Roadmap desarrollo Full Stack

programación roadmap

FRONTEND STARTS HERE

- [x] HTML.
- [x] CSS.

CHECKPOINT - Static Webpages.

Now that you have learnt HTML and CSS, you should be able to build static webpages. I recommend you to build as many test projects at each yellow step of the roadmap as possible to solidify what you learn.

The practice that I used to follow when I was learning was this:

- While you are watching a course or reading a book, make sure to code along with the instructor/author — pause the video at regular intervals and code what you are being taught.
- Search on YouTube and watch a few project based tutorials on the topic that you are learning. Apart from coding along with the instructor:
  - Try to build the same project at least 2 to 3 times on your own without looking at the video. If you get stuck, refer to the section of the video where the instructor builds that part of the project.
  - Build something else that is similar to the project that you just built. For example, if you just built a todo app, try to build a notes app or a reminder app.
  - [ ] Here is a simple blog design in figma that you can try to copy.

- [x] JavaScript. (JS Roadmap)

CHECKPOINT - Interactivity.

At this point you should be able to add interactivity to your web pages using JavaScript. You should make sure that you have learnt the following:

- Know about variables, loops, data types, conditionals, functions.
- Know about arrays and objects and different ways to access their data.
- Know how to select DOM elements.

- Add event listeners to DOM elements (e.g. click, focus, form submission).
- Use JavaScript to add and remove DOM elements
- Add and remove classes from DOM elements
- Use JavaScript to make HTTP requests to external APIs (i.e. `fetch`)
- Use JavaScript to store data in the browser's local storage.

Here are few ideas to practice your skills:

- [ ] Create a simple to-do list app that allows users to search, add, edit, and delete items. Use local storage to store the data.
- [ ] Create a simple webpage where user can put in anyone's GitHub username and see their profile information. You can use GitHub's API to fetch the data. For example, here is the sample URL to fetch my data. Make sure to add validation and error handling.
- [ ] Create a basic calculator app that allows users to perform basic arithmetic operations.

- [x] npm.

CHECKPOINT - External Packages.

At this point, you should be able to install and use external packages using `npm` . You probably know about npmjs.com where you can search for packages and read their documentation. You should also be familiar with the `package.json` file and how to use it to manage your project dependencies.

You don't need to get into the module bundlers and build tools just yet. Just make sure that you are able to use the dependencies installed in the node_modules folder using simple link and script tags in your HTML.

Regarding projects, here are a few ideas that you can try:

- [ ] Create a simple webpage that shows the current time of user. You can use use dayjs to get the current time and display it on the page. Here is the sample design for homepage.
- [ ] Install the micromodal library. Create a button on the page clicking which should open a modal and let the user select a timezone from a dropdown. Once the user selects a timezone, the modal should close and the time on the page should be updated to show the time in the selected timezone. Here is the sample design for the modal.

- [x] Git.
- [x] GitHub.

CHECKPOINT - Collaborative Work.

Now that you have learnt git and GitHub you should be ready to work with others. You should now setup your GitHub profile and push all the projects that you have built so far to your

GitHub profile. Here are some of my recommendations for your GitHub profile:

- Keep the repository names lowercase and use hyphens to separate words e.g. todo-app instead of TodoApp or Todo-App.
- Add a README.md file to each repository that you create. This file should contain a description of the project. Put some effort into the readme and make sure it clearly details what the project is about and how anyone can run it locally.
- Add snapshots of your project to the readme file so that anyone can see what the project looks like without having to run it locally.
- Add a LICENSE file to each repository that you create. This file should contain the license that you want to use for the project. You can use choosealicense.com to help you choose a license.

- [x] Tailwind.
- [ ] React.

CHECKPOINT - Frontend Apps.

