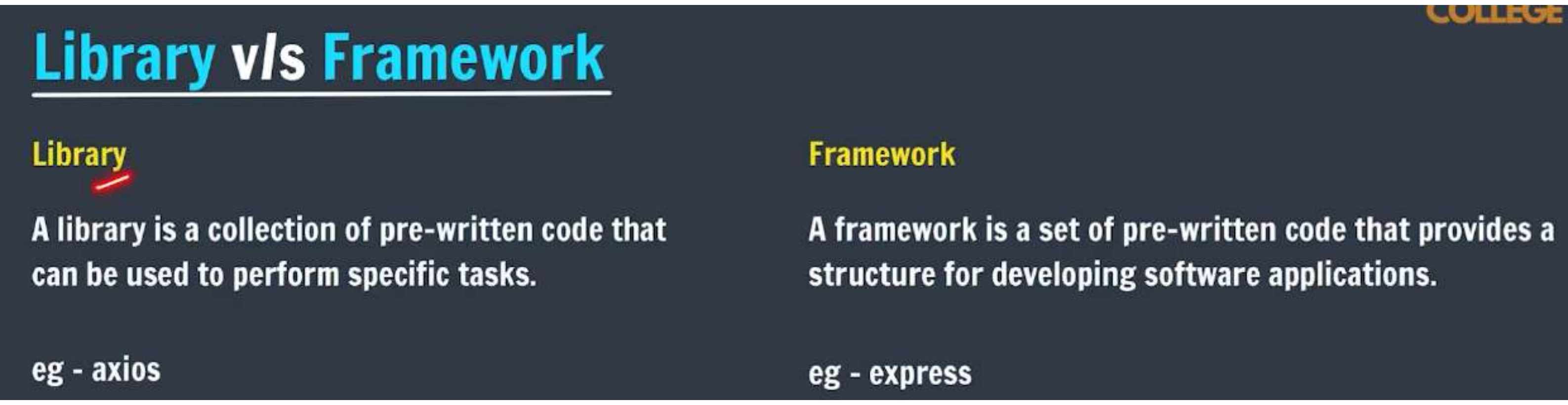


Library v/s Framework

Library

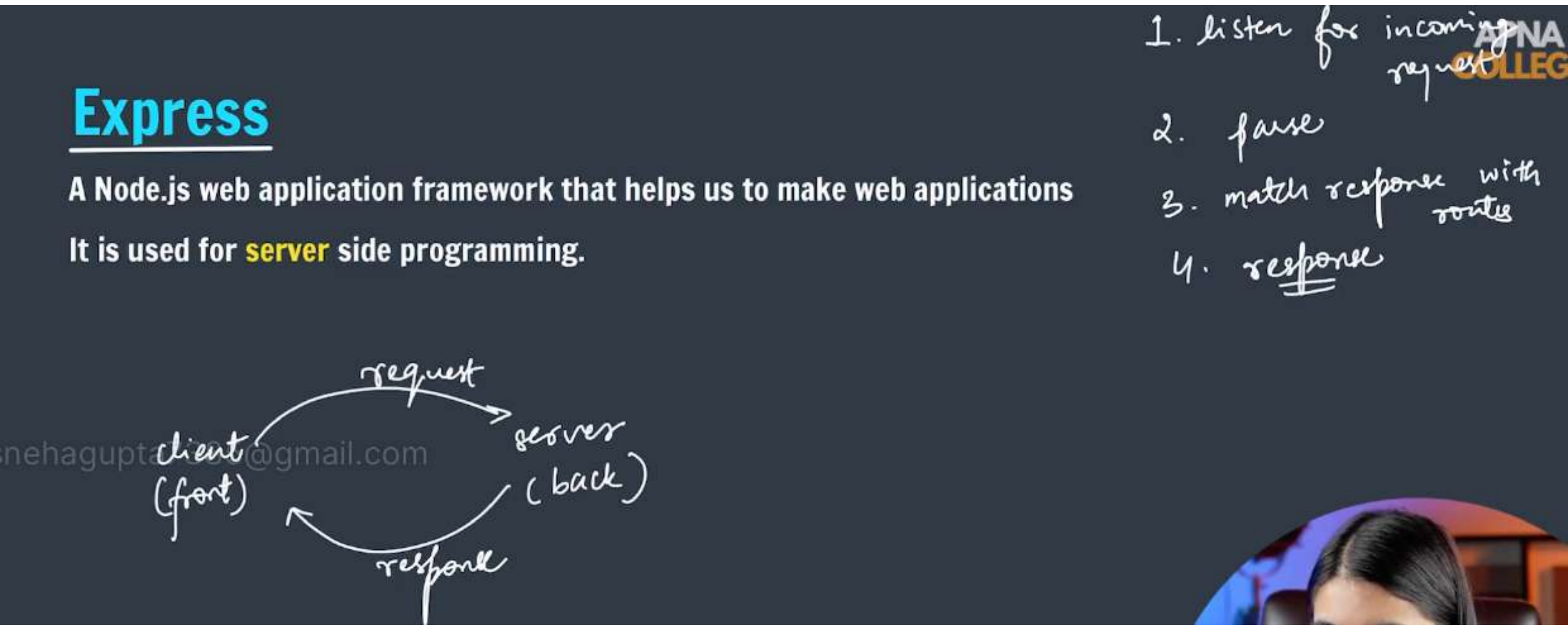
A library is a collection of pre-written code that can be used to perform specific tasks.

eg - axios

Framework

A framework is a set of pre-written code that provides a structure for developing software applications.

eg - express

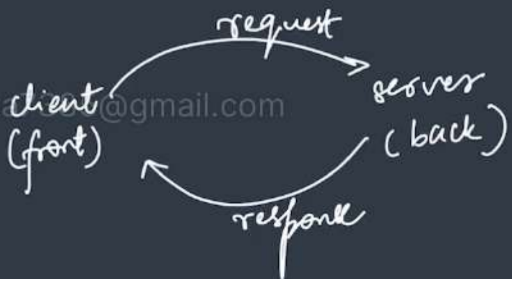


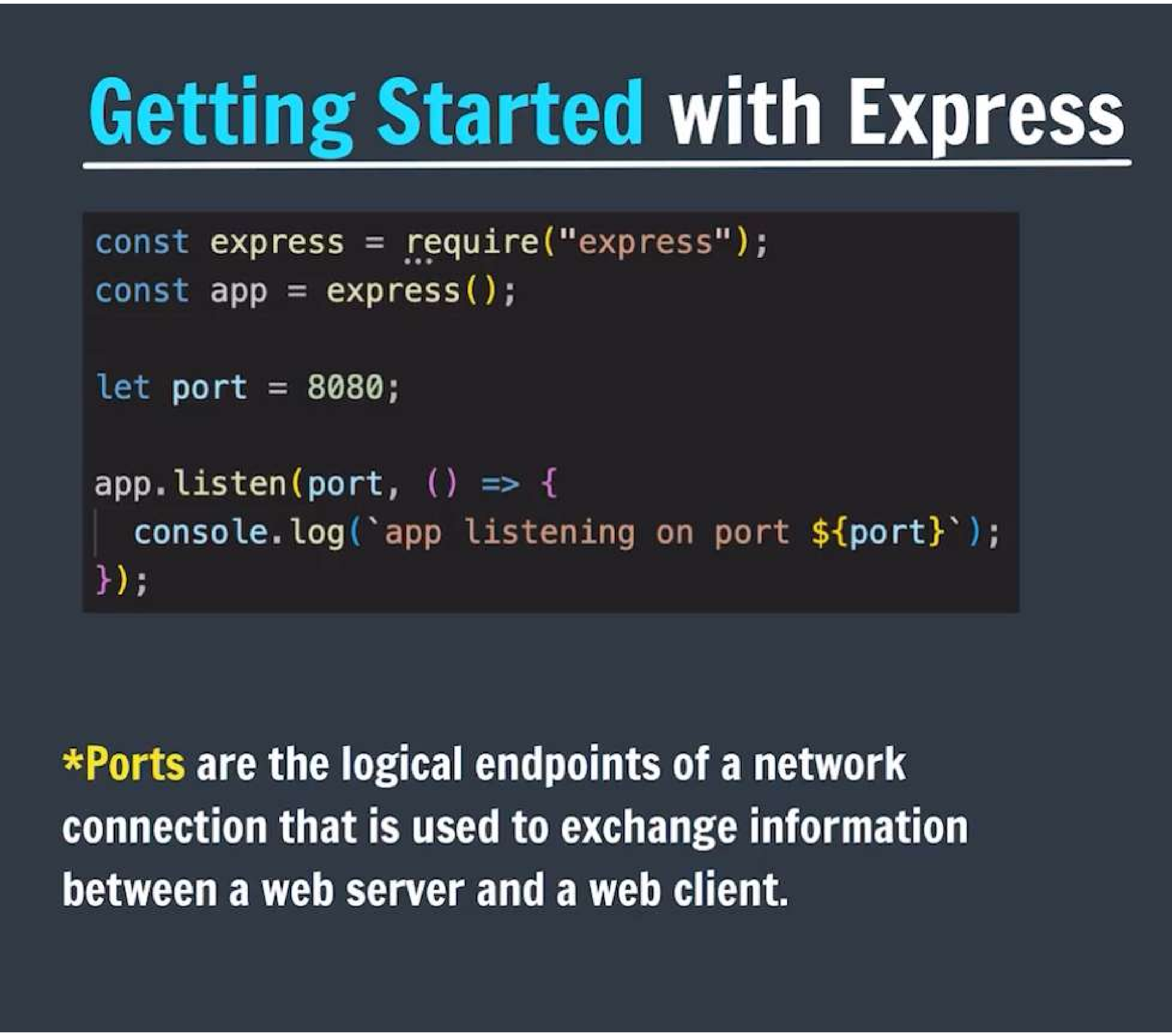
Express

A Node.js web application framework that helps us to make web applications

It is used for server side programming.

1. listen for incoming request
2. parse
3. match response with routes
4. response

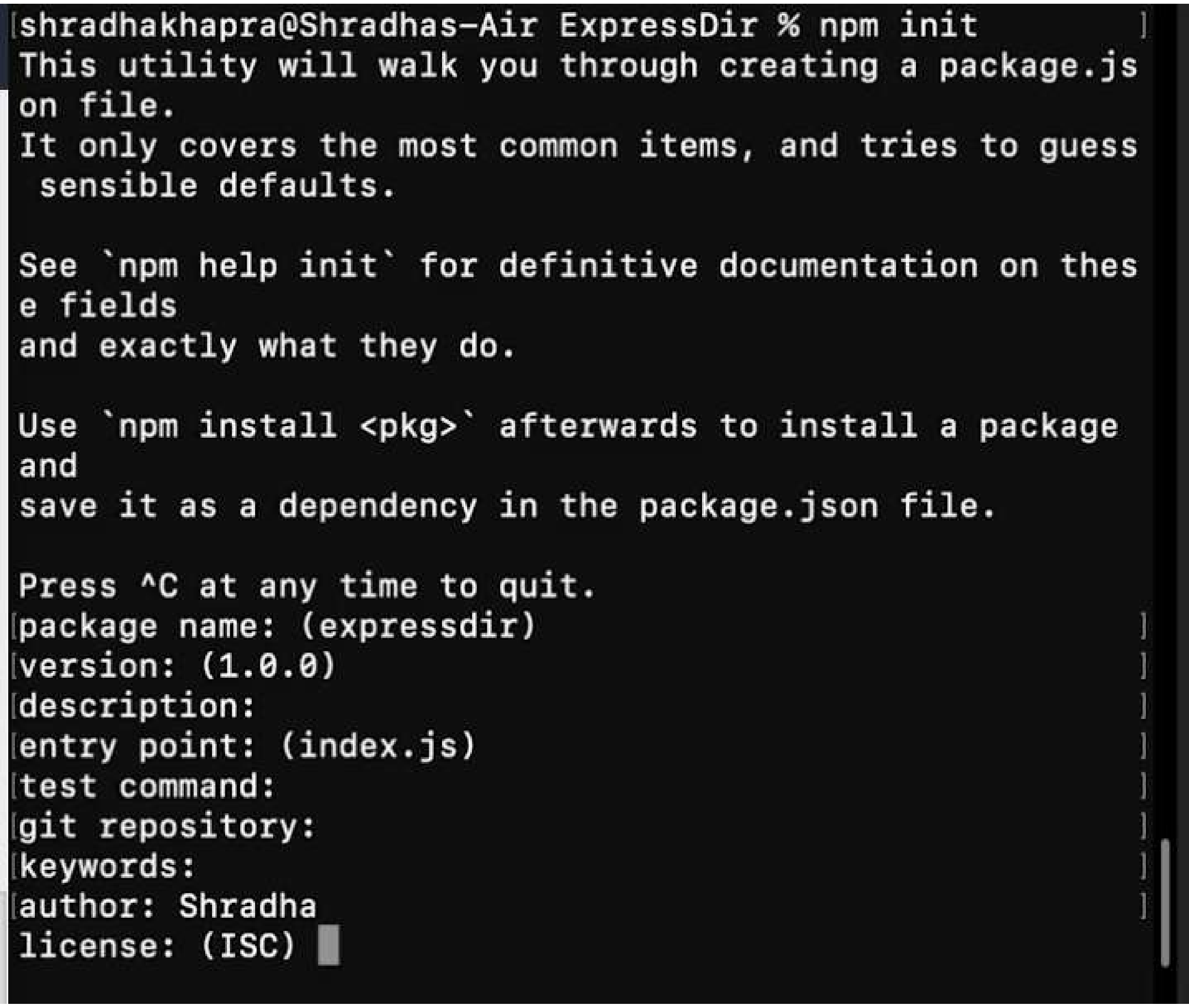




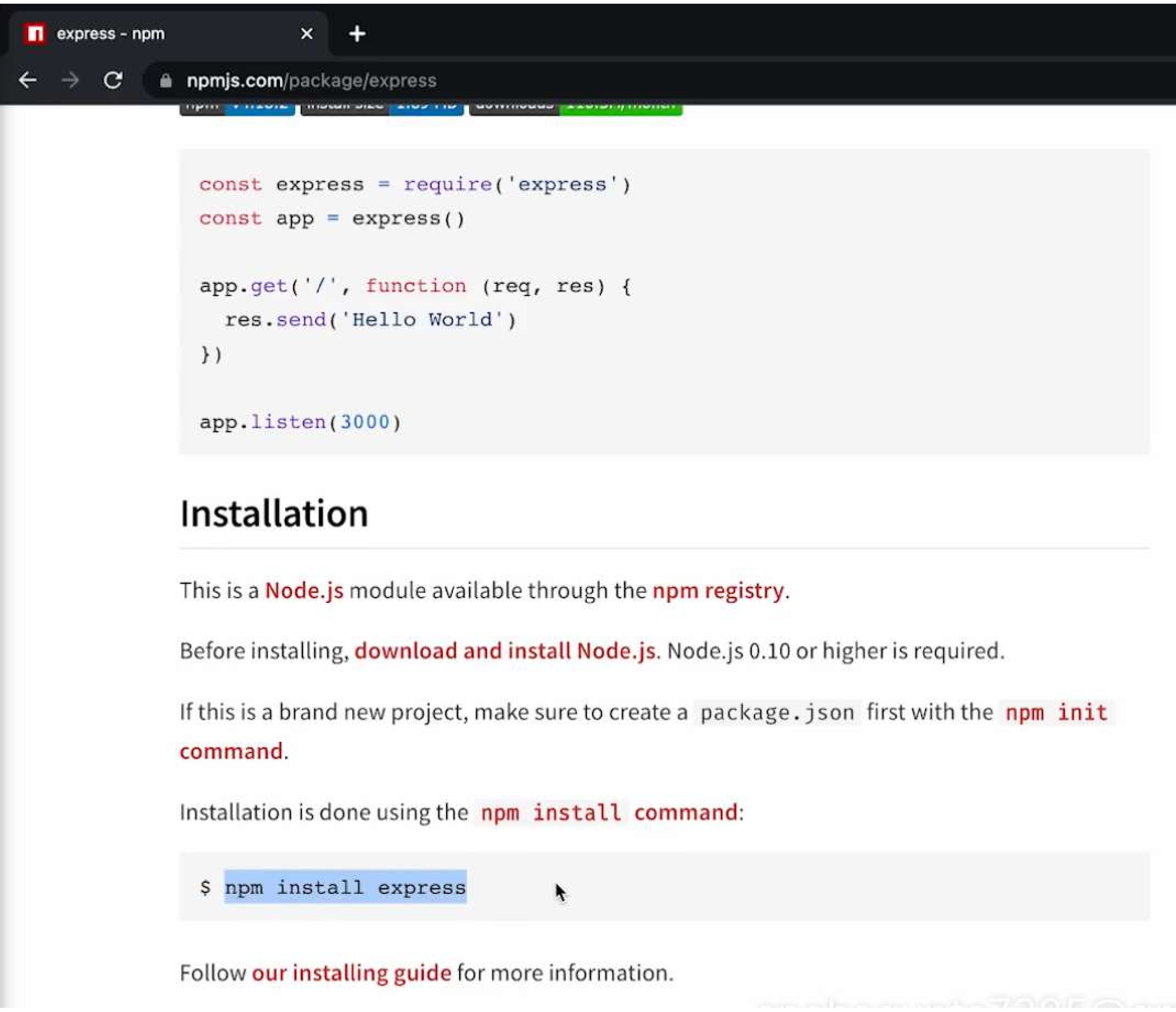
Getting Started with Express

```javascript
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app listening on port ${port}`);
});
```

*Ports are the logical endpoints of a network connection that is used to exchange information between a web server and a web client.



```
shradhakhapra@Shradhas-Air ExpressDir % npm init
This utility will walk you through creating a package.js
on file.
It only covers the most common items, and tries to guess
 sensible defaults.

See `npm help init` for definitive documentation on thes
e fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package
and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (expressdir)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author: Shradha
license: (ISC)
```



```
const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

Installation

This is a Node.js module available through the npm registry.

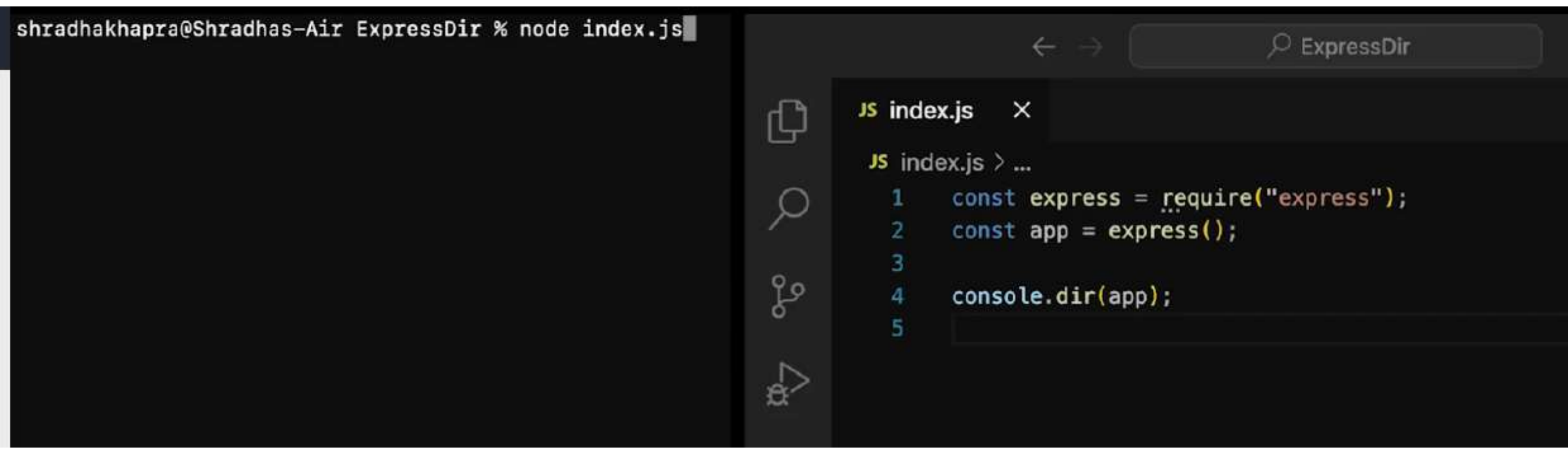
Before installing, download and install Node.js. Node.js 0.10 or higher is required.

If this is a brand new project, make sure to create a `package.json` first with the `npm init` command.

Installation is done using the `npm install` command:

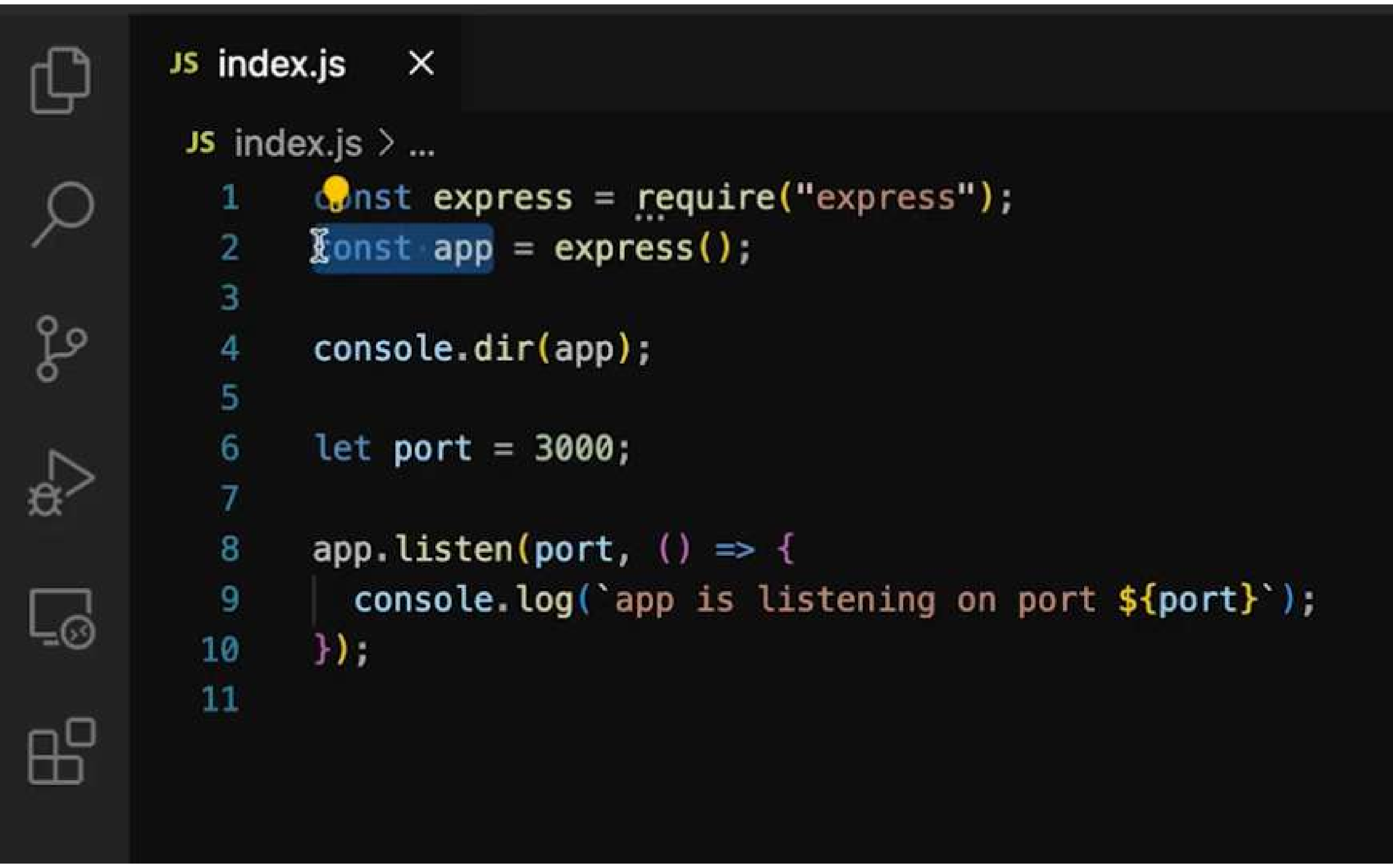
```
$ npm install express
```

Follow our installing guide for more information.

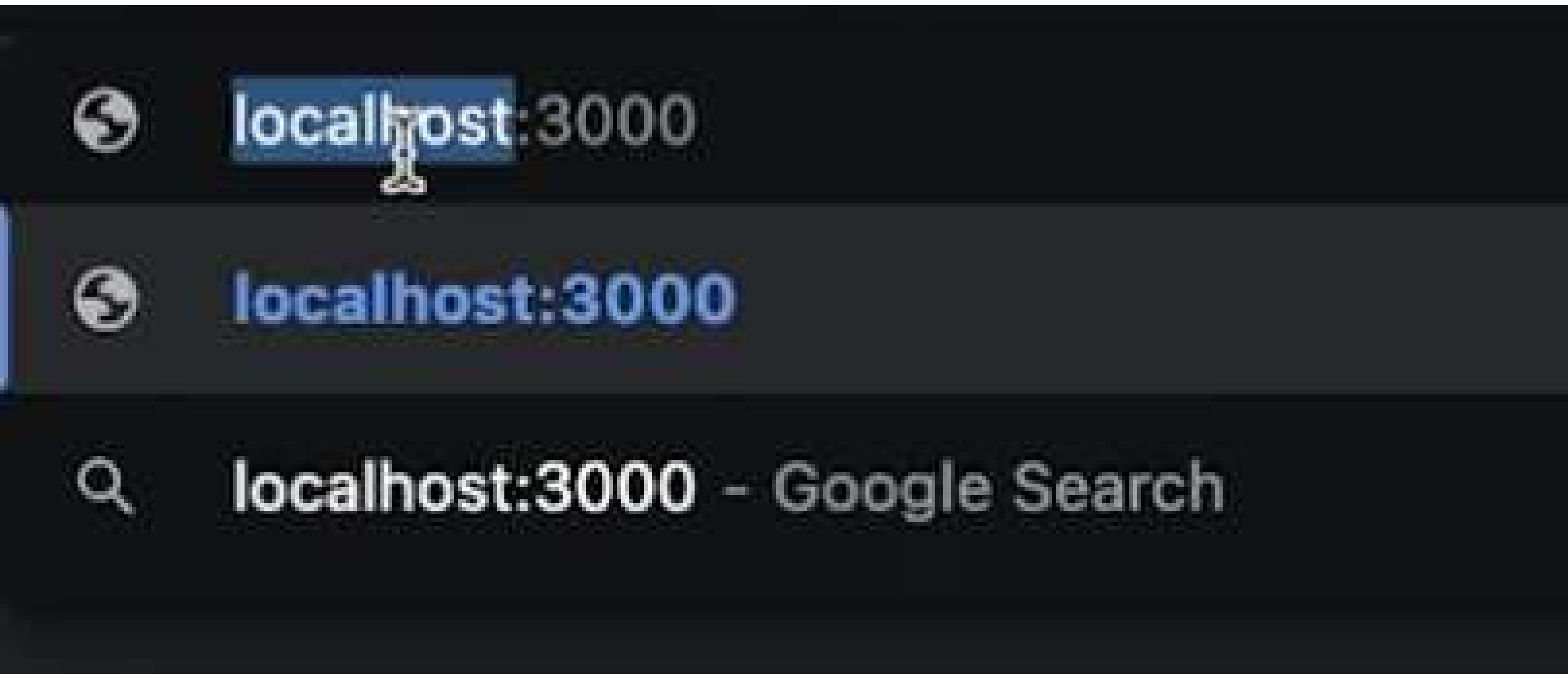


