Day5

Callbacks
Event Loop
Synchronous and Single threaded
NodeJs architecture- Single threaded event loop

Callbacks
(Asynchronous JS)

- In programming languages like c, ruby there is the expectation that whatever happens on line 1 will finish before the code on line 2 starts running and so on down the file, but Js is different.

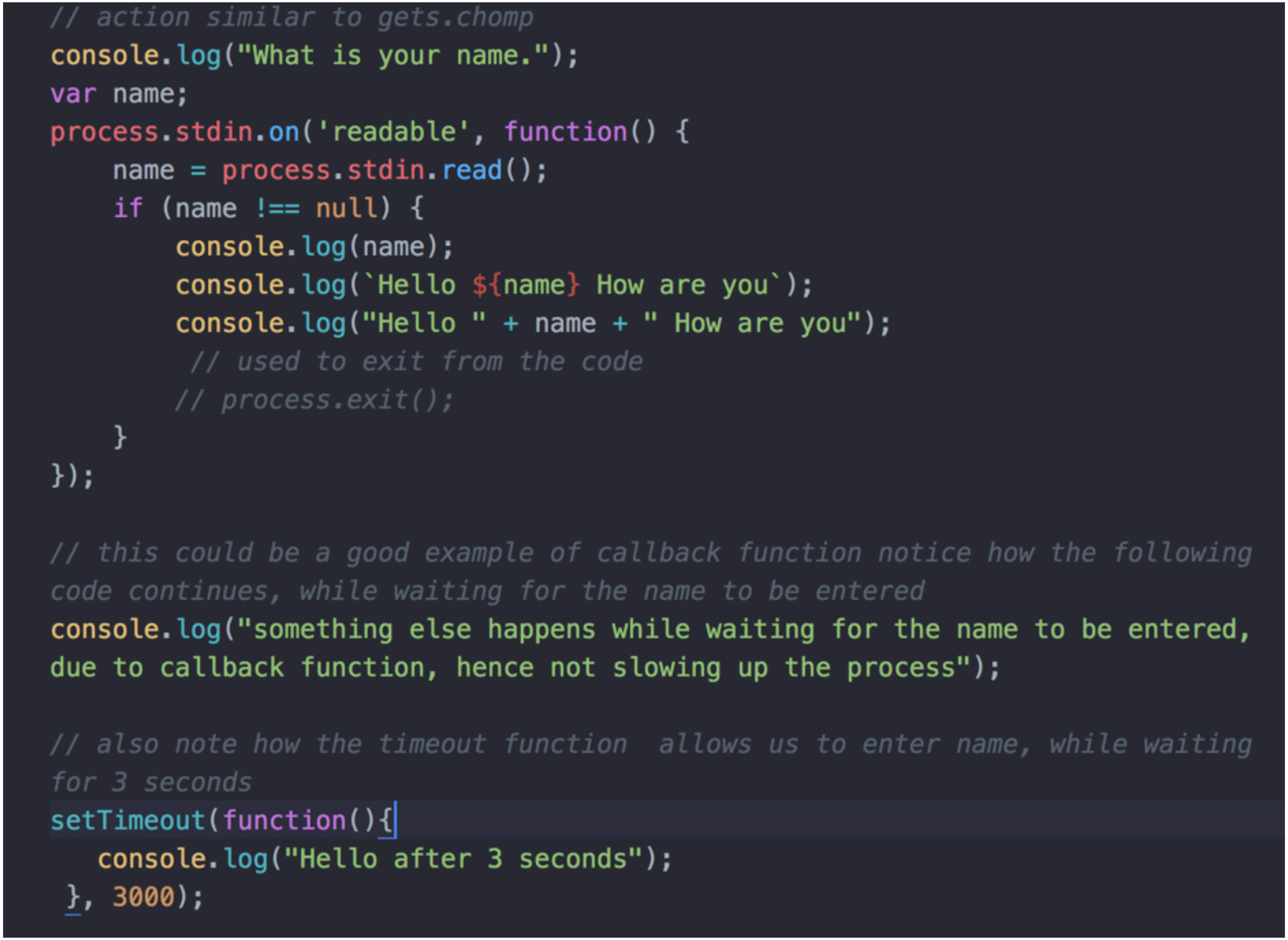
- Callback functions are the functions that allows you to perform other tasks while waiting for a time-consuming task to be completed.

