LAYING THE FOUNDATION

1.What is JSX? What are the Superpowers of JSX?

JSX is a HTML like Syntax but not HTML inside JavaScript. JSX is not a package. It was built by Facebook as a way to make easier for developers to build User interfaces (UI) using React, a JavaScript library for building user interfaces. Babel reads the whole code and converts it into React.

JSX is used mainly to write less code with more functionality.

> **Syntax of JSX is:- Const element = <h1 > Hello Kiddo!! </h1>;**

Ex:-

 Const writer = <h1 id=" wall" > Build Curiosity </h1>;

We can also write this as const writer = (

<h1 id =" wall">

Build Curiosity

</h1>) ;

Here, we've used `( )` parentheses around the JSX. Because it is mandatory to use it when we write code in multiple lines. And moreover, this is now called a JSX Expression.

When we run this JSX Expression in our Browser, it throws us **Syntax Error: Unexpected token `<`** They don't understand this code and this Angular Brackets (</>).

Then who understands this code and executes it???

BABEL understands this code. BABEL is one JavaScript Library which reads our code and gives us another code.

How does JSX gets Executed? / How JSX Works?

Our JSX Code is taken into Babel, which understands the JSX code and Babel converts the code into React Element which gives OBJECT at the end of any eons and that object is converted to HTML which gets rendered to DOM.

How does Babel come into our project? Should we install it?

Babel comes along with Parcel. When we used command **npm i parcel,** Babel comes along with parcel as a dependency.

We can find Babel in **node_modules => package-lock.json** . Babel has a lot of helpers and a lot of Transitive dependencies.

Super-Powers OF JSX:

1.Readability

2.Maintainability

3.Less Code

4.Developer Friendly

5.No Repetition

6.Syntactical Sugar

7.Faster than Normal JS as it performs optimizations while translating to regular JS

8.simple and elegant while writing large pieces of Code.

2. Role of type attribute in script tag? What options can I use there?

The type attribute identifies the scripting language of code embedded within a script element or referred via the element's src attribute. This is specified as a MIME(MULTIPURPOSE INTERNET MAIL EXTENSIONS) type(previously), Now Internet Media type.

A Media type is nothing but a two-part identifier for file formats and format contents transmitted on the Internet. "text, audio , application , image , message , model , multipart, example" are some internet media types used in web apps.

Examples of supported Internet Media types(MIME) and some common values for type attribute include text/javascript , text/ecmascript , application/javascript , application/ecmascript , text/babel , text/typescript.

According to HTML 4.01

https://www.w3.org/TR/html401/interact/scripts.html#h-18.2.1

The type attribute specified the scripting language of the element's contents and overrides the default scripting language. The scripting language is specified as a content type(Ex- "text/javascript"). Users must apply a value for this attribute. There's no default value for this attribute.

But in HTML5 text/javascript is the default type, so we can omit it.

The type attribute gives the language of the script or format of the data. If the attribute is present, it's value must be a valid Internet media type. The charset parameter must not be specified. The default which is used if the attribute is absent, is "text/javascript".

Moreover HTML5 doesn't need "type = "text/javascript" . It's the default .

Fortunately, at present maximum browsers support JS. In HTML, it is better to leave

"type = "text/javascript". The browsers know what to do. So, it's best to simply write **<script>** and wrap our code between it without any type attributes until and unless there're any JAVASCRIPT modules in our file.

(FOR CDATA and SVG search in detail……. type requires for them)