```
shradhakhapra@Shradhas-Air ExpressDir % node index.js
```

```js
const express = require("express");
const app = express();

console.dir(app);
```



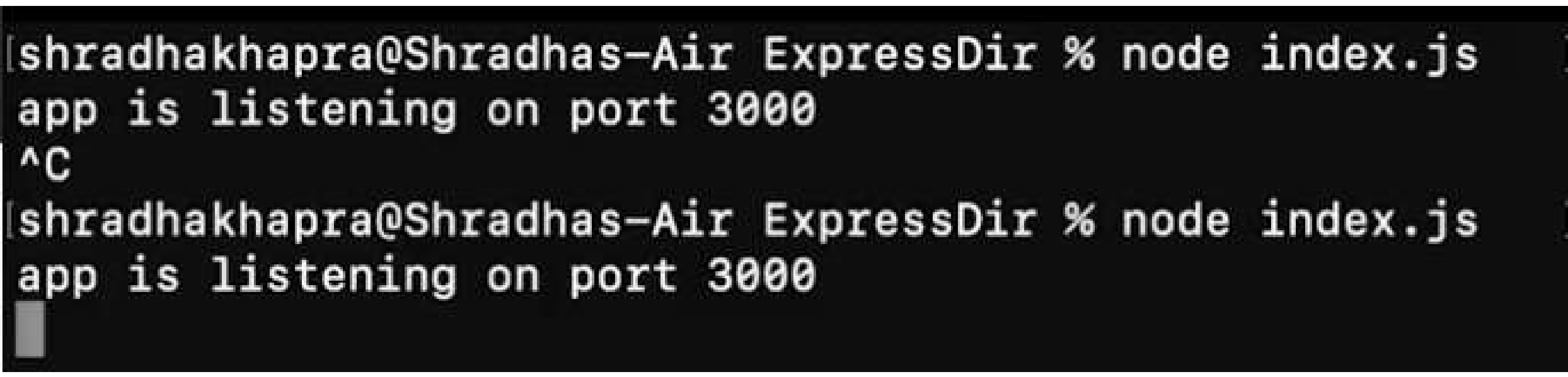
```javascript
const express = require("express");
const app = express();

console.dir(app);

let port = 3000;

app.listen(port, () => {
  console.log(`app is listening on port ${port}`);
});
```



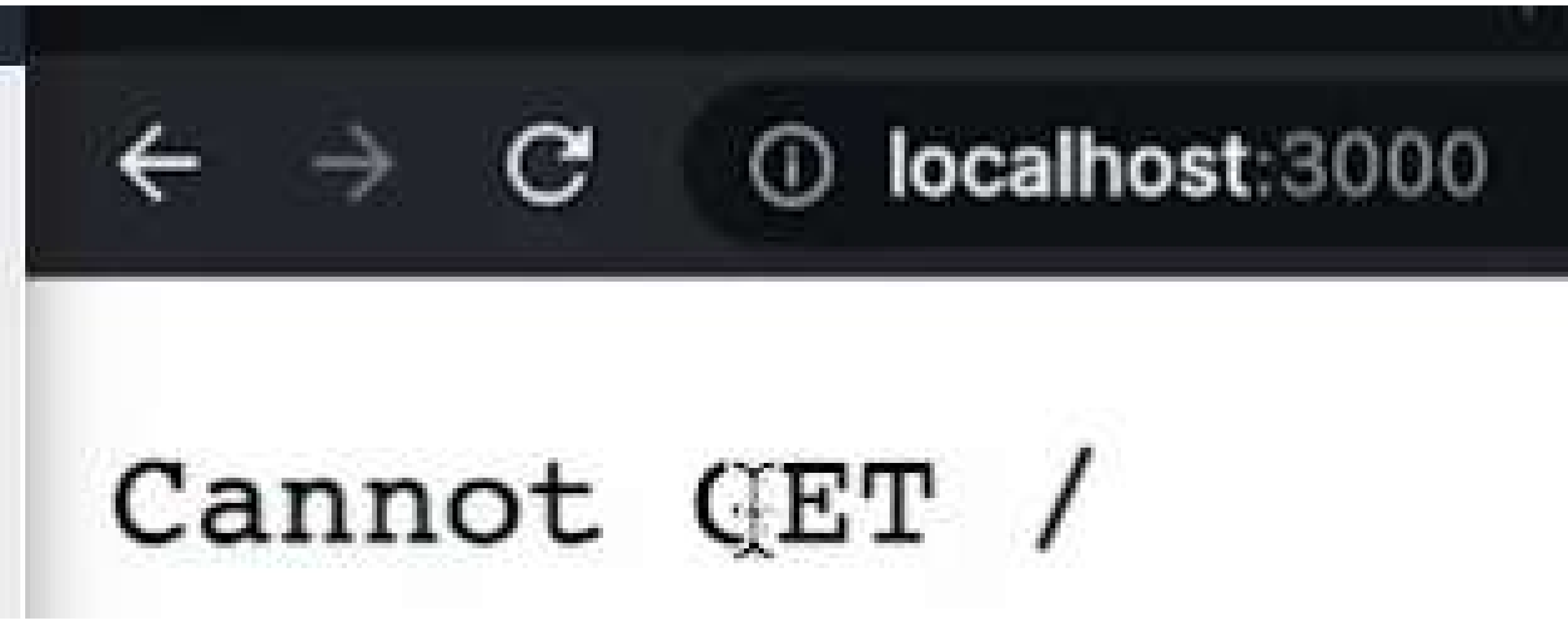
localhost:3000

localhost:3000

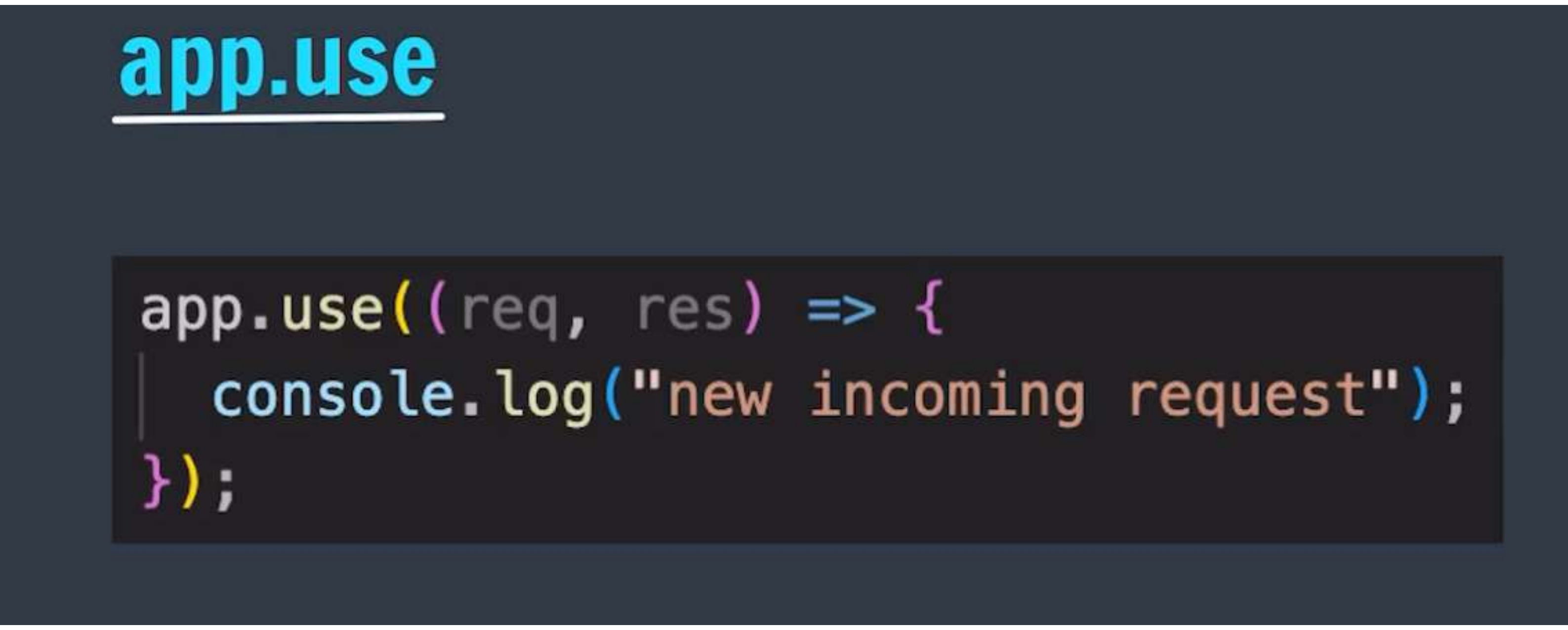
localhost:3000 - Google Search



```
shradhakhapra@Shradhas-Air ExpressDir % node index.js
app is listening on port 3000
^C
shradhakhapra@Shradhas-Air ExpressDir % node index.js
app is listening on port 3000
```

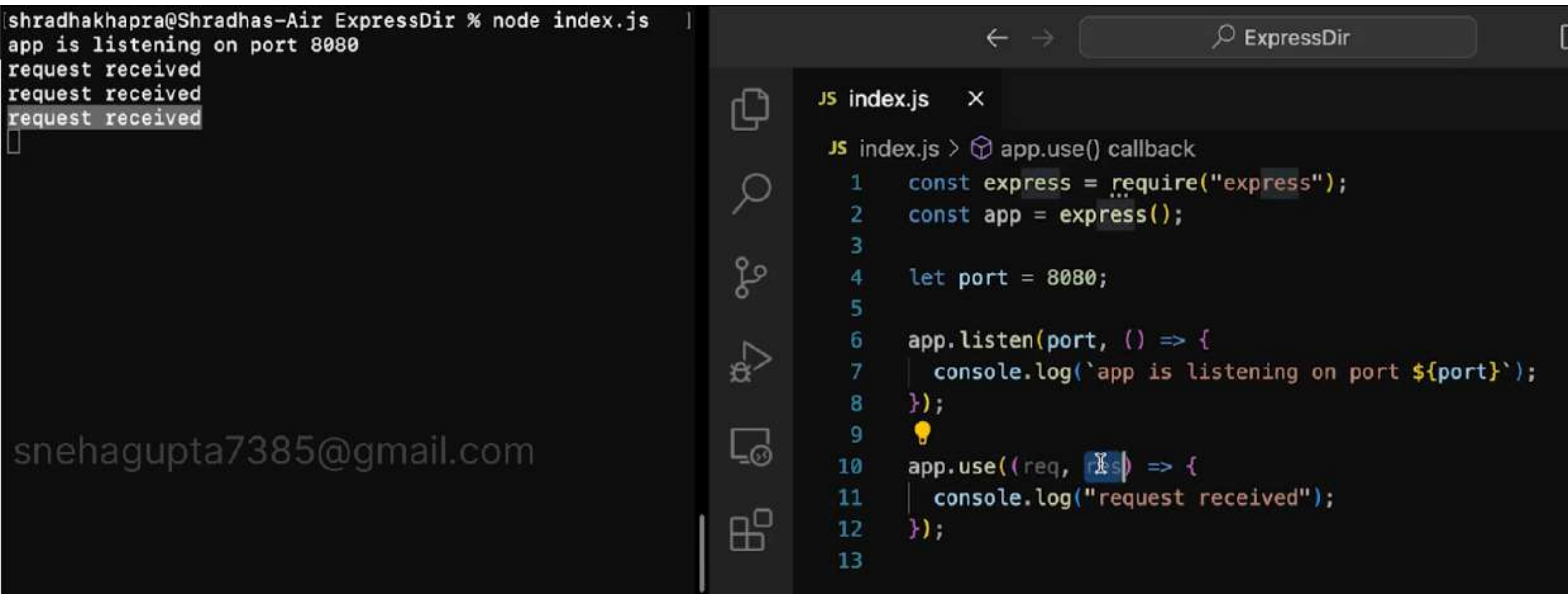


```
Cannot GET /
```

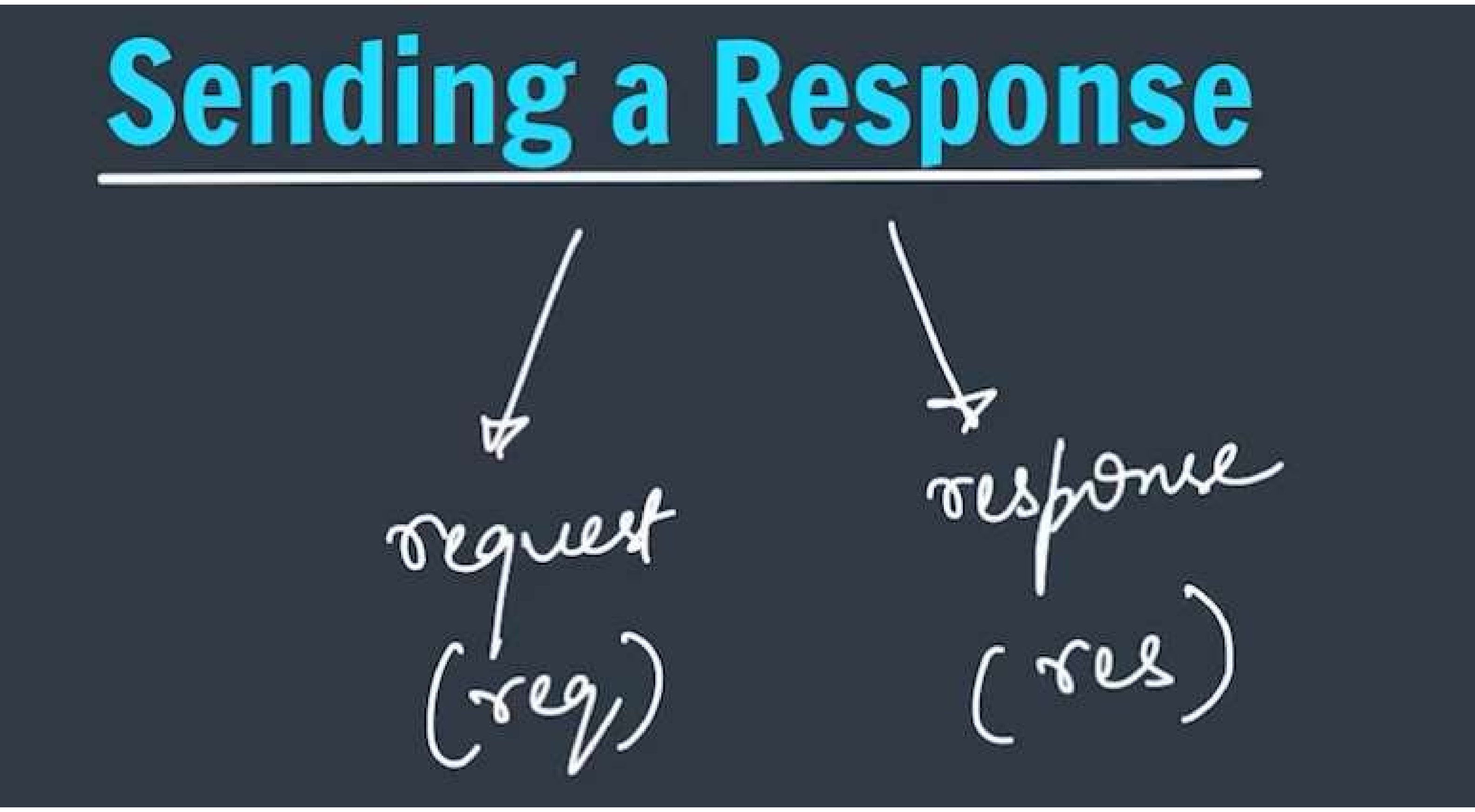


app.use

```javascript
app.use((req, res) => {
  console.log("new incoming request");
});
```

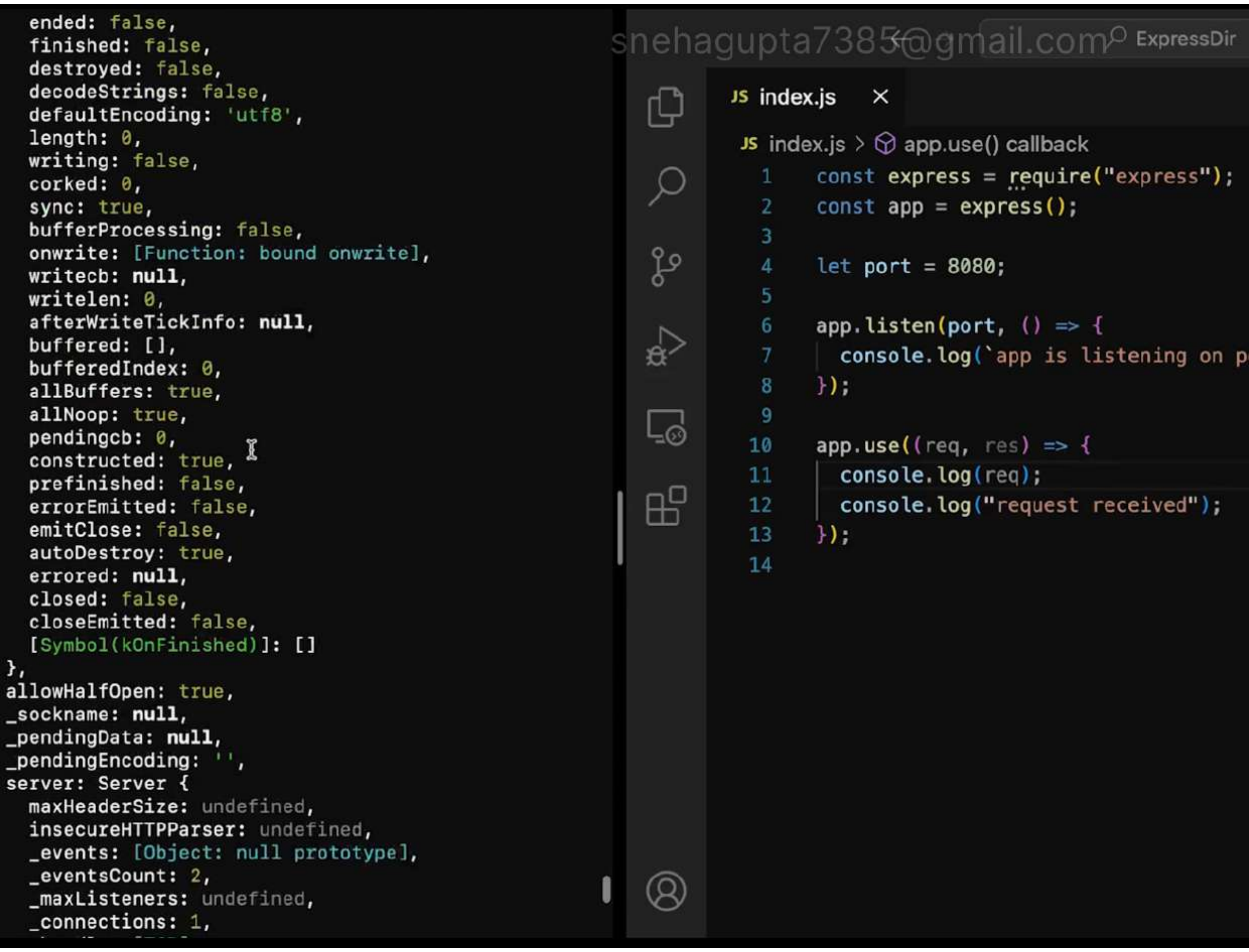

```
shradhakhapra@Shradhas-Air ExpressDir % node index.js
app is listening on port 8080
request received
request received
request received
```

