

# HTTP API HW II

This assignment is due by the date on the dropbox.

## **Assignment:**

Fork & Download the ***HTTP-API-Assignment-II*** starter files from our Github organization.  
<https://github.com/IGM-RichMedia-at-RIT/http-api-assignment-ii>

In this assignment, you will use Node.js to build a simple HTTP API using the GET, HEAD & POST methods. Responses will be in json and use the appropriate status codes. You do not need to implement xml responses or client code to handle xml for this assignment. You will still need to write custom client code for sending requests, etc.

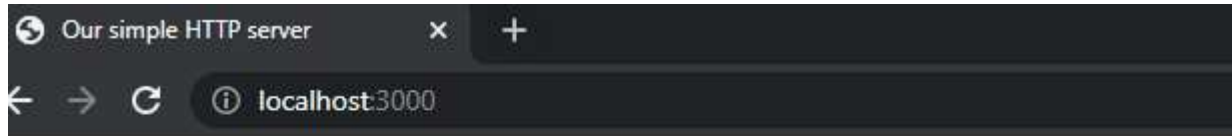
You will need to build an HTTP API that satisfies the following criteria:

- Loads the client page provided as the home page.
- Loads the css file into the client page as text/css.
- Loads the following URLs, both with fetch() when hitting the 'send' button **and** directly to the URL (directly to URL will fire a GET request). Client fetch() requests should be made to the selected url with the selected method when the "Get User" button is pressed.
  - /getUsers with GET retrieves 200 success with results.
  - /getUsers with HEAD retrieves 200 success without results.
  - /notReal with GET retrieves a 404 not found with error message.
  - /notReal with HEAD retrieves a 404 not found without error message.
  - Any other page with a 404 status code
- POST to the following URL with fetch() when the "Add User" button is pressed.
  - /addUser: Should return a 201 status code when adding a new user, a 204 when updating the age of an existing user, and a 400 if the request is missing a name, age, or both.
- In fetch requests, client should NOT parse JSON responses for HEAD requests.
- In fetch requests, client should parse JSON responses for other responses.
- In fetch requests, status codes should be used/handled appropriately.
- JSON failures should have a human readable "message" field, as well as an unique text "id" field that is useful for developers.

# HTTP API HW II

## Examples of Correct Output:

The home page '/' (client.html and style.css)



### POST Status Code Tests

Name:  Age:

/getUsers GET request after hitting "Get User" *BEFORE* adding a user. Should get a 200 status code in the inspector's network tab. Note that to get the "{}" output for the users object, you may need to use the JSON.stringify function, otherwise it will print out as [object Object].

### POST Status Code Tests

Name:  Age:

## Success

{}

Example output of the JSON response printed to the console.

```
▼ Object ⓘ  
  ▼ users: Object
```

# HTTP API HW II

/getUsers HEAD request after hitting 'Send' with /getUsers and 'HEAD' *BEFORE* adding a user. (HEAD requests do not have a body, just status codes and meta data like content-type). Should be a 200 status code.

HEAD requests can be used to see the size or file type/format of a file before downloading. Similarly, HEAD requests can see if a file has changed since last time it was downloaded (images, CSS files, JS files, etc). If it's the same file as other places on the site, there may be no point in downloading the same file again.

## POST Status Code Tests

Name:  Age:

## Success

---

/addUser after hitting 'Add User' with an appropriate name and age. Should get a 201 status.

## POST Status Code Tests

Name:  Age:

## Created

Message: Created Successfully

Example output of JSON response.

```
▼ Object 1
  message: "Created Successfully"
```

---

# HTTP API HW II

/addUser after hitting 'Add User' with an *existing* name and new age.  
This will throw a 204. There is no response body in a 204 message.

## POST Status Code Tests

Name:  Age:

## Updated (No Content)

---

/addUser after hitting 'Add User' without a name or age. Should get a 400 status.

## POST Status Code Tests


Name:  Age:

## Bad Request

Message: Name and age are both required.

Example output of JSON response.

```
POST http://localhost:3000/addUser 400 (Bad Request)
```

Object 

```
id: "addUserMissingParams"
message: "Name and age are both required."
```

---

# HTTP API HW II

/getUsers GET request after hitting 'Get User' *AFTER* adding a user. Should be a 200 status code.

In this example, objects are indexed by the user's name, which is why the 'Test' object has the name 'Test'. If a 'test2' is added, then there would be another object called 'test2' with the name field set to 'test2'.

*You do not have to structure your objects quite like that as long as you get this to work correctly. This output is achieved by calling `JSON.stringify` on the `users` element of the response object and adding it to the content section.*

## POST Status Code Tests

Name:  Age:

## Success

```
{"Test": {"name": "Test", "age": "2"}}
```

Example output of JSON response.

```
▼ Object
  ▼ users: Object
    ▼ test: Object
      age: "2"
      name: "test"
```

---

/getUsers HEAD request after hitting 'Get User' *AFTER* adding a user. Status will be a 200.

HEAD requests do not have a body, just status codes and meta data like content-type. This means that even after adding, there will not be a body. The response will contain content-type and all headers that a GET request should have.

