Node.js

TABLE OF CONTENTS

01

Event Loop

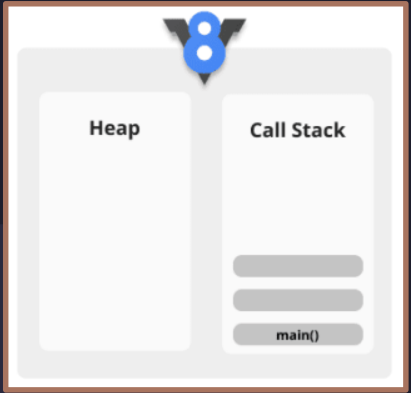
What is Event Loop – ⚙ JavaScript Engine

- A program that executes your JavaScript code.
- A popular example is Google's V8 engine.

What is Event Loop - ⚙ V8 Engine

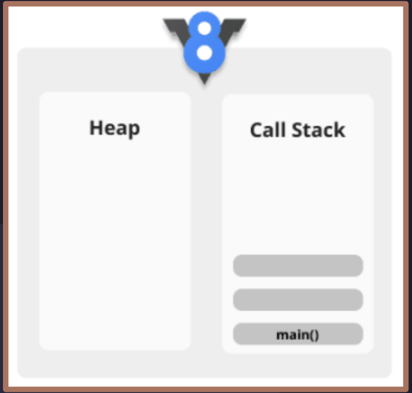
- Open-source.
- High-performance JavaScript engine.
- Written in C++
- V8 engine is used inside Google Chrome, **Node.js**.

What is Event Loop – V8 Engine



- The V8 engine has two main components:
- Heap
  - Unstructured memory that is used for memory allocation of the variables and the objects.
- Call Stack
  - LIFO data structure that is used for function calls that record where we are in the program.

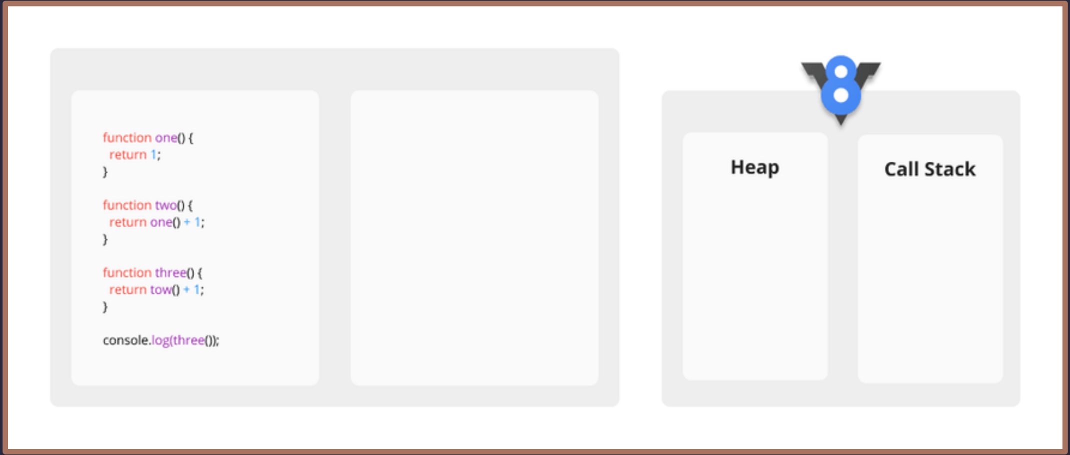
What is Event Loop – Call stack



- JavaScript is a single-threaded programming language.
- Which means it can do one thing at a time.
- And it has one Call Stack.
- Single threaded = single call stack = single task at a time.