```js
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app is listening on port ${port}`);
});

app.use((req, res) => {
  console.log("request received");
});

```



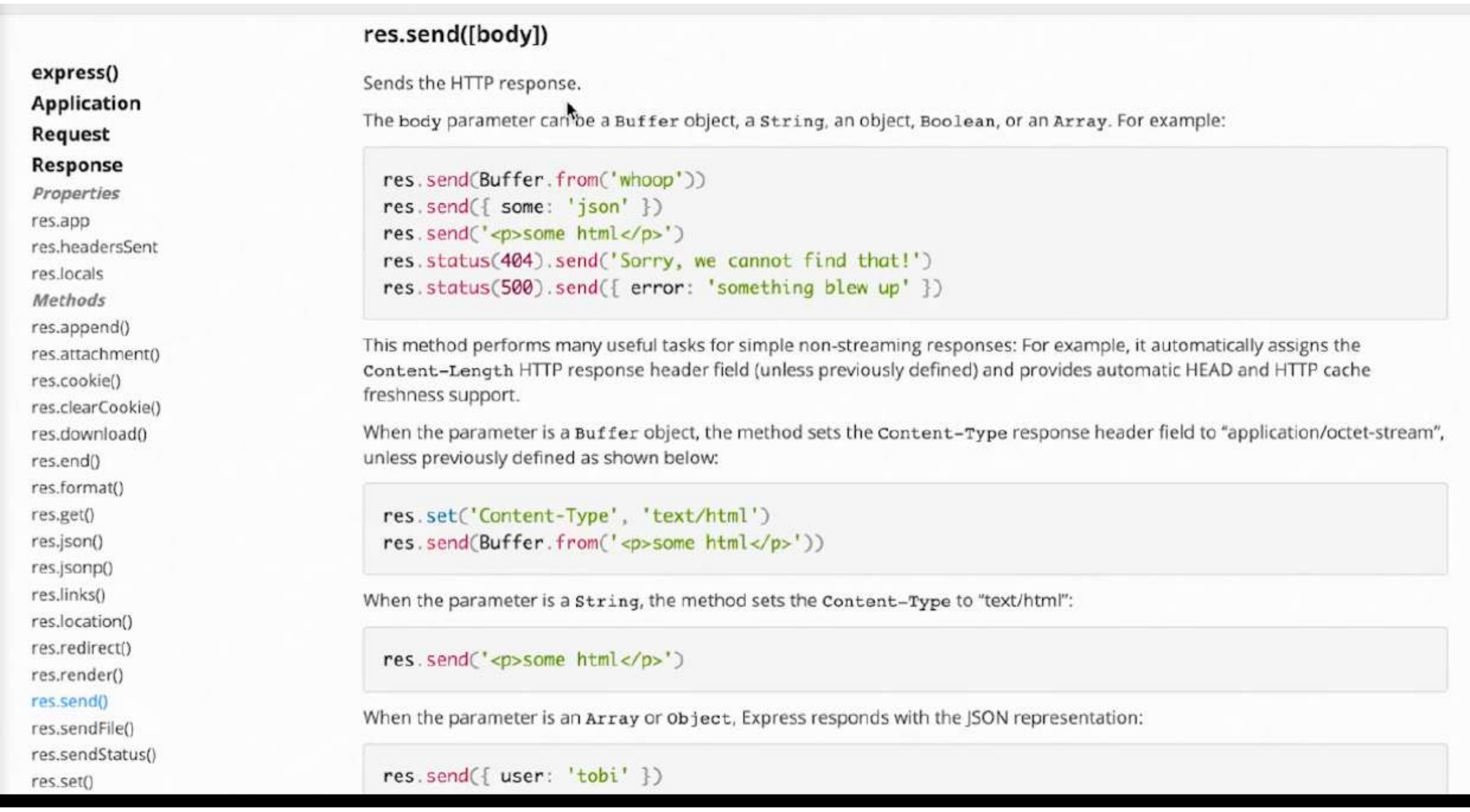
Sending a Response

$$\begin{tikzcd}
 & \text{Sending a Response} \arrow[dl] \arrow[dr] & \\
\text{request (req)} & & \text{response (res)}
\end{tikzcd}$$



```
  ended: false,
  finished: false,
  destroyed: false,
  decodeStrings: false,
  defaultEncoding: 'utf8',
  length: 0,
  writing: false,
  corked: 0,
  sync: true,
  bufferProcessing: false,
  onwrite: [Function: bound onwrite],
  writecb: null,
  writelen: 0,
  afterWriteTickInfo: null,
  buffered: [],
  bufferedIndex: 0,
  allBuffers: true,
  allNoop: true,
  pendingcb: 0,
  constructed: true,
  prefinished: false,
  errorEmitted: false,
  emitClose: false,
  autoDestroy: true,
  errored: null,
  closed: false,
  closeEmitted: false,
  [Symbol(kOnFinished)]: []
},
allowHalfOpen: true,
_sockname: null,
_pendingData: null,
_pendingEncoding: '',
server: Server {
  maxHeaderSize: undefined,
  insecureHTTPParser: undefined,
  _events: [Object: null prototype],
  _eventsCount: 2,
  _maxListeners: undefined,
  _connections: 1,
```

```js
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app is listening on p
});

app.use((req, res) => {
  console.log(req);
  console.log("request received");
});

```



express()
Application
Request
Response
Properties
res.app
res.headersSent
res.locals
Methods
res.append()
res.attachment()
res.cookie()
res.clearCookie()
res.download()
res.end()
res.format()
res.get()
res.json()
res.jsonp()
res.links()
res.location()
res.redirect()
res.render()
res.send()
res.sendFile()
res.sendStatus()
res.set()

res.send([body])

Sends the HTTP response.

The `body` parameter can be a `Buffer` object, a `String`, an object, `Boolean`, or an `Array`. For example:

```
res.send(Buffer.from('whoop'))
res.send({ some: 'json' })
res.send('<p>some html</p>')
res.status(404).send('Sorry, we cannot find that!')
res.status(500).send({ error: 'something blew up' })
```

This method performs many useful tasks for simple non-streaming responses: For example, it automatically assigns the `Content-Length` HTTP response header field (unless previously defined) and provides automatic HEAD and HTTP cache freshness support.

When the parameter is a `Buffer` object, the method sets the `Content-Type` response header field to "application/octet-stream", unless previously defined as shown below:

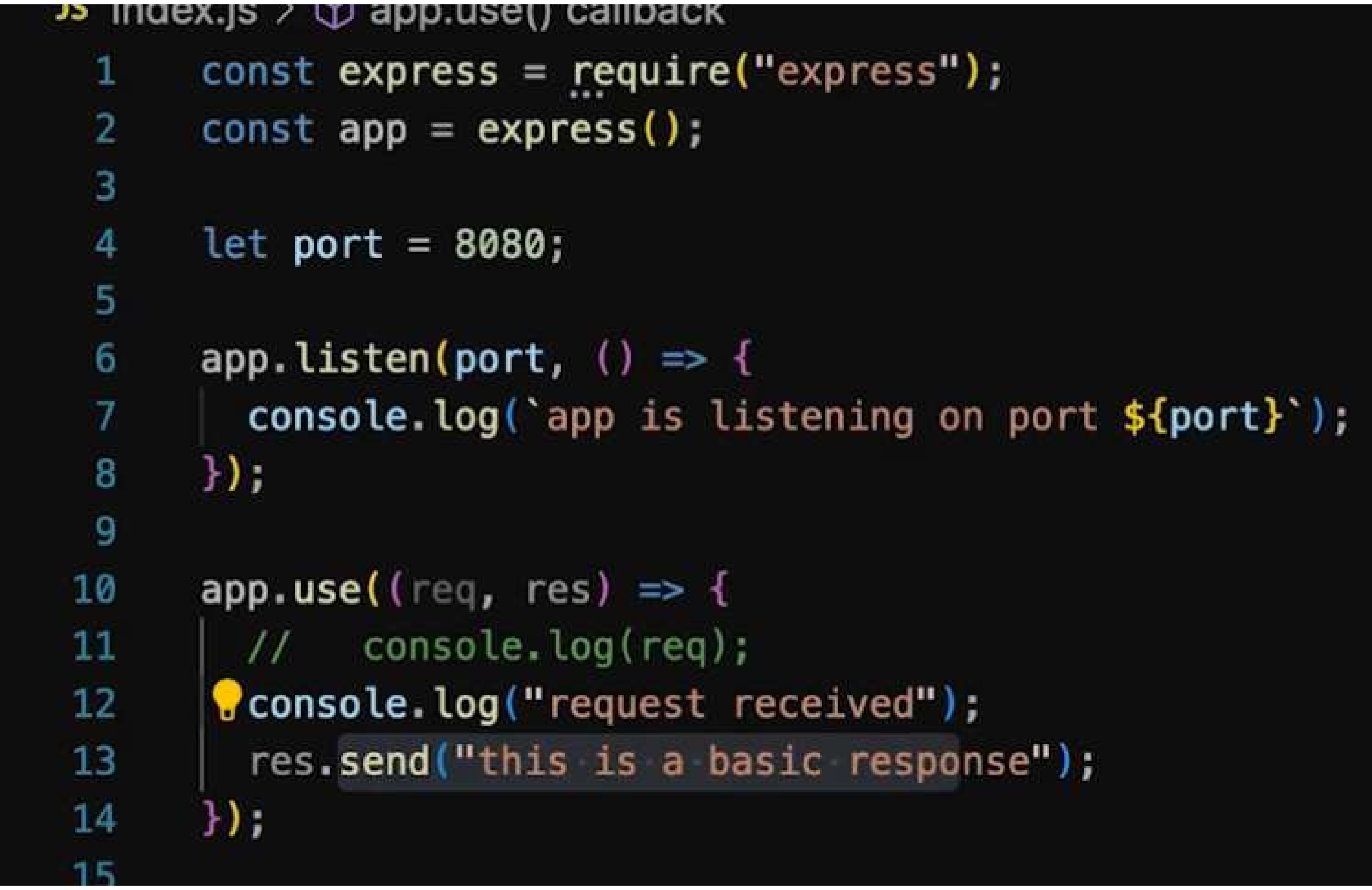
```
res.set('Content-Type', 'text/html')
res.send(Buffer.from('<p>some html</p>'))
```

When the parameter is a `String`, the method sets the `Content-Type` to "text/html":

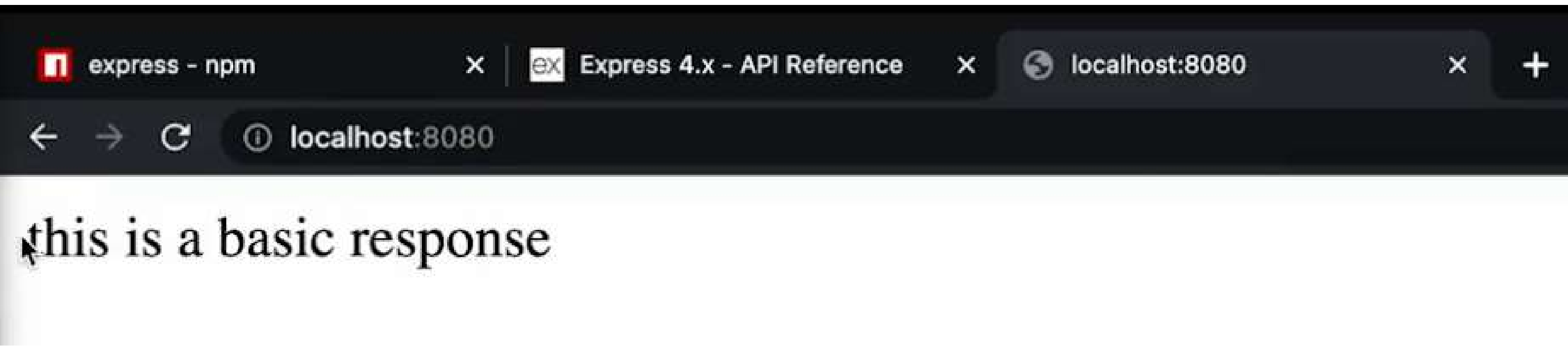
```
res.send('<p>some html</p>')
```

When the parameter is an `Array` or `Object`, Express responds with the JSON representation:

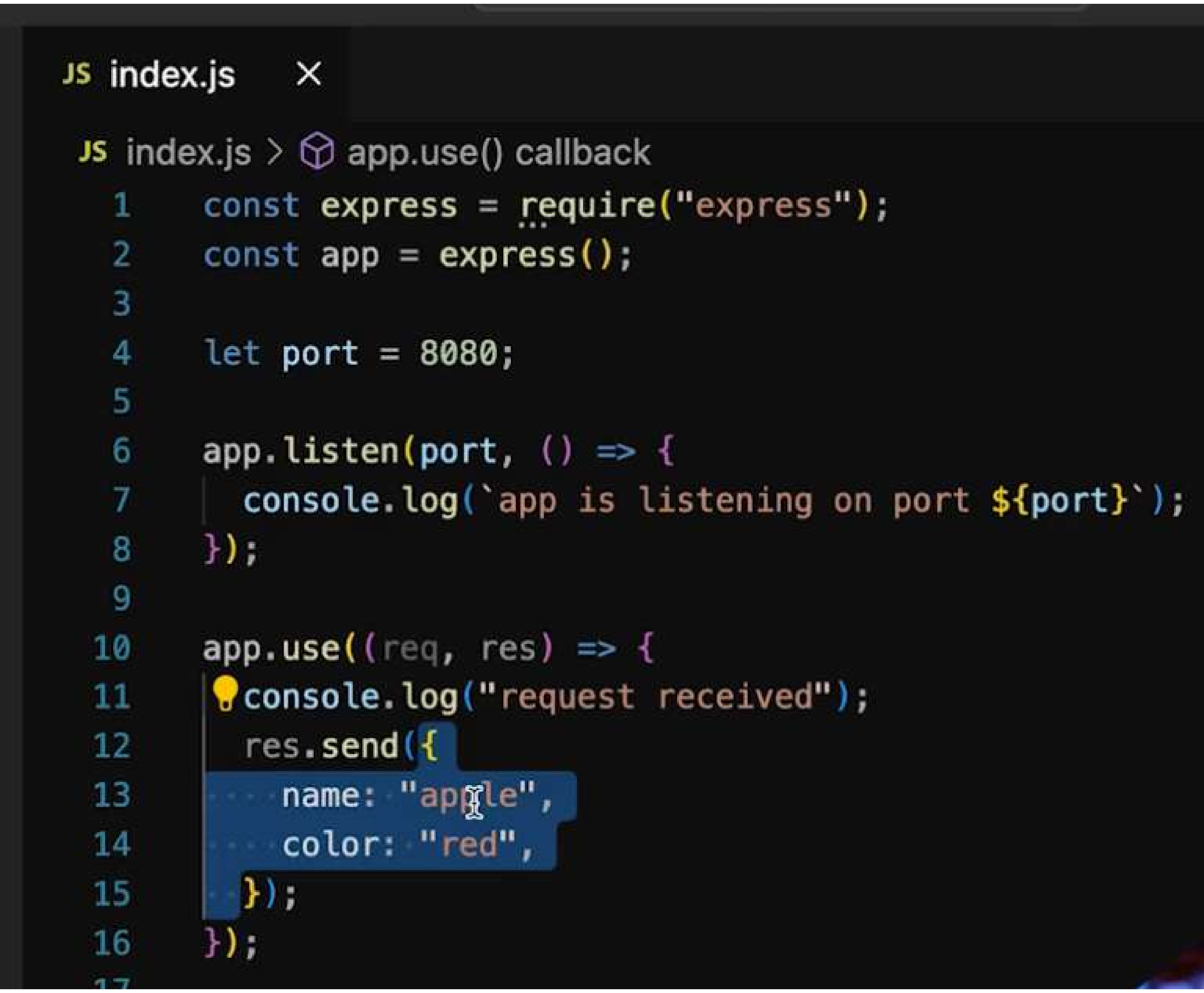
```
res.send({ user: 'tobi' })
```



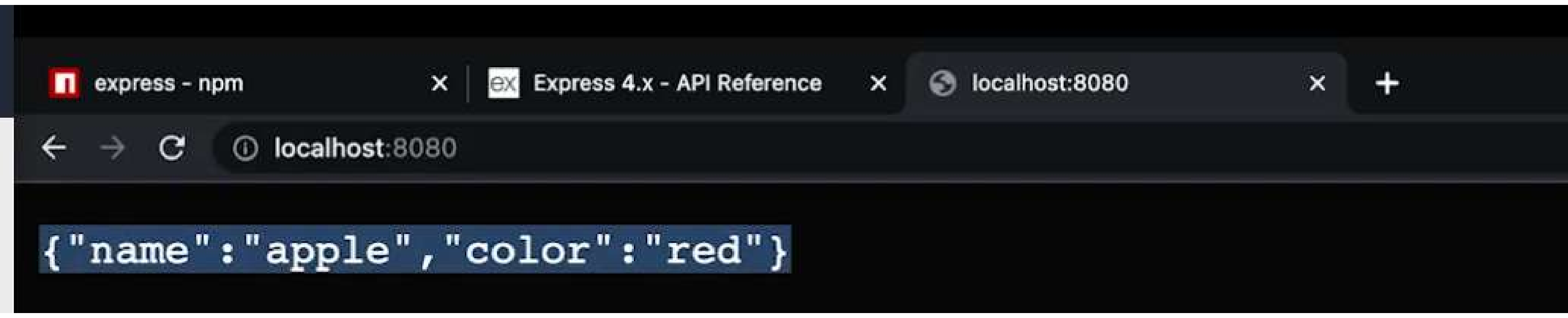
```
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app is listening on port ${port}`);
});

app.use((req, res) => {
  //   console.log(req);
  console.log("request received");
  res.send("this is a basic response");
});
```



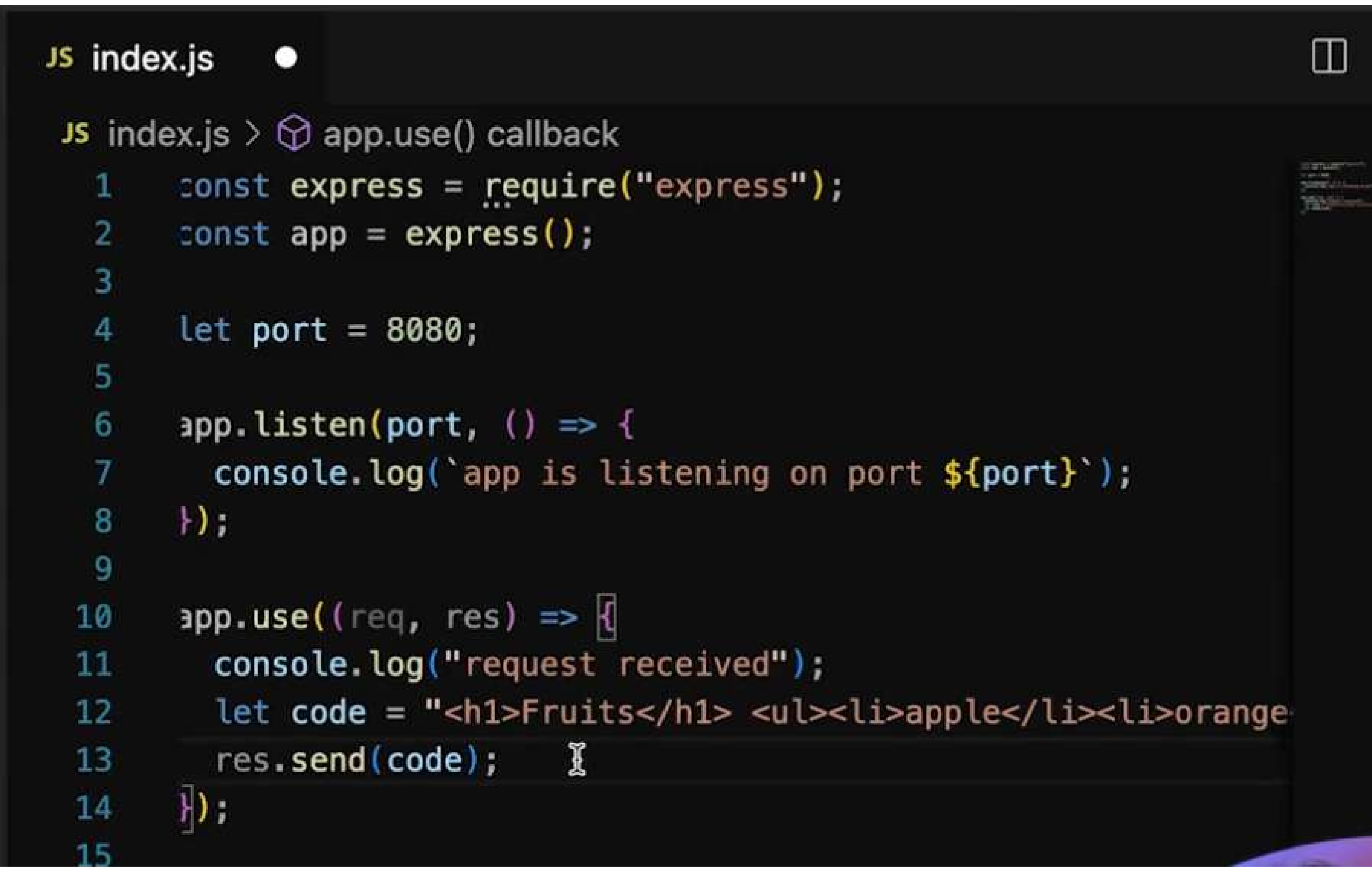
```
this is a basic response
```



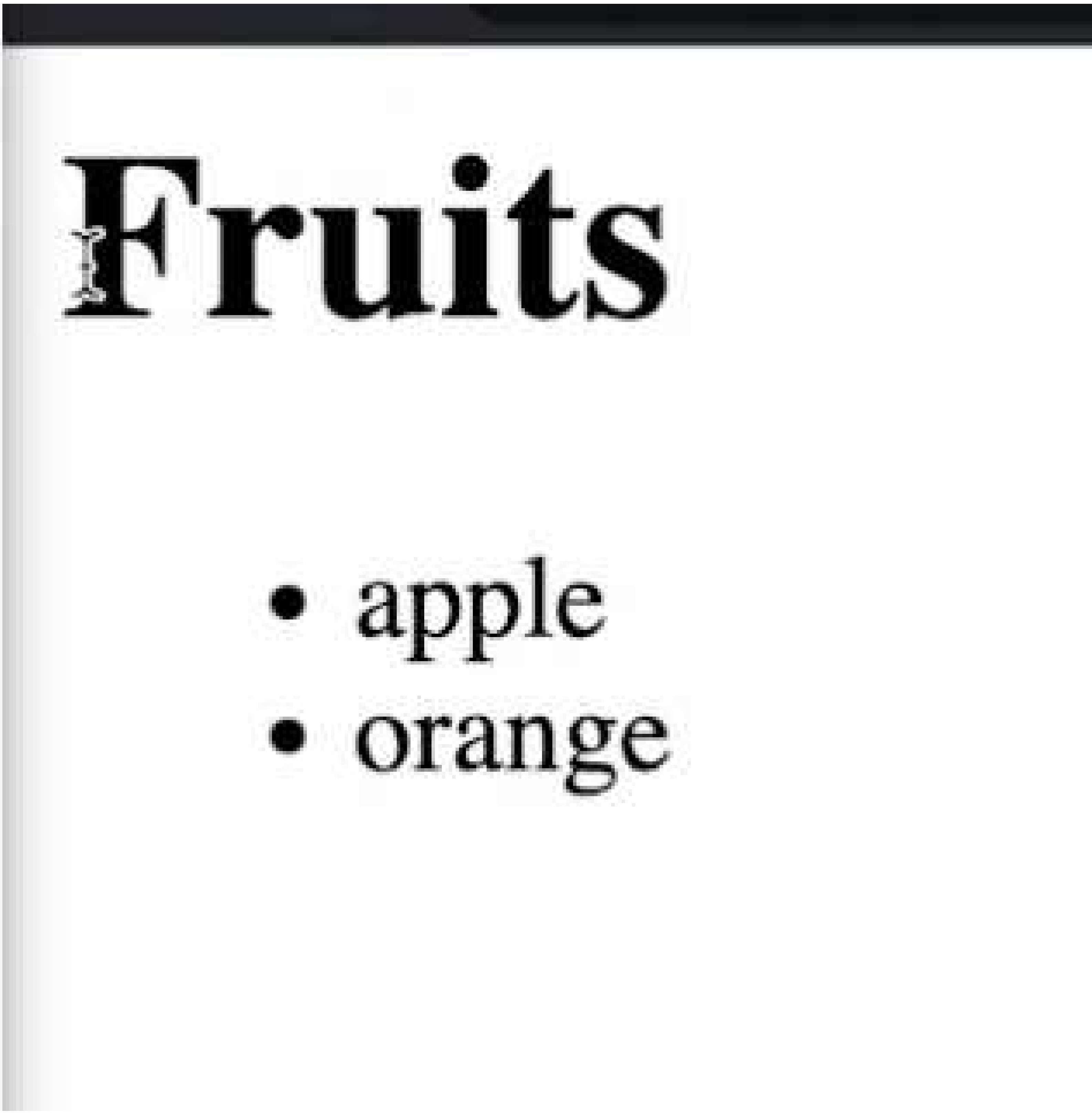
```javascript
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app is listening on port ${port}`);
});

app.use((req, res) => {
  console.log("request received");
  res.send({
    name: "apple",
    color: "red",
  });
});
```



