**Node.js Callback Function**

**Node.js Callback Function** : Asynchronism is one of the fundamental factor for Node.js to have become popular. And Callback is the realization of asynchronism for functions. Generally, in Node.js, most of the functions that work on resources have callback variants.

When an asynchronous function is called upon a resource for some task, the control is let immediately to continue with the subsequent statements after the function. The task on the resource would start in parallel. This helps Node.js continue with other tasks while the function is working with the resource. Once the task with the resource is completed, Node.js resumes with the callback function. Callback function is called with arguments : data object, result object and (or) error object containing information regarding the task.

**Blocking Function**: In contrast to asynchronous function, a synchronous function blocks the execution until the task on the resource is completed. Therefore synchronous function is also called as blocking function.

**Node.js Nested Callback Function**: If there are multiple operations to be done on the resource sequentially, and also has to be done asynchronously, you may nest the callback functions, one in another.

Now we shall see the working of a callback function in comparison with a blocking function, on reading (task) a file (resource).

* [Example](https://www.tutorialkart.com/nodejs/node-js-callback-function/#Example-1) for Node.js Blocking Function
* [Example](https://www.tutorialkart.com/nodejs/node-js-callback-function/#Example-2) for Node.js Callback Function
* [Example](https://www.tutorialkart.com/nodejs/node-js-callback-function/#Example-3) for Node.js Nested Callback Function

**Example 1 – Blocking Function**

In this example, we will read a file “sample.txt” synchronously using readFileSync() method.

**read-file-sync.js**

|  |
| --- |
| var fs = require('fs');    // read file sample.txt  var data = fs.readFileSync('sample.txt');  console.log("Reading file completed : " + new Date().toISOString());    console.log("After readFileSync statement : " + new Date().toISOString()); |

Run this program using node in command-prompt/terminal, and you will get the following output.

**Output**

|  |
| --- |
| arjun@arjun-VPCEH26EN:~/nodejs$ node read-file-sync.js  Reading file completed : 2017-10-19T12:21:40.103Z  After readFileSync statement : 2017-10-19T12:21:40.105Z |

TheAfter readFileSync statement  is always executed only after reading the file is completed.fs.readFileSync  is blocking the execution flow.

**Example 2 – Node.js Callback Function**

Following is an example node script which reads “sample.txt” file asynchronously, with the help of **Node.js Callback Function**.

**read-file-async.js**

|  |
| --- |
| var fs = require('fs');    // read file sample.txt  fs.readFile('sample.txt',      // callback function that is called when reading file is done      function(err, data) {          if (err) throw err;          // data is a buffer containing file content          console.log("Reading file completed : " + new Date().toISOString());  });    console.log("After readFile asynchronously : " + new Date().toISOString()); |

Run this program using node in command-prompt/terminal, and you will get the following output.

**Output**

|  |
| --- |
| **arjun@arjun-VPCEH26EN:~/nodejs$ node read-file-async.js**  **After readFile asynchronously : 2017-10-19T12:25:36.982Z**  **Reading file completed : 2017-10-19T12:25:36.987Z** |

You may observe that even after executing console.log(“After readFile asynchronously  statement, it took around 5ms to complete reading the file. fs.readFile(‘sample.txt’, callback function{..}) has not blocked the execution, instead a new process is started in parallel with the main control flow, to read the file (perform the task on resource).

**Example 3 – Node.js Nested Callback Function**

To demonstrate Node.js Nested Callback Function, we shall consider a scenario of [renaming a file](https://www.tutorialkart.com/nodejs/node-fs-rename-file/) and then [deleting](https://www.tutorialkart.com/nodejs/delete-a-file-in-nodejs-using-node-fs/) it using asynchronous functions.

**nodejs-nested-callback.js**

|  |
| --- |
| var fs = require('fs');    fs.rename('sample.txt', 'sample\_old.txt',      // 1st call back function      function (err) {          if (err) throw err;          console.log('File Renamed.');          fs.unlink('sample\_old.txt',              // 2nd call back function              function (err) {                  if (err) throw err;                  console.log('File Deleted.');              }          );      }  ); |

Run this program using node in command-prompt/terminal, and you will get the following output.

**Output**

|  |
| --- |
| arjun@arjun-VPCEH26EN:~/nodejs$ node nodejs-nested-callback.js  File Renamed.  File Deleted. |