Namaste React

Episode 1 Inception

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <div id="root"></div>

    <script>

        let heading=document.createElement('h1');

        heading.innerHTML='My name is Zaid';

        const root=document.getElementById('root');

        root.appendChild(heading);

    </script>

</body>

</html>

Output

My name is Zaid

Now perform the same thing with the help of react first of all import the cdn of react

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <div id="root">

    </div>

    <!-- this is for core react liberary  -->

    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

    <!-- this is used for dom operations   -->

<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<script>

// this line is using the react core cdn link

    const heading=React.createElement('h1’, {id:"name"},"This is react"); // (name of element, attributes, content)

    // this line is using the react dom cdn link

    const root=ReactDOM.createRoot(document.getElementById('root'));

    root.render(heading);

</script>

</body>

</html>

Output

This is react

If we try to console the heading then it will give an object which type is h1 heading is not the h1 tag it is the object. Heading is a react element React.createElement will create a js object. It will not create a h1 tag.

If I want to create a nested elements in react

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <div id="root">

    </div>

    <!-- this is for core react liberary  -->

    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

    <!-- this is used for dom operations   -->

<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<script>

const parent=React.createElement('div',{id:"parent"},React.createElement('div',{id:"child"},React.createElement('h1',{id:"name"},"This is nested Eleemnts")));

const root=ReactDOM.createRoot(document.getElementById('root'));

root.render(parent);

</script>

</body>

</html>

Output

This is nested Elements.

React.createElement will react a object but when it will render on the browser the object will convert into html.

If wanted to create a sibling element then write a code like this

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <div id="root">

    </div>

    <!-- this is for core react liberary  -->

    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

    <!-- this is used for dom operations   -->

<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<script>

const parent=React.createElement('div',{id:"parent"},

React.createElement('div',{id:"child"},

[React.createElement('h1',{id:"name"},"This is nested Eleemnts"),

React.createElement('p',{id:"para"},"this is para ")]

));

const root=ReactDOM.createRoot(document.getElementById('root'));

root.render(parent);

</script>

</body>

</html>

Note

So, we have imported the react cdn in our file and also write the react code inside the script tag so if I will change the order of the scripts tags is I use the react code script tag before the cdn it will not work.

I am writing the code inside the root div with the help of react but if there is already some content inside the root div then that content will be replaced by the react code value.

Npm is used for installing the package and npx is used for executing the package.

Cdn link of react will create an another network and also we have to update the version in cdn link so because of these two reasons cdn are not preferable.

Episode 2 Igniting Our App

Package.json is the configuration for npm.

There are two types of dependencies Normal dependencies and the development dependencies. The normal dependencies will be used in development as well as in the production but the development will only be used in the development.

Npm install -D parcel: this is a development dependency. And parcel is a bundler which bundled our code.

Parcel: ^2.8.3 This ^ is caret if we use caret before our version, it will automatically upgrade to the latest.

When we install parcel in our app the node modules will create and the node modules will contain the parcel and its dependencies and their dependencies. Which is called transitive dependencies.

Parcel uