**React**

**Day 8th**

**How to create your own web server??**

Webpack provides another feature as dev server. It can be used to quickly develop an application. This set of options is picked up by webpack dev server and can be used to change its behavior in various ways. Here’s a rudimentary example that gzips and serves everything from our public directory in the project root. **Loaders??**

Out of the box , webpack only understands JavaScript and JSON files. Loaders allow webpack to process other types of files and convert them into valid modules that can be consumed by our application and added to the dependency graph.

**Hot Module Replacement (HMR) / (hot property in configuration)**

Hot Module Replacement (HMR) exchanges, adds or removes modules while an application is running, without a full reload. This can significantly speed up development in a few ways.

* Retain application state which is lost during a full reload.
* Save valuable development time by only updating what’s changed.
* Instantly update the browser when modifications are made to CSS or JS in the source code, which is almost comparable to changing styles directly in the browser’s dev tool.

**How it works**

The following steps allow modules to be swapped in and out of an application:

1. The application asks the HMR runtime to check for updates
2. The runtime asynchronously downloads the updates and notifies the application
3. The application then asks the runtime to apply the updates.
4. The runtime synchronously applies the updates.

You can set up the HMR so that this process happens automatically, or you can choose to require user interaction for updates to occur.

**Open property in configuration file / Webpack open**

Whenever you are serving web application automatically browser will be opening. Tells dev-server to open browser after server had been started. Set it to true to open your default browser.

Here one question will arise like if we are not providing any type equals to module but still I can see the changes on my UI how so all the things is taken care by webpack server along with dependencies.

**HTML webpack plugin (To update the content dynamically)**

The HTML Webpack plugin simplifies the creation of HTML files to serve your webpack bundles. This is especially useful for webpack bundles that include a hash in the filename which changes every compilation. You can either let the plugin generate an HTML file for you, supply your own template using lodash templates, or use your own loader.

To install html plugin you need to run the command as shown below:

`npm install –save-dev html-webpack-plugin`

**@babel/preset-env**

It is a smart preset that allows you to use the latest JavaScript without needing to micromanage which syntax transforms are needed by your target environments. This both makes your life easier and JS bundles smaller.

**Babel loader**

This package allows transpiling JS files using **Babel** and **webpack**. You can install it using `**npm i babel-loader**`

**To create react app without having create-react app or vite CLI**

1. **Webpack configuration**
2. **Babel**
3. **Dependencies**
   1. **React and react DOM**
4. **DevDependencies**
   1. **Webpack**
   2. **Webpack-cli**
   3. **@babel/core**
   4. **@babel/preset-env**
   5. **@babel/preset-react**
   6. **Babel loader**
   7. **Html-webpack Plugin**
   8. **Webpack dev-server**

So if you wanna install all these things at a time then you can go with as follows

Npm install –save-dev webpack webpack-cli @babel/core @babel/preset-env @babel/preset-react babel-loader html-webpack-plugin webpack-dev-server