* Webpack *

Webpack is a module bundler for JavaScript application.
It allows developers to manage dependencies, optimize code, & bundle multiple modules into a single file.

N-P-M (Node Package Manager) →

- Npm is a package manager for JavaScript & NodeJS.
- It is used to manage dependencies for JavaScript projects, & provide a central repository for open-source package.
- NPM allows developers to easily install-uninstall & update packages for their projects.
- It uses a file called 'package.json' to define the dependencies for a project, & can automatically download & intall the necessary packages when a project is initialized or updated.
- NPM is included with Node.js, which means that it is available on most operating Systems & can be used to manage packages for both client-side and Server side, Javascript projects.

Note* To use 'npm' you should have NodeJS installed, Node brings npm by default.

Start Webpack:

Step1 → Create a folder. 'project'.
Step2 → Open terminal in the directory of project.
Step3 → Write command — ` npm init -y `.

(npm init -y) is a command to initialize an npm package manager you will get a file named 'package.json' this package json file is a dependency manager.

Step4 → Install webpack using npm & webpack CLI

Command → npm install webpack webpack-cli --save

if the command not works → npm cache clean --force & then run above command.

Step 7 → Create a file index.html [illegible]

npm i --save //this is not a command. I am just Explaining don't run this

i short for install | --save → is used to save the installed package as a dependency in the project's package.json.

When you use 'npm install' to install a package without the '--save' option, the package is installed locally in the 'node-modules' folder but it is not added to the project's 'package.json'.

After Step 4 → You will see a folder named → Node Modules

Q. What is Node Modules?

Ans → node-modules directory is a folder that contains all the dependencies (modules) that are required by your application.

When you use 'npm' to install a package, the package and all of its dependencies are downloaded & installed into the 'node-modules' directory of your package.

After installing webpack & webpack-cli, you can find these two inside package.json dependency array.

Step 5 → Create a folder named → 'dist' in the root of your application.

Step 6 → Create another folder named → 'src' in the root.

Note* → you must have 2 folders dist & src in your project to work with webpack

* Your Project Structure will look like →

```
Project_name
          |→ node_modules
folder -  |→ dist
folder -  |→ src
          |→ package-lock.json
          |→ package.json
```

Step 7 → Create a file index.html inside dist folder.

Step 8 → Create a file index.js inside src folder.

index.html & index.js will be the entry point of dist & src folders respectively. These entry points are for our webpack. Our webpack will search these files in our project.

* Start using Webpack *

Create another file in src → named second.js

src
- index.js
- second.js

second.js → code

```
export function other(){
    return "Other function called inside second.js";
}
```

index.js → code

```
import {other} from "./second"
console.log(other());
```

dist
- index.html → <!-- Write html boiler-plate & some text inside body -->

Now go to package.json → Create a start script inside "script" property available.

package.json →

```
{
  ...
  scripts: {
    "build": "webpack"
  }
```

① Open Terminal Run the command → npm run build...

This command will start our webpack & our webpack will do some processing.

If you run the command and it executes without error →
Webpack have generated a file → main.js inside dist folder automatically

This main.js file is created by webpack & its a compressed/bundled form of our code.
lets suppose index.js & other.js/second.js contains 1000s of lines of codes that complex code might become very heavy for browser to load this much, our webpack minifies all that code & creates a new file that is little light-weighted & easy to be loaded by our browser.

You can relate it by min.js & .js files → min.js is minified without space so our file is small!

We always release our minified file in production.

Now in final step to this I will link my main.js file to my index.html

Dist →
index.html ⟶

```
<html>
  !
  <Body>
       Hi There!
  <Script src="./main.js" ></script>
  </Body>
</Html>
```

Thats the basic idea of "why to use Webpack". It helps us creating minified files those are optimized & we can use this file to improve performance!

* Define Mode *

In Webpack we can define the mode.!
mode tells webpack whether we are building our app for 'production' or 'development'

Create a file in root → webpack.config.js

code inside the file } → ↳ ~~export~~

```
module.exports = {
    mode : "production"
}
```

Run build command → npm run build

This will set our mode to production & build file i.e main.js will still look same as earlier cause previously also when we have not defined mode webpack made a production mode app only.

lets change mode to "development" →

Webpack.config.js →

```
module.exports = {
    mode: "production"
    mode: "development"
}
```

Run build command → npm run build.

Now you can see our main.js have lot more code with lot more explaination. It is not recomended to use this developer mode file in production. cause it will ultimately take more time.

* To change entry point JS file name *

By default as I have mentioned webpack will take index.js as entry point. But In case you want to change name or want some other file to be the entry point.

Then we need to configure it in webpack.config.js →

webpack.config.js →

```
module.exports = {
    mode: "production",
    entry: "./src/app.js"
}
```

Src → rename index.js to app.js

* To change output file name from main.js to out.js *

webpack.config.js →

```
const path = require('path');
module.exports = {
    output: { path: path.resolve(__dirname, 'dist'),
              filename: 'out.js'
    }
}
```

don't forget to rename your file you're linking inside index.html.

* Create a Web Server *

I want to create a web server.

That means I don't want to build using 'npm run build' & then open my project to check the latest changes.

Instead I want whenever I (Ctrl+S) save my file our webpack should build the code & run our project on open our project.

Step 1 → Install Webpack Dev

Write command → ~~npm i~~ `npm install webpack-dev-server --save-dev`

~~`npm install webpack-dev --save-dev`~~

This command will install webpack-dev & save it as a dev-dependency.

Step 2 → Create a new script inside package.json "scripts"

package.json

```
{
  "scripts": {
    "build": "webpack",
    "start": "webpack-dev-server --mode-development --open",
  }
}
```

this start script will create webpack server, build our output file in development mode & --open will open our app on some port.

Step 3 → Add configuration to webpack.config.js file →

webpack.config.js →

```
const path = require('path');    // from node_module

module.exports = {
  devServer: {
    static: path.join(__dirname, 'dist'),
    compress: true,
    port: 3000
  }
}
```