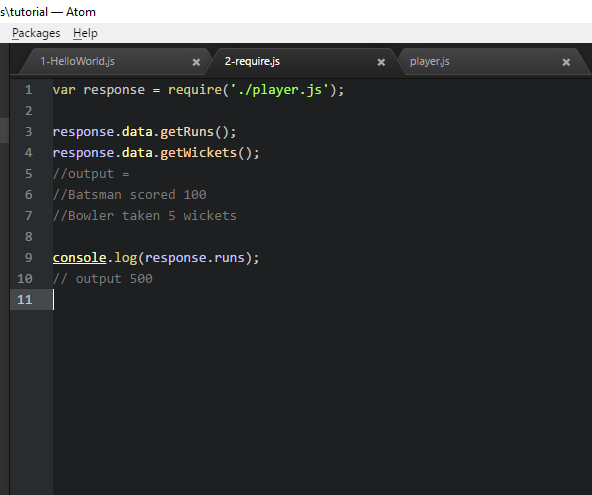
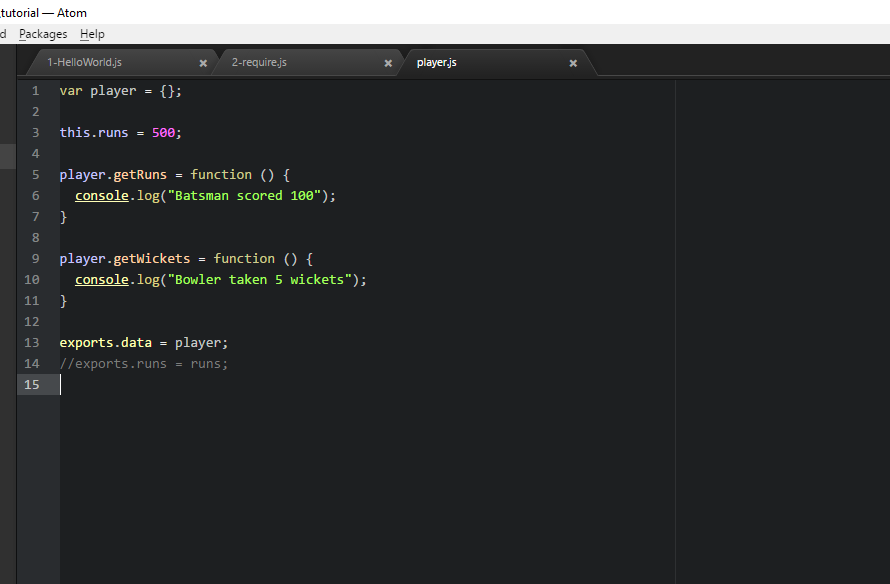
NodeJS Step-By-Step guide

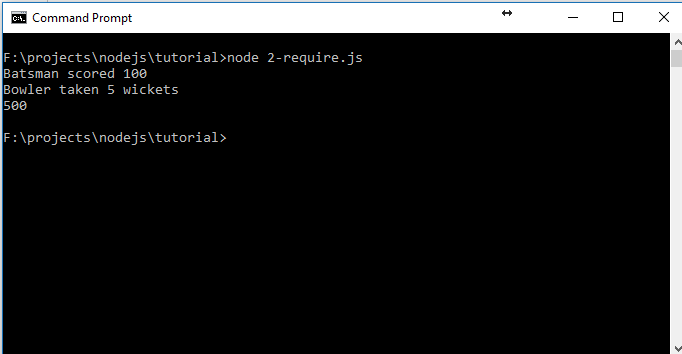
# Use of require

* Require allows piece of js code written in some other file (player.js) to be inserted into main 2-require.js file
* Inside player.js we define player {}
* Export allows methods or variables written in player.js file to be accessed from 2-require.js file
* In this example, either we declare var runs = 500 and then export.runs = runs or we can declare this as this.runs = 500 , then only runs can be accessed from main file