What is Event Loop – Call stack

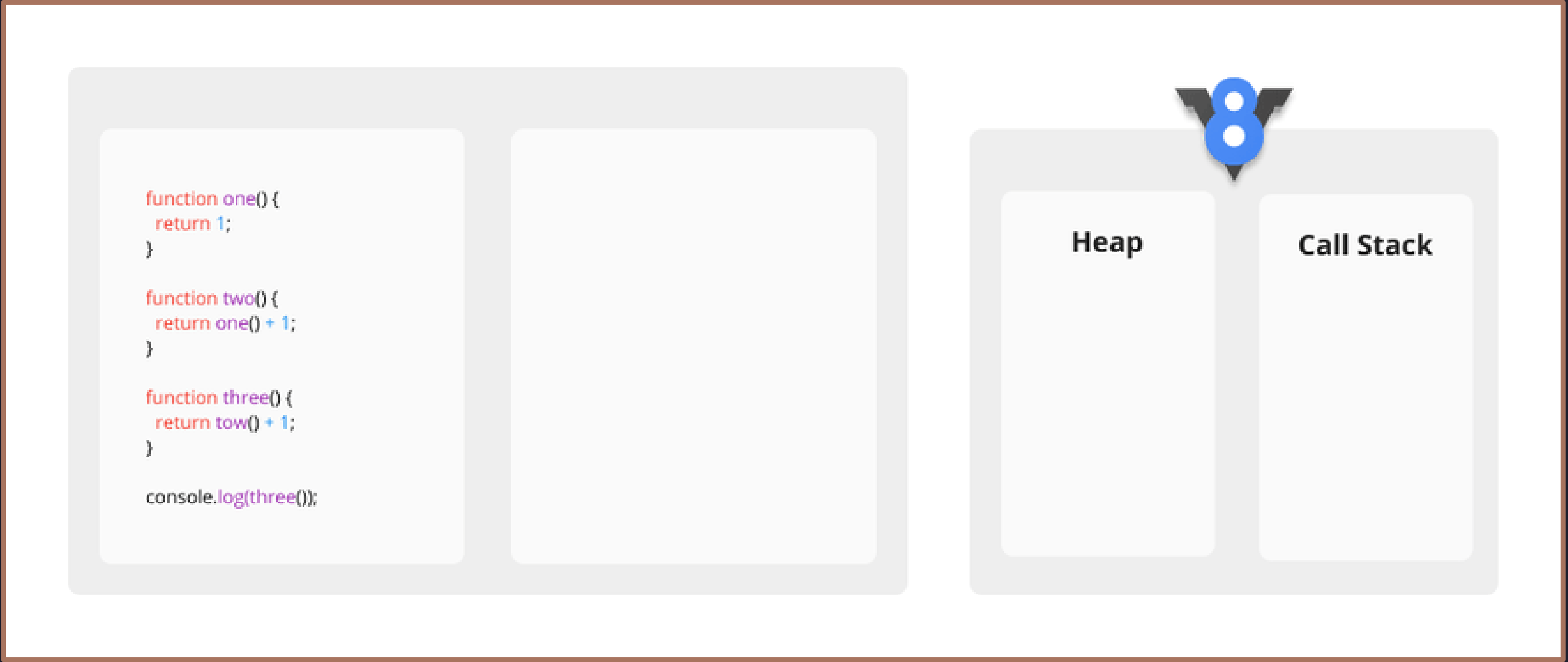
- If you call a function.
- It's pushed on the top of the Call Stack.
- And when the function returns, it's popped from the top of the Call Stack.



```
function one() {
  return 1;
}

function two() {
  return one() + 1;
}

function three() {
  return tow() + 1;
}

console.log(three());
```

Please choose slide show to visualize animation above.