- Time consuming tasks could be like loading pictures, downloading files etc.

- Callback behaviour makes Js faster:

- Eg: A task similar to gets.chomp in ruby, which halts further execution unless an input is provided, on the contrary continues execution in case of Js.

- https://github.com/NandiniNayak/javascript-LessonPlan/tree/master/lesson8-callbacks



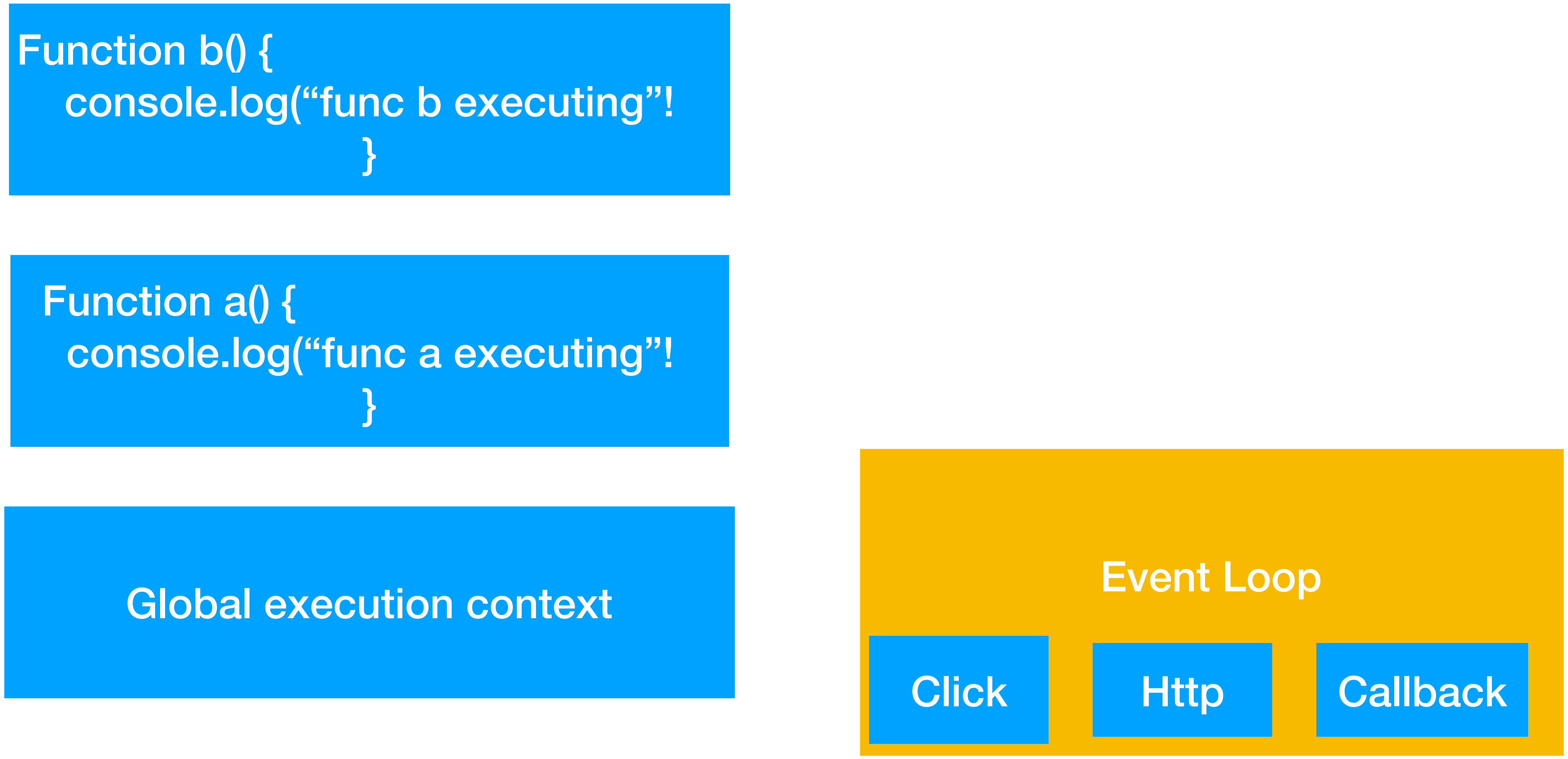
```
// action similar to gets.chomp
console.log("What is your name.");
var name;
process.stdin.on('readable', function() {
    name = process.stdin.read();
    if (name !== null) {
        console.log(name);
        console.log(`Hello ${name} How are you`);
        console.log("Hello " + name + " How are you");
         // used to exit from the code
        // process.exit();
    }
});

// this could be a good example of callback function notice how the following
code continues, while waiting for the name to be entered
console.log("something else happens while waiting for the name to be entered,
due to callback function, hence not slowing up the process");

// also note how the timeout function  allows us to enter name, while waiting
for 3 seconds
setTimeout(function(){
   console.log("Hello after 3 seconds");
 }, 3000);
```

