|  |  |  |
| --- | --- | --- |
| Interacting with Node in the command-line | Type-in node | This will open Node REPL (Read-Evaluate-Print-Loop)  *Test JavaScript code snippet* |
|  | Type-in .exit or CTRL+D | To exit REPL |
| Core Modules | All kind of functionality is stored in Modules | const fs = require('fs');    // Read from file synchronosly:  const textIn = fs.readFileSync('./txt/input.txt','utf-8'); |
| Asynchronous Nature of Node.js | Heavy work is done in background. All other code is executed on one thread | The **callback** model - function is called once the one before has finished its work; this can quickly lead to some hard to read and manageable code **– Callback Hell**.  To solve this problem, we can use **ES6 Promises or ES8 async/await.** |

const fs = require('fs');

// callback hell example:

fs.writeFile('start.txt', 'utf-8', (err,data1) => {

fs.writeFile(`${data1}.txt`,'utf-8',(err,data2) => {

fs.readFile('append.txt','utf-8',(err,data3) => {

fs.writeFile('final.txt', `${data2} ${data3}` ,'utf-8', (err) => {

if(err) throw err;

});

});

});

});

|  |  |  |
| --- | --- | --- |
| **Creating a Web Server** | Include the **http-module** | const http = require('http'); |
|  | Create the Server | // create server  const server = http.createServer((req,res) => {  res.end('Hello from the server');  });  // use the server listen method()  server.listen(8000, '127.0.0.1', () => {  console.log('Listening to requests on port 8000');  }); |
| **Routing** | Include the url-module | const url = require('url'); |
|  | A **HttpHeader** is metadata about the response. | We can use **writeHead**()to send back headers by adding objects as a parameter to the method. One of the standard headers is to inform the browser of the Content Type. When we set to Content Type to text/html, the browser will now expect some html. We can also add our own headers in here and pass on data about the response. |

const server = http.createServer((req,res) => {

const pathName = req.url;

if(pathName === '/' || pathName === '/overview'){

res.end('This is the OVERVIEW');

}else if(pathName === '/product'){

res.end('This is the PRODUCT');

}else{

res.writeHead(404, {

'Content-type':'text/html',

'my-own-header':'hello-world'

});

res.end('<h1>Page not found!</h1>')

}

});

|  |  |  |
| --- | --- | --- |
| **Build a Simple API** | A service from which we can request some data |  |
|  | **JSON** is a text format that looks like a JavaScript Object. | A common use of JSON is to exchange data to/from a web server.  When receiving data from a web server, the data is always a string. Parse the data with **JSON.parse()** , and the data becomes a JavaScript object. |
|  | - add another route(/api) to our project | }else if(pathName === '/api'){  res.end('This is an API');  }else{ |
|  | - read data from the data.json file | use the synchronous version of the file steam reader - fs.readFileSync(), and place it to the top level. **Top Level code** is only executed once in the beginning, because it is outside the callback function. |
|  | - parse JSON into a JavaScript object |  |
|  | - send back the result to the client |  |
|  | When a file is requested, to find its location in the file system, we could access it with this type of code: | fs.readFile('./dev-data/data.json') The dot (.) in node referse to the directory from which we run the node command in the terminal.  If we run the node command somewhere else the dot would mean something else. |
|  | A better approach to locate the script that we want to execute in the files system : | ***\_\_dirname*** variable  All node.js scripts get access to this variable. This variable always translates to the directory in which the script is located.(Note that only exception for this rule is when used with the required function). To access the \_\_dirname variable use a template string. |

// top-level code - use synchronous read function.

const data = fs.readFileSync(`${\_\_dirname}/dev-data/data.json`, 'utf-8');

const dataObj = JSON.parse(data);

const server = http.createServer((req,res) => {

const pathName = req.url;

if(pathName === '/' || pathName === '/overview'){

res.end('This is the OVERVIEW');

}else if(pathName === '/product'){

res.end('This is the PRODUCT');

}else if(pathName === '/api'){

res.writeHead(200,{ 'Content-type': 'application/json'});

res.end(data); // sending back data from top-level.

}else{

res.writeHead(404, {

'Content-type':'text/html',

'my-own-header':'hello-world'

});

res.end('<h1>Page not found!</h1>')

}

});