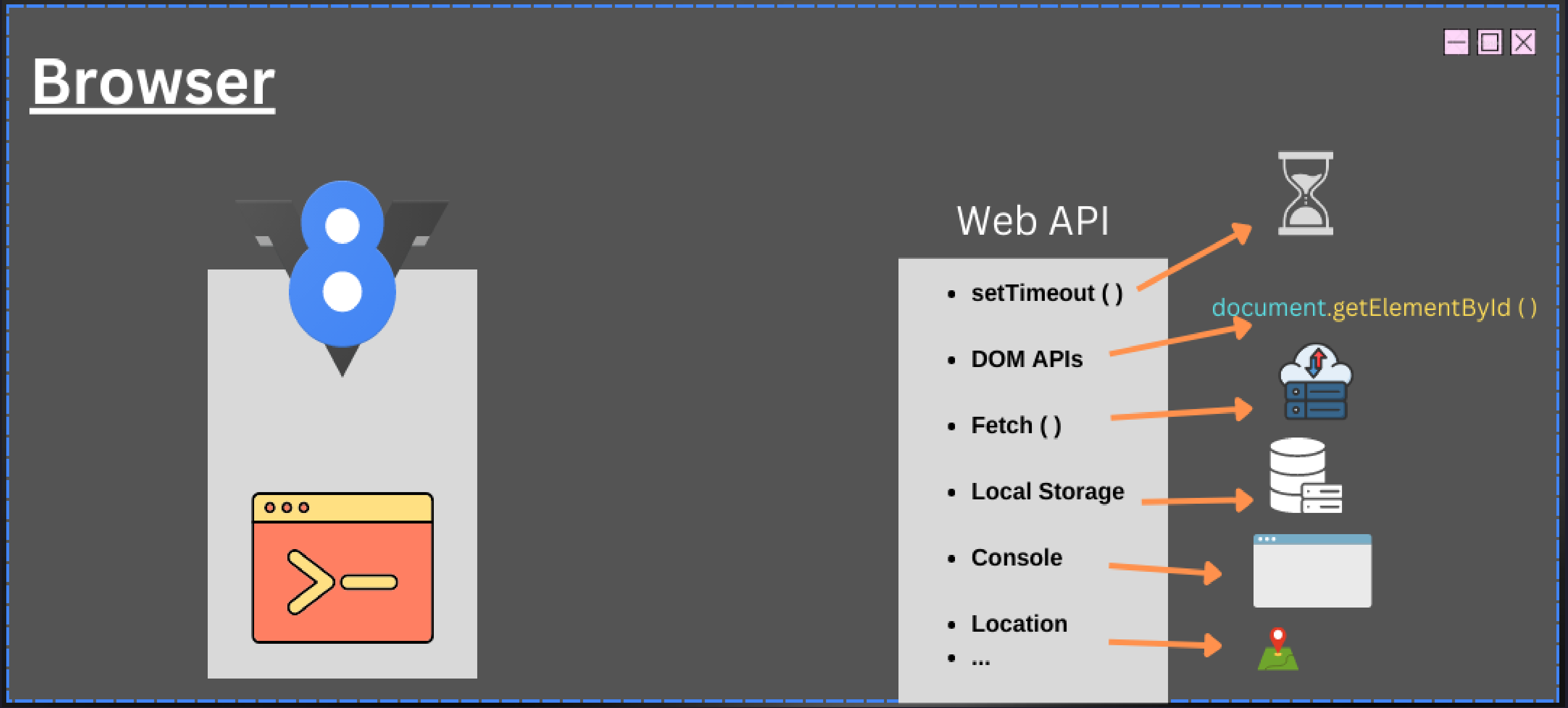
What is Event Loop – Call stack



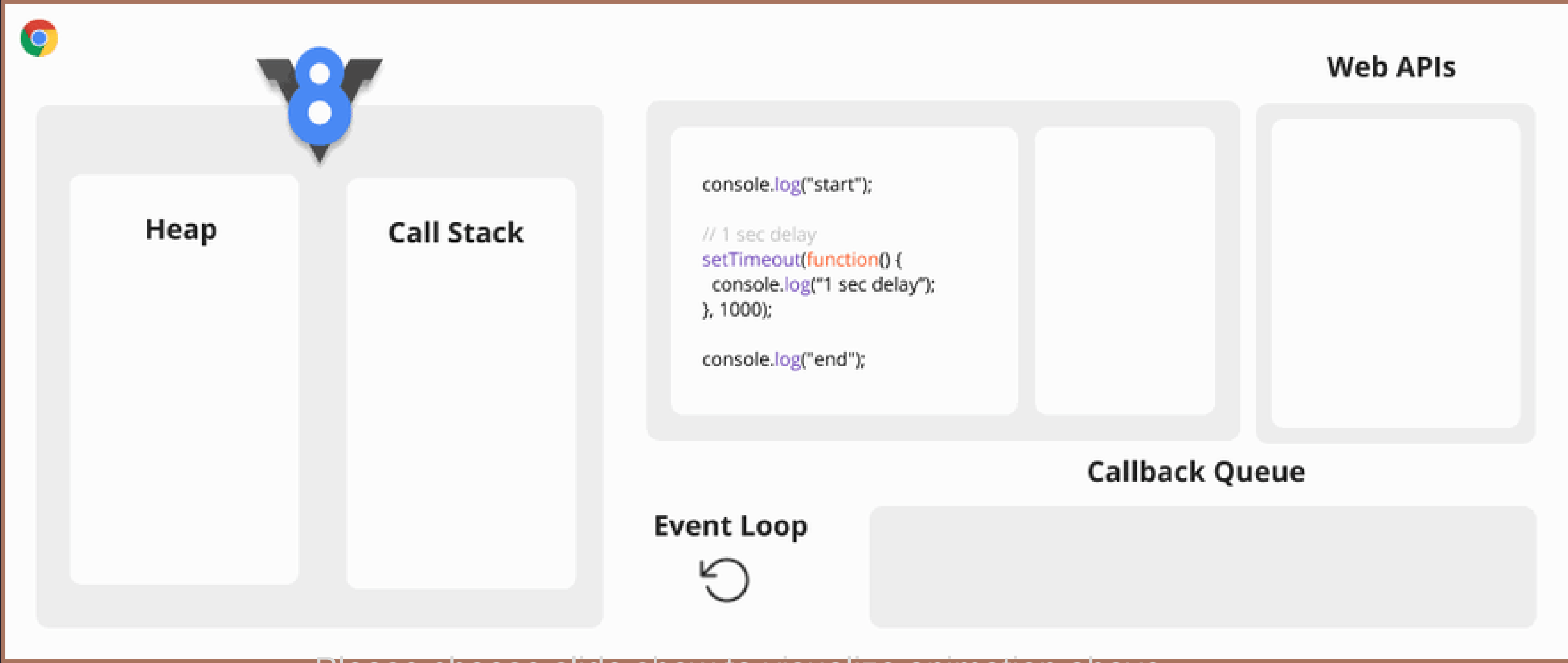
```
function one() {
  return 1;
}

function two() {
  return one() + 1;
}

function three() {
  return tow() + 1;
}

console.log(three());
```

Please choose slide show to visualize animation above.

