**How to run Frontend and Backend with one command?**

[Ravithamara](https://medium.com/@rwijayabandu?source=post_page-----55d5f2ce952c--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F8e2e1831b91c&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40rwijayabandu%2Fhow-to-run-frontend-and-backend-with-one-command-55d5f2ce952c&user=Ravithamara&userId=8e2e1831b91c&source=post_page-8e2e1831b91c----55d5f2ce952c---------------------post_header-----------)

4 min read

·

Jan 30, 2024

9

When you develop full stack you maybe manually start each server using separate terminal, but this approach is tedious and inefficient

Let’s assume you typically run your node.js backend by opening a new terminal and and running the command **node server** or **nodemon server** Similarly, the React frontend by opening a separate terminal and running the command **npm start**. **Why not run both in single terminal using single command?**

There are two effective ways. Use the **“concurrently”** library or **“npm-run-all”**CLI tools designed to run multiple npm-scripts in parallel.

**Concurrently Library**

Let’s delve into the Concurrently Package. The Concurrently allows you to run multiple commands or scripts simultaneously from a single terminal offering a more efficient workflow. This means you can define separate scripts for starting the React and node servers in your package.Json file, and “concurrently” will execute them together.

First you have to install **“concurrently”** in your project with the command **npm install concurrently** or **npm install concurrently --save-dev**. It’s recommended to add **concurrently** as devDependencies since it's usually used for developing purposes. If you have created separate folders for frontend and backend, ensure that you install “concurrently” in root folder which containing both.

npm install concurrently   
  
npm install concurrently --save-dev

Next Find the “start” script in package. Json file in main folder and modify it. The basic syntax looks like this:

"start": "concurrently \"script1\" \"script2\""

Replace “script1” and “script2” with the actual commands you want to run concurrently.

Or, you can customize further, like this:

"scripts": {  
 "start": "concurrently \"npm run start:frontend\" \"npm run start:backend\"",  
 "start:frontend": "cd path/to/frontend && npm start",  
 "start:backend": "cd path/to/backend && node server"  
}

Replace “path/to/frontend” and “path/to/backend” with the actual paths.

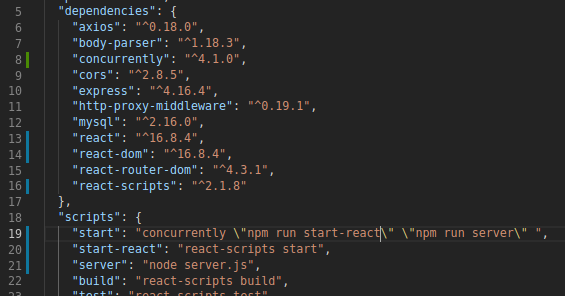
Here’s an example if you have separate frontend and backend folders:

"start": "concurrently \"cd frontend && npm start\" \"cd backend && node server\""

Look, now if you run **npm start** it will call concurrently first. Then if you have a frontend folder with frontend codes, first you need to move onto the frontend directory (**cd frontend**) (use necessary command to move onto your relevant folder), and run frontend with the **npm start** command and then similarly if you have a backend folder with backend codes, move onto the backend directory (**cd backend**) (use necessary command to move onto your relevant folder), and run **node server**

If you want to add more command you have to add the commands like the above format with backslashes and quote marks for each scripts. **Now you can simply open a new terminal and run npm start from your main/root project folder to get both your frontend and backend up and running concurrently.** This streamlined approach enhances productivity, enables rapid iteration, and ultimately leads to better Developer Experience.

Using npm “concurrently” simplifies the development workflow by allowing you to manage multiple tasks seamlessly, improving efficiency and making it easier to maintain complex projects. It’s a valuable tool for developers working on projects with diverse requirements for concurrent processes.



“Concurrently” provides additional customization and flexibility options to enhance the development experience. There are more features in Concurrently to make development much easier.



**1. Named Commands:**

"start": "concurrently --names 'FE,BE' \"npm run start:frontend\" \"npm run start:backend\""

* The --names option allows you to assign names to the commands being executed.
* This makes the console output more readable and helps you easily identify which process corresponds to the frontend and which one to the backend.

**2. Exiting on Error:**

By default, if any of the commands exit with a non-zero status code (indicating an error), concurrently will terminate all the other commands. However, you can modify this behavior with the following options:

* --kill-others: If one command fails, this option prevents the termination of other commands.
* --success first|last: Specifies whether to consider the entire execution a success if the first or last command succeeds.

This can be particularly useful in scenarios where you want to continue running other processes even if one encounters an error.

**3. Color-Coded Output:**

The output of each command is color-coded by default.

"start": "concurrently -n \"FRONTEND,BACKEND\" -c \"red,blue\" \"npm run start:frontend\" \"npm run start:backend\""

* -c "red,blue" specifies the colors for the named commands. In this case, the frontend command will have red output, and the backend command will have blue output.
* Color-coded output makes it easy to visually distinguish between the different processes, aiding in quickly identifying the source of logs or errors.