JS - Synchronous and Single Threaded

- Is JS synchronous (one task at a time and in order) or Asynchronous(multiple tasks at a time) ??
  - **JS Engine is synchronous and single threaded(**one command at a time**)**
  - **Thread is a smallest unit of execution to which processor(cpu) allocates time**



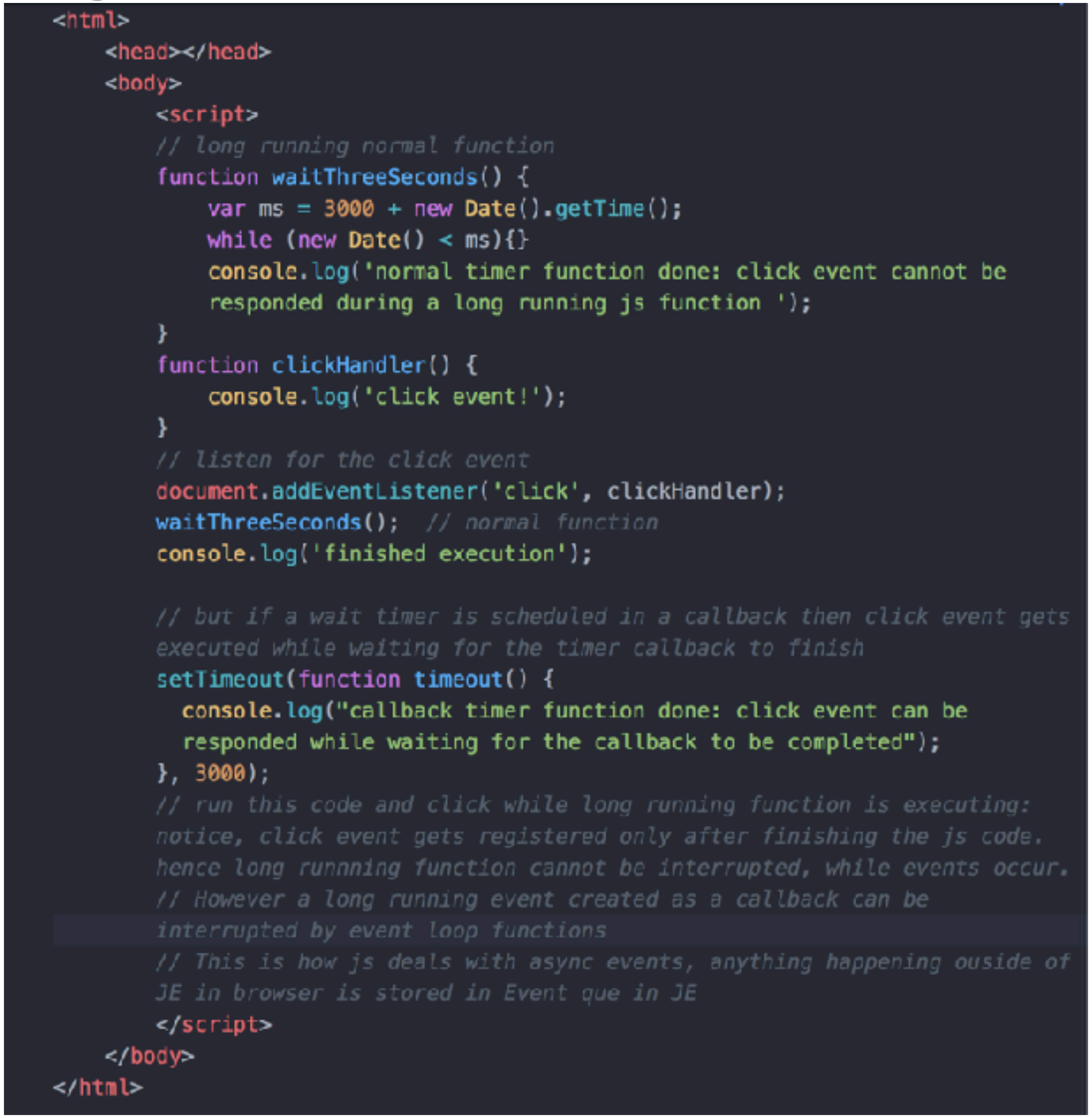
```
Function b() {
    console.log("func b executing"!
}
```

```
Function a() {
    console.log("func a executing"!
}
```

Node JS Architecture :
Single Threaded Event Loop

- If it is synchronous?? How does it handle external events such as click function?
  - Js handles async events(external trigger such as a click event) through a **event loop**, which executes **only after Js code is executed**
  - **Event loop** and **callbacks** results in non-blocking behaviour of node.js, resulting in not blocking input or output device
    - http://latentflip.com/loupe/?code=JC5vbignYnV0dG9uJywgJ2NsaWNrJywgZnVuY3Rpb24gb25DbGljaygpIHsKICAgIHNldFRpbWVvdXQoZnVuY3Rpb24gdGltZXIoKSB7CiAgICAgICAgY29uc29sZS5sb2coJ1lvdSBjbGlja2VkIHRoZSBidXR0b24hJyk7ICAgIAogICAgfSwgMjAwMCk7Cn0pOwoKY29uc29sZS5sb2coIkhpISIpOwoKc2V0VGltZW91dChmdW5jdGlvbiB0aW1lb3V0KCkgewogICAgY29uc29sZS5sb2coIkNsaWNrIHRoZSBidXR0b24hIik7Cn0sIDUwMDApOwoKY29uc29sZS5sb2coIldlbGNvbWUgdG8gbG91cGUuIik7!!!PGJ1dHRvbj5DbGljayBtZSE8L2J1dHRvbj4%3D