ES6 Supports the Modules Concept. We can see the role of type attribute in script tag with an example so, that it would be clearer. OK.
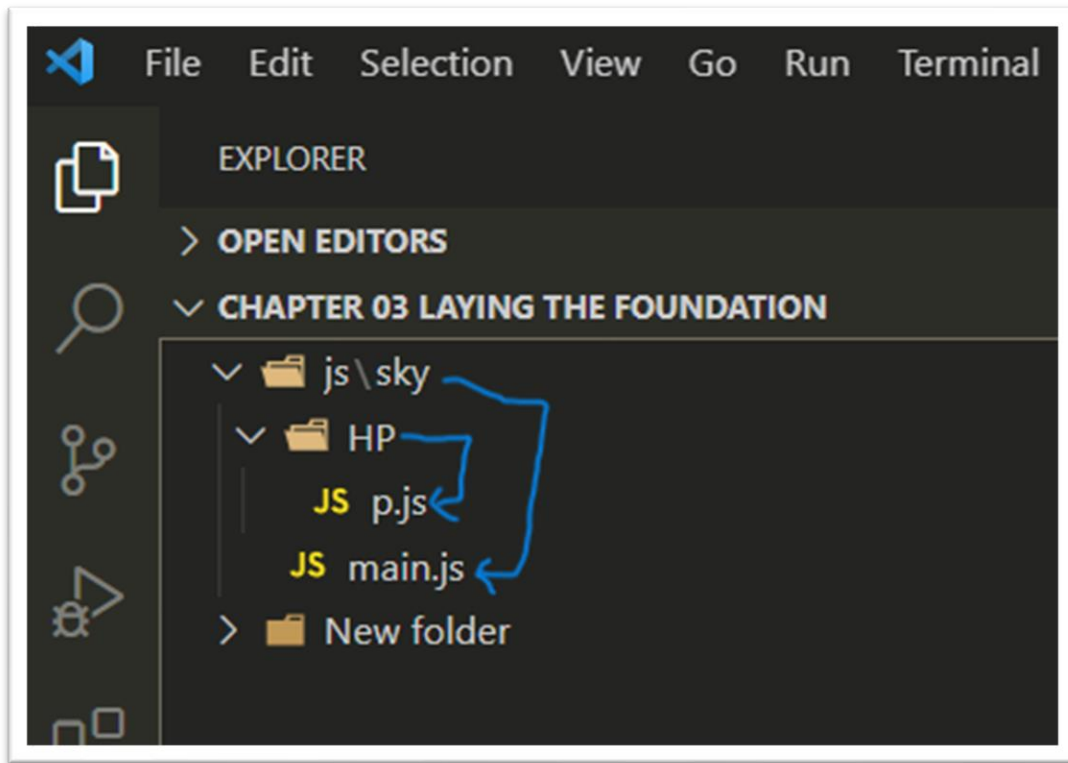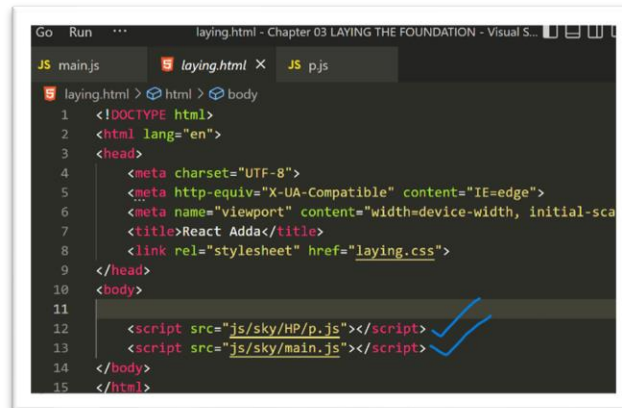

Here, we can see there's a folder named ` js ` and one folder ` sky ` inside " js", and inside that("sky") folder 'main.js' file and ` HP ` folder are present. And inside the ` HP ` folder we can see ` p.js` file.

Now we've a two files p.js and main.js. We'll create a value in p.js and Now I'll try to use this variable in main.js. We can see that p.js is present inside HP and HP is inside sky. So, I'll be using this value in main.js and logs it with the assigned variable.
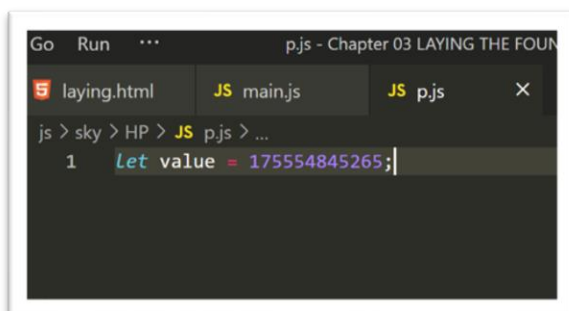
Now in order to use this variable in main.js we'll go to our html and enter the script file. We'll enter our both the files path in this way and specify their location.
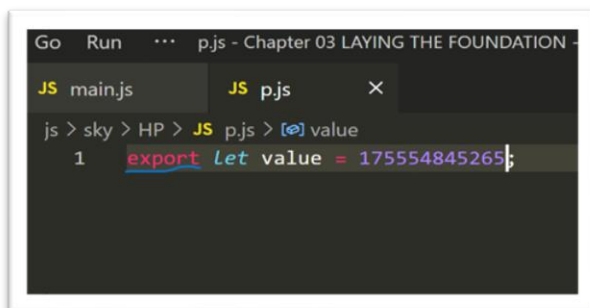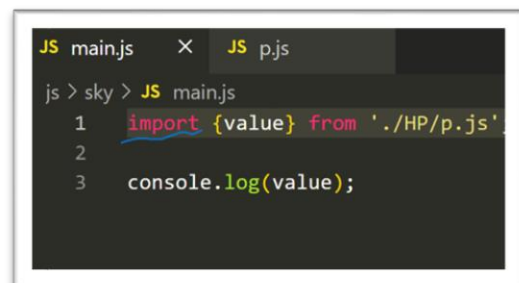

Here, we gonna specify the main.js file and p.js , as it contains the main key value and we're using the value in main.js, So we need to import the value to script variables. We can find our value in console when we run the file. It is clear till now but when we're working on some projects and there'll be scenarios where the number of script files rise and writing them at the bottom of HTML also be not that good. So, to avoid that complexity **ES6 comes with modules concept**.

