If you pushed your code to github and if you delete git so if you want to push again other code to that cuurrent repo so firstly clone repo then push your add by as usual.

<https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/>

<https://www.geeksforgeeks.org/python-convert-html-pdf/>

<https://pythonexamples.org/python-convert-html-to-pdf/>

<https://www.thepythoncode.com/article/convert-html-to-pdf-in-python>

<https://www.techiediaries.com/convert-html-pdf-python/>

<https://www.tutorialspoint.com/how-to-convert-html-to-pdf-using-python>

[jaik01861@gmail.com](mailto:jaik01861@gmail.com)

Intern@2022

EventEmitter is a Node.js class that includes all the objects that are capable of emitting events. These objects contain an eventEmitter.on() function through which more than one function can be attached to the named events that are emitted by the object. Whenever an EventEmitter object throws an event, all the attached functions to that specific event are invoked synchronously

Node.js uses a single threaded model in order to support async processing. With async processing, an application can perform better and is more scalable under web loads. Thus, Node.js makes use of a [single-threaded model](https://www.edureka.co/blog/nodejs-tutorial/#nodearchitecture) approach rather than typical thread-based implementation.

The [**package.json file**](https://www.edureka.co/blog/nodejs-tutorial/#json) in Node.js is the heart of the entire application. It is basically the manifest file that contains the metadata of the project where we define the properties of a package.

An event loop in Node.js handles all the asynchronous callbacks in an application. It is one of the most important aspects of Node.js and the reason behind Node.js have non-blocking I/O. Since Node.js is an event-driven language, you can easily attach a listener to an event and then when the event occurs the callback will be executed by the specific listener. Whenever functions like setTimeout, http.get, and fs.readFile are called, Node.js executed the event loop and then proceeds with the further code without waiting for the output. Once the entire operation is finished, Node.js receives the output and then executes the callback function. This is why all the callback functions are placed in a queue in a loop. Once the response is received, they are executed one by one.

 node-googlesheet@coherent-bay-336012.iam.gserviceaccount.com

1BM9jF\_W1MWQtiY50-8o6UX1cSXVeMvjjj6vHmRwKvOU/edit#gid=0

https://docs.google.com/spreadsheets/d/1BM9jF\_W1MWQtiY50-8o6UX1cSXVeMvjjj6vHmRwKvOU/edit#gid=0

In general, middleware is a function receives the Request and Response objects. In other words, in an application’s request-response cycle these functions have access to various request &  response objects along with the next function of the cycle. The next function of middleware is represented with the help of a variable, usually named next. Most commonly performed tasks by the middleware functions are:

* Execute any type of code
* Update or modify the request and the response objects
* Finish the request-response cycle
* Invoke the next middleware in the stack

In Node.js, the control flow function is basically the code that is executed between the asynchronous function calls. Below are the steps that must be followed for executing it:

1. Firstly, the order of execution must be controlled.
2. Then, the required data need to be collected.
3. Next, the concurrency must be limited.
4. Once done, the next step of the program has to be invoked.

Javascript Interview Questions

1. Find 2nd larget number in array in javascript

2.