Async event handling

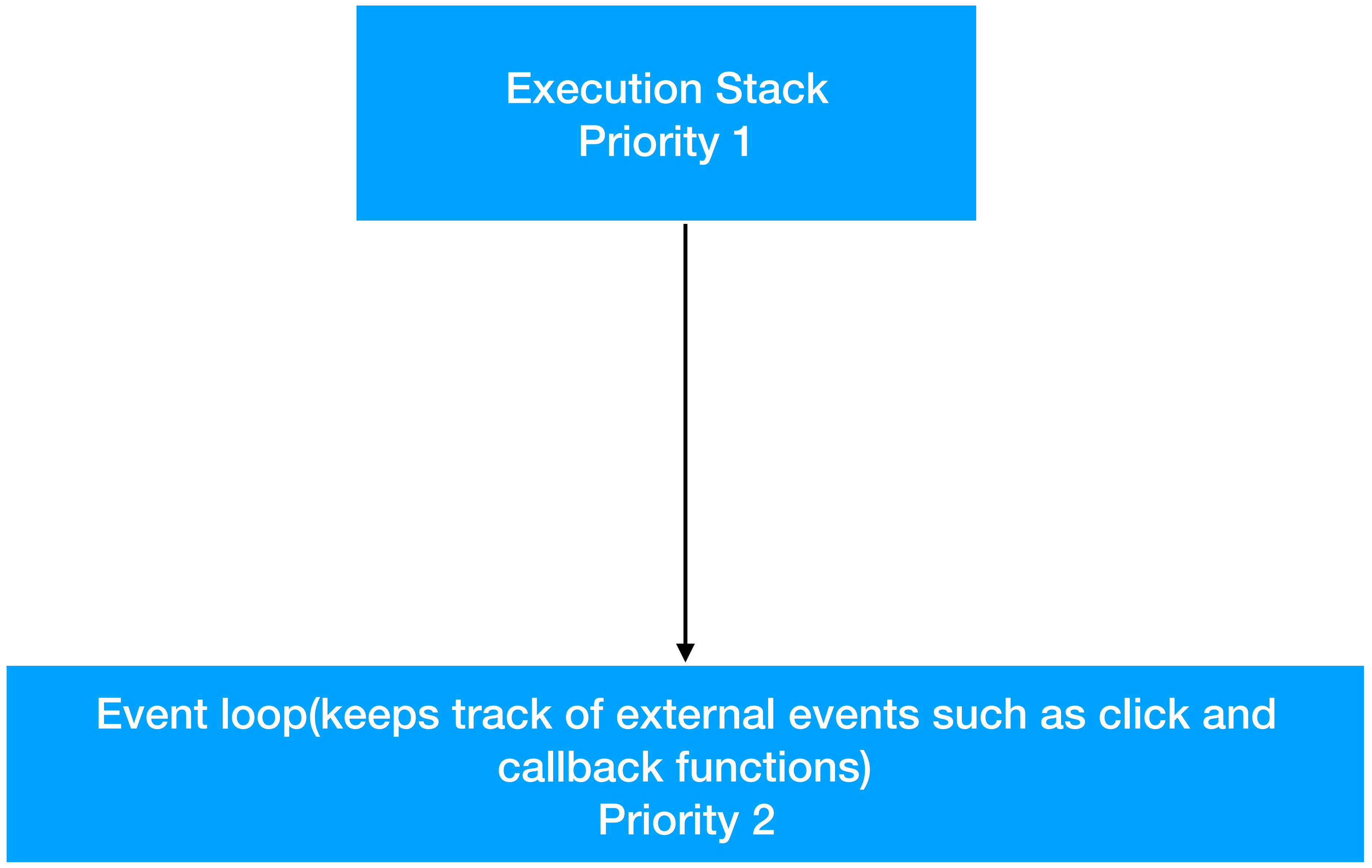


```
<html>
    <head></head>
    <body>
        <script>
        // long running normal function
        function waitThreeSeconds() {
            var ms = 3000 + new Date().getTime();
            while (new Date() < ms){}
            console.log('normal timer function done: click event cannot be
            responded during a long running js function ');
        }
        function clickHandler() {
            console.log('click event!');
        }
        // listen for the click event
        document.addEventListener('click', clickHandler);
        waitThreeSeconds();  // normal function
        console.log('finished execution');

        // but if a wait timer is scheduled in a callback then click event gets
        executed while waiting for the timer callback to finish
        setTimeout(function timeout() {
          console.log("callback timer function done: click event can be
          responded while waiting for the callback to be completed");
        }, 3000);
        // run this code and click while long running function is executing:
        notice, click event gets registered only after finishing the js code.
        hence long runnning function cannot be interrupted, while events occur.
        // However a long running event created as a callback can be
        interrupted by event loop functions
        // This is how js deals with async events, anything happening ouside of
        JE in browser is stored in Event que in JE
        </script>
    </body>
</html>
```

