**NPM vs NPX**

NPM(Node Package Manager) We get it by default when installing Node.js and it is a dependency/package manager; it also help developers to install packages both globally and locally . NPM doesn’t run any packages ;for this the package must be specified in the package. Jason file. When executable files install by npm packages , npm creates a link to them.

For files which installed locally there is a link at the ./node\_ modules /.bin/ directory and, for globally installs at bin/ directory.

We can execute global packages by typing their name in command line but it doesn’t work for local packages so, for this we must type the local path:$./node\_modules/.bin/some package or we can edit package.jason file and adding the package in the script section and use npm run. We should do this steps each time want to run a package for solving it npx comes

NPX(Node Package Executor) it is an npm binary package executor ,whit it the locally installed packages run simply by this command :$npx package . npx will check whether <command> or <package> exists in the path or in the local project binaries, and if so it will execute it. Another advantage of npx is executing packages that are not previously installed it gets and run packages simultaneously. With npx, we will not have to install global packages to be able to run them moreover, we will not have to create an npm script to run local binaries.

In addition, with npx, we can run commands with different Node.js versions it means we can easily test node command using different node versions, without having to use a version manager like nvm. Therefore, we can run js scripts directly from git.

**To put it in nutshell**, **NPM manages packages and it is not always easy to run them easily. NPX is a tool to run node packages easily without installing binaries.** NPX gives us a clear and easy way to run packages and helps us to avoid versioning, dependency issues, and installing unnecessary packages that we just want to test.

**Components and state in react**

A **Component** is one of the core building blocks of React. In other words, we can say that every application we will develop in React will be made up of pieces called components. Components make the task of building UIs much easier. we can break down an UI into multiple individual pieces called components and work on them independently and merge them all in a parent component which will be our final UI.

There are two types of components:

**functional components** which are simply JavaScript functions. We can create a functional component in React by writing a JavaScript function. These functions may or may not receive data as parameters. They are **functional**because they are basically functions, **stateless**because they do not hold and/or manage state and, **presentational**because all they do is output UI elements

**Class components** which are a little more complex than the functional components. The functional components are not aware of the other components in your program whereas the class components can work with each other. We can pass data from one class component to other class components. They are **class**because they are basically classes, **smart**because they can contain logic, **Stateful**because they can hold and/or manage local state and, **container**because they usually hold/contain numerous other (mostly functional) components

The **state** is an instance of React Component Class can be defined as an object of a set of **observable** properties that control the behavior of the component. In other words, the State of a component is an object that holds some information that may change over the lifetime of the component.

**Causes component re-rendering**

React components automatically re-render whenever there is a change in their state or props.

A simple update of the state, from anywhere in the code, causes all the User Interface (UI) elements to be re-rendered automatically.  Re-render can be caused due to any of the three reasons listed:

1. **Update in state:** The state change can be from a prop or setStatechange to update a variable(say). The component gets the updated state and react re-renders the component to reflect the change on the app.
2. **Update in prop:** Likewise, the change in prop leads to state change and state change **leads to re-rendering of the component by React.**
3. **Re-rendering of parent component:**Whenever the components render function is called, all its subsequent child components will re-render, regardless of whether their props have changed or not.

**References:**

[**https://www.freecodecamp.org/news/npm-vs-npx-whats-the-difference/**](https://www.freecodecamp.org/news/npm-vs-npx-whats-the-difference/)

[**https://www.geeksforgeeks.org/what-are-the-differences-between-npm-and-npx/#:~:text=Npm%20is%20a%20tool%20that,pollution%20for%20the%20long%20term**](https://www.geeksforgeeks.org/what-are-the-differences-between-npm-and-npx/#:~:text=Npm%20is%20a%20tool%20that,pollution%20for%20the%20long%20term)**.**

[**https://medium.com/codex/are-npm-and-npx-the-same-567104f13bf0**](https://medium.com/codex/are-npm-and-npx-the-same-567104f13bf0)

[**https://medium.com/edonec/state-in-react-an-overview-a182675cee2c**](https://medium.com/edonec/state-in-react-an-overview-a182675cee2c)

[**https://www.w3schools.com/react/react\_state.asp**](https://www.w3schools.com/react/react_state.asp)

[**https://medium.com/the-andela-way/understanding-react-components-37f841c1f3bb**](https://medium.com/the-andela-way/understanding-react-components-37f841c1f3bb)

[**https://www.w3schools.com/react/react\_components.asp**](https://www.w3schools.com/react/react_components.asp)

[**https://reactjs.org/docs/react-component.html**](https://reactjs.org/docs/react-component.html)

[**https://www.geeksforgeeks.org/reactjs-components/**](https://www.geeksforgeeks.org/reactjs-components/)

[**https://www.geeksforgeeks.org/re-rendering-components-in-reactjs/#:~:text=React%20components%20automatically%20re%2Drender,in%20their%20state%20or%20props.&text=A%20simple%20update%20of%20the,to%20be%20re%2Drendered%20automatically**](https://www.geeksforgeeks.org/re-rendering-components-in-reactjs/#:~:text=React%20components%20automatically%20re%2Drender,in%20their%20state%20or%20props.&text=A%20simple%20update%20of%20the,to%20be%20re%2Drendered%20automatically)**.**

### 