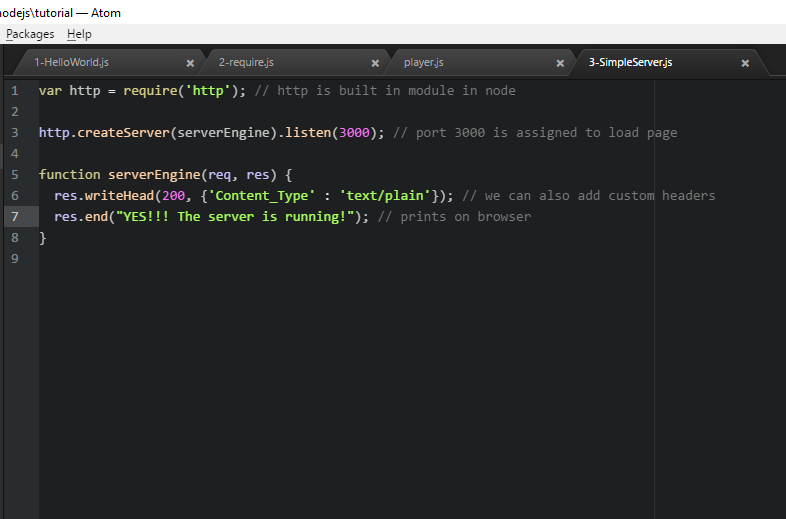


Output:



# Create Simple Server

* http is built-in/default module provided by nodejs
* createServer api called on http accepts callback function as serverEngine(req, resp) and listens to port 3000, we can assign any port
* **res.writeHead(200, {'Content\_Type' : 'text/plain'});** will have **StatusCode: 200** and object {} wherein we can add other Key\_value pairs such as custom headers
* **res.end(“YES!!! The server is running!”)** would run on browser with <http://localhost:3000/>



Output:

|  |
| --- |
|  |

# Modules and Module States

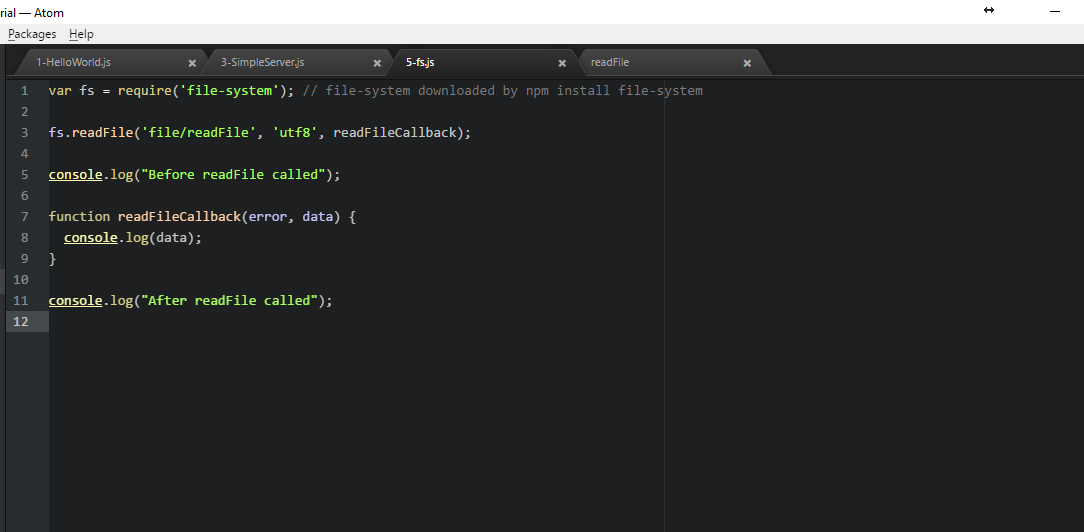
* In progress for ref see -- <https://www.youtube.com/watch?v=YBQTGWzF10c&list=PLYxzS__5yYQmHbpKMARP04F344zYRX91I&index=7>