At this point you should be able to build a complete frontend application including:

- Structuring your webpages with HTML
- Styling your webpages with CSS
- Adding interactivity to your webpages with JavaScript
- Using the DOM API to manipulate your webpages
- Using the Fetch API to make HTTP requests
- Understand promises and use async/await syntax to write asynchronous code
- Installing and using external libraries with npm
- Version controlling your code with Git
- Pushing your code to GitHub

If you decided to skip React and Tailwind for now, that is fine also but you should be able to build a complete frontend application using vanilla HTML, CSS, and JavaScript. However, keep in mind that the modern frontend applications are mostly built with frameworks like React, Vue, and Angular. So, you should learn at least one of them at any point of time.

This marks the end of frontend basics that you needed, we will now be moving to the backend development. While you continue with the backend development, know that there is more to frontend development and remember to checkout the frontend roadmap later in your journey.

BACKEND STARTS HERE

- [ ] Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project!
Node.js runs the V8 JavaScript engine, Google Chrome's core, outside the browser. This allows Node.js to be very performant.
Node.js app runs in a single process, without creating a new thread for every request.
Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

Visit the following resources to learn more:

- Official Website.
- Learn Node.js Official Website.
- Node.JS Introduction.
- Node.js and Express.js Full Course (YouTube).

CHECKPOINT - CLI Apps.

At this point you should be able to build CLI applications using Node.js or whatever backend programming language you picked.

You should be able to build a CLI application that can:

- Read and write files
- Parse command line arguments
- Make HTTP requests
- Parse JSON
- Use a third-party library (e.g. a library for parsing CSV files)
- Use a third-party API

Here are some ideas for CLI applications you can build:

- [ ] Create a CLI application that takes a URL and a CSS selector arguments and prints the text content of the element that matches the selector. Hint you can use cheerio. (Mirar también puppeteer).
- [ ] An application that takes optionally takes two dates and prints the most starred GitHub projects in that date range. Hint you can use GitHub's search API.
- [ ] Bulk rename files in a directory. Hint you can use fs and path.
- [ ] Write a CLI application that takes a path as input and compresses all the images in that directory. It should accept an option for output path; if the output path is not given it should compress images in place otherwise write the compressed images to the output path. Hint you can use sharp (High performance Node.js image processing).

- [ ] PostgreSQL.

- [ ] SQLite.

CHECKPOINT - Simple CRUD.

CRUD stands for Create, Read, Update, and Delete. These are the four basic operations you can perform on any data when working with web applications, databases, and APIs.

Now that you know about programming language and the databases, you should be able to build a simple CLI application that interacts with database. We haven't talked about the APIs yet but you don't need an API to practice CRUD operations. Here are some of the CLI applications you can build to practice CRUD operations:

A simple todo list application for the CLI with the following options:

- `--new` to add a new todo item
- `--list [all|pending|done]` to list the todo items
- `--done [id]` to update a todo item
- `--delete [id]` to delete a todo item
- `--help` to list all the available options
- `--version` to print the version of the application

Recursos: Ejemplo de CLI - CRUD en FreeCodeCamp.

- [ ] RESTful APIs.
- [ ] JWT Auth.
- [ ] Redis.

CHECKPOINT - Complete App.

At this point, you should have everything that you need to build a complete application that:

- Has a responsive frontend that users can interact with.
- Has a backend API that is secured with JWT authentication.
- Has a database that stores data.

At this point you should practice building as much as you can on your own to solidify your knowledge. If you need inspiration, here are some ideas:

- [ ] Build a simple blogging application where users can register, login, setup their blog and write posts.
- [ ] A single page site builder where users can pick a template, modify it and publish it. Hint you will need filesystem to store the design templates. Template files will have placeholders that you will need to replace with user data.
- [ ] Build a simple e-commerce application which will have two types of users i.e. Sellers who can: Register as Seller, Login, Setup their store, Add products, Edit products, Delete products, View Received Orders, Update Order Status (Pending, Shipped, Delivered), Buyers

who can register, Login, Browse products by all sellers, Add products to cart, Checkout, View order history, View order status, Cancel order, View seller profile, View seller products

These are just some ideas to get you started. You can build anything you want. The goal is to practice building a complete application from scratch.