Event loop with a restaurant analogy

- https://www.youtube.com/watch?v=s9Zy8ISjxIw
- https://www.youtube.com/watch?v=h_HwkHobfs0


Execution Stack
Priority 1
Event loop(keeps track of external events such as click and callback functions)
Priority 2

Callback hell

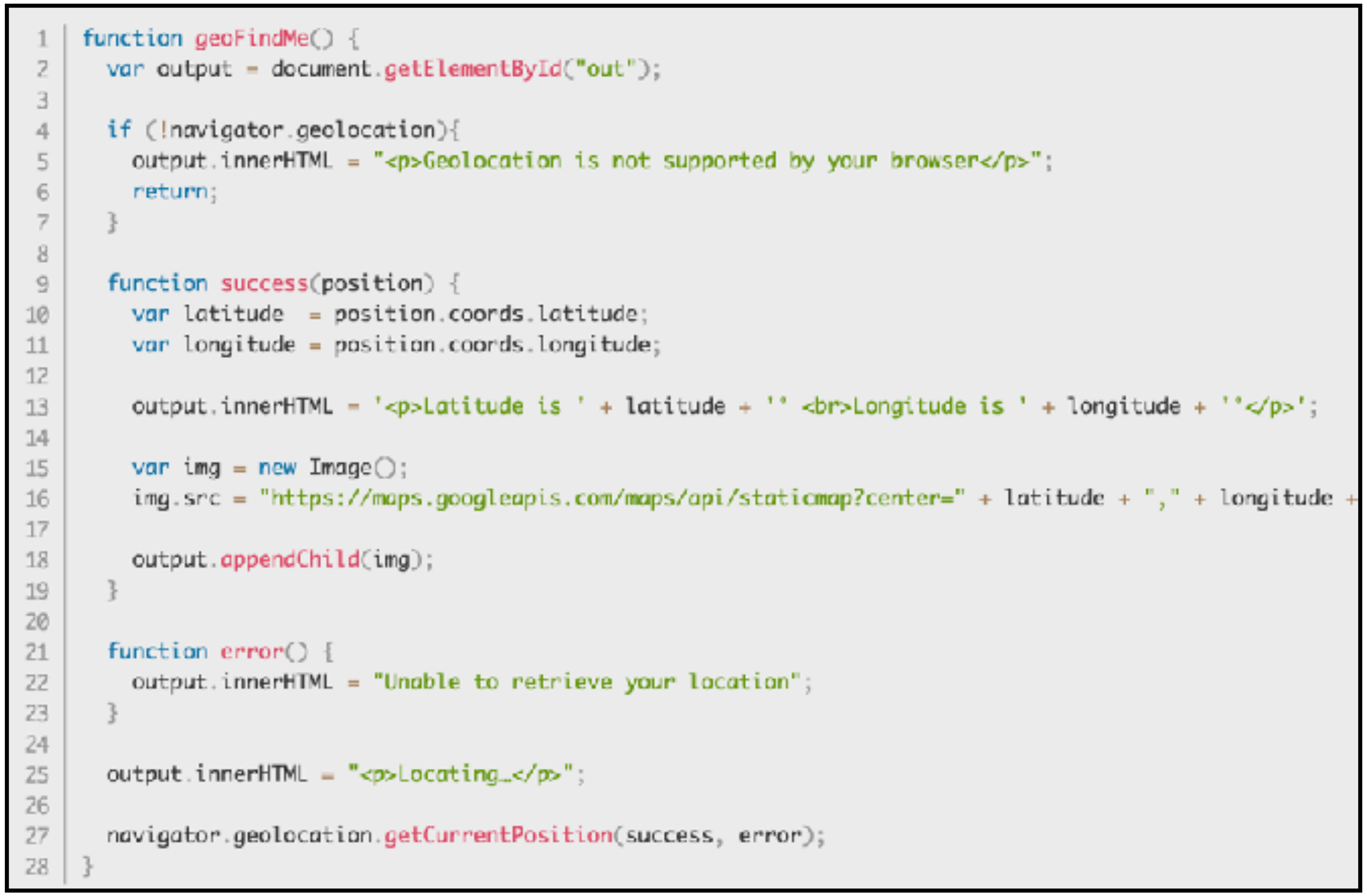
- http://callbackhell.com/
- **Don’t nest functions**, give them names and place them at top level of the program
- Handle **every single error** in every one of your callbacks
    - With callbacks the most popular way to handle errors is to reserve first argument of callback for an error
- Create reusable functions and place them in a module