# File System

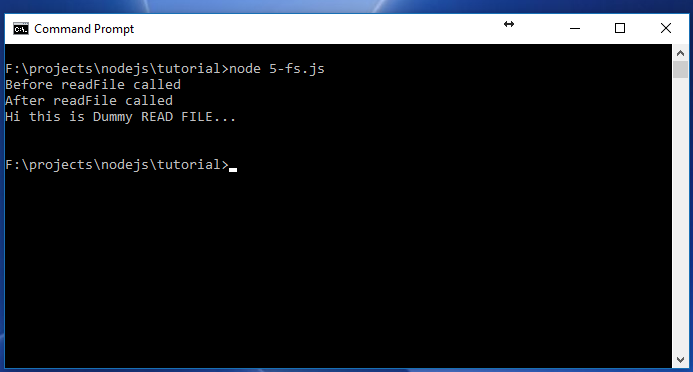
* Used to read or write into file
* Need to download file-system module by **npm install file-system --save**

Read File

* **fs.readFile (“name and path of file”, “encoding”, callback Fn)**
* readFile is **async** function
* in order to make synchronous we need to **write var data = fs.readFileSync(“name and path”, “encoding”)** and then output the data as **console.log(data)**

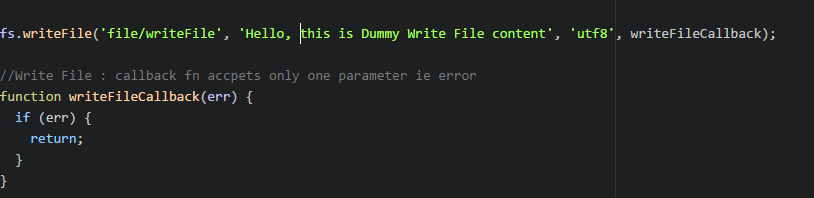


Output:

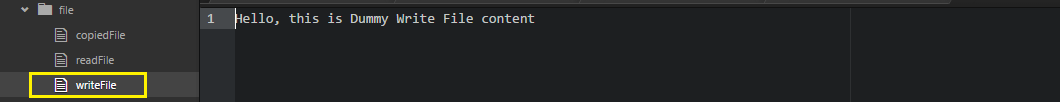


Write File

* **fs.writeFile(“name and path of file to which we write our content”, “content”, “utf8”, callbackFn)**
* writeFile is **async** function
* in order to make synchronous we need to **write var data = fs.writeFileSync(“name and path”, “encoding”)** and then output the data as **console.log(data)**
* **fs.writeFile callback** fn accepts only one parameter i.e. **error**

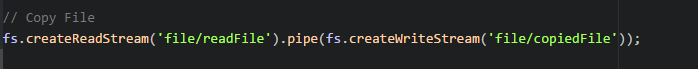


writeFile output: new file writeFile created in file folder with new content in it



# Copy files content to another file

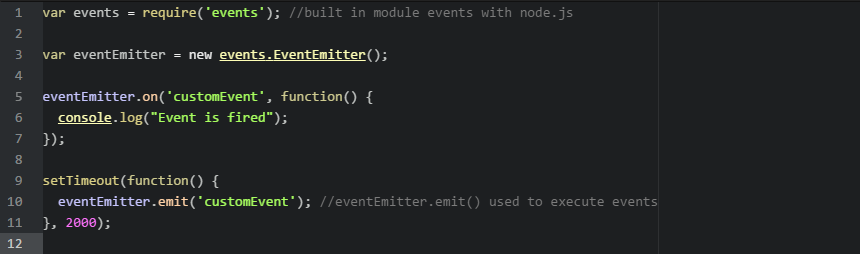
* to copy we use file stream mechanism
* streaming allows read data from source or write data to destination file in **continuous** manner



