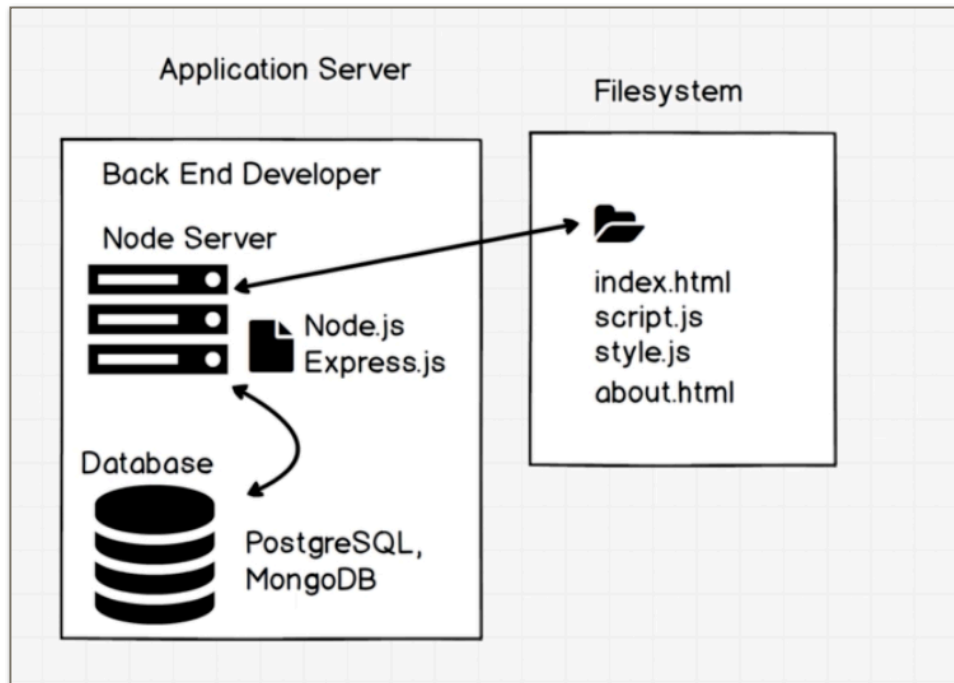


Node.js



Check node version in terminal

```
node -v
```

Use node in terminal

```
node
```

No window object, instead we have `global` and `process`

```
> window
```

```
ReferenceError: window is not defined
```

Exit node and return to terminal

```
> process.exit()
```

(`Ctrl-c` for general exit)

Create file from terminal

```
touch script.js
```

Run file from terminal using node (it will exit automatically)

```
node script.js
```

Example:

`setTimeout` to wait a few seconds before execution

`__dirname` to get name of current path

script.js

```
const a = 4;
const b = 5;

setTimeout(() =>{
  console.log(a+b);
}, 3000) // wait 3 seconds

console.log(__dirname);
```

Run in terminal

```
~/git/node:master$ node script.js
```

```
/Users/stepha/git/node
```

```
9 <<<< waits 3 seconds to show up
```

Exports & Imports between Modules

Each file is called a module. Must export and import to access data from other modules.

```
script2.js  
const largeNumber = 356;  
  
// export from this module  
module.exports = {  
  largeNumber: largeNumber  
};  
// end of export
```

```
script.js  
// import from other module  
const script2 = require('./script2.js')  
// NOTE: const script2 can have any name  
// end of import  
  
const a = script2.largeNumber;  
const b = 5;  
  
console.log(a+b);
```

Types of Modules

Three types of modules:

1. Modules you create yourself
2. Built - in modules, like ' fs ' and ' http '
3. Packages from npm

' fs ' module - allows you to read the file system

```
const c = require('fs'); // import statement  
console.log(c);
```

Use with these functions:

```
const c = require('fs').readFile; // import
```

```
const c = require('fs').readFileSync; // import
```

' http ' module - used to build server

```
const c = require('http'); // import statement  
console.log(c);
```

nodemon - npm package used so you don't have to refresh while developing modules

Run from terminal to generate package.json file (-y skips questions) in the node folder...

```
npm init -y
```

Install 'nodemon' and add it to devDependencies in package.json file:

```
npm install nodemon --save-dev
```

Note: devDependencies are only used while developing, not released with final app or server.

