



Integrify Preparation Materials for Full Stack Program

1. INTRODUCTION

This document will tell you about the basic concepts of Full Stack Development.

Make sure you learn from the documentation first – only then do the quiz. 💡

MATERIALS

[Github](#)

[HTML](#)

[CSS](#)

[JavaScript](#)

[React](#)

[Backend](#)

PRACTICAL ASSIGNMENTS

[Front-end // React](#)

[Back-end // Node.JS](#)


Happy coding! And remember: debugging is part of the fun! 🐛

After going through the whole preparation material, you can:

1. Declare and assign variables in JavaScript
2. Know the difference between let and const

3. Be able to write JavaScript functions, with parameters
4. Be able to use and manipulate objects
5. How to use "this" keyword
6. Manipulate HTML elements with JavaScript
7. How to fetch data in JavaScript
8. Understand and how to use array methods: map, filter, find
9. Able to write react functional components with hooks
10. Understand software architecture style, NodeJS
11. NodeJS principle: single thread, non blocking IO
12. Create a simple serve using NodeJS or Express

AND

13. Create your own portfolio website?! Please do! Start for instance with HTML & CSS:  Build your portfolio using only HTML & CSS - Tutorial

Find some guidance & inspiration here:

<https://blog.ioanatiplea.dev/dos-and-donts-of-creating-your-portfolio-website>

2. MATERIALS

2.1. Github

- Git and GitHub Full course: <https://youtu.be/nvSszqsQYGQ>
- Useful document about the use of GitHub with tips and tricks: <https://integrify.academy/blogs/github-101-introduction-to-github-for-a-better-portfolio>


2.2. HTML

- [HTML Crash Course](#)
- HTML Full course with project: <https://youtu.be/TWUf6eSi2K8>

Done? Then test your skills:

 Quiz: https://www.w3schools.com/html/html_quiz.asp



2.3. CSS

- [CSS Crash Course](#)
- CSS Full course with project:
 [CSS Full Course with a project | Zero to Hero | English Tutorial](#)
- CSS flex box: <https://css-tricks.com/snippets/css/a-guide-to-flexbox>

Done? Then test your skills:

 Quiz: https://www.w3schools.com/css/css_quiz.asp

2.4. JavaScript

- The introduction to JavaScript: <https://www.w3schools.com/js/>
- Javascript basics:
 [JavaScript Basic Syntax – Tutorial](#) &
<https://www.w3schools.com/js/DEFAULT.asp>
- Javascript Basic Syntax / Functions:
 [JavaScript Basic Syntax: Functions](#) &
https://www.w3schools.com/js/js_functions.asp
- Javascript Basic Syntax / Array:
[JavaScript Basic Syntax – Arrays – YouTube](#)
- A crash course on JavaScript:
<https://www.youtube.com/watch?v=hdl2bqOjy3c>
- Important Javascript concepts: data types, operators, variables, let and const, synchronous and asynchronous, “this” keyword, promises

Done? Then test your skills:

 Quiz: https://www.w3schools.com/js/js_quiz.asp

2.5. React

- React: <https://reactjs.org/docs/getting-started.html>
- React concepts: virtual DOM, functional components.
 > You should learn and focus more on functional components.

- React hooks: useState, useEffect.
> How and when are they used?
- React: render props, pass props.

Done? Then test your skills by coding a simple to-do application!

💡 Small assignment / using React in practice!

See section 3 below.

2.6. Backend

- REST API introduction: <https://restfulapi.net/>
- REST API: <https://www.geeksforgeeks.org/rest-api-introduction/>
- REST API:
https://www.youtube.com/watch?v=SLwpqD8n3d0&ab_channel=ProgrammingwithMosh
- Node JS: <https://www.w3schools.com/nodejs/default.asp>
- Node JS:
https://www.youtube.com/watch?v=TIB_eWDSMt4&ab_channel=ProgrammingwithMosh
- Express: <https://expressjs.com/en/5x/api.html#express>
- Express: <https://www.youtube.com/watch?v=pKdORpw7O48>

Done? Then test your skills by coding a simple http server!

💡 Small assignment using NodeJS or Express in practice!

See section 3 below.

3. PRACTICAL ASSIGNMENTS

3.1 Front-end // React

Assignment requirements:

1. Create a page (component) that displays a button that once clicked will bring up a form where the user can enter information for the todo item (title, deadline, and status – in progress, not started, or done).
2. Once the user has entered the item, dismiss the page and show the item in the list on the page.
3. If the user clicks on any items in the list, the user can edit the information of that item.

