**NODE JS**

**Basics**

Core Modules from node js:

http ->Launch a server, send requests

https-> Launch SSL server

fs, path, os

require -> keyword of node js to import a file in node js.

./ looks for local file, plain text looks for node global module

Acces methods from the object required through ob.something

createServer takes a request listener takes a callback, that receives a request and a response. That function inside will be called whenever a request is sent to the server.

IF a request comes, please use the function. If x happens, then do y.

The create server method returns a server, me have to store it in a server constant. We have to make listen the server, it makes it keep listening. Takes the port parameter input (Eg.: 3000).There is an ongoing looping process.

Process.exit quits the loop and the server doesn t listen for incoming requests anymore.

The callback function form createServer takes this req and res parameters that can then me used to acces url, method, or headers: req.url,req.method,req.headers.

This same callback has res, and this can be changed on every request: res.setHeader(“content-type”,”text/html”). This tells the type of the content is html. There is a set of supported headers that the browser understand. Res.write(“<html code>”)

When we end writing, we just call res.end();

Route requests:

Just control flow if req.url===”/”, and return res.end(); to quit the function execution after writing the response. If not, the other block inside the global scope of the function will be called.

Another if to take url “/message” and req.method==”POST”.

When handling the POST request, have to change res.statusCode =302;, and change header res.setHeader(“location”,”/”) to set meta information of the response. Location is a default header understood by the browser, and we pass also the url location to which the response will redirect, “/”.

Const body = [];

Req.on(“data”, (chunk)=>{

Body.push(chunk);

}

🡪Listen to event, whenever any data comes in as a request, a callback is used. The callback receives a chunk of data.

To register another event listener, we add

Req.on(“end”,()=>{

Const parsedBody = Buffer.concat(body).toString();

Const parseBody=Buffer.concat(body).toString();

Console.log(parsedBody);

}

🡪Callback is called when other incoming requests finish, in this case when the parse is done (the function that parses the data received in the req). To interact with all this chunks we need to buffer, the bus stop to interact with it. We use .toString(); method from buffer

Node js registes functions internally, and when a request comes, does functions are called. Same with forms, callbacks are set to be called after a request is sent. When it is sent, my node code choses which route this should go. We don’t want to block the event execution for a long time.

We assign whatever is exported in the file required into the variable routes. The route variable will hold the function exported. If multiple functions are called, an object is assigned into routes.