Note

- When a callback is passed as a parameter to the function, it **does not have parenthesis** following it. We do not want it to be executed immediately.

- Eg: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API

- Notice **success callback** in the code snippet has no parenthesis following it in line 27, as the code will not be invoked immediately



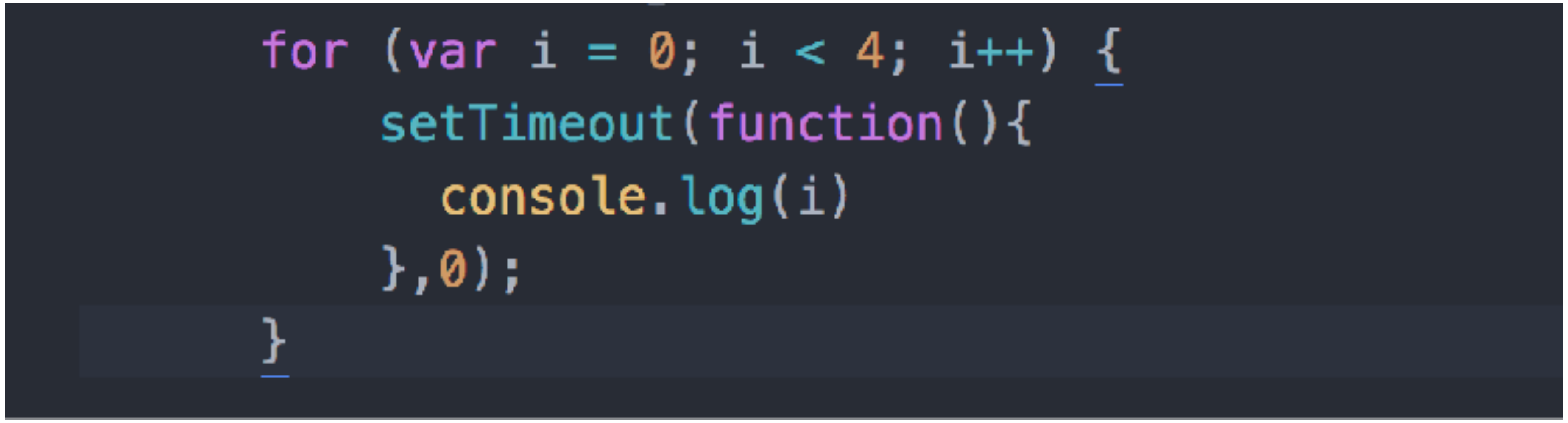
```
1  function geoFindMe() {
2    var output = document.getElementById("out");
3
4    if (!navigator.geolocation){
5      output.innerHTML = "<p>Geolocation is not supported by your browser</p>";
6      return;
7    }
8
9    function success(position) {
10     var latitude  = position.coords.latitude;
11     var longitude = position.coords.longitude;
12
13     output.innerHTML = '<p>Latitude is ' + latitude + '° <br>Longitude is ' + longitude + '°</p>';
14
15     var img = new Image();
16     img.src = "https://maps.googleapis.com/maps/api/staticmap?center=" + latitude + "," + longitude +
17
18     output.appendChild(img);
19   }
20
21   function error() {
22     output.innerHTML = "Unable to retrieve your location";
23   }
24
25   output.innerHTML = "<p>Locating…</p>";
26
27   navigator.geolocation.getCurrentPosition(success, error);
28 }
```