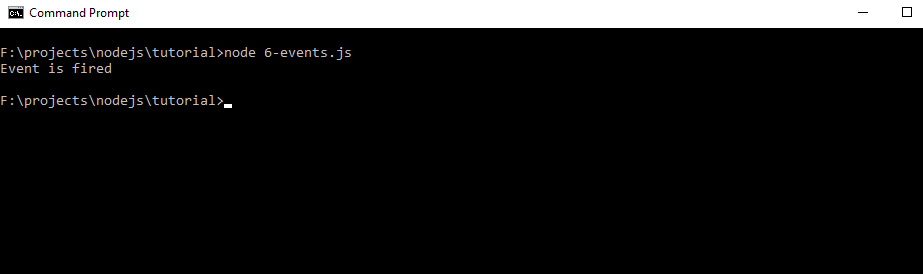
* **fs.createReadStream** would read file content from readFile
* **pipe** : Piping is a mechanism where we provide the output of one stream as the input to another stream
* **fs.createWriteStream** would create/override the existing file with destination as file/copiedFile and copy the contents of readFile into it

# Events in NodeJS

* **events** is built-in module in nodejs and can import as **require(‘events’)**
* **var eventEmitter = new events.EventEmitter();** used to create and emit new/custom/built-in events
* **customEvent** is new custom event and can be initialized on eventEmitter()

****

* this will output **Event is fired** in command line console and execute after 2 secs as we have wriiten setTimeout for 2 secs

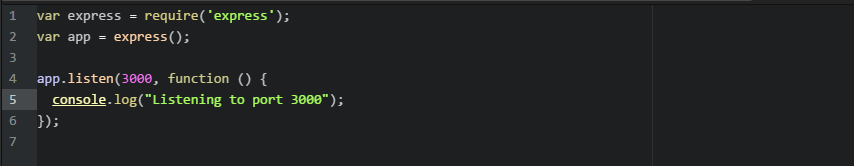


# Express.js

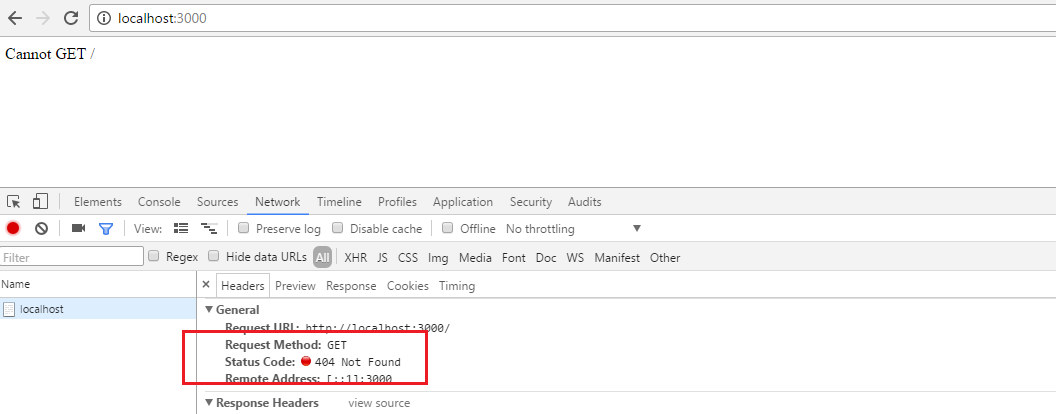
* Express is node.js framework and its popularly used one
* Its built in module in node package
* Need to add to project as **npm install express –save** which will install all the express js dependencies and can be observed n package.json

HelloWorld in express

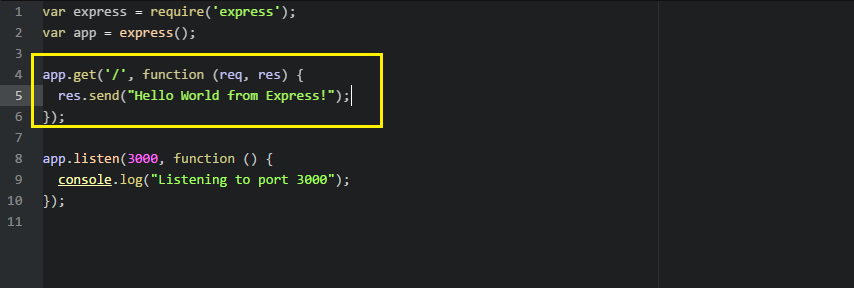
* **Var express = require(‘express’)** will return us a function and we can use that with **var app = express();**
* App.listen(3000, callback) will start function and once we see that on browser it will give Cannot GET error