What is Event Loop – Web API



What is Event Loop – Call back queue



```
console.log("start");

// 1 sec delay
setTimeout(function() {
  console.log("1 sec delay");
}, 1000);

console.log("end");
```

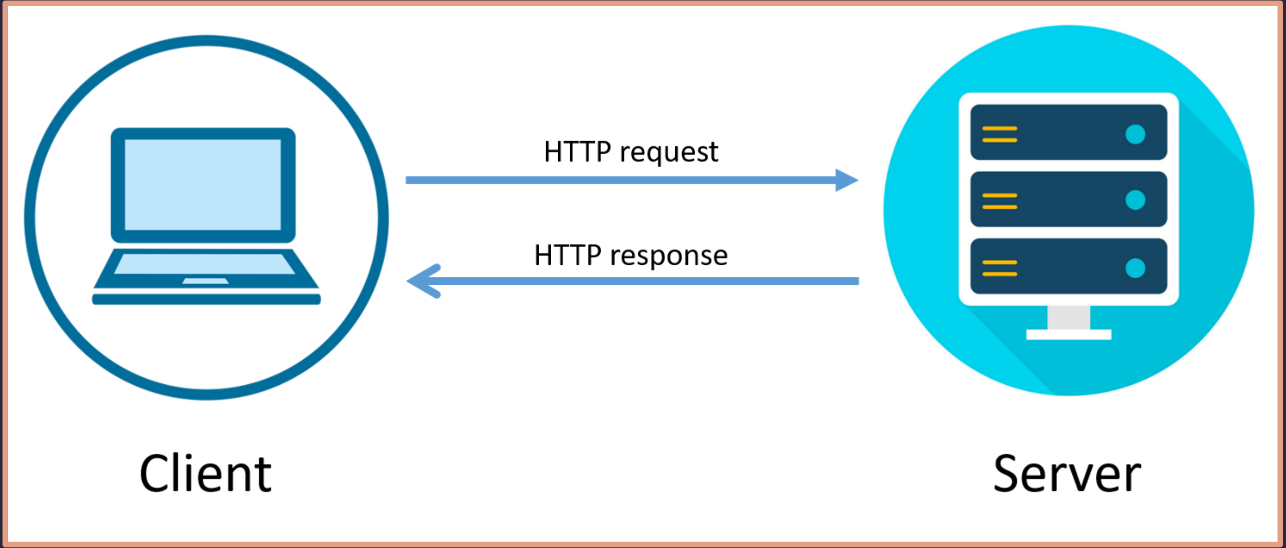
Please choose slide show to visualize animation above.