We can use nodemon because it is in:

```
node_modules > .bin > nodemon
```

To use it, go to the package.json file and find the scripts section:

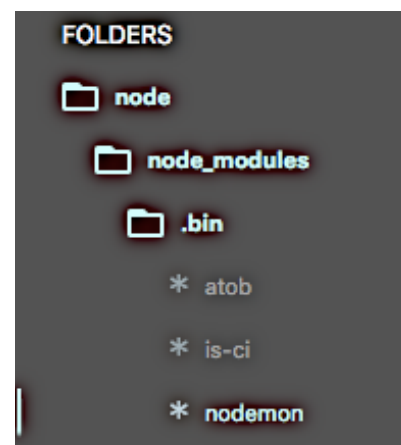
```
"scripts": {  
  "test": "echo \"Error: no test specified\" &&  
exit 1"  
},
```

Change that to this:

```
"scripts": {  
  "start": "nodemon"  
},
```

Go to terminal and run:

```
npm start
```



We have a clean exit but see that something is still running... nodemon. As we develop, nodemon will listen for changes and automatically output them so we don't have to keep running the "node script.js" command. **Ctrl-c** to exit

Building a Server

From terminal, create new file:

```
touch server.js
```

server.js

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

const server = http.createServer(() => {
  console.log('I hear you, thnks for the request')
})

server.listen(3000); // port number can be anything
```

In the package.json file, edit the scripts section:

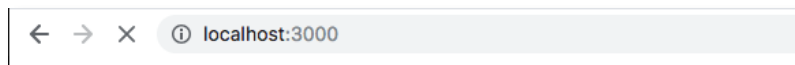
```
"scripts": {
  "start": "nodemon server.js"
},
```

Run from terminal:

```
node server.js
```

It looks like its just hanging there.

Let's go to port 3000 on our local host (my machine). Go to browser, navigate to <http://localhost:3000/>



Every time you connect to port 3000 (includes refreshing browser), this pops up in terminal:

```
I hear you, thnks for the request
```

This is happening because `server.js` is running in node, listening for connections to port 3000. However, the server is not responding to the request with anything.



This page isn't working

localhost didn't send any data.

ERR_EMPTY_RESPONSE

`Ctrl-c` to exit.

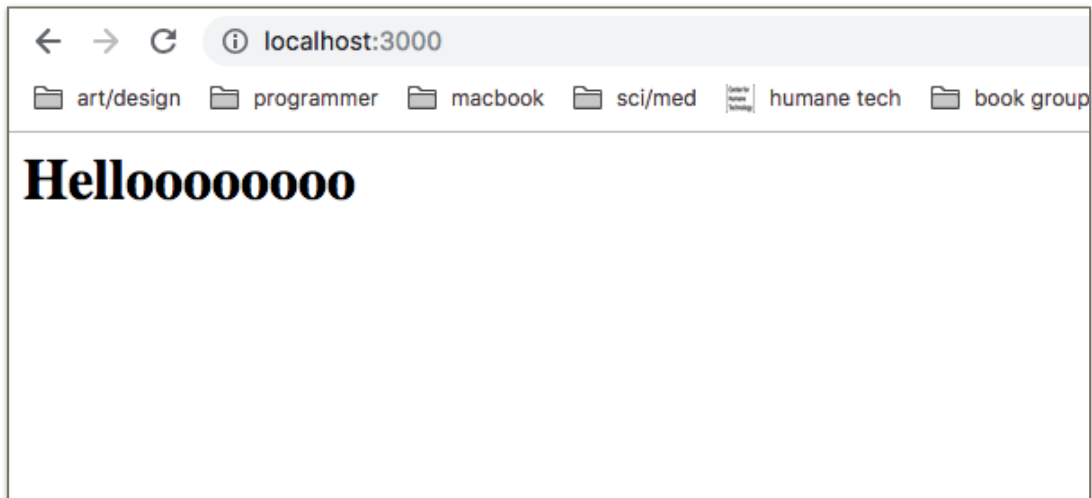
Let's edit our server file:

```
server.js  
  
const http = require('http');  
  
const server = http.createServer((request, response) => {  
  response.setHeader('Content-Type', 'text/html');  
  response.end('<h1>Helloooooooooo</h1>');  
})  
  
server.listen(3000); // port number can be anything
```

`npm start`

To track changes with **nodemon**, run `npm start` in terminal.
(`package.json` file set up to watch `server.js` in previous step)

Refresh in browser to get:



In Network tab of dev tools:

Name	Status	Type
localhost	200	document

localhost

favicon.ico

General

Request URL: http://localhost:3000/
Request Method: GET
Status Code: 200 OK
Remote Address: [::1]:3000
Referrer Policy: no-referrer-when-downgrade

Response Headers

view source

Connection: keep-alive
Content-Length: 21
Content-Type: text/html
Date: Sun, 28 Apr 2019 17:34:13 GMT



localhost

favicon.ico

1

Hellooooooooo

Can get information about server requests...