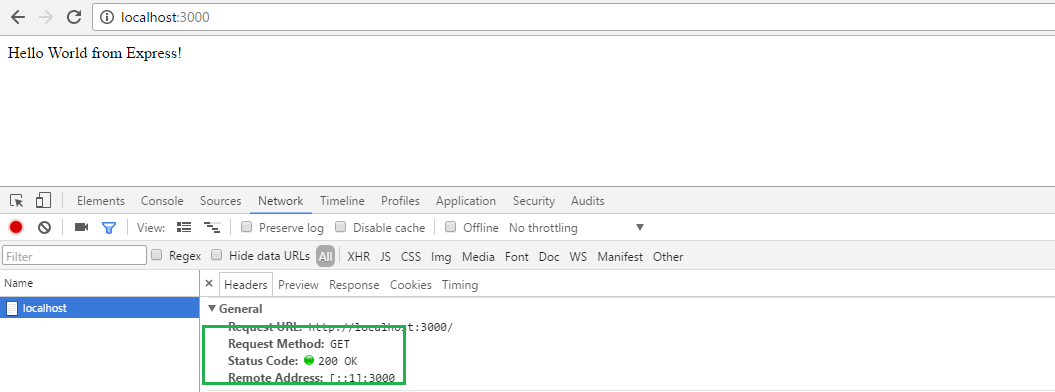


* This will output in browser as



* Need to add app.get() to GET requested URL and corresponding response



* This will output as 

# ExpressJS serving static files

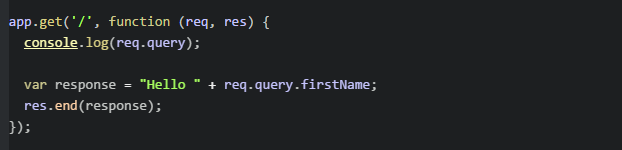
* Used to access particular file from folder by defining its path
* Improved security as all the other folders except for assets are secured or not exposed , this can be achieved as



* If we console.log(\_\_dirname) it gives path till current working directory,
* so in this case even if the assets folder is not accessible from outside, the css file inside assets can be accessed on /cssFile/someCSSFile.css where in someCSSFile.css resides in assets folder
* and the content of someCSSFIle.css can be seen on browser as [**http://localhost:3000/cssFile/someCSSFile.css**](http://localhost:3000/cssFile/someCSSFile.css)

# ExpressJS GET method

* GET is http method to data
* **req.query** is used to get query params in this case since the url is <http://localhost:3000/?firstName=aaa> **req.query** will return json object as **{firstName: aaa}**
* when we **end** or send response to html page as **Hello + req.query.firstName** this would output as **Hello aaa**

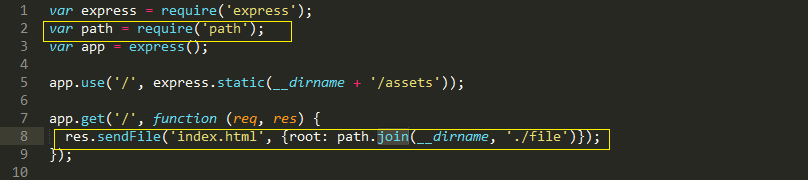


* this shall output as



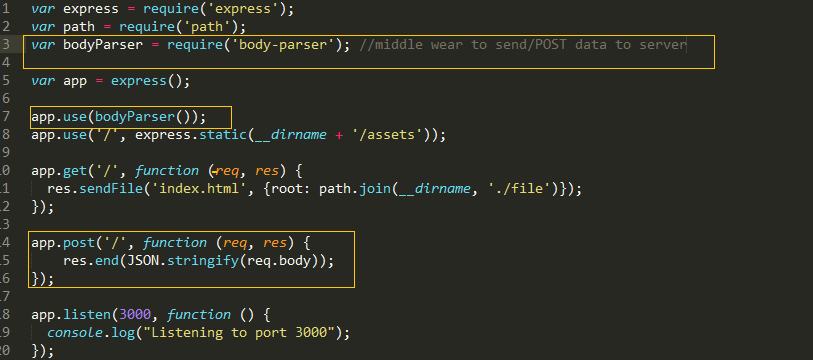
# GET index.html page

* Need to add path module as var path = require(‘path’)
* Create index.html page inside folder say (in this eg inside file folder)
* On app.get res will send index.html file to server via res.sendFile with the location where it resides



