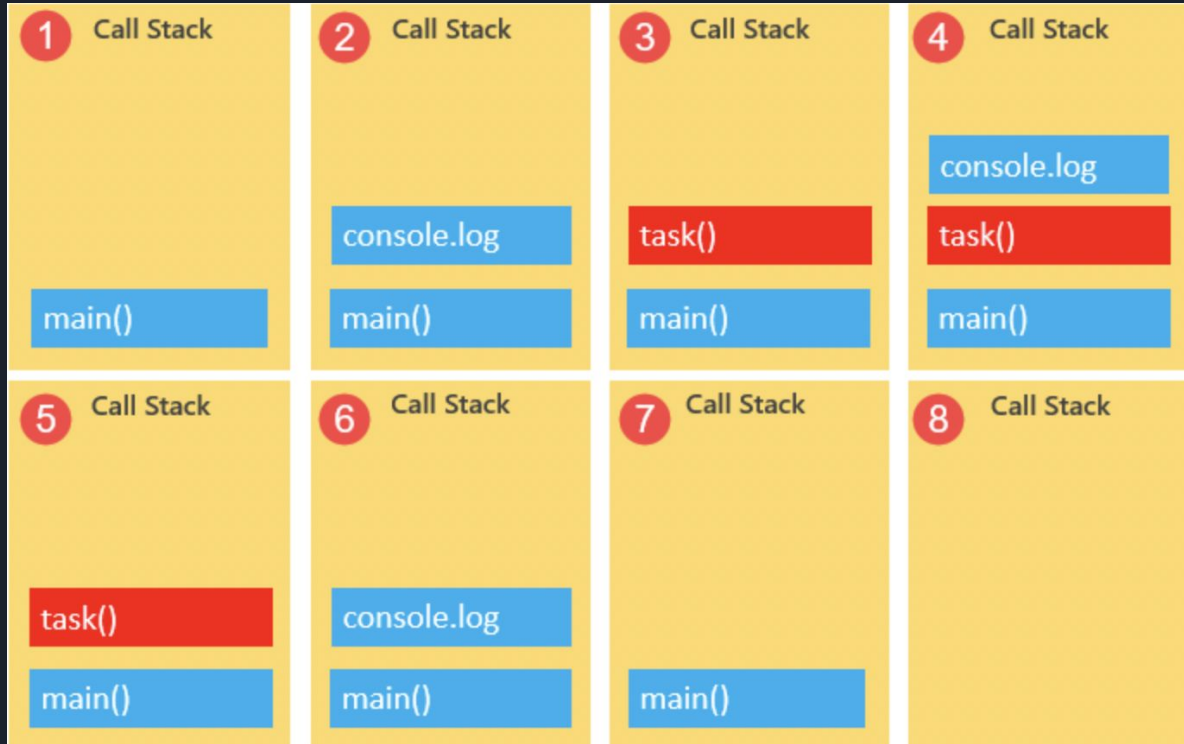
“Common widely used example of a blocking function is a function that calls an API from a remote server”

What does this look like? (what a hassle)

```
function task(message) {  
  // emulate time consuming task  
  let n = 10000000000;  
  while (n > 0){  
    n--;  
  }  
  console.log(message);  
}
```

```
console.log('Start script...');  
task( Download a file  
console.log('Done!');
```

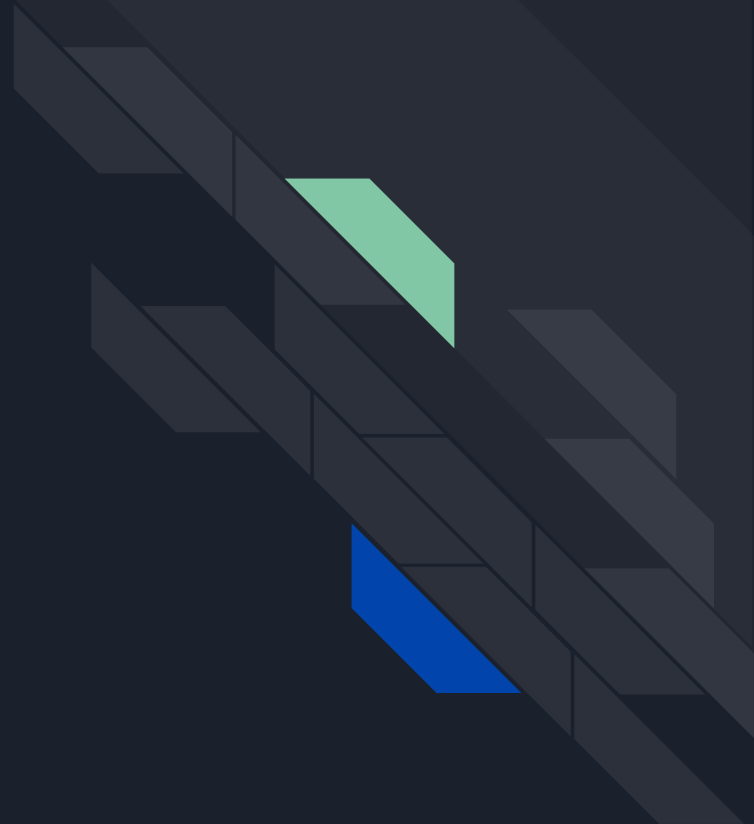
```
Start script...  
Download a file.  
Done!
```






Is there anything I can do about this?