```
{"name":"apple","color":"red"}
```

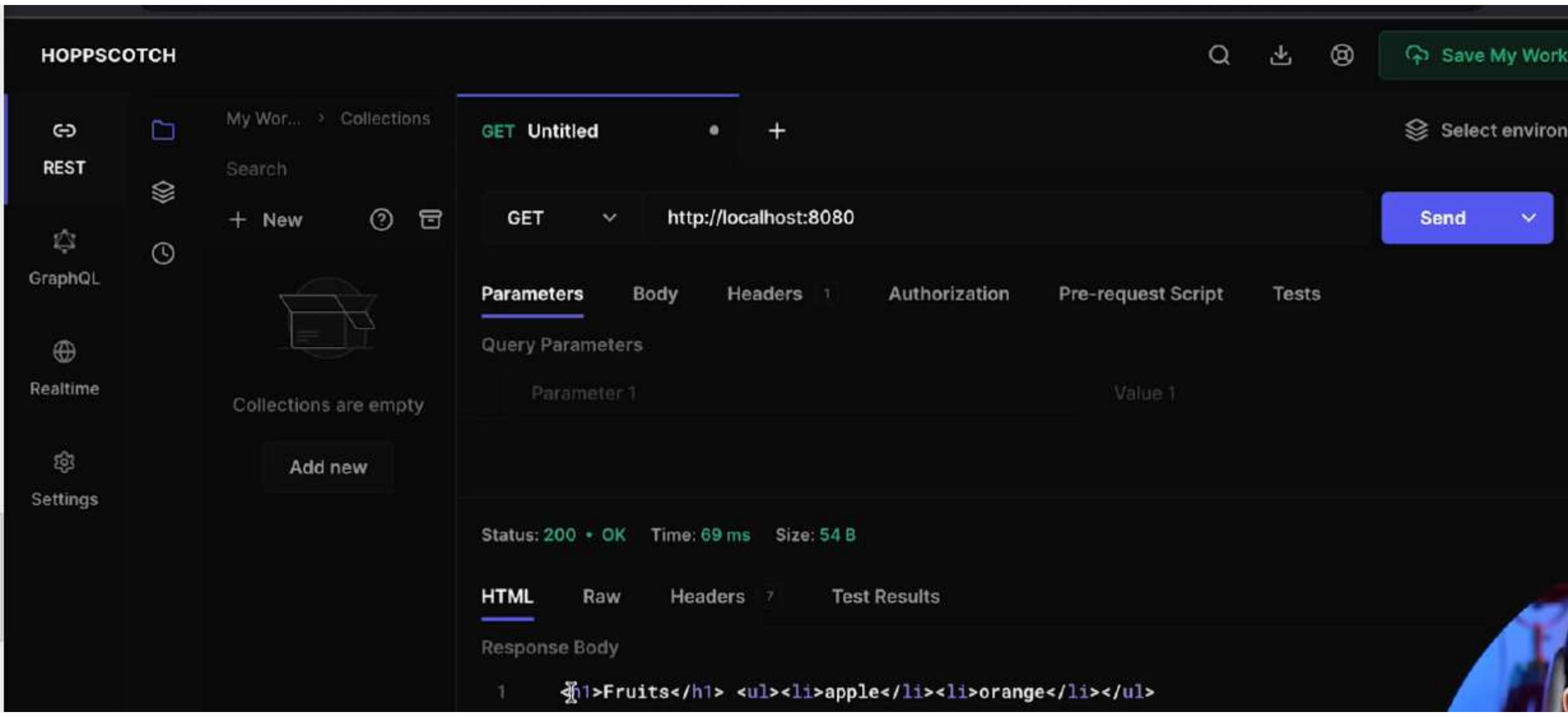


```
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app is listening on port ${port}`);
});

app.use((req, res) => {
  console.log("request received");
  let code = "<h1>Fruits</h1> <ul><li>apple</li><li>orange
  res.send(code);
});
```



Fruits

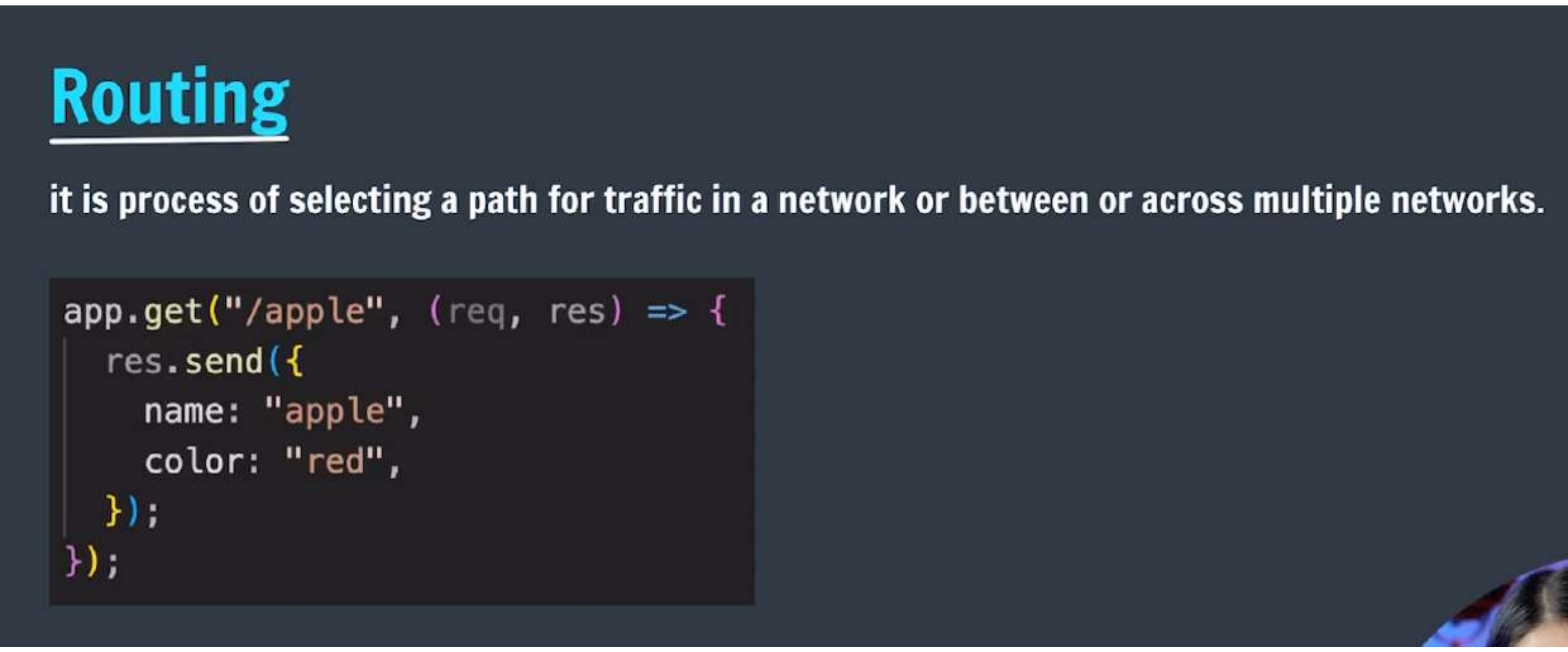
- apple
- orange



```
GET http://localhost:8080
```

Status: 200 • OK   Time: 69 ms   Size: 54 B

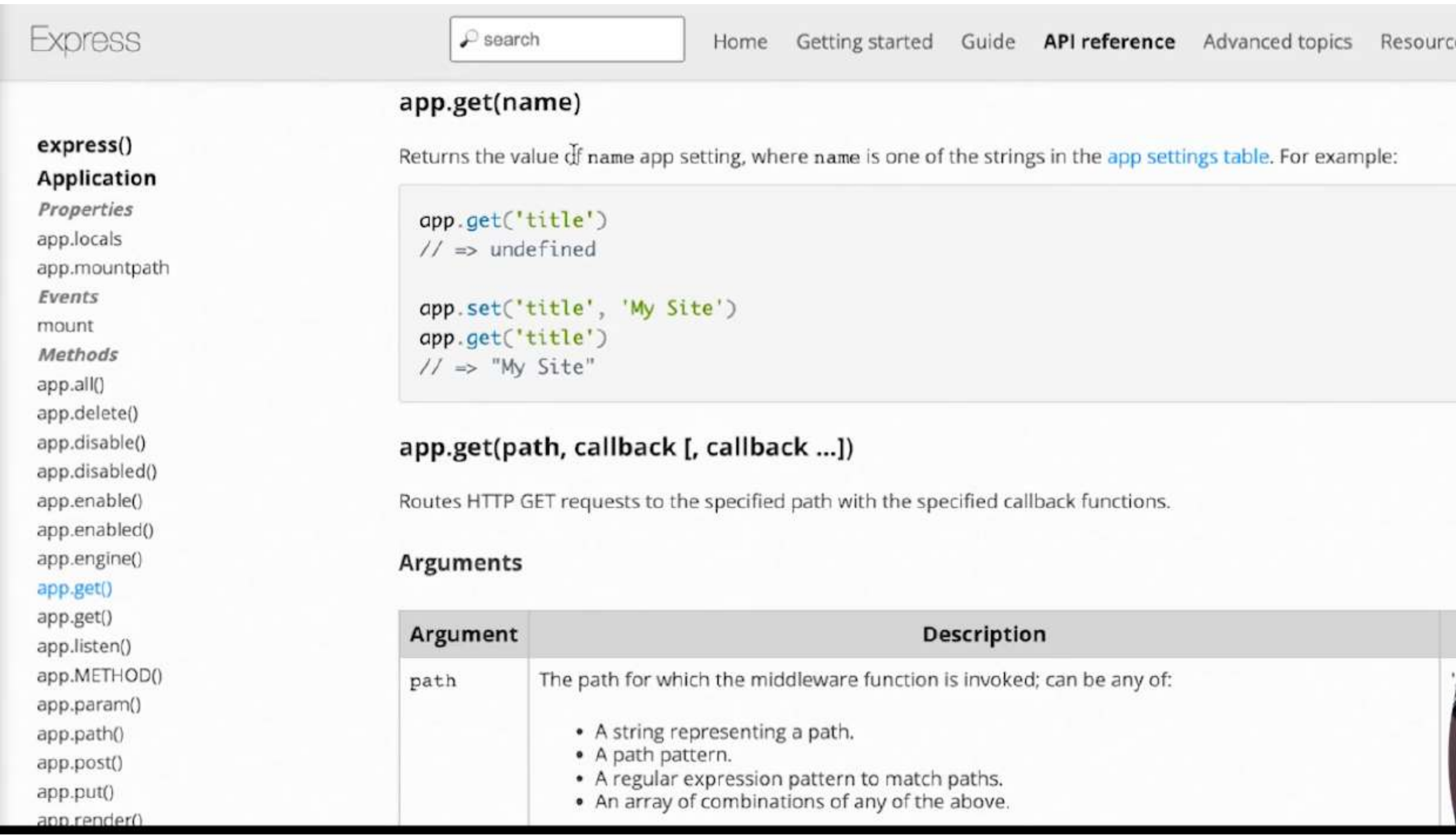
```
<h1>Fruits</h1> <ul><li>apple</li><li>orange</li></ul>
```



Routing

it is process of selecting a path for traffic in a network or between or across multiple networks.

```
app.get("/apple", (req, res) => {
  res.send({
    name: "apple",
    color: "red",
  });
});
```



app.get(name)

Returns the value of `name` app setting, where `name` is one of the strings in the app settings table. For example:

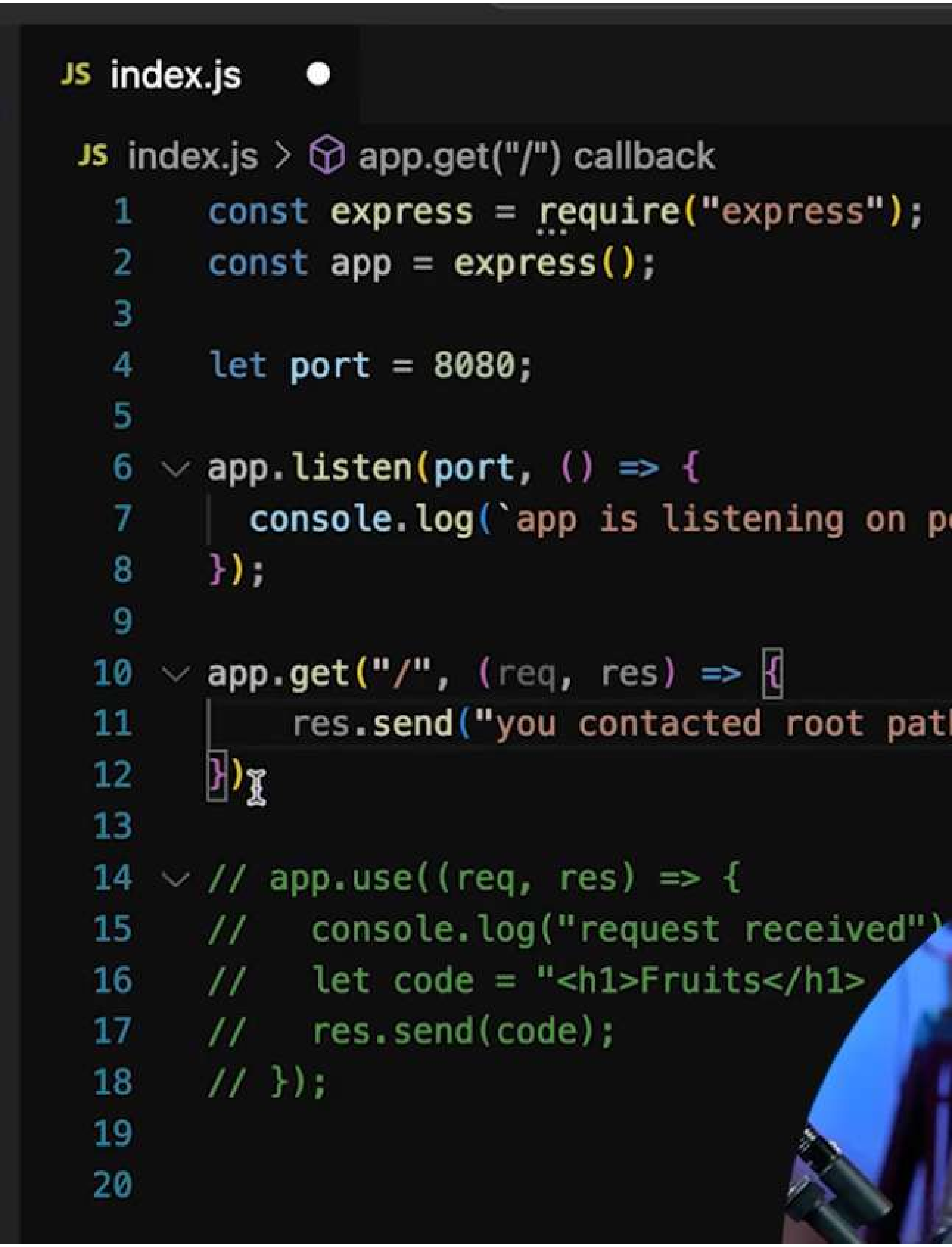
```
app.get('title')
// => undefined

app.set('title', 'My Site')
app.get('title')
// => "My Site"
```

app.get(path, callback [, callback ...])

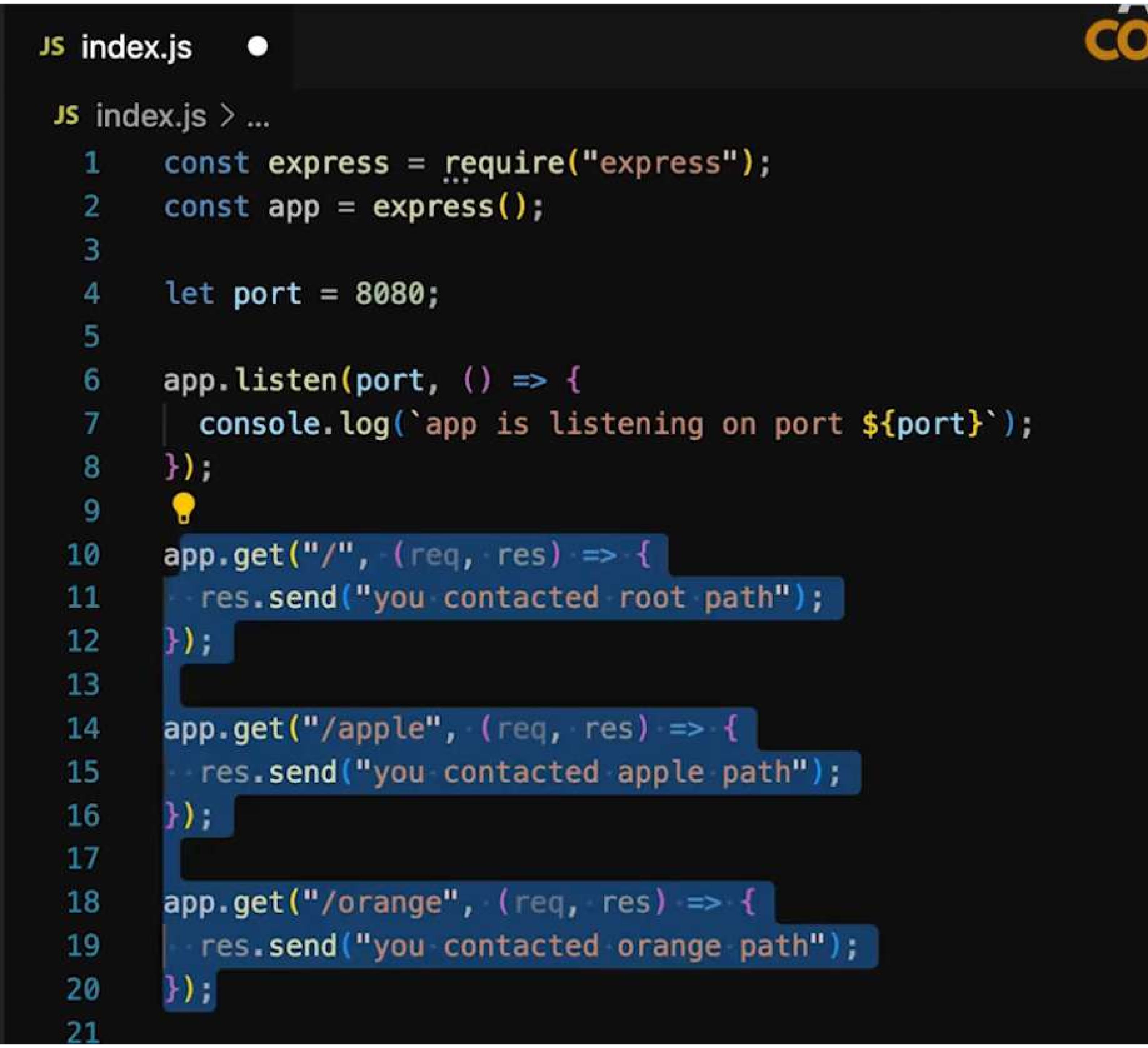
Routes HTTP GET requests to the specified path with the specified callback functions.

Arguments

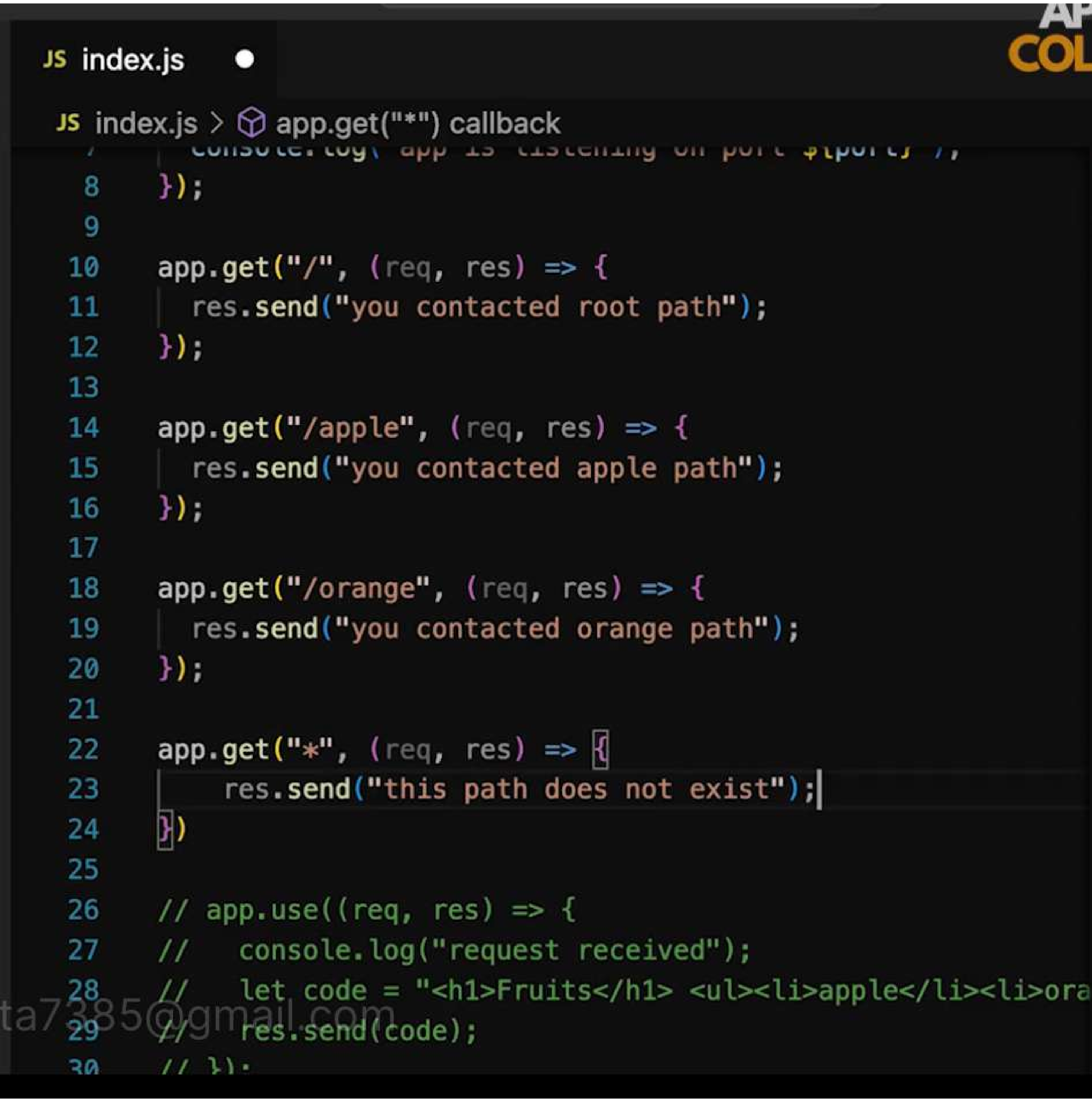
Argument	Description
`path`	The path for which the middleware function is invoked; can be any of: • A string representing a path. • A path pattern. • A regular expression pattern to match paths. • An array of combinations of any of the above.



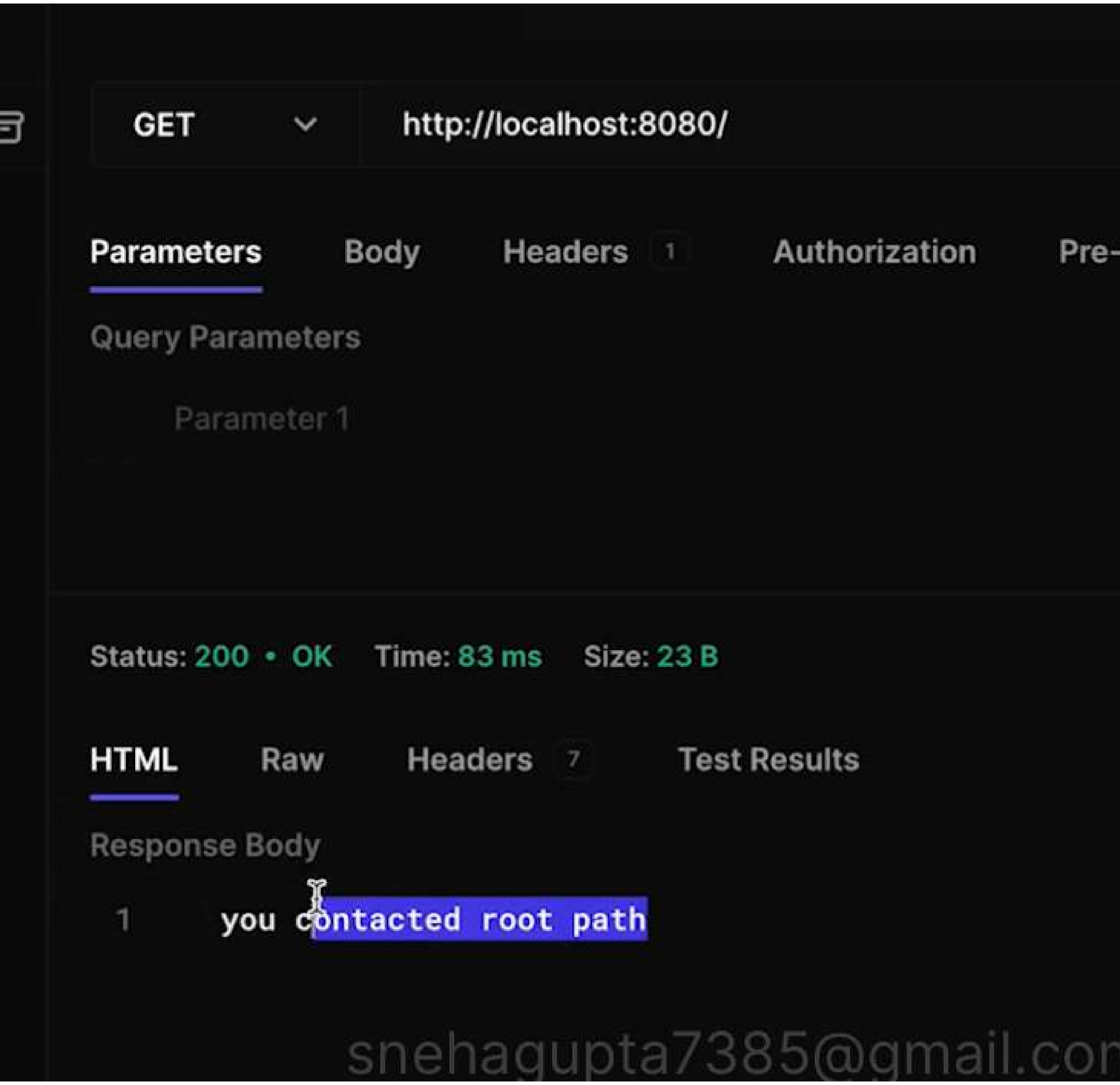
```
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app is listening on p
});

