NODE JS

Aruna S
Department of
Computer Science and Engineering

NODE JS

HTTP Module

S. Aruna
Department of Computer Science and Engineering

Web Server

- A Web Server is a software application which handles HTTP requests sent by the HTTP client, like web browsers, and returns web pages in response to the clients.
- Web servers usually deliver html documents along with images, style sheets, and scripts.
- Most of the web servers support server-side scripts, using scripting languages or redirecting the task to an application server which retrieves data from a database and performs complex logic and then sends a result to the HTTP client through the Web server.
- Apache web server is one of the most commonly used web servers. It is an open source project.

Web Application Architecture

A Web application is usually divided into four layers −

Client − This layer consists of web browsers, mobile browsers or applications which can make HTTP requests to the web server.

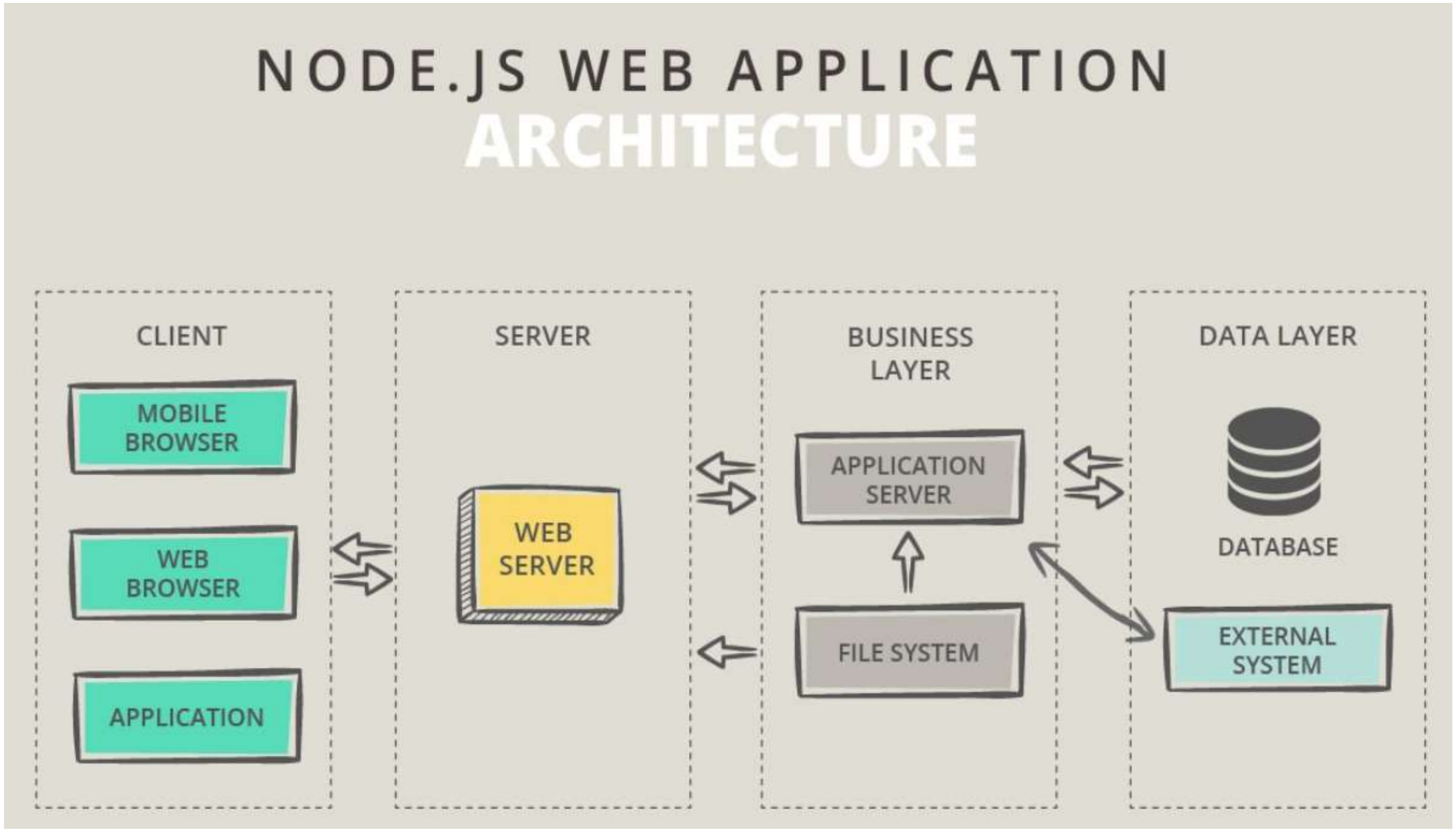
Server − This layer has the Web server which can intercept the requests made by the clients and pass them the response.