So , now we can remove the above script file (`<script src="js/sky/HP/p.js"></script>`)

In our source file we'll add export in front of the variable which we want to export. In our main.js file we write **import {variable name} from filepath**.
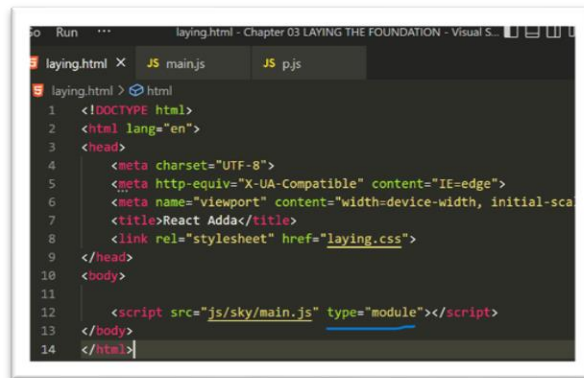
When we run this file now. We'll get a **SyntaxError : Cannot use import statement outside a module**

Here , it saying that We cannot use import statement outside module because scripts doesn't support the import by default. Now we need to introduce " **type = "module" instead** of
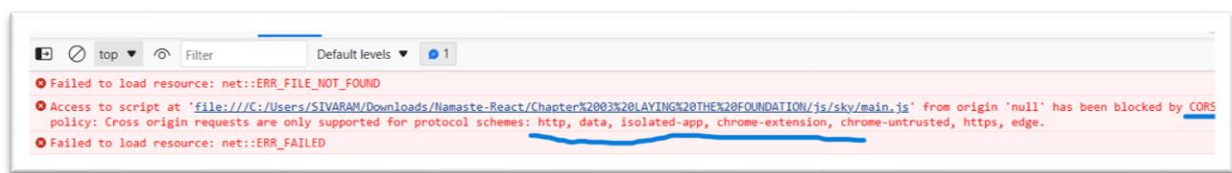
type = " text/javascript". As we're using module concept from JavaScript we need to write

type = "module" . Wherever and whenever we're using import statements in JS(ES6) we need to use type = "module".

When we execute our file now. We will get this error in our console.

If we're getting this problem, mean we're not using our file in local host and using it as normal file/directly. We've to host it on local host. We can host it on Node, or vs live server or Xampp server (PHP tool) or any. We'll get the output desired after we run it on localhost.

JUST A SMALL RECAP –

We've removed HP/p.js from HTML and added type="module" in **script file.** Why, because we're using the ES6 module concept in the file. That's the reason we've to have it a localhost **.** and because of the Cross-origin policy, we cannot directly use it with the file and we can use the localhost and in console we got the output.

We Know that Some browsers don't support the ES6 and so, our code won't work on it. And to overcome it we'll use BABEL trans piler to convert our code to older version browser compatible code. But what we need to see here, is babel doesn't support the modules concept. Babel converts the code into ES2015/ES6. In order to combine multiple files (module bundling thing) we've to use webpack, **parcel**, roll ups , browserify etc. Module bundlers are ought to be used to combine all the files and work for us , as babel does the compile line by line, it can't' combine the files and give us the desired output. So, we use Module Bundlers here.

FOR DETAILED READ THESE DUDE 👇 👇

Intro to JS Bundlers and Babel –

https://blog.bolajiayodeji.com/introduction-to-babel-and-javascript-bundlers

JS MODULE BUNDLER GUIDE - https://snipcart.com/blog/javascript-module-bundler

3. {TitleComponent} vs {< TitleComponent/>} vs {< TitleComponent> </ TitleComponent>} in JSX?

Curly braces {} are special syntax in JSX. In JSX, curly braces {} are used to insert variables/expressions in the JSX Code. The {} are used to evaluate the expression and include its value in the resulting JSX element. The expression can be a variable, function, an object or any code that resolves into a value.

Let's see how these braces are used in JSX :-

{TitleComponent}:

This syntax is used to include a variable or expression in the JSX element. The given value in the {} will be evaluated and included in the JSX element.

{< TitleComponent/>}

This syntax is not valid in JSX. We cannot include a JSX element inside curly braces. Instead, we should write the JSX element directly in the JSX code.

Ex:- <TitleComponent/>

{< TitleComponent> </ TitleComponent>}

This syntax is used to create a JSX element that has a starting and ending tag. The content between the starting and ending tags will be included in the JSX element.

Finally, we should use curly braces {} to include variables or expressions in your JSX code and you should write JSX elements directly in your JSX code, using starting and ending tags if necessary.

//

In Small context we can wrap this.

{} play major role in the JSX. For calling Component, to use React Element inside React Component , to use functional component inside React Component etc.

{TitleComponent}

This syntax is used to include a variable or expression or element in the React Component . The given element in {} will be evaluated in JSX Element.

Along with that we can use {} to write plain/any JavaScript inside JSX, and to call functions in it. ({Title()}.

{< TitleComponent/>}

This syntax is not valid. Because we can use <Title/> without curly braces {} when element/expression/component are used inside a React component. To use the functional component in React Component we use either **<Title/> or {Title ()}.**

At the end functional component is a function. So, **{Title ()} too shall work.**

Whenever we're referring to functional component/Class Based/Any Component we shall use the syntax of JSX <JSX/>.

Direct Functions are not executed as React Child. So, we've to use {Title ()} / <Title/>.

{< TitleComponent> </TitleComponent>}

This syntax is used to create a JSX element that has as starting and ending tag. The content between the starting and ending tags will be included in the JSX element.