app.get("/", (req, res) => {
    res.send("you contacted root pat
})

// app.use((req, res) => {
//   console.log("request received")
//   let code = "<h1>Fruits</h1>
//   res.send(code);
// });


```



```
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app is listening on port ${port}`);
});

app.get("/", (req, res) => {
  res.send("you contacted root path");
});

app.get("/apple", (req, res) => {
  res.send("you contacted apple path");
});

app.get("/orange", (req, res) => {
  res.send("you contacted orange path");
});

```

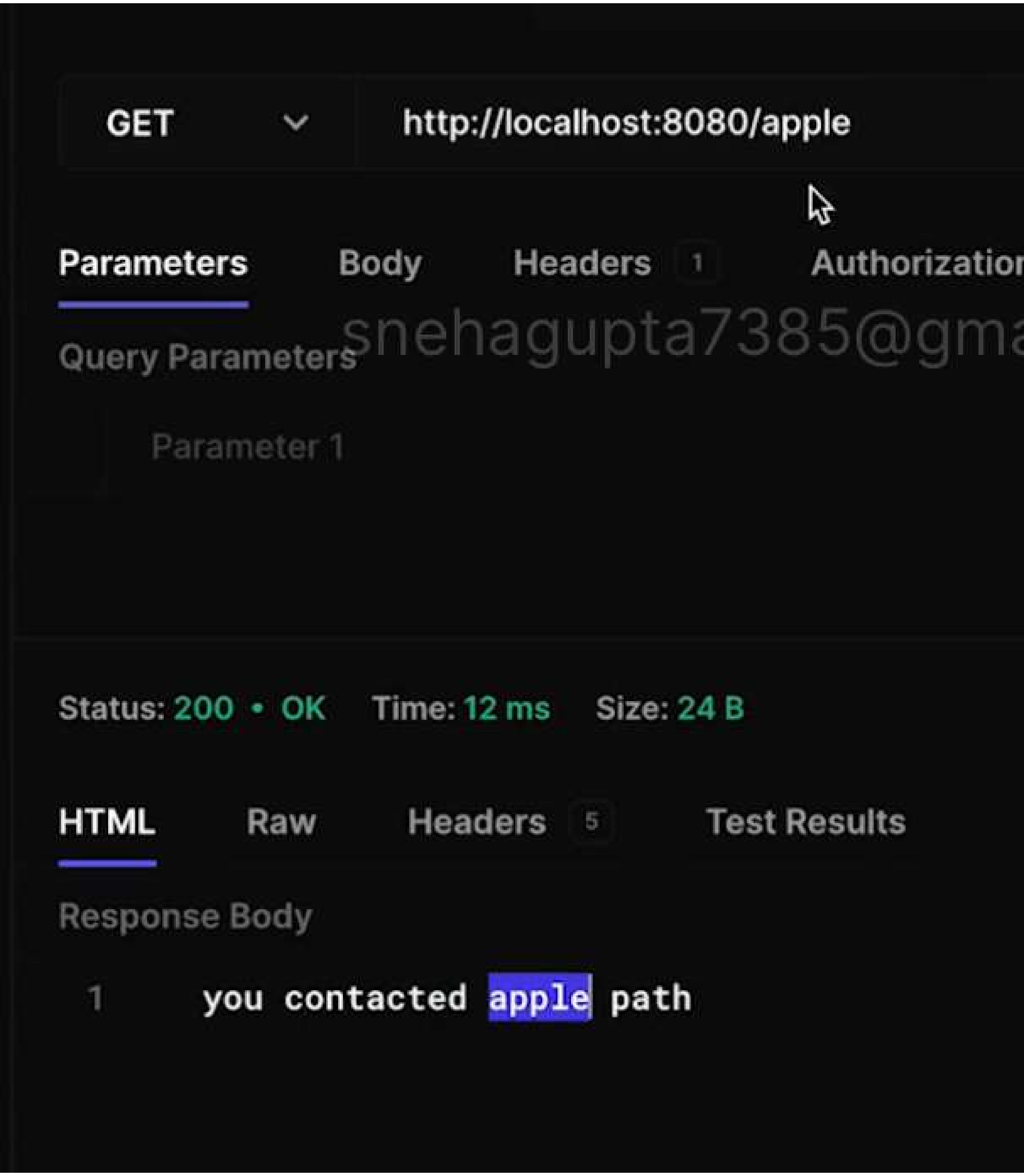


```
  console.log(`app is listening on port ${port}`);
});

app.get("/", (req, res) => {
  res.send("you contacted root path");
});

app.get("/apple", (req, res) => {
  res.send("you contacted apple path");
});

app.get("/orange", (req, res) => {
  res.send("you contacted orange path");
});

app.get("*", (req, res) => {
    res.send("this path does not exist");
})

// app.use((req, res) => {
//   console.log("request received");
//   let code = "<h1>Fruits</h1> <ul><li>apple</li><li>ora
//   res.send(code);
// });
```



```
Status: 200 • OK    Time: 83 ms    Size: 23 B
```

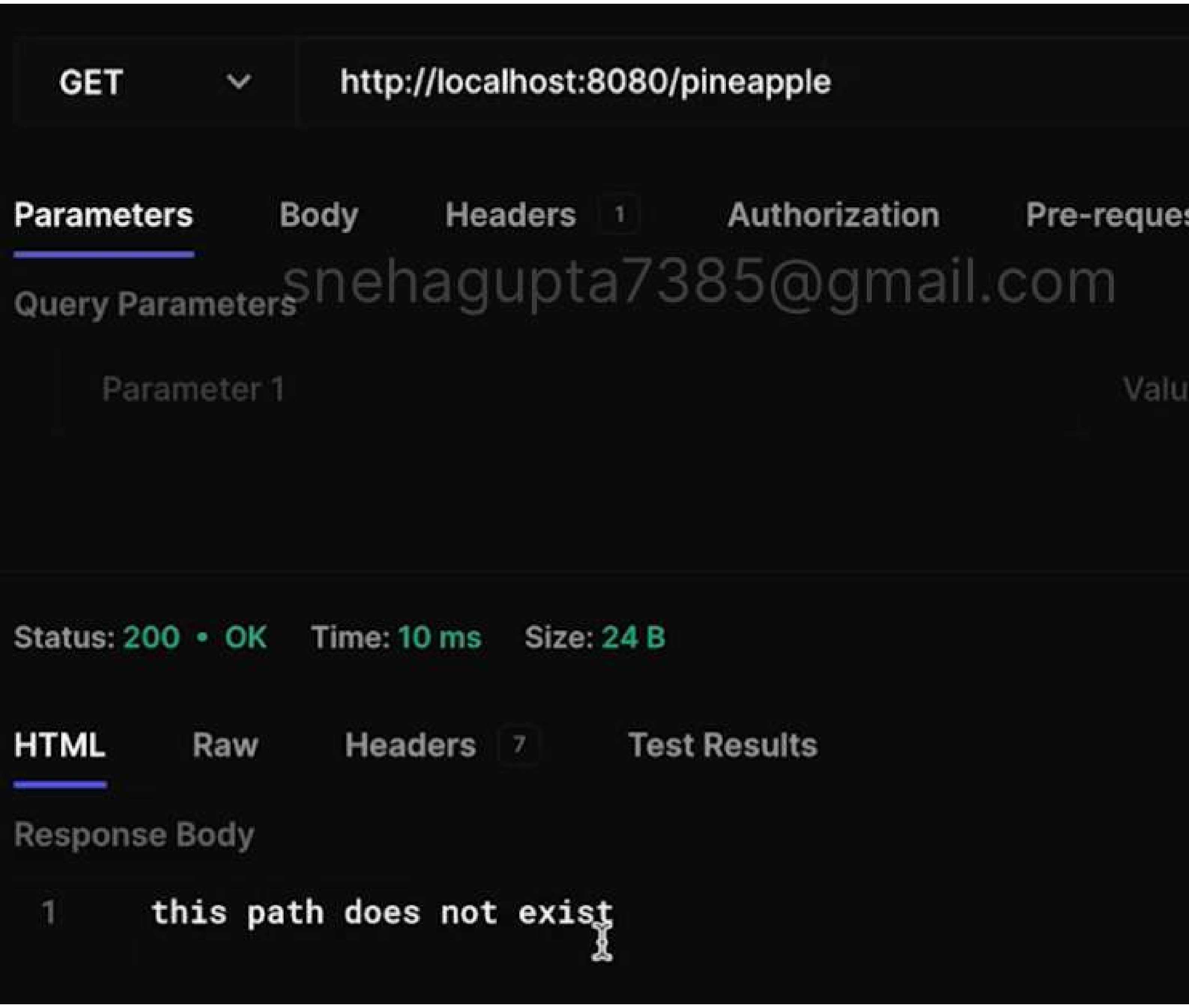
```
you contacted root path
```



http://localhost:8080/apple

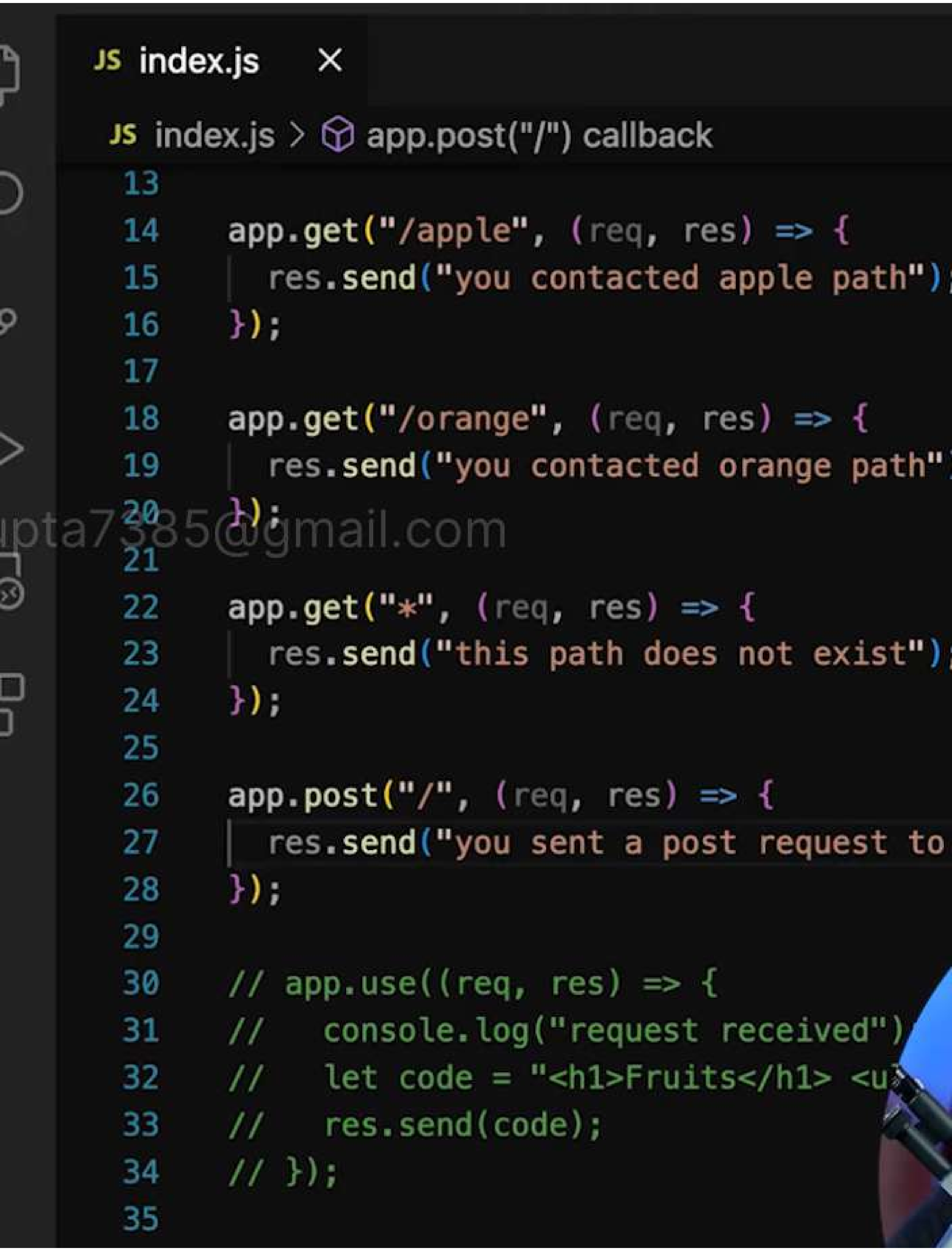
Status: 200 • OK Time: 12 ms Size: 24 B

```
you contacted apple path
```

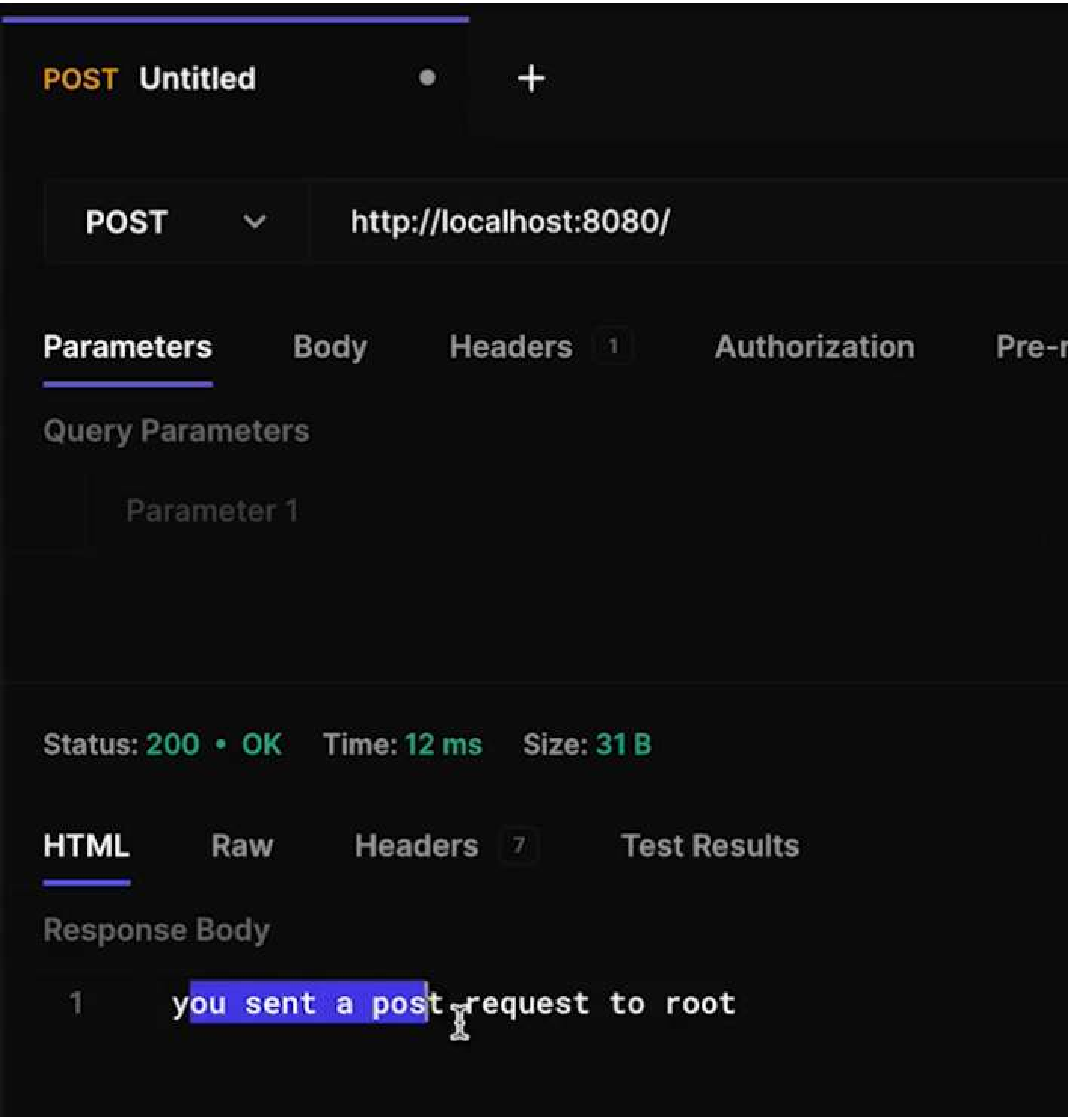


Status: 200 • OK Time: 10 ms Size: 24 B

```
this path does not exist
```



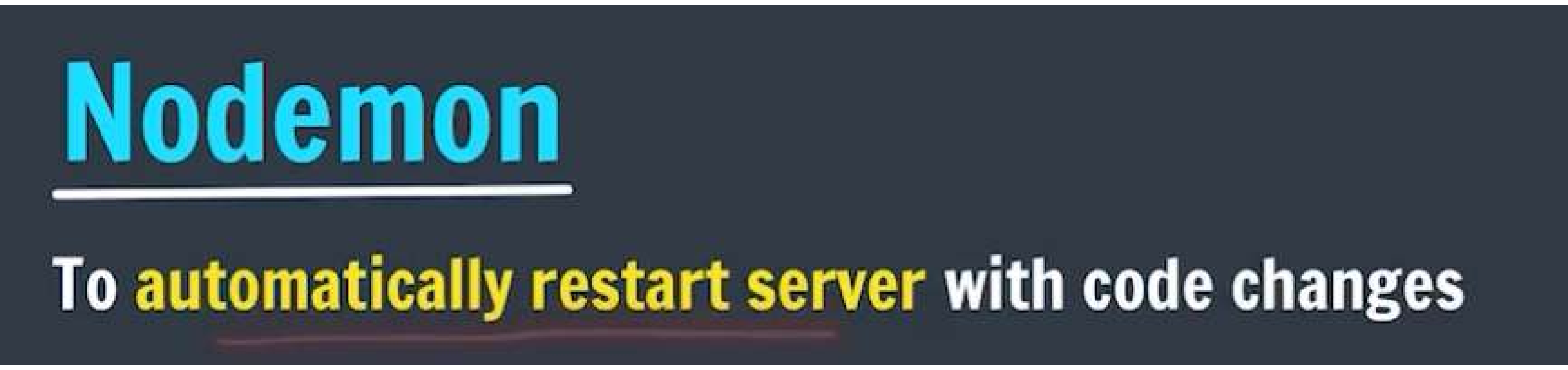
```javascript

app.get("/apple", (req, res) => {
  res.send("you contacted apple path");
});

app.get("/orange", (req, res) => {
  res.send("you contacted orange path"
});

app.get("*", (req, res) => {
  res.send("this path does not exist");
});

app.post("/", (req, res) => {
  res.send("you sent a post request to
});