Business − This layer contains the application server which is utilized by the web server to do the required processing. This layer interacts with the data layer via the database or some external programs.

Data − This layer contains the databases or any other source of data.

NODE JS

Web Application Architecture



HTTP Module – Web Server using Node

The components of a Node.js application.

Import required modules – We use the **require** directive to load Node.js modules.

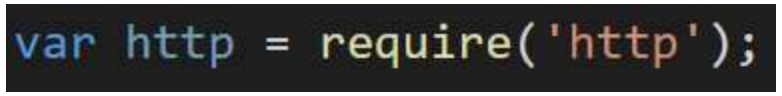
Create server – A server which will listen to client's requests similar to Apache HTTP Server. Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).

Read request and return response – The server created in an earlier step will read the HTTP request made by the client which can be a browser or a console and return the response.

Creating Node.js Application

Step 1 - Import Required Module

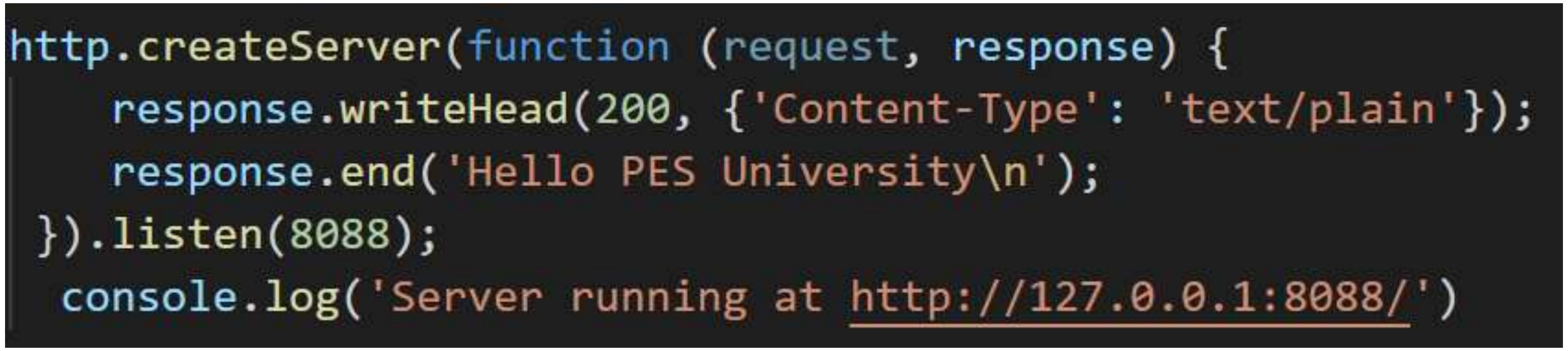
We use the **require** directive to load the http module and store the returned HTTP instance into an http variable as follows



```
var http = require('http');
```

Step 2 - Create Server

- We use the created http instance and call **http.createServer()** method to create a server instance.
- Then we bind it at port 8088 using the **listen** method associated with the server instance.



```
http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/plain'});
    response.end('Hello PES University\n');
 }).listen(8088);
  console.log('Server running at http://127.0.0.1:8088/')
```

Step 3 - Testing Request & Response

```
$node app.js
```

Verify the Output. Server has started.