No.





Just Kidding. Enter callback functions.
(higher order functions)

Callback functions: a
function as a *parameter*
of another function, to
be executed later on the
callback function **queue**

```
function isOdd(number) {  
  return number % 2 !== 0;  
}  
  
function filter(numbers, fn) {  
  let results = [];  
  for (const number of numbers) {  
    if (fn(number)) {  
      results.push(number);  
    }  
  }  
  return results;  
}  
  
let numbers = [1, 2, 4, 7, 3, 5, 6];  
console.log(filter(numbers, isOdd));
```

From Previous Example:

```
console.log('Start script...');

setTimeout(() => {
  task('Download a file.');
}, 1000);

console.log('Done!');
```

Start script...

Done!

Download a file.

WARNING!
SHORTHAND!

```
() => {
  console.log("I am awake now");
}

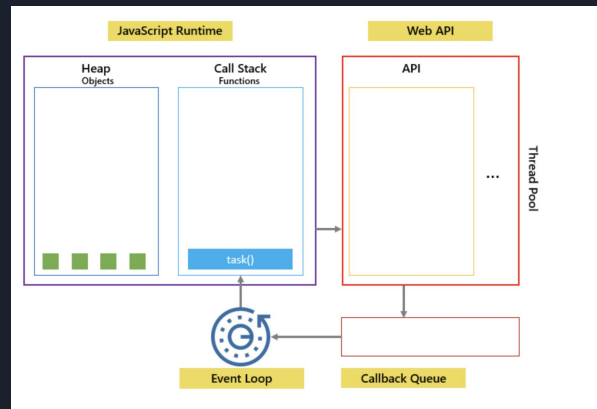
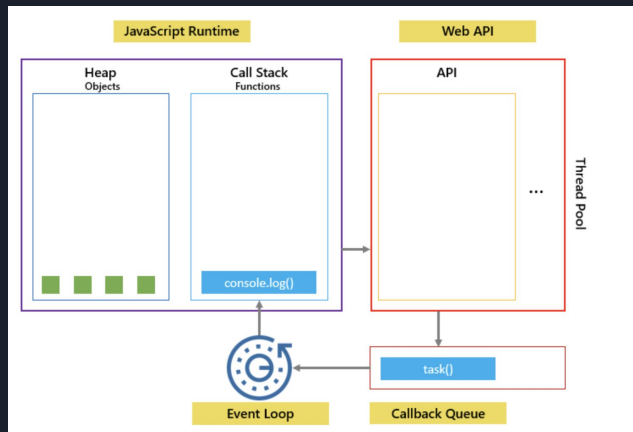
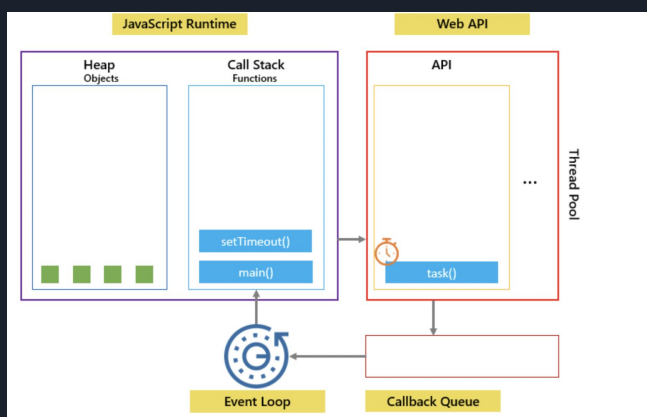
which is exactly equivalent to

function() {
  console.log("I am awake now");
}
```

The `setTimeout()` is a method of the `window` object. The `setTimeout()` sets a timer and executes a [callback function](#) after the timer expires.

How does this work with the function call stack then?

Just because the **JavaScript** engine is single threaded, it doesn't mean the **browser** is. The web browser can do activities like fetch requests, timeouts, and button presses concurrently.



Exercise: What does the following code output?

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

setTimeout(() => {
    console.log('Execute immediately.');
```



```
}, 0);

console.log('Bye!');
```



Answer:

```
Hi!
```

```
Bye!
```

```
Execute immediately.
```



Some Code!

Here is some code that uses callbacks to execute a query!