// app.use((req, res) => {
//   console.log("request received")
//   let code = "<h1>Fruits</h1> <u
//   res.send(code);
// });

```



POST http://localhost:8080/

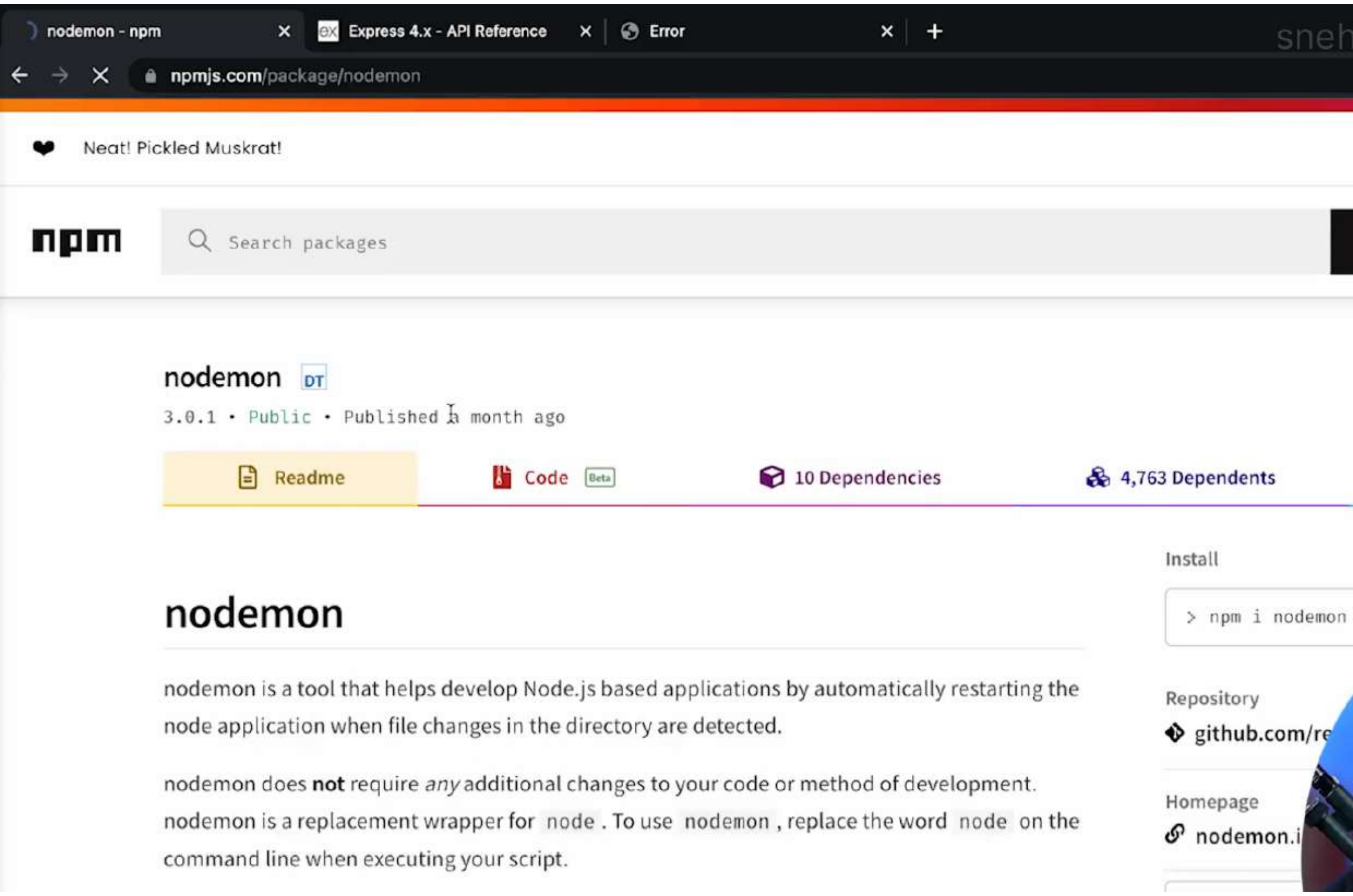
Status: 200 • OK Time: 12 ms Size: 31 B

```
you sent a post request to root
```



Nodemon

To automatically restart server with code changes



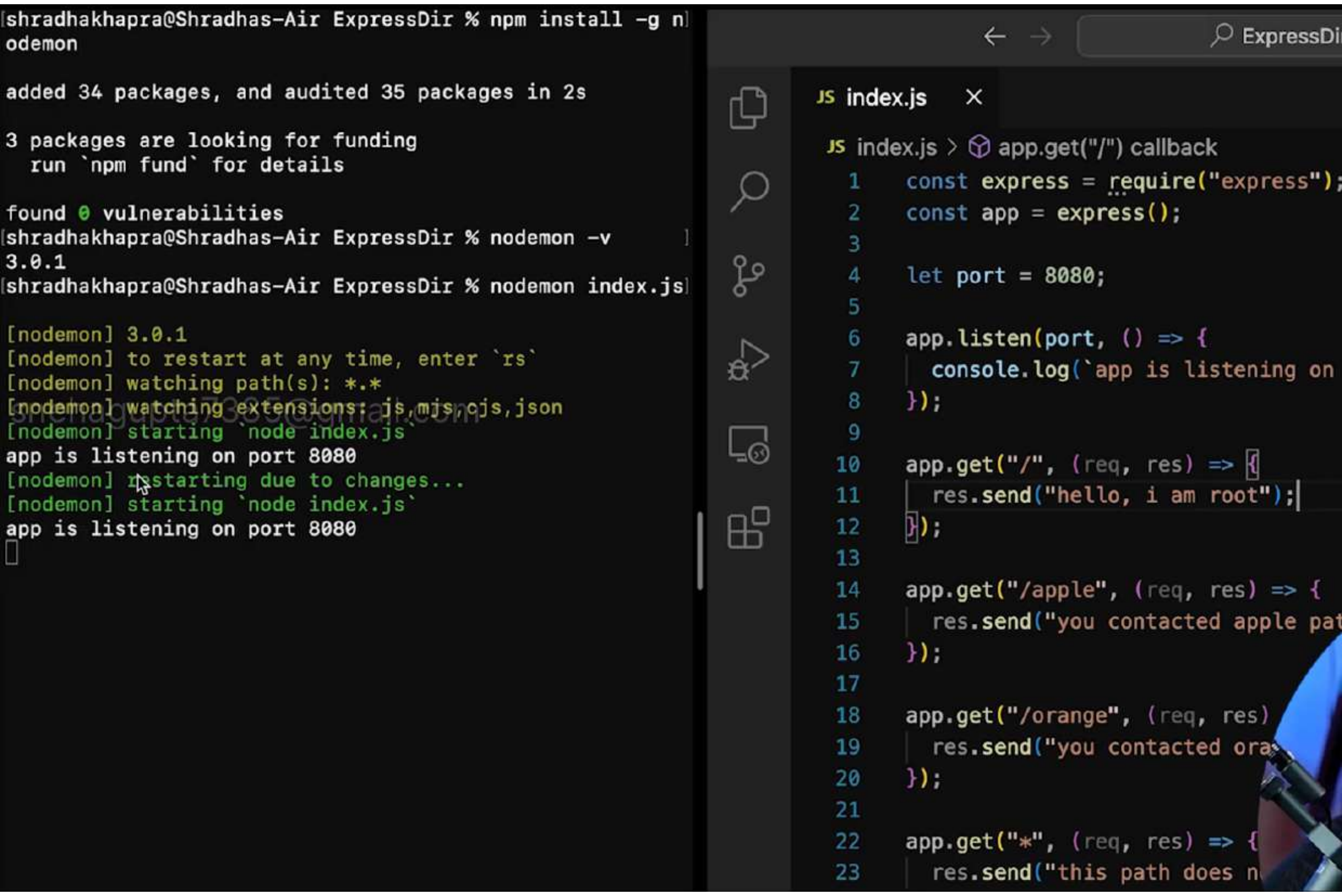
nodemon DT

3.0.1 • Public • Published a month ago

nodemon

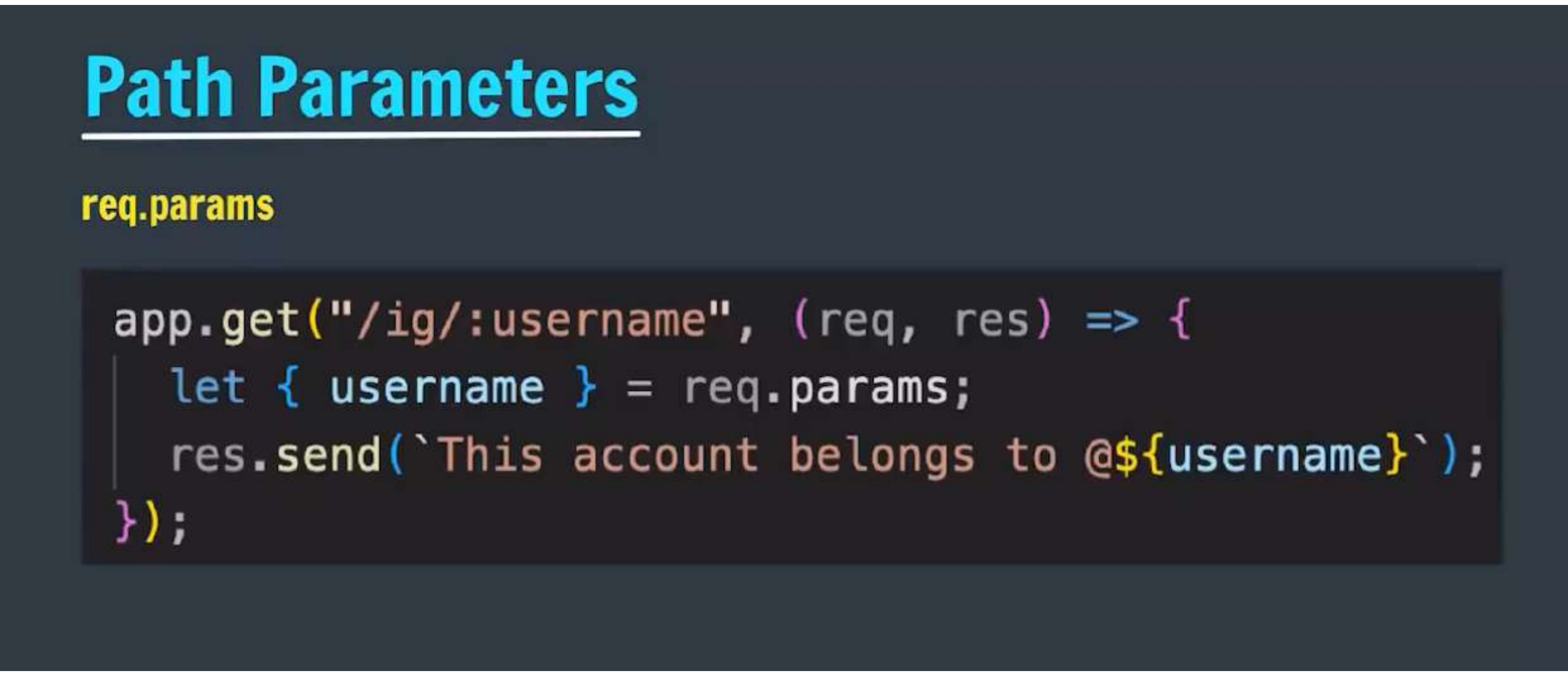
nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.

nodemon does **not** require *any* additional changes to your code or method of development. nodemon is a replacement wrapper for `node`. To use `nodemon`, replace the word `node` on the command line when executing your script.



```
shradhakhapra@Shradhas-Air ExpressDir % npm install -g n
odemon

added 34 packages, and audited 35 packages in 2s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
shradhakhapra@Shradhas-Air ExpressDir % nodemon -v
3.0.1
shradhakhapra@Shradhas-Air ExpressDir % nodemon index.js

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
app is listening on port 8080
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
app is listening on port 8080
```

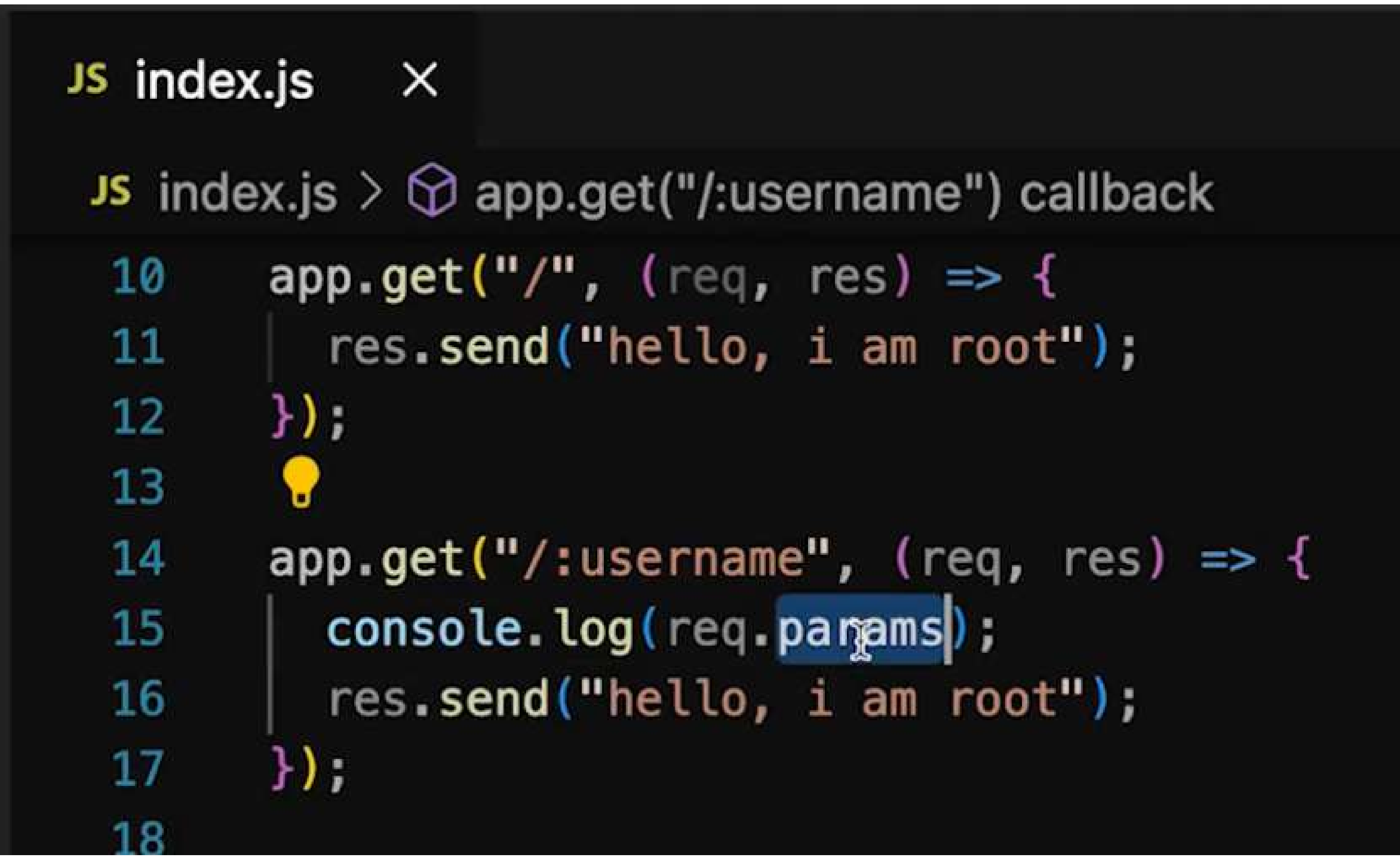
```javascript
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
  console.log(`app is listening on
});

app.get("/", (req, res) => {
  res.send("hello, i am root");
});

app.get("/apple", (req, res) => {
  res.send("you contacted apple pat
});

app.get("/orange", (req, res)
  res.send("you contacted ora
});

app.get("*", (req, res) => {
  res.send("this path does n
```



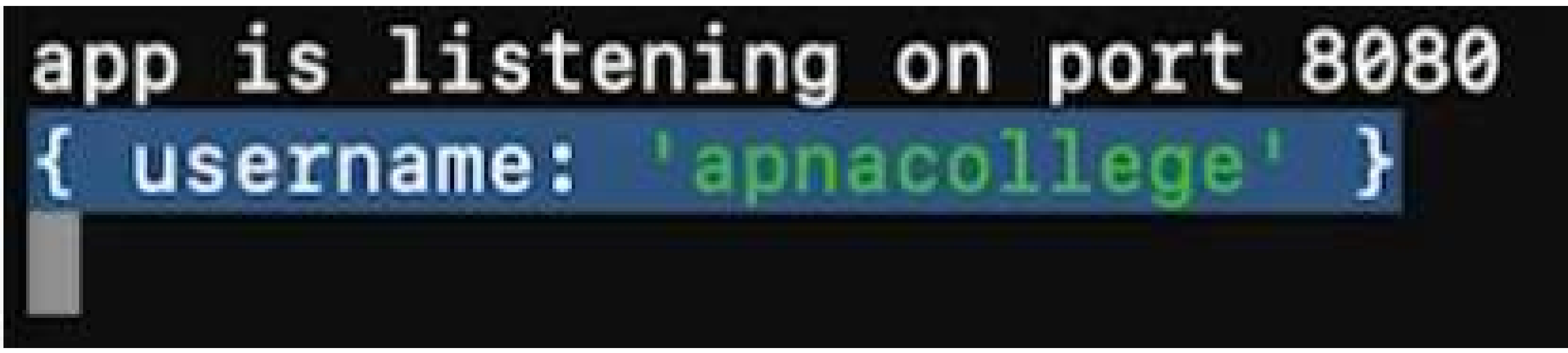
Path Parameters

req.params

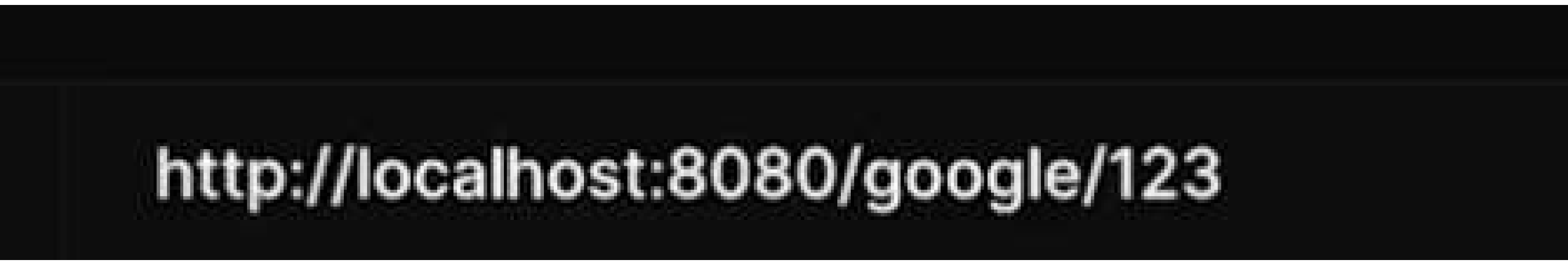
```
app.get("/ig/:username", (req, res) => {
  let { username } = req.params;
  res.send(`This account belongs to @${username}`);
});
```



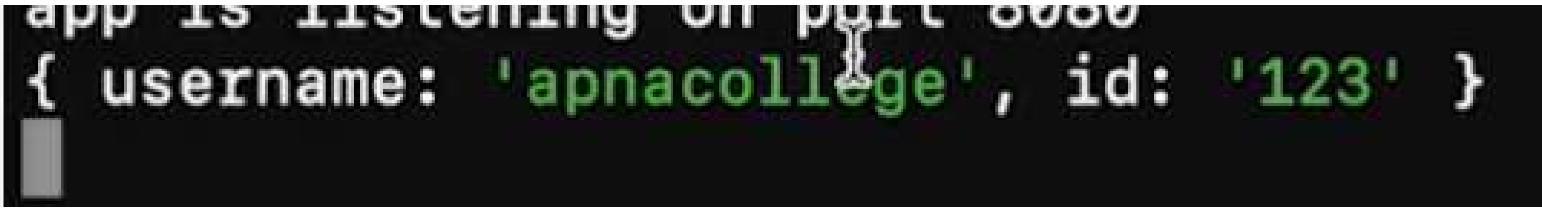
```
app.get("/", (req, res) => {
  res.send("hello, i am root");
});