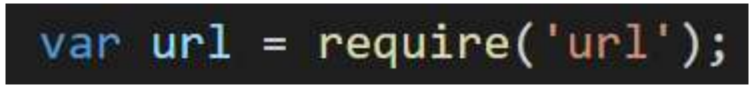
Server running at http://127.0.0.1:8088/

NODE JS

URL Module

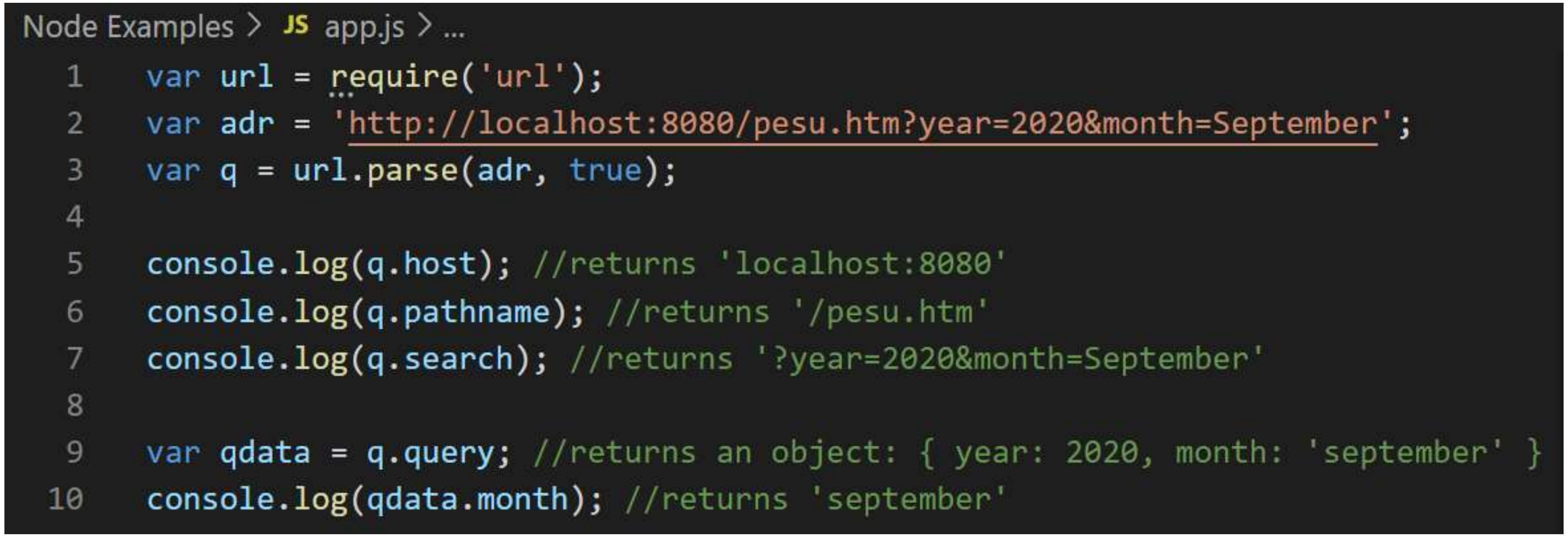
The URL module splits up a web address into readable parts.

To include the URL module, use the require() method:



```
var url = require('url');
```

Parse an address with the url.parse() method, and it will return a URL object with each part of the address as properties:

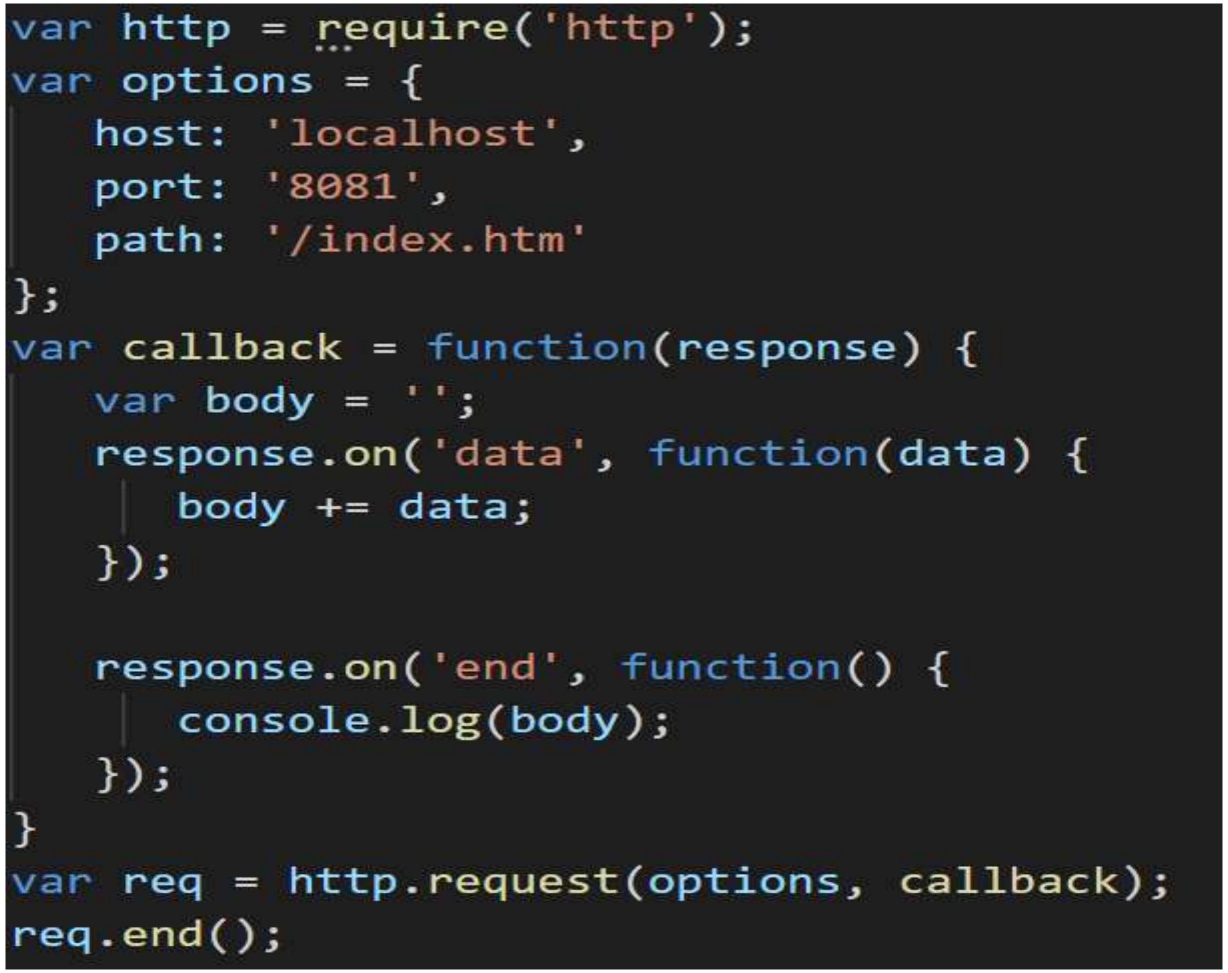


```
var url = require('url');
var adr = 'http://localhost:8080/pesu.htm?year=2020&month=September';
var q = url.parse(adr, true);

console.log(q.host); //returns 'localhost:8080'
console.log(q.pathname); //returns '/pesu.htm'
console.log(q.search); //returns '?year=2020&month=September'

var qdata = q.query; //returns an object: { year: 2020, month: 'september' }
console.log(qdata.month); //returns 'september'
```

NODE JS

Web Client Using Node

A web client can be created using **http** module.



```
var http = require('http');
var options = {
   host: 'localhost',
   port: '8081',
   path: '/index.htm'
};
var callback = function(response) {
   var body = '';
   response.on('data', function(data) {
      body += data;
   });

   response.on('end', function() {
      console.log(body);
   });
}
var req = http.request(options, callback);
req.end();
```

THANK YOU

Aruna S
Department of
Computer Science and Engineering
arunas@pes.edu