02

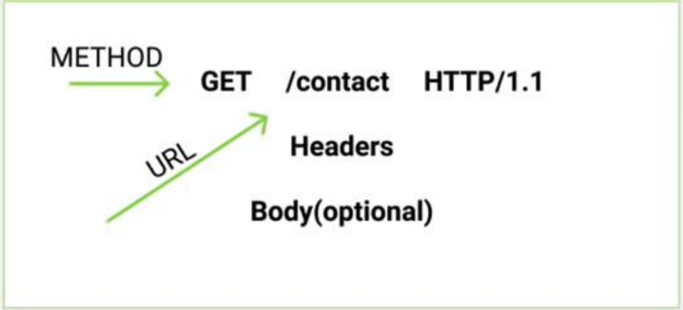
HTTP request and response message

HTTP Request and Response

- Request and Response Cycle



Request Message



Request URL: `https://www.course-api.com/slides/`

Request Method: `GET`

Status Code: 🟢 `200 OK`

Remote Address: `138.68.239.6:443`

Referrer Policy: `strict-origin-when-cross-origin`

Headers

Pragma: `no-cache`

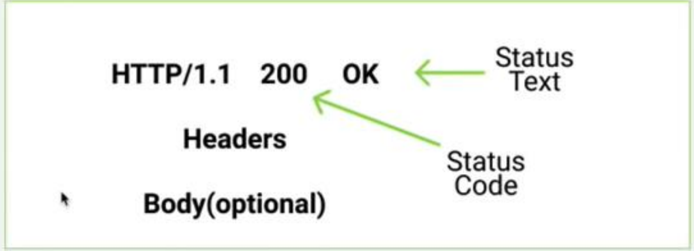
Referer: `https://www.course-api.com/`

Body

▾ **Request Payload** view source

```
▾ {email: "hello@hello.com"}
    email: "hello@hello.com"
```