app.get("/:username", (req, res) => {
  console.log(req.params);
  res.send("hello, i am root");
});
```



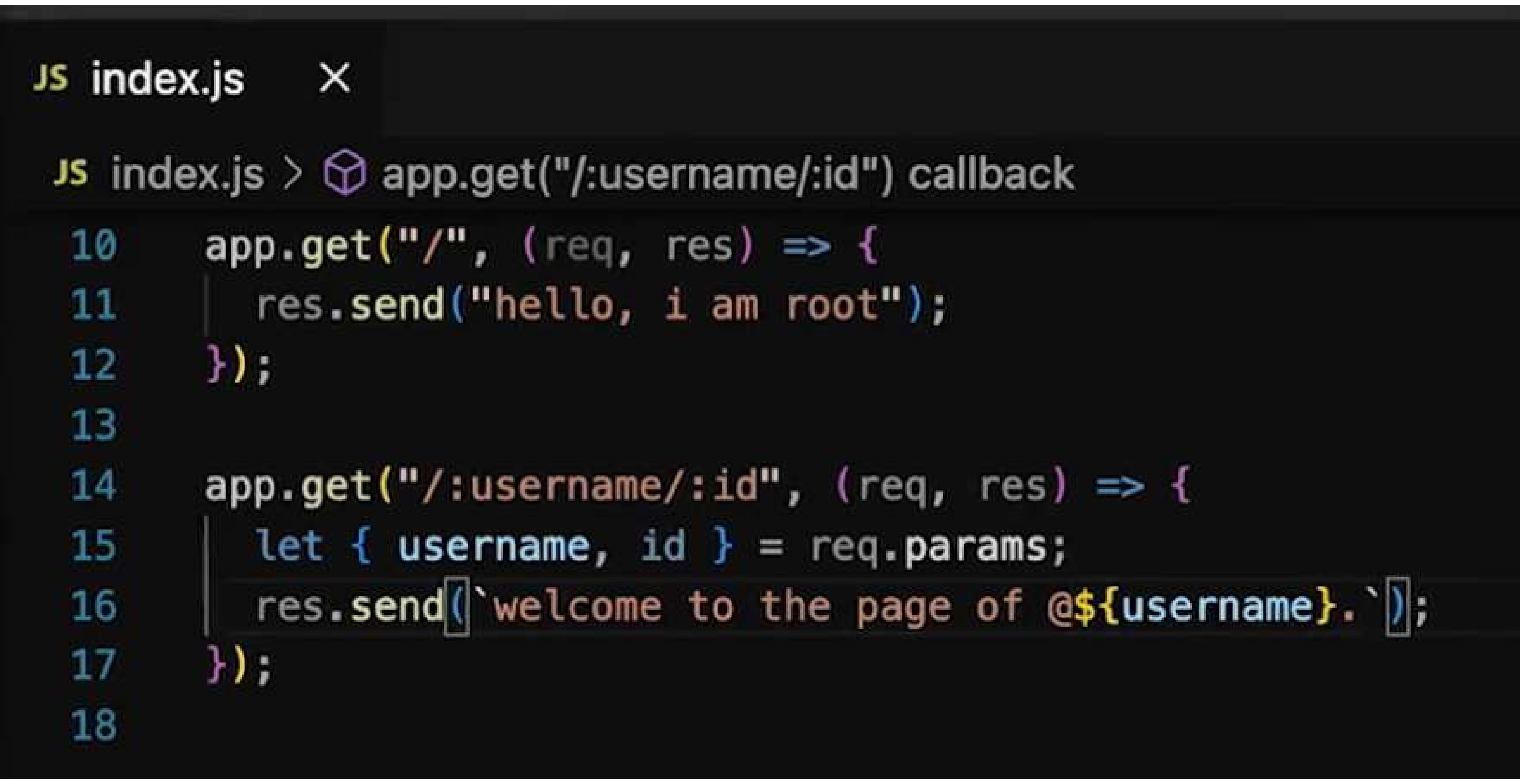
```
app is listening on port 8080
{ username: 'apnacollege' }
```



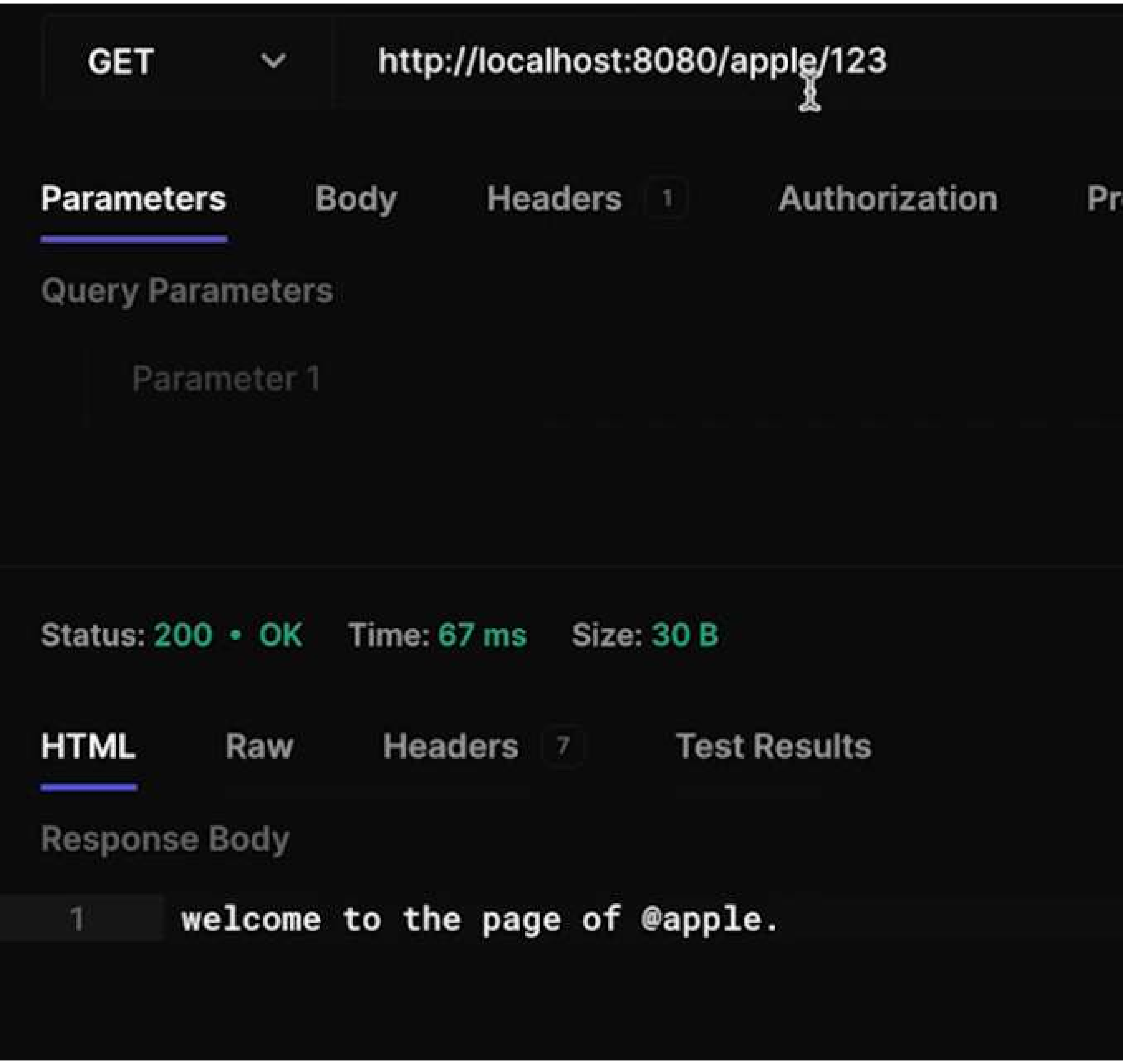
http://localhost:8080/google/123



```
app is listening on port 8080
{ username: 'apnacollege', id: '123' }
```



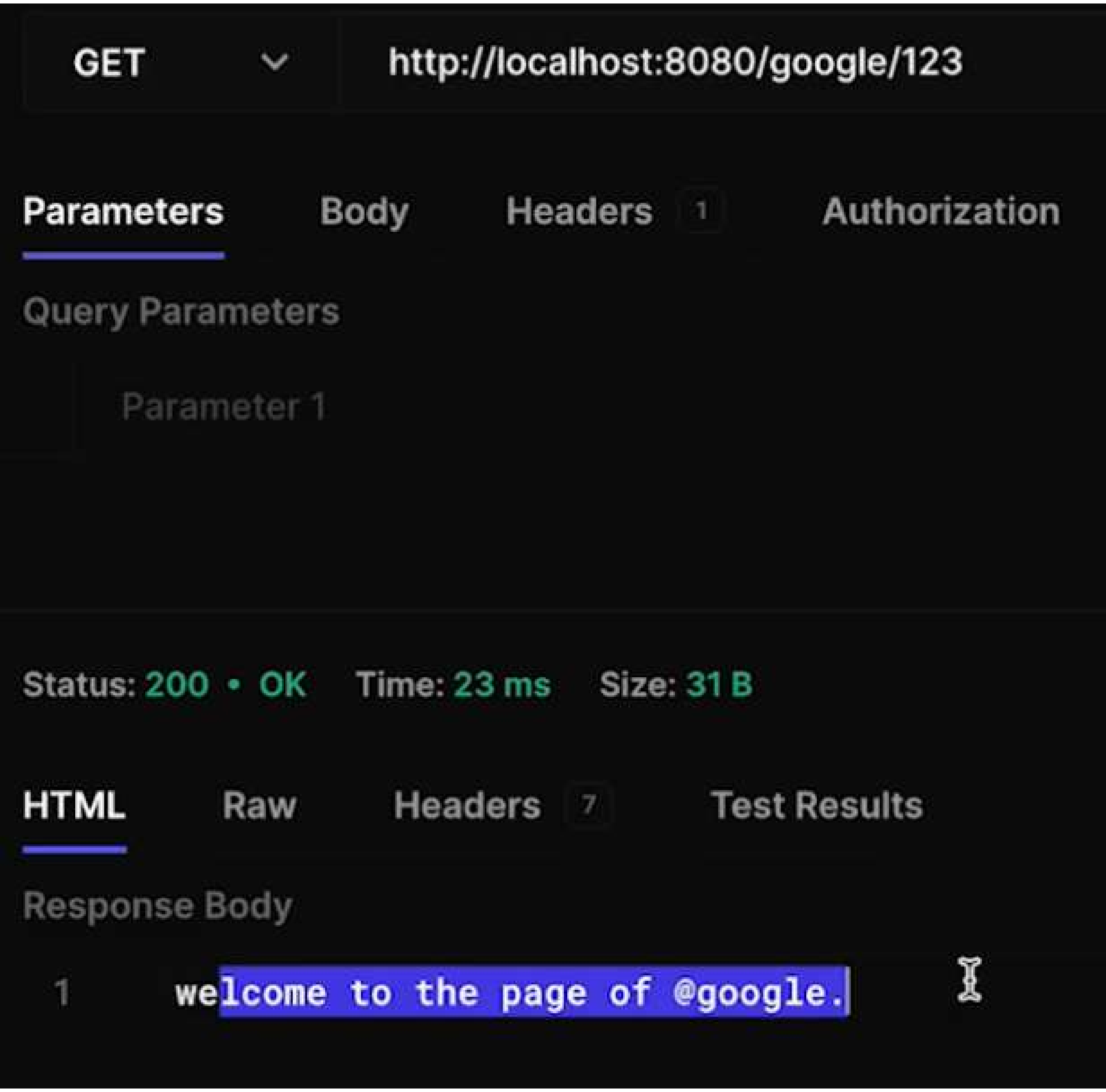
```js
app.get("/", (req, res) => {
  res.send("hello, i am root");
});

app.get("/:username/:id", (req, res) => {
  let { username, id } = req.params;
  res.send(`welcome to the page of @${username}.`);
});

```



```
GET http://localhost:8080/apple/123
```

Status: 200 • OK Time: 67 ms Size: 30 B

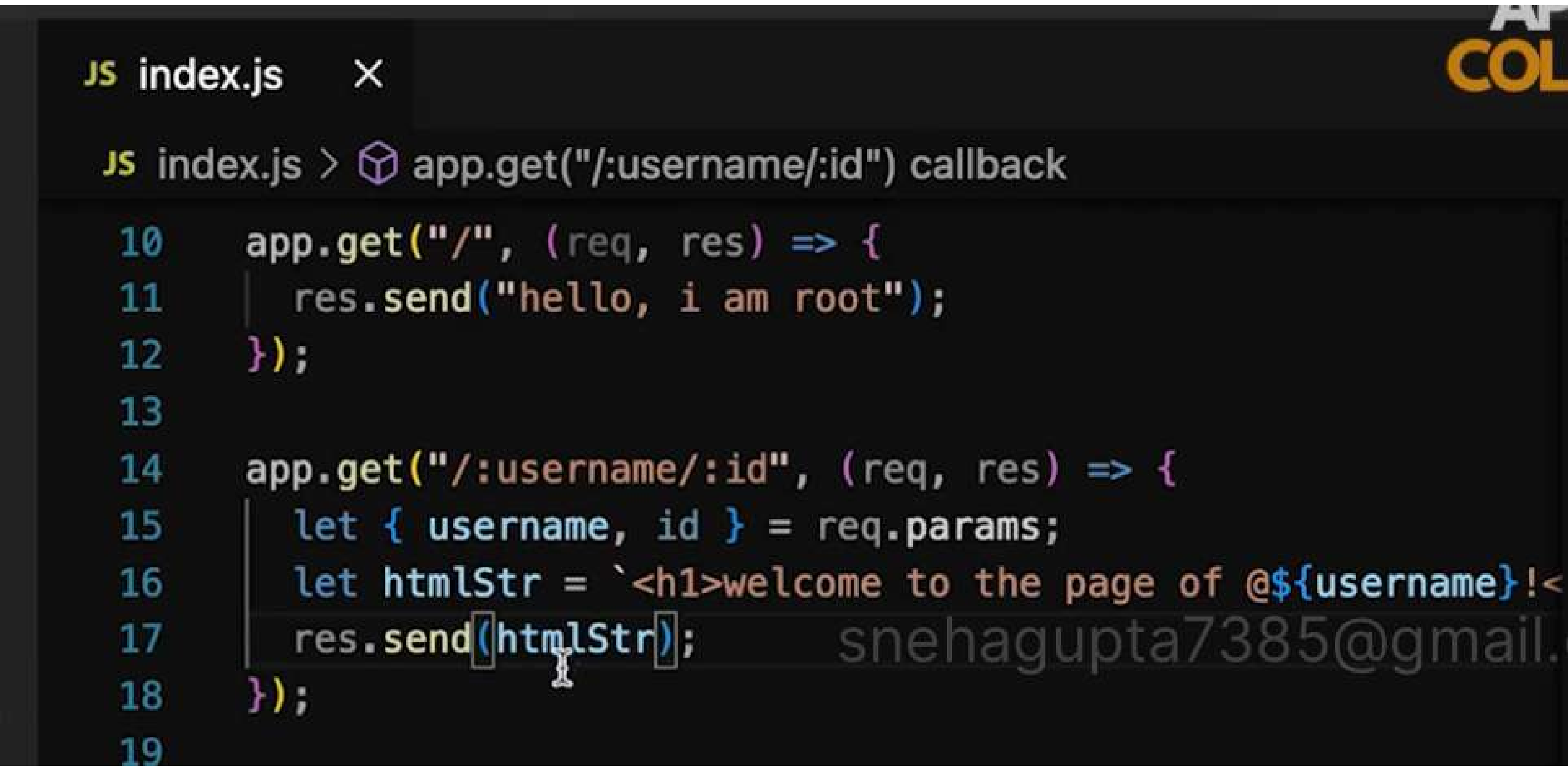
```
welcome to the page of @apple.
```



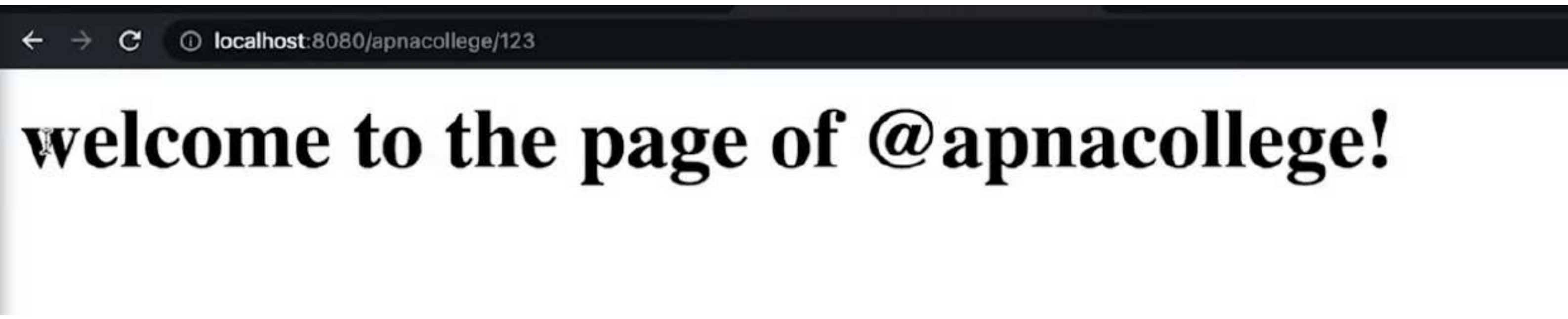
```
http://localhost:8080/google/123
```

Status: 200 • OK Time: 23 ms Size: 31 B

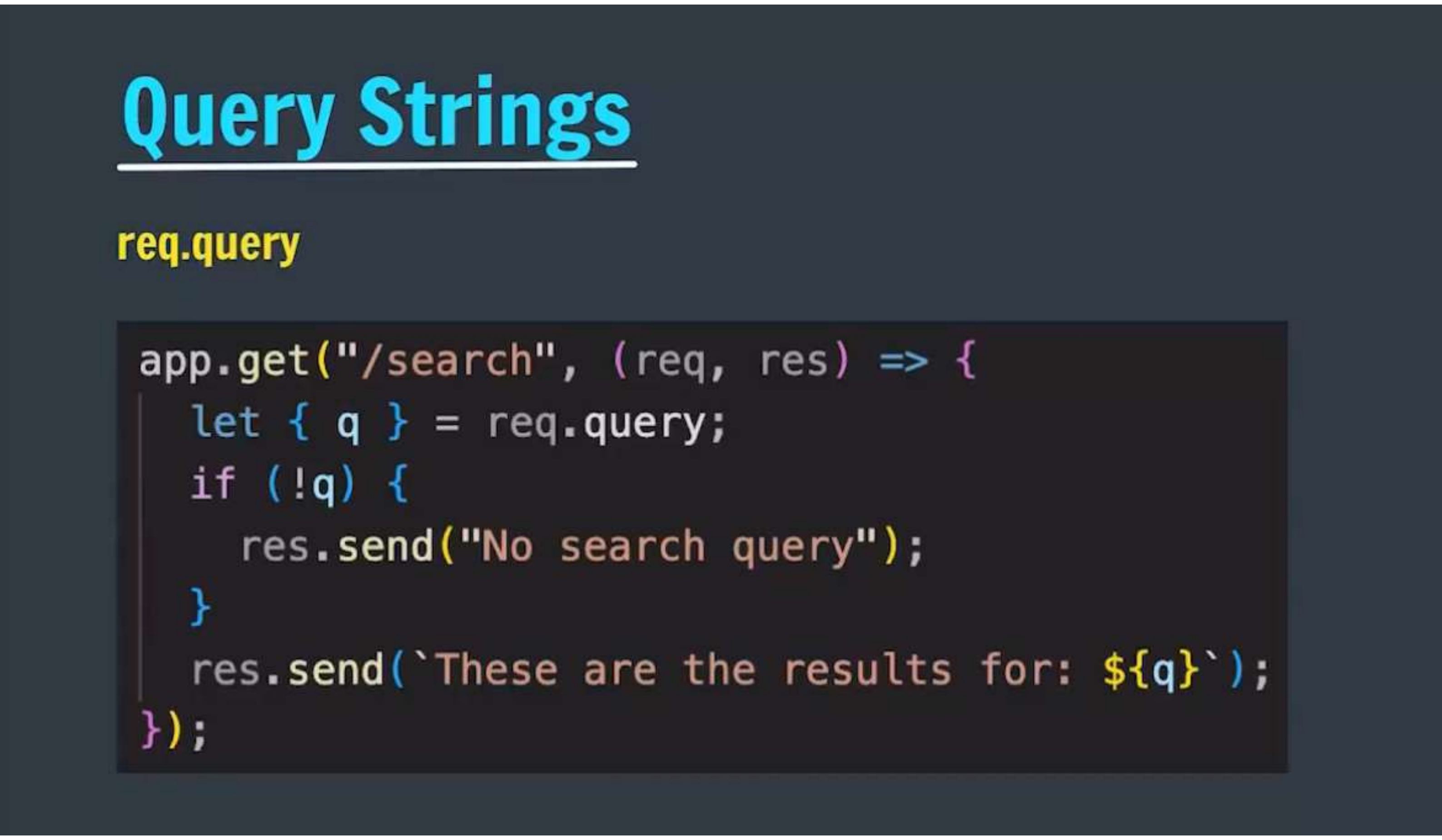
```
welcome to the page of @google.
```



```
app.get("/", (req, res) => {
  res.send("hello, i am root");
});

app.get("/:username/:id", (req, res) => {
  let { username, id } = req.params;
  let htmlStr = `<h1>welcome to the page of @${username}!<
  res.send(htmlStr);
});
```


localhost:8080/apnacollege/123

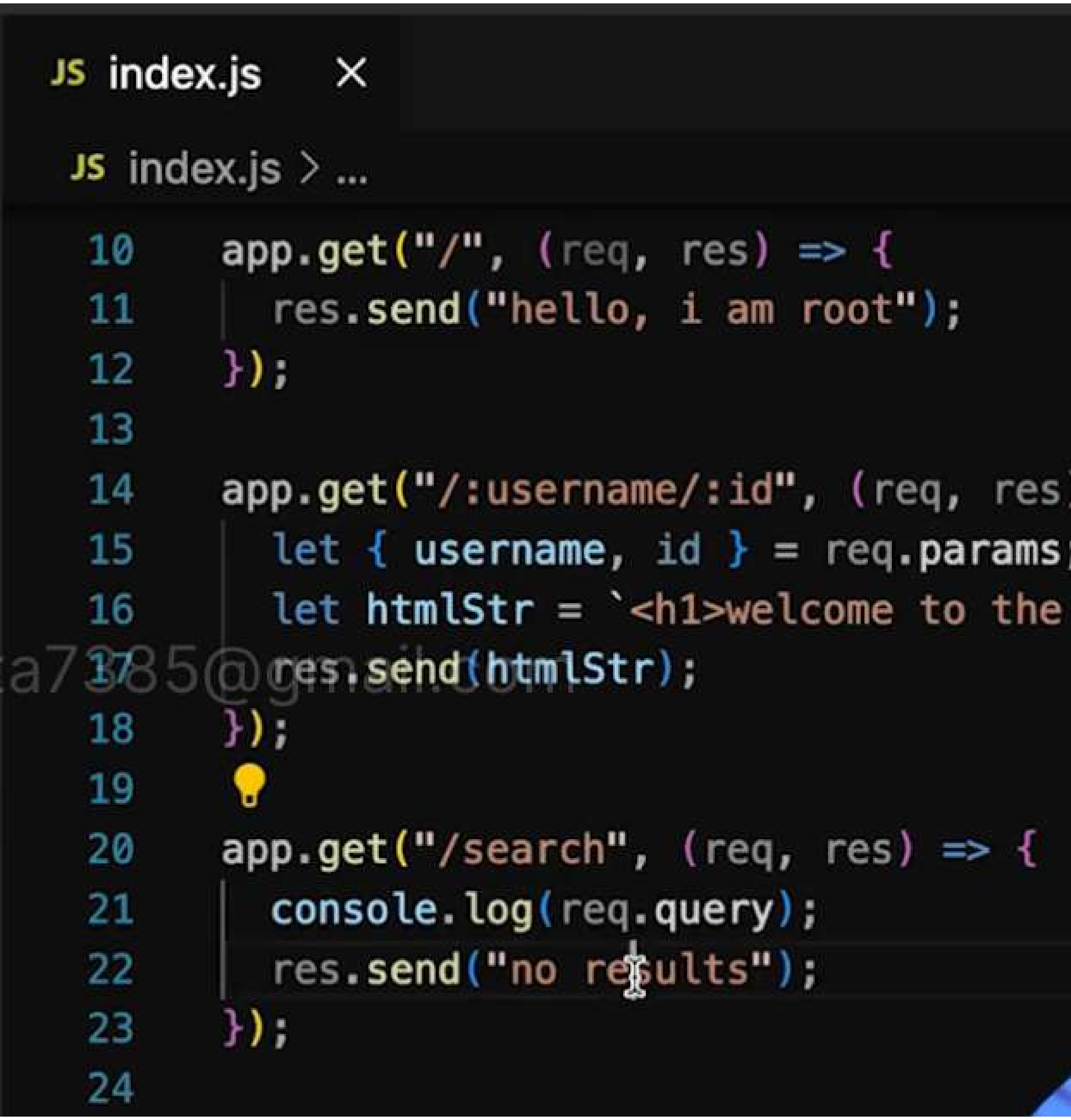
welcome to the page of @apnacollege!



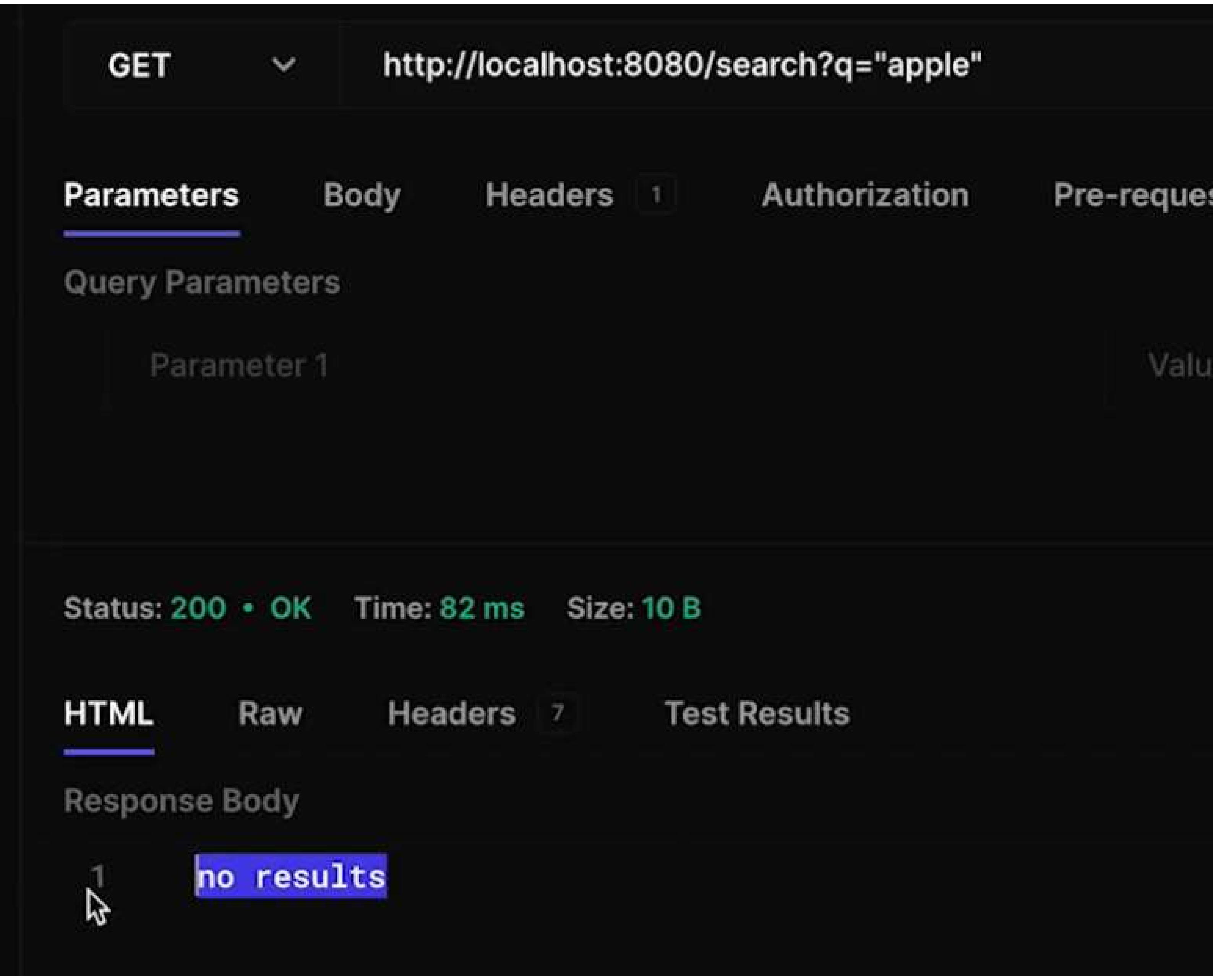
Query Strings

req.query