Resources

- https://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop
- https://medium.freecodecamp.org/javascript-callbacks-explained-using-minions-da272f4d9bcd
- https://www.youtube.com/watch?v=s9Zy8ISjxIw
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop

Interview question

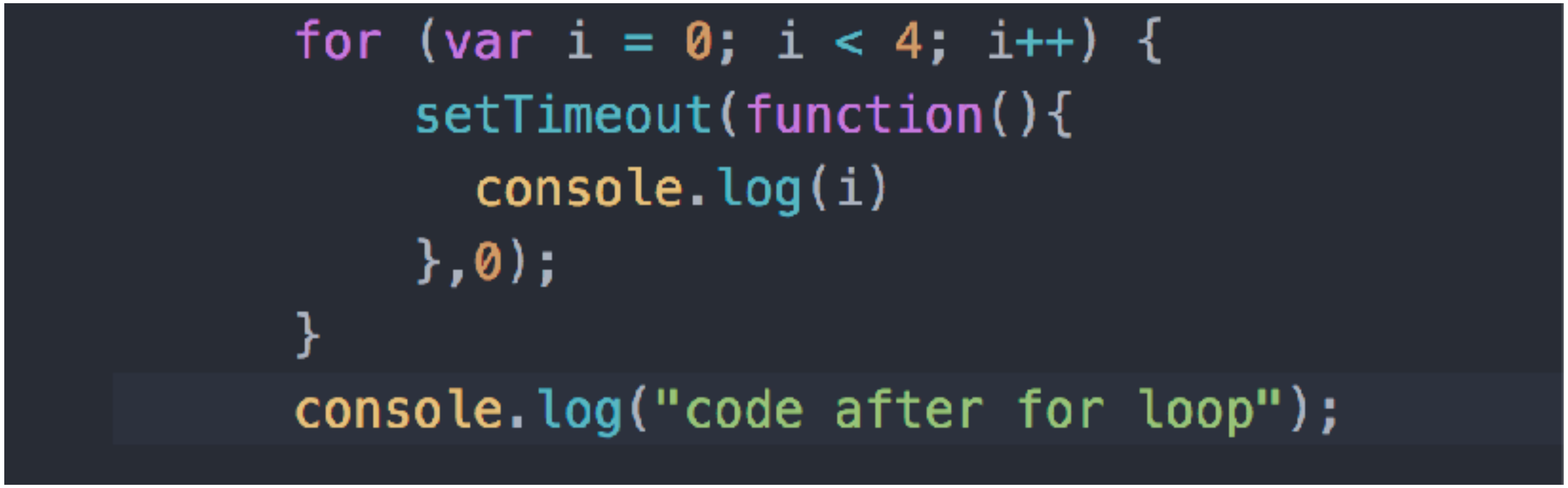
- What is the value of I ?



```
for (var i = 0; i < 4; i++) {
    setTimeout(function(){
      console.log(i)
    },0);
}
```

Interview question

- What is the sequence of console.log



```
for (var i = 0; i < 4; i++) {
    setTimeout(function(){
      console.log(i)
    },0);
}
console.log("code after for loop");
```