```
server.js
const http = require('http');

const server = http.createServer((request, response) => {

  console.log('I hear you, thnks for the request')

  // request
  console.log('Headers:', request.headers)
  console.log('Method :', request.method)
  console.log('Url    :', request.url)

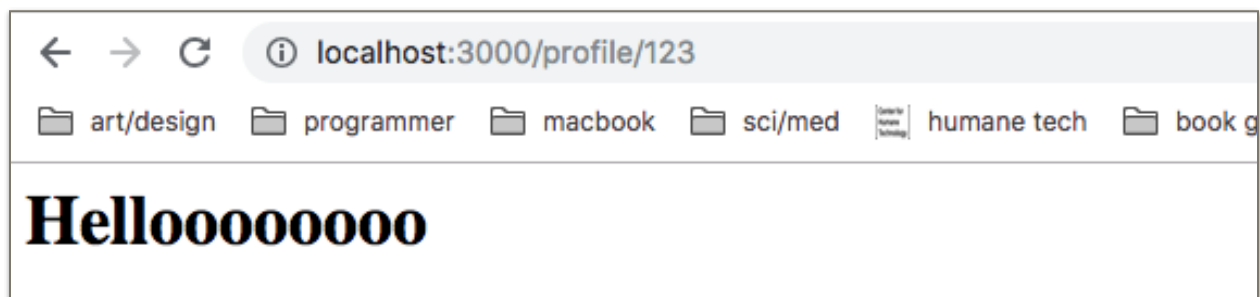
  // response
  response.setHeader('Content-Type', 'text/html');
  response.end('<h1>Helloooooooooo</h1>');

})

server.listen(3000); // port number can be anything
```

Now whenever we connect to the port, information about the request (headers, method, url) appears in the terminal.

If I connect to <http://localhost:3000/profile/123>



the terminal shows:

```
Url    : /profile/123
```

Instead of

```
Url    : /
```

Using JSON for responses

In the previous example, the **Content-Type** of our response is **'text/html'**. JSON is great for AJAX requests and responses, so let's do that.

```
server.js

const http = require('http');

const server = http.createServer((request, response) => {

  console.log('I hear you, thnks for the request')

  // request
  console.log('Headers:', request.headers)
  console.log('Method:', request.method)
  console.log('Url:', request.url)

  // response

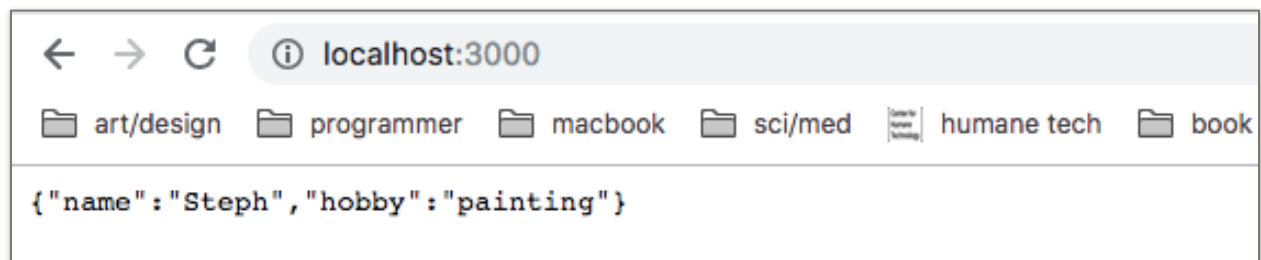
  const user1 = { // javascript object
    name: 'Steph',
    hobby: 'painting'
  }

  response.setHeader('Content-Type', 'application/json');
  response.end(JSON.stringify(user1)); // convert object
to JSON so we can send it over