Hints:

- Use functional react components.
- Divide the application into small components, for instance: Button, ToDoItem, and ToDoList.
- React hooks: useState, useParams, useEffect
- For styling: use any bootstrap or library that you feel comfortable with. Recommend MUI: <https://mui.com/>
- After finishing the project, push the code to Github, also try to deploy it to Netlify, or take a screenshot of your application. (<https://www.netlify.com/>)

Example of a successful to-do app:

Add a new todo

Click

Add new todo

Title

Deadline

Status ▼

Cancel Add

Add a new todo

Learn JavaScript closure
Deadline: tonight

Pay the Internet bill
Deadline: tomorrow morning

Finalize the React assignment
Deadline: This Friday

Done Not started In progress

3.2 Back-end // Node.JS

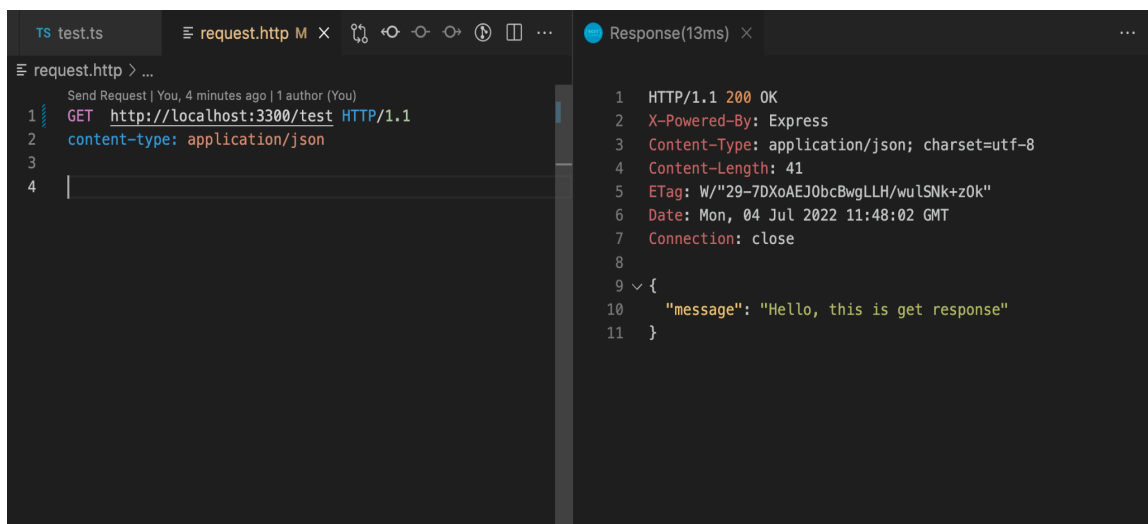
Assignment requirements:

- HTTP GET request to localhost:5000 return {"message": "Hello, this is get response"}
- HTTP POST request to localhost:5000 return {"message": "Hello, this is post response"}
- HTTP PUT request to localhost:5000 return {"message": "Hello, this is put response"}
- HTTP DELETE request to localhost:5000 return {"message": "Hello, this is delete response"}
- HTTP PATCH request to localhost:5000 return {"message": "Hello, this is patch response"}

Hints:

- Using nodemon (<https://www.npmjs.com/package/nodemon>) to run the project.
- To test the server: using one of the following ways:
 - Request.http
 - Postman: <https://www.postman.com/>
 - Insomnia: <https://insomnia.rest/>

Example of a successful GET request using request.http :



The screenshot shows a code editor with a file named `test.ts` and a terminal window. The terminal output shows a successful GET request to `http://localhost:3300/test` using `request.http`. The response is a JSON object: `{ "message": "Hello, this is get response" }`.

```
TS test.ts  request.http M x  Response(13ms) x ...
request.http > ...
Send Request | You, 4 minutes ago | 1 author (You)
1 GET http://localhost:3300/test HTTP/1.1
2 content-type: application/json
3
4
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 41
5 ETag: W/"29-7DXoAEJ0bcBwgLLH/wuLSNk+z0k"
6 Date: Mon, 04 Jul 2022 11:48:02 GMT
7 Connection: close
8
9 {
10   "message": "Hello, this is get response"
11 }
```

3.3 Project with API, React, and TypeScript

Inspiration featured by Integrify:

How to build your own MEME GENERATOR !

<https://www.youtube.com/watch?v=coAUyL8ezSE>