# HTTP API HW II

## POST Status Code Tests

Name:  Age:

## Success

---

/notReal GET request after hitting 'Get User'. Should get a 404 status code.

## POST Status Code Tests

Name:  Age:

## Not Found

Message: The page you are looking for was not found.

Example output of JSON response.

```
GET http://127.0.0.1:3000/notReal 404 (Not Found)
Object
  id: "notFound"
  message: "The page you are looking for was not found."
```

---

/notReal HEAD request after hitting 'Get User'. HEAD requests do not have a body, just status codes and meta data like content-type. Should get a 404 status code.

# HTTP API HW II

## POST Status Code Tests

Name:  Age:

## Not Found

Example output in console

```
HEAD http://127.0.0.1:3000/notReal 404 (Not Found)
```

RICH MEDIA WEB APP DEV II

# HTTP API HW II

Direct call to /style.css

```
← → ↻ ⓘ 127.0.0.1:3000/style.css

html, body {
  height: 100%;
  width: 100%;
  margin: 0;
  padding: 0;
}

#top {
  text-align: center;
  margin: 0 auto;
}

#top h1 {
  font-size: 2em;
}

#content {
  margin-top: 100px;
  font-size: 2em;
  text-align: center;
}
```

Direct call to /getUsers BEFORE adding a user.

```
← → ↻ ⓘ 127.0.0.1:3000/getUsers

{"users":{}}
```

Direct call to /getUsers AFTER adding a user.

```
← → ↻ ⓘ 127.0.0.1:3000/getUsers

{"users":{"test":{"name":"test","age":"2"}}
```

Direct call to anything else

```
← → ↻ ⓘ 127.0.0.1:3000/theVoid

{"message":"The page you are looking for was not found.,"id":"notFound"}
```



# HTTP API HW II

## ESLint & Code Quality

You are required to use ESLint with the AirBnB configuration for code quality checks. The `eslint` config file was given to you in the starter files. Your **server** code must pass ESLint to get credit for this part. Warnings are okay, but errors must be fixed. I won't require ESLint on the client side code in this assignment. ***If you cannot fix an error or need help, please ask us!***

You are also required to use the "pretest" script in order to run ESLint. I should be able to run "`npm test`" and have ESLint scan your code for me.

## Continuous Integration

You are required to use CircleCI for continuous integration. Your build must succeed and be handed in with the assignment. The `circle.yml` config file was given to you in the starter files. CircleCI will build based on your `npm test` and ESLint configuration.

For instructions on how to do this, look at the ***CircleCI Setup*** instructions on mycourses.

RICH MEDIA WEB APP DEV II

# HTTP API HW II

## Rubric

DESCRIPTION	SCORE	VALUE %
<b>Loads the client page</b> – Loads the client.html page as the home page.		5
<b>Loads the css as text/css</b> – Loads the style.css file as text.css for the home page and directly.		5
<b>/getUsers with GET works correctly</b> – /getUsers with GET retrieves 200 success with results (even if empty). Should also work directly to URL.		10
<b>/getUsers with HEAD works correctly</b> – /getUsers with HEAD retrieves 200 success without results.		15
<b>/notReal or any other page with GET works correctly</b> – /notReal with GET retrieves a 404 not found with error message. Should also work directly to the URL.		10
<b>/notReal with HEAD works correctly</b> – /notReal with HEAD retrieves a 404 not found without error message.		15
<b>/addUser with POST and new valid data works correctly</b> – /addUser with POST and new valid data, adds a user and retrieves 201 with message.		15
<b>/addUser with POST and existing data works correctly</b> – /addUser with POST and existing valid data, updates a user and retrieves 204 without message.		15
<b>/addUser with POST and invalid data works correctly</b> – /addUser with POST and invalid data retrieves 400 with message.		5
<b>JSON responses formatted correctly</b> – JSON messages are formatted correctly. JSON errors contain a message field and error id field.		5
<b>Heroku Penalty</b> – If your app is not functional on Heroku, there will be a penalty.		-20% (this time)
<b>CircleCI Penalty</b> – If your build is not successful on CircleCI, there will be a penalty.		-20% (this time)
<b>ESLint Penalties</b> – I should be able to run 'npm test' and have ESLint scan your code without errors. Warnings are acceptable.  <b>Check your code with ESLint often during development!</b>  <b>If you cannot fix an error, please ask us!</b>	1 Error	-5%
	2 Errors	-10%
	3 Errors	-15%
	4 Errors	-20%
	5+ Errors	-30%
<b>Additional Penalties</b> – These are point deductions for run time errors, poorly written code or improper code. There are no set values for penalties. The more penalties you make, the more points you will lose.		
<b>TOTAL</b>		100%

## Submission:

By the due date please submit a zip of your project to the dropbox. **Do not** include the node\_modules folder in the zip or in your git repo.

In the submission comments include the following:

- a link to your Heroku App (not the dashboard but the working web link from 'open app'),
- a link to your Github repo (if private, please add the TA and myself).
- a link to your circleCI build *in the submission comments*.