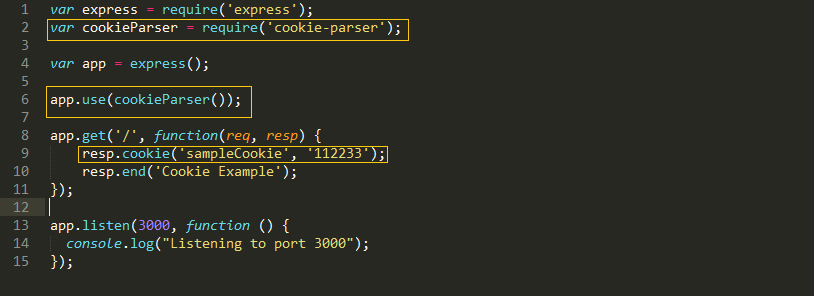
# Express POST method

* http POST is used to send data to server
* **app.post** is used to accomplish this
* we need **body-parser as middleware** to implement POST
* **body-parser:** body parser is used to extract entire body portion of an input stream and makes it available or exposes the same on req.body
* **req.body** will have all the form information that is to be send to server , we make it in readable json format by enclosing in **JSON.stringify(req.body)**
* index.html has the form which data we need to send

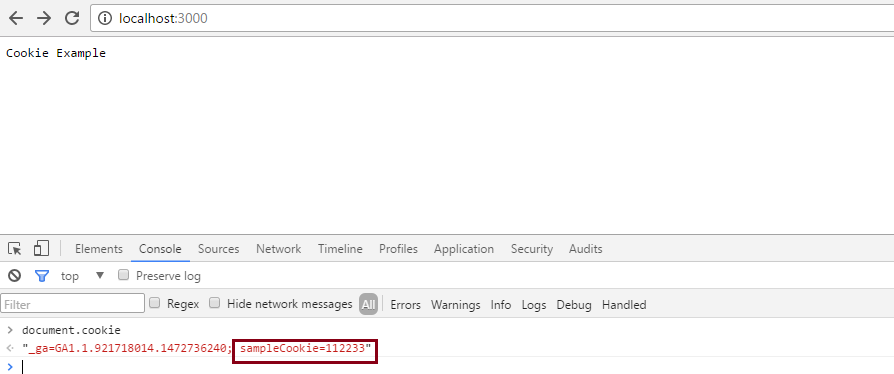


# Cookies

* used to store data in cookies
* A [**cookie**](http://www.webopedia.com/TERM/C/cookie.html) is the term given to describe a type of message that is given to a [Web browser](http://www.webopedia.com/TERM/B/browser.html) by a [Web server](http://www.webopedia.com/TERM/W/Web_server.html).  The main purpose of a cookie is **to identify users** and possibly **prepare customized Web pages or to save site login information** for you.
* We need to include **cookie parser** to achieve this
* **resp.cookie('sampleCookie', '112233')** will set sampleCookie with value 112233 which can be verified on browse console **document.cookie**
* in the same way we can remove cookie as **resp.clearCookie('sampleCookie');**

****

* output this as:

****

# Nodemon

* nodemon: node monitor is used in project so that we don’t have to go to command/node prompt every single time we update the server.js file, instead nodemon will take care of restarting server, this will save good amount of time
* once we install nodemon in project, and when we start server instead of writing node server.js write it as **nodemon server.js**

# Routing

* used to implement routing mechanism
* var router = express.Router(); to initialize routing
* <http://localhost:3000/home> will print This is Home page! on browser
* <http://localhost:3000/stat> will print This is stat page! on browser