```
const oracledb = require('oracledb');

function getEmployee(empId, getEmployeeCallback) {
  oracledb.getConnection(function(err, conn) {
    if (err) {
      console.log('Error getting connection', err);
      getEmployeeCallback(err);
      return;
    }

    console.log('Connected to database');

    conn.execute(
      `select *
       from employees
       where employee_id = :emp_id`,
      [empId],
      {
        outFormat: oracledb.OBJECT
      },
      function(err, result) {
        if (err) {
          console.log('Error executing query', err);

          getEmployeeCallback(err);

          conn.close(function(err) {
            if (err) {
              console.log('Error closing connection', err);
            } else {
              console.log('Connection closed');
            }
          });

          return;
        }

        console.log('Query executed');

        getEmployeeCallback(null, result.rows[0]);

        conn.close(function(err) {
          if (err) {
            console.log('Error closing connection', err);
          } else {
            console.log('Connection closed');
          }
        });
      }
    );
  });
}
```



Javascript examples

<http://loin.cs.brynmawr.edu/~hbritton/js6.html>

Especially 6,7,8,10,12 (navigate by change the number in url after js)

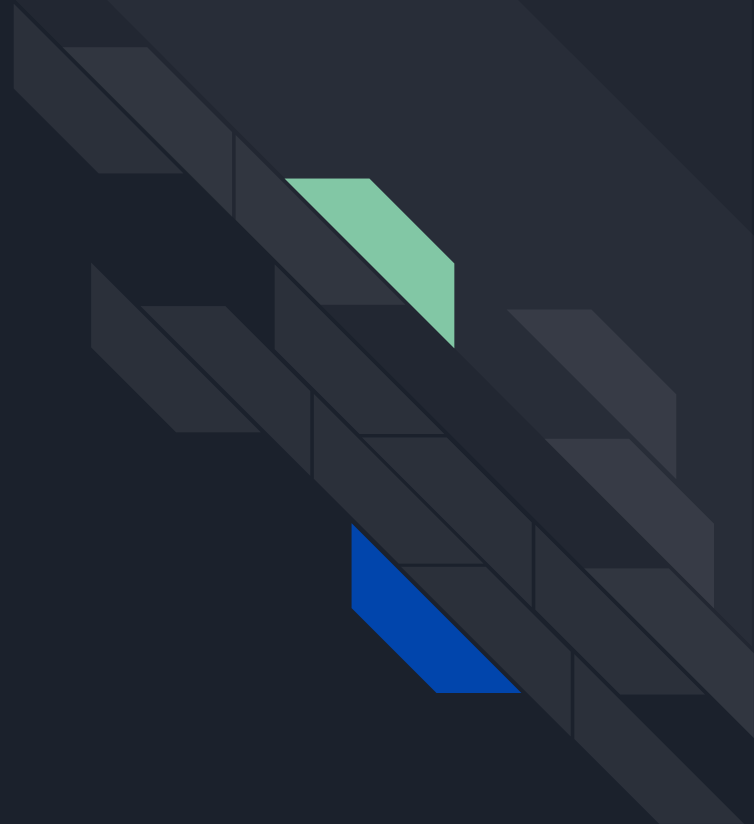


Quick Check-in:

What is the benefit of callback functions in JavaScript?

What is an example of when this process would be needed?

Part 2: JSON





What is JSON??

- JSON stands for JavaScript Object Notation
- It is a text format that sort of resembles a dictionary/hashmap
- JSON is a TEXT FORMAT, so while it was derived from a JavaScript object to be easily parsable in JS, other languages have code for parsing it as well



JSON String

```
'{"name":"John", "age":30, "car":null}'
```

Here is an **object** with 3 properties:
name, age, and car

```
let personName = obj.name;  
let personAge = obj.age;
```



Usages

- It is fully text based, so easy to send over servers
- It is easily parsable by JS, receive JSON data then immediately put it into JS object
- Can store JavaScript objects as text



Syntax Overview:

```
"name": "John"
```

- key/value pair separated by a colon
- Strings must use double quotes
- Slight difference between JavaScript object and JSON formatting- key must be string in JSON

JavaScript

```
{name: "John"}
```

JSON

```
{"name": "John"}
```



Legal JSON and JavaScript Object Values

JSON Values:

- String
- Number
- Object
- Array
- Boolean
- null

```
'{"name":"John", "age":30, "city":"New York}"'
```

JavaScript Values

- Above and
- Function
- Date
- undefined

```
person = {name:"John", age:31, city:"New York"};
```



Parsing JSON in JS

JavaScript has an inbuilt JSON parser that converts JSON formatted text into a JS object

```
const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
```

```
const text = '[ "Ford", "BMW", "Audi", "Fiat" ]';  
const myArr = JSON.parse(text);  
document.getElementById("demo").innerHTML = myArr[0];
```

Exceptions to this are the datatypes illegal in JSON like dates and functions, the parser cannot read these



Creating JSON

The function “stringify” converts a JS object to JSON, which can then be sent or store.

```
const myJSON = JSON.stringify(obj);
```



Check-in:

What are the key features of JSON?

What are the valid data types for a JSON key and value?



Sources

<https://www.javascripttutorial.net/javascript-event-loop/>

<https://webreference.com/javascript/basics/callbacks/>

<https://www.javascripttutorial.net/javascript-callback/>

Professor Towell's Examples: <http://cs.brynmawr.edu/~gtowell/383/jsXX.html>

https://www.w3schools.com/js/js_json_intro.asp

<https://dzone.com/articles/how-to-interact-with-a-database-using-callbacks-in>