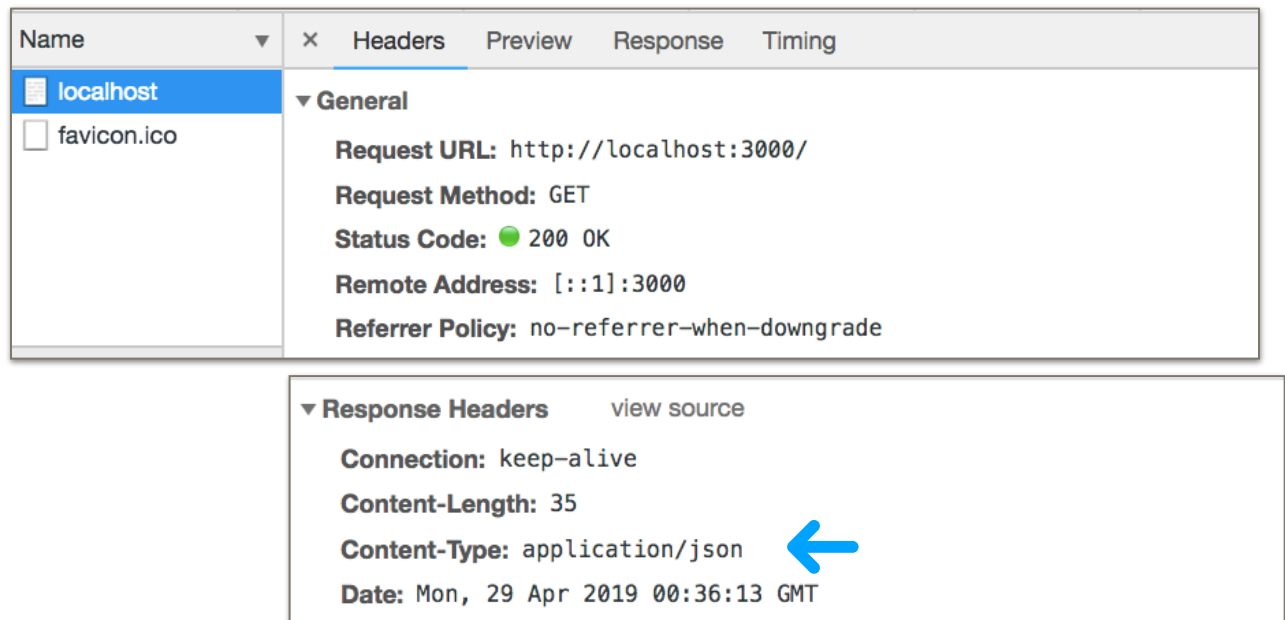
})

server.listen(3000); // port number can be anything
```

See response in JSON format below:



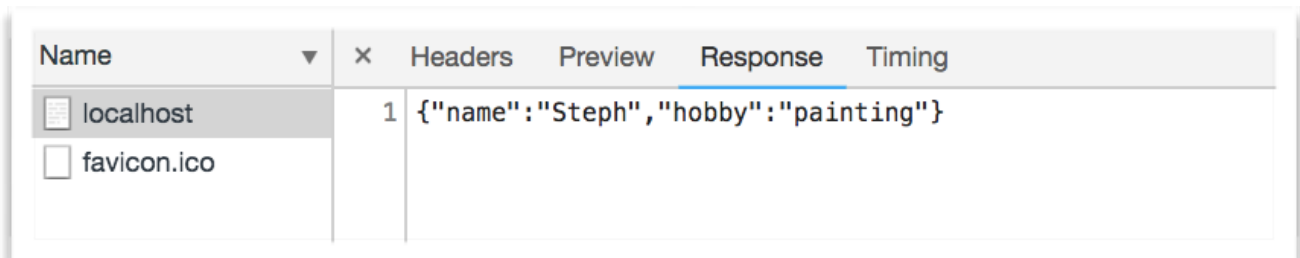
In Network tab of Chrome dev tools, we see content-type is JSON:



The screenshot shows the Chrome DevTools Network tab. The left sidebar lists the network resources: 'localhost' (selected) and 'favicon.ico'. The main panel is divided into two sections. The top section, 'General', displays request details: Request URL: http://localhost:3000/, Request Method: GET, Status Code: 200 OK, Remote Address: [::1]:3000, and Referrer Policy: no-referrer-when-downgrade. The bottom section, 'Response Headers', shows: Connection: keep-alive, Content-Length: 35, Content-Type: application/json (highlighted with a blue arrow), and Date: Mon, 29 Apr 2019 00:36:13 GMT. A 'view source' link is also present next to the Response Headers section.

Name	Headers	Preview	Response	Timing
localhost	General Request URL: http://localhost:3000/ Request Method: GET Status Code: 200 OK Remote Address: [::1]:3000 Referrer Policy: no-referrer-when-downgrade			
favicon.ico	Response Headers view source Connection: keep-alive Content-Length: 35 Content-Type: application/json Date: Mon, 29 Apr 2019 00:36:13 GMT			

Response is in JSON format:



The screenshot shows the Chrome DevTools Network tab with the 'Response' tab selected. The left sidebar shows 'localhost' and 'favicon.ico'. The main panel displays the response body as a JSON object: {"name":"Steph","hobby":"painting"}.

Name	Headers	Preview	Response	Timing
localhost	1		{"name":"Steph","hobby":"painting"}	
favicon.ico				