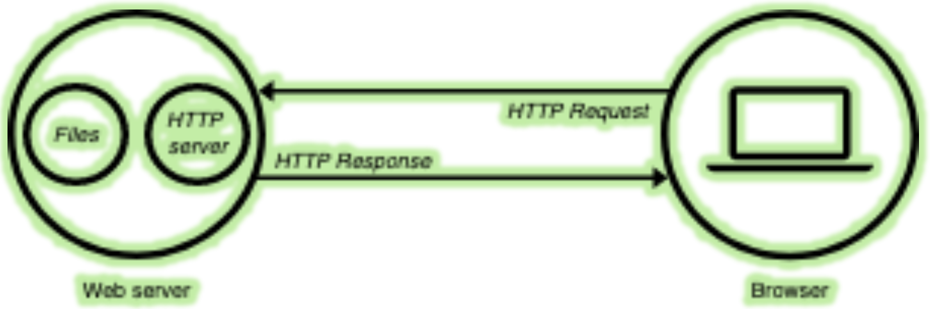
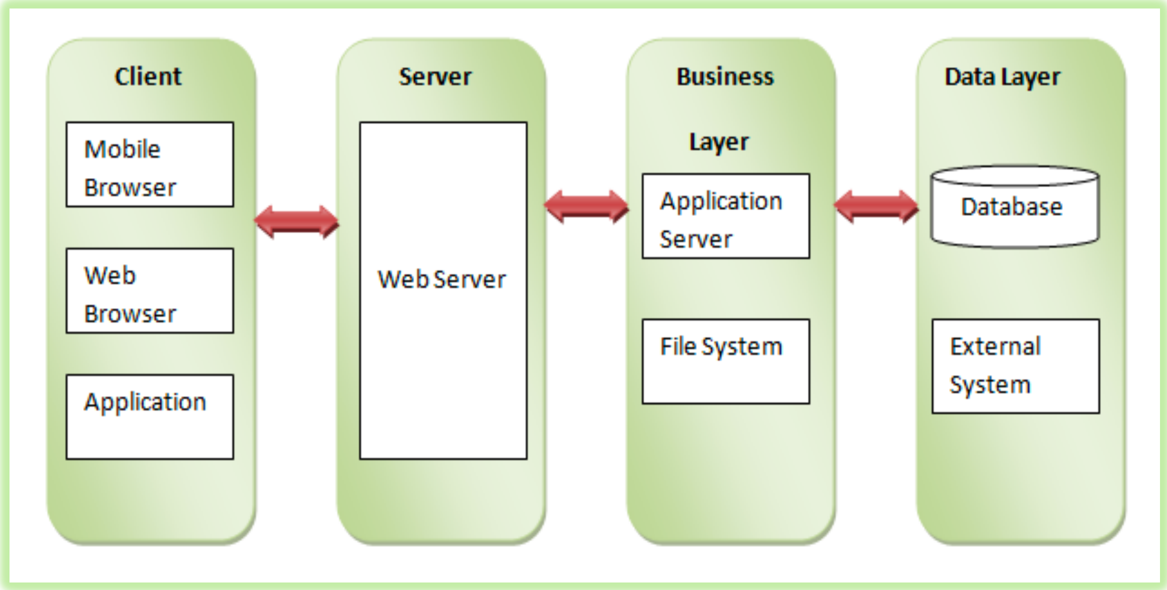
COMP3123 - Full Stack Development I

Web Server ?

- To access web pages of any web application, you need a web server.
- The web server will handle all the http requests for the web application.
- E.g. IIS is a web server for ASP.NET web applications and Apache is a web server for PHP or Java web applications.
- Node.js provides capabilities to create your own web server which will handle HTTP requests asynchronously.





Node as a Web Server

- The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.
- Use the createServer() method to create an HTTP server:

```
var http = require('http');
//create a server object:
http.createServer(function (req, res) {
res.write('Hello World!'); //write a response to the client
res.end(); //end the response
}).listen(8080); //the server object listens on port 8080
```

- The function passed into the http.createServer() method, will be executed when someone tries to access the computer on port 8080.
- Save the code above in a file called "demo_http.js", and initiate the file:
- Initiate demo_http.js:

```
node demo_http.js
```

Web reference Link

Add an HTTP Header

- If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type:

```
var http = require('http');
http.createServer(function (req, res) {
 // add a HTTP header:
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Hello World!');
  res.end();
}).listen(8080);
```

Read the Query String

The function passed into the http.createServer() has a req argument that represents the request from the client, as an object (http.IncomingMessage object).

This object has a property called "url" which holds the part of the url that comes after the domain name:

demo_http_url.js

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(req.url);
  res.end();
}).listen(8080);
```

Node.js File System Module

- The Node.js file system module allows you to work with the file system on your computer.
- To include the File System module, use the require() method:

    var fs = require('fs');

<u>Common use for the File System module:</u>

- Read files
- Create files
- Update files
- Delete files
- Rename files

- The ***fs.readFile()*** method is used to read files on your computer.
- The File System module has methods for creating new files:

    fs.appendFile()

    fs.open()

    fs.writeFile()

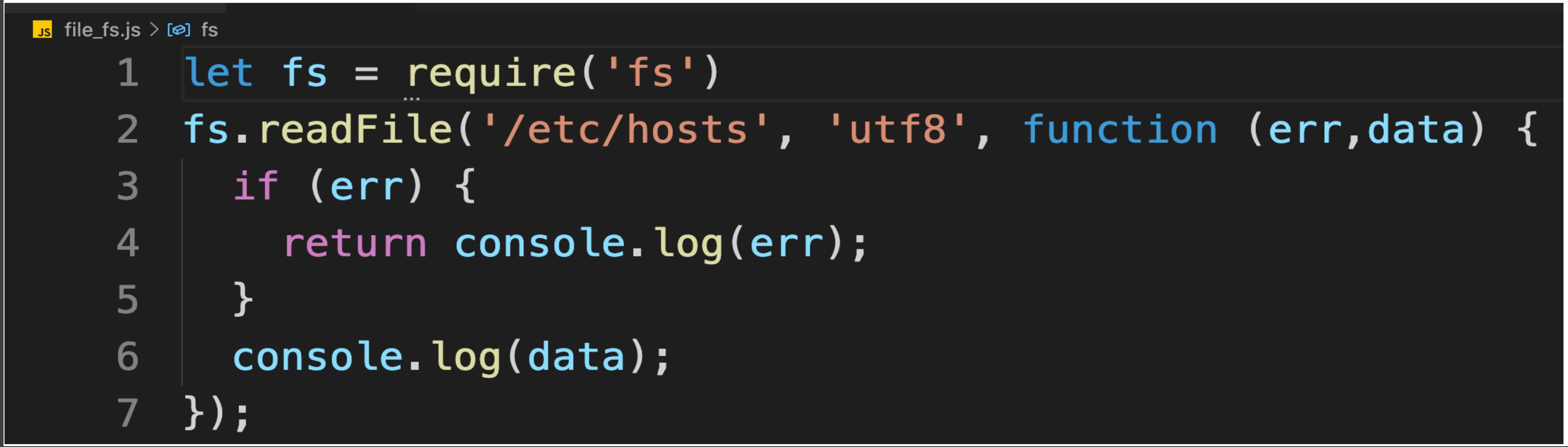
- The fs.appendFile() method appends specified content to a file. If the file does not exist, the file will be created:

Node.js File System Module

- The File System module has methods for updating files:

  fs.appendFile()

  fs.writeFile()
- The fs.appendFile() method appends the specified content at the end of the specified file.
- To delete a file with the File System module, use the ***fs.unlink()*** method.
- To rename a file with the File System module, use the ***fs.rename()*** method.

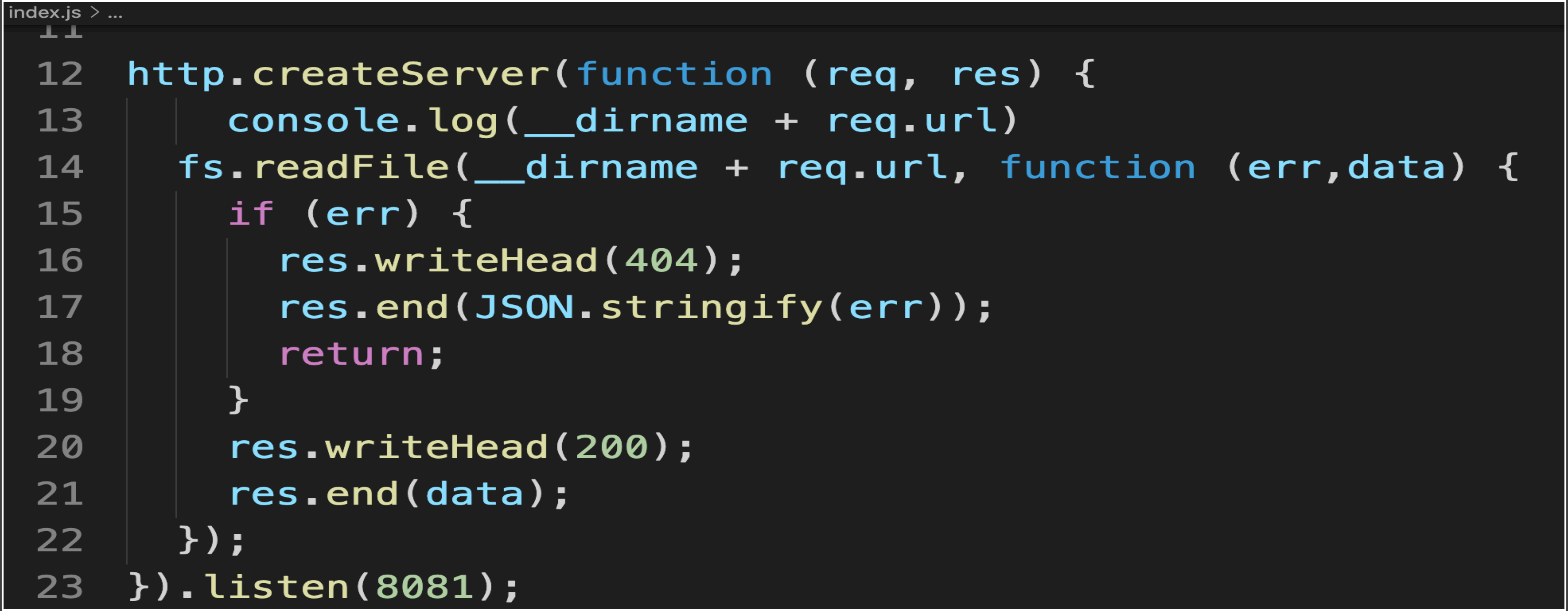

file_fs.js > fs

```
let fs = require('fs')
fs.readFile('/etc/hosts', 'utf8', function (err,data) {
  if (err) {
    return console.log(err);
  }
  console.log(data);
});
```

Web Reference Link

Serving Static Files

- A basic necessity for most http servers is to be able to serve static files.



index.js > ...

```
http.createServer(function (req, res) {
    console.log(__dirname + req.url)
  fs.readFile(__dirname + req.url, function (err,data) {
    if (err) {
      res.writeHead(404);
      res.end(JSON.stringify(err));
      return;
    }
    res.writeHead(200);
    res.end(data);
  });
}).listen(8081);
```

Thank You