```
app.get("/search", (req, res) => {
  let { q } = req.query;
  if (!q) {
    res.send("No search query");
  }
  res.send(`These are the results for: ${q}`);
});
```

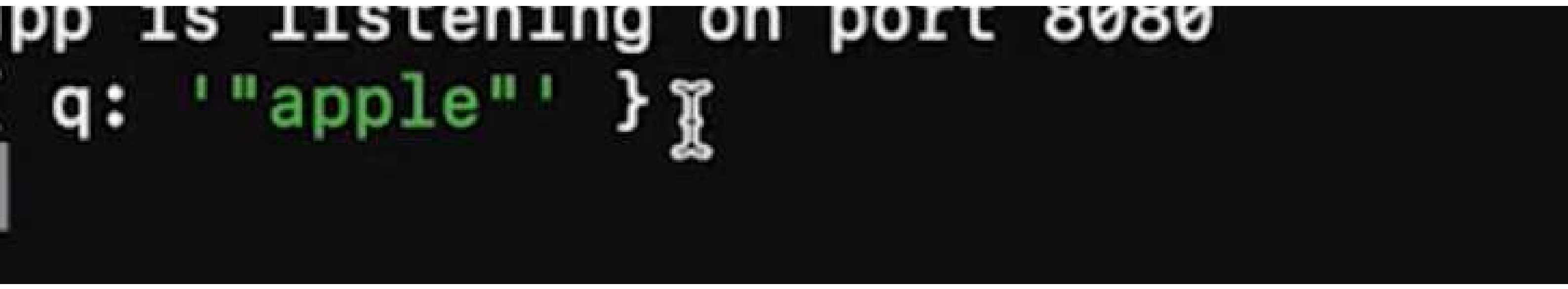


```javascript
app.get("/", (req, res) => {
  res.send("hello, i am root");
});

app.get("/:username/:id", (req, res
  let { username, id } = req.params
  let htmlStr = `<h1>welcome to the
  res.send(htmlStr);
});

app.get("/search", (req, res) => {
  console.log(req.query);
  res.send("no results");
});

```

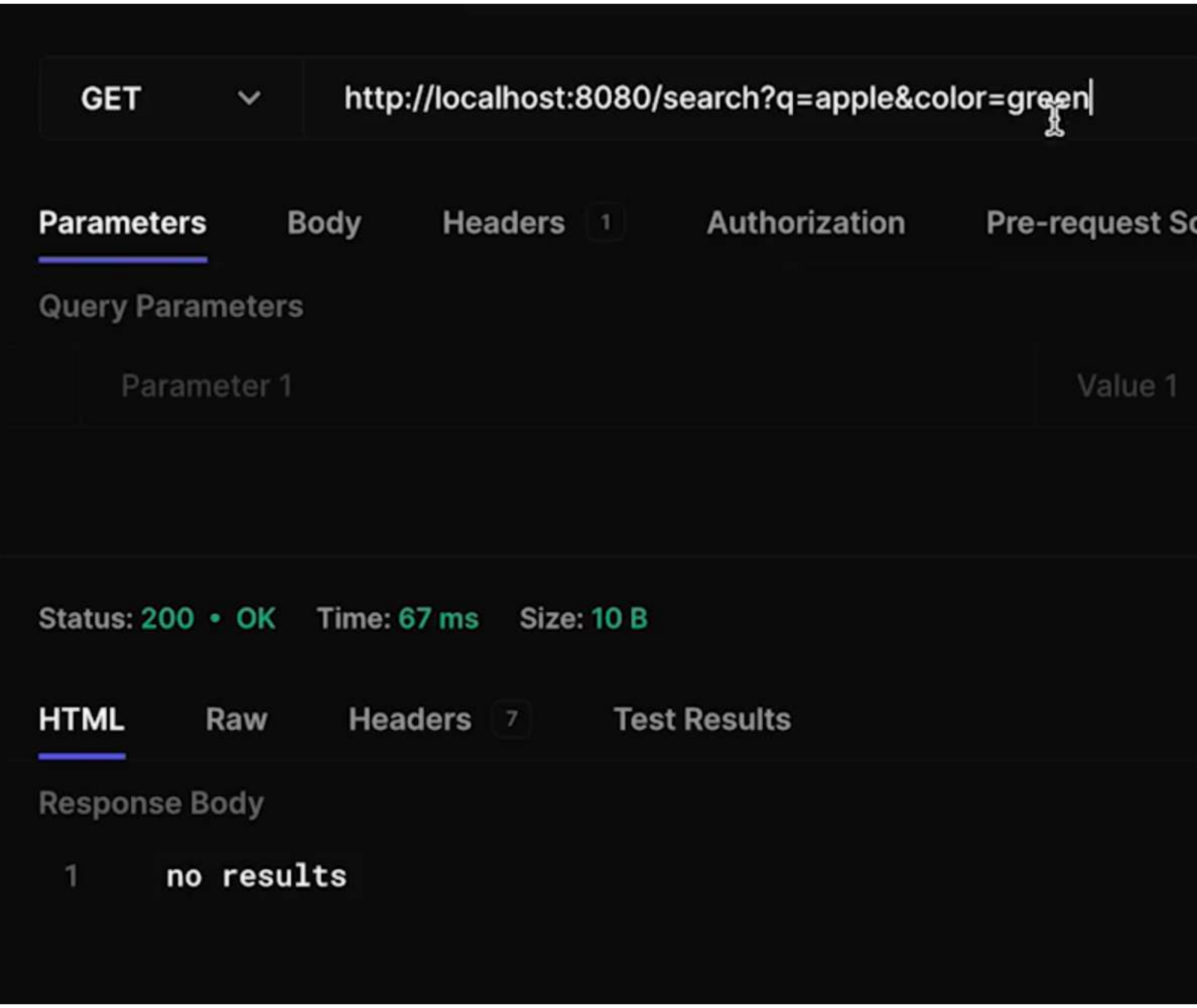

GET http://localhost:8080/search?q="apple"

Status: 200 • OK Time: 82 ms Size: 10 B

```
no results
```



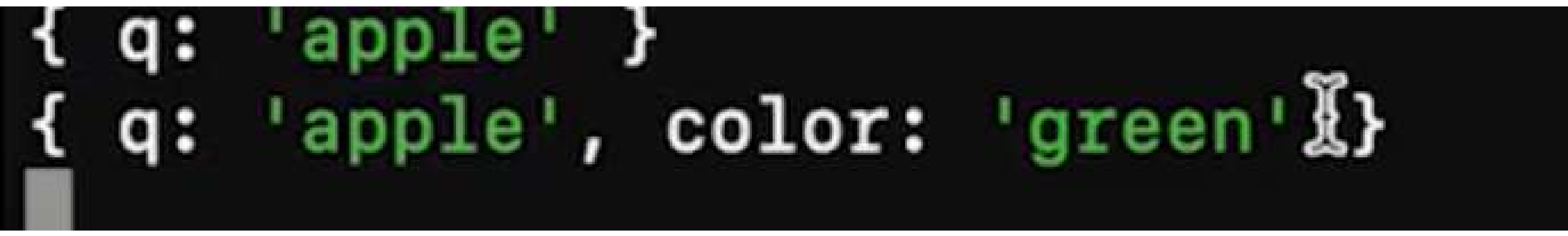
```
pp is listening on port 8080
 q: '"apple"' }
```



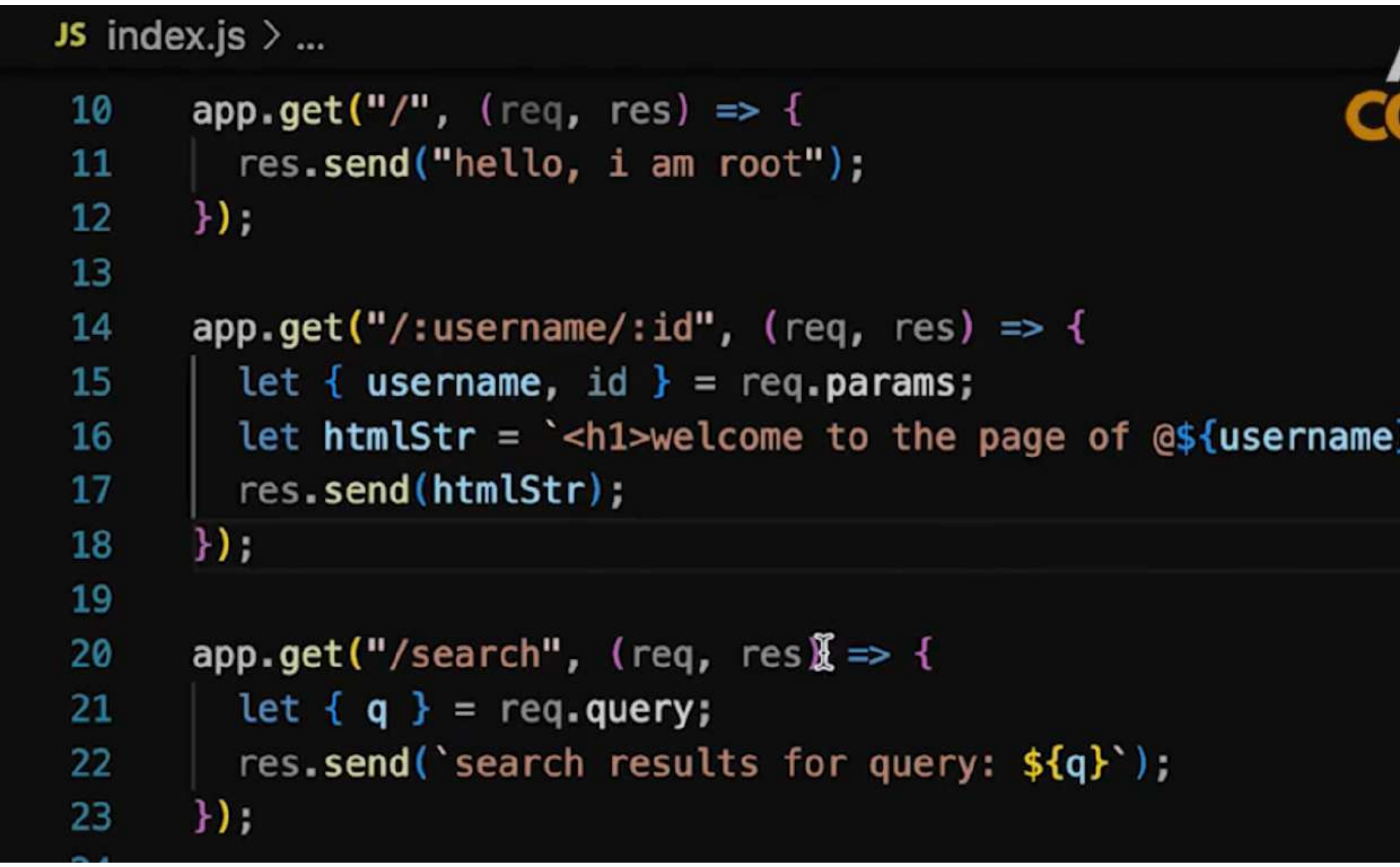
```
GET    http://localhost:8080/search?q=apple&color=green
```

Status: 200 • OK   Time: 67 ms   Size: 10 B

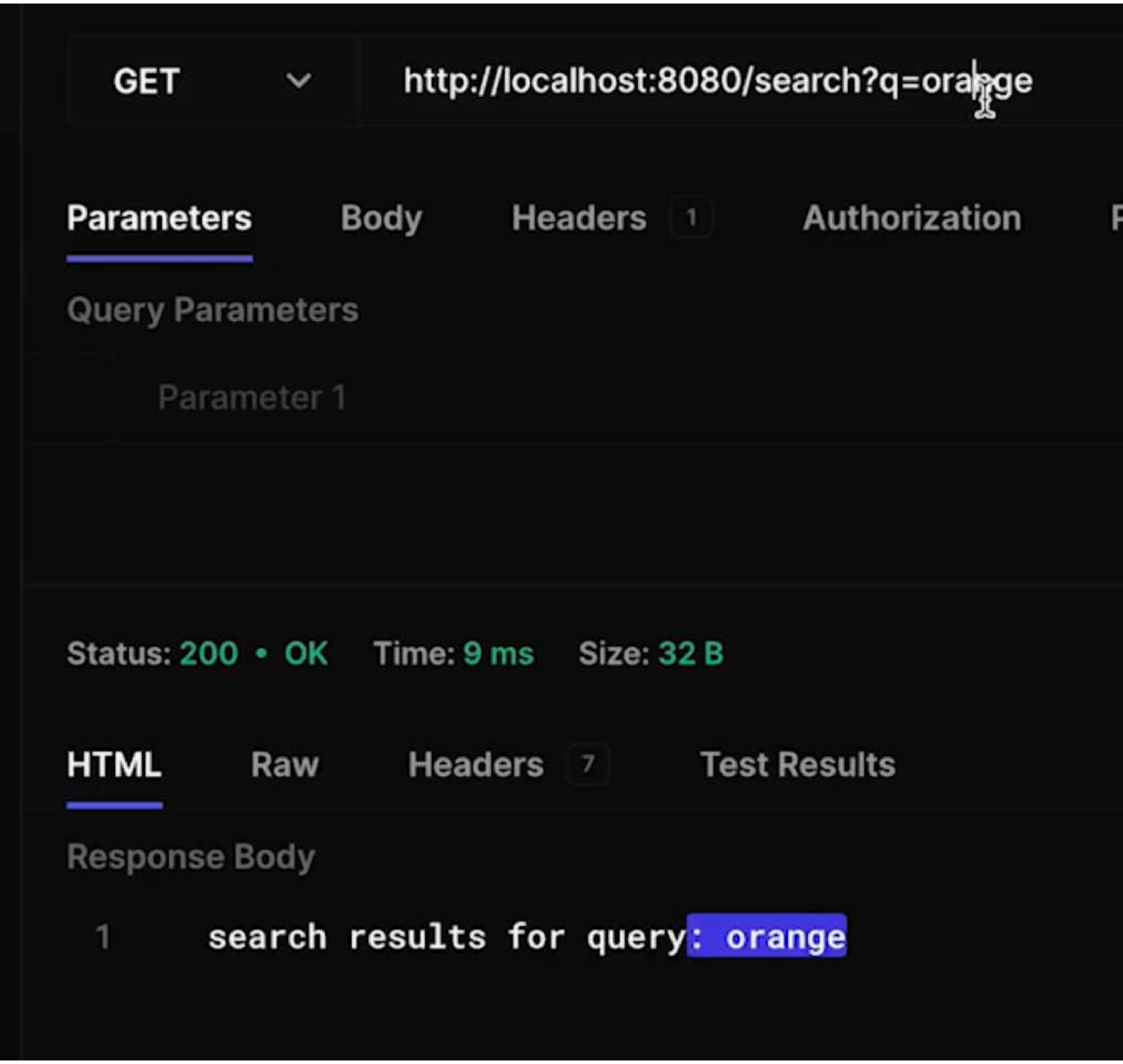
```
no results
```



```
{ q: 'apple' }
{ q: 'apple', color: 'green' }
```


JS index.js > ...

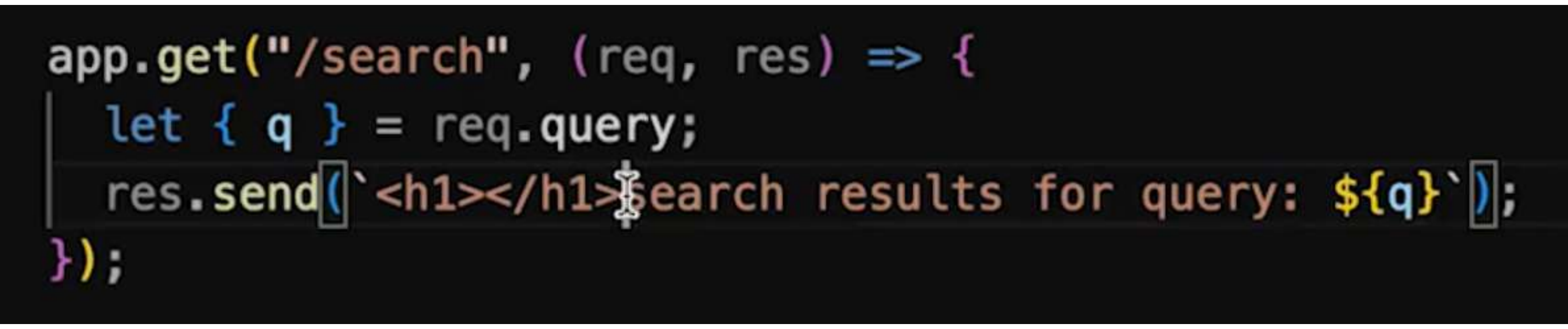
```
app.get("/", (req, res) => {
  res.send("hello, i am root");
});

app.get("/:username/:id", (req, res) => {
  let { username, id } = req.params;
  let htmlStr = `<h1>welcome to the page of @${username
  res.send(htmlStr);
});

app.get("/search", (req, res) => {
  let { q } = req.query;
  res.send(`search results for query: ${q}`);
});
```



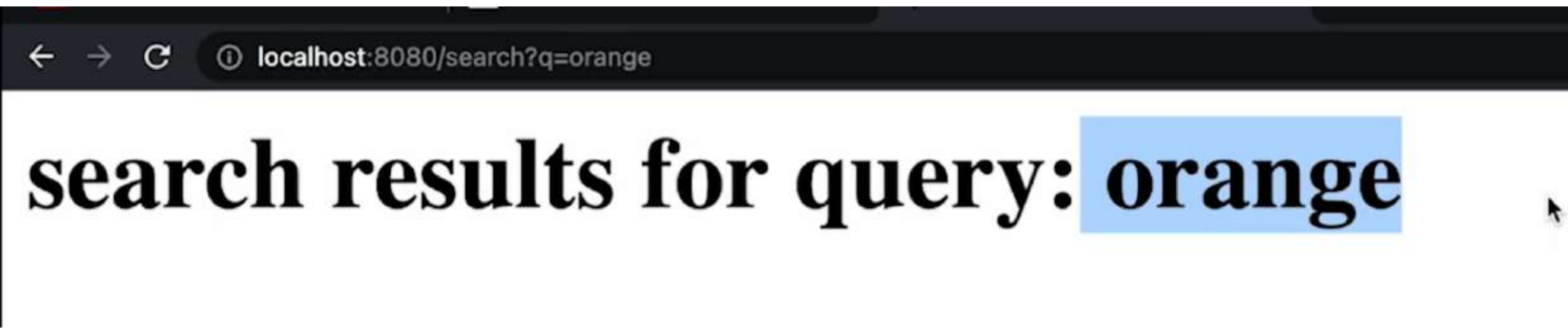
```
http://localhost:8080/search?q=orange
```

Status: 200 • OK Time: 9 ms Size: 32 B

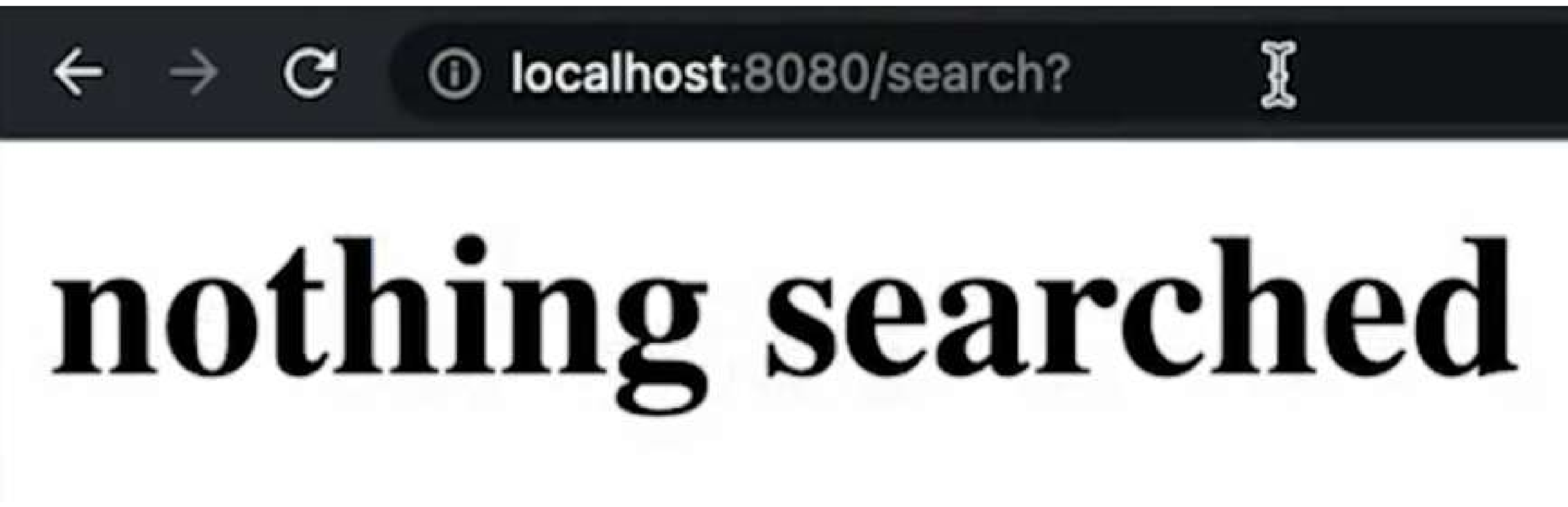
```
search results for query: orange
```



```
app.get("/search", (req, res) => {
  let { q } = req.query;
  res.send(`<h1></h1>Search results for query: ${q}`);
});
```


localhost:8080/search?q=orange

search results for query: orange



nothing searched