Response Message



Request URL: `https://serverless-functions-course.netlify.app/api/6-newsletter`

Request Method: `POST`

Status Code: 🔴 `400`

Remote Address: `104.248.78.24:443`

Referrer Policy: `strict-origin-when-cross-origin`

Headers

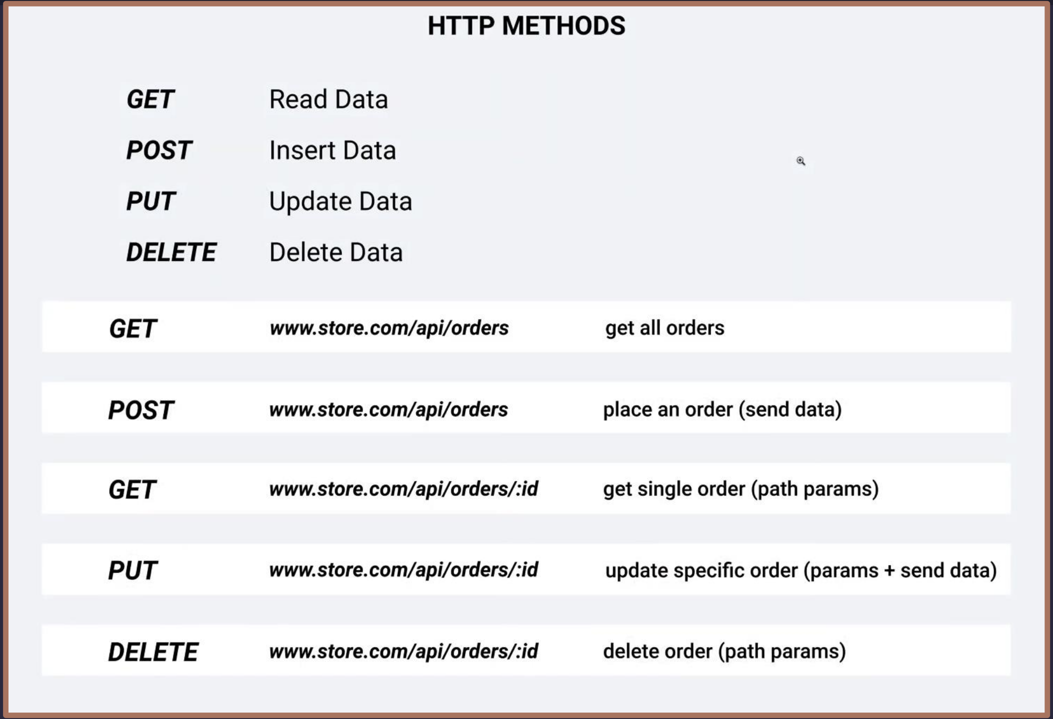
Content-Type: `text/html; charset=UTF-8`

Content-Type: `application/json; charset=utf-8`

Body

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-wi
    <title>Slides</title>
    <link rel="stylesheet" href="./styles.css" />
  </head>
  <body>
```

HTTP Request methods



HTTP METHODS

GET	Read Data
POST	Insert Data
PUT	Update Data
DELETE	Delete Data

GET	***www.store.com/api/orders***	get all orders
POST	***www.store.com/api/orders***	place an order (send data)
GET	***www.store.com/api/orders/:id***	get single order (path params)
PUT	***www.store.com/api/orders/:id***	update specific order (params + send data)
DELETE	***www.store.com/api/orders/:id***	delete order (path params)

03

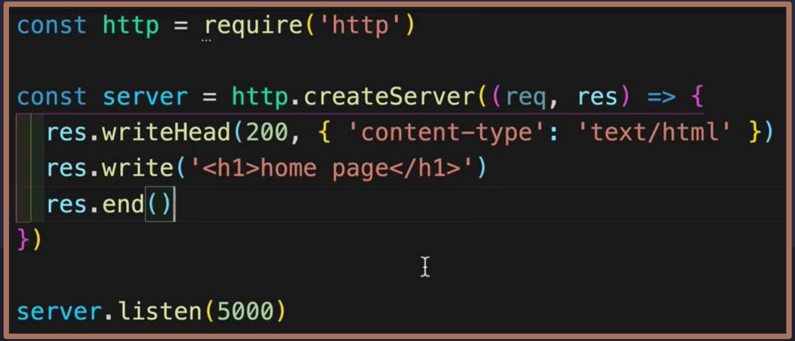
HTTP Basics

HTTP Response message

- res.end( )
    - This signals the server that all the headers and body have been sent.
    - Must be called on each response

HTTP Request and Response - Headers

- res.writeHead(200, { ‘content-type’ : ‘type/html’ })
- HTTP headers are key-value pairs.
- That are included in the header of an HTTP request or response.
- Passes additional context and metadata about the request or response.



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

const server = http.createServer((req, res) => {
    res.writeHead(200, { 'content-type': 'text/html' })
    res.write('<h1>home page</h1>')
    res.end()
})

server.listen(5000)
```

HTTP Response – status code

- Specifies whether a specific HTTP request has been successfully completed.
- Responses are grouped in five classes:
- Visit following link to explore more:
    - https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

<QnA>

Thanks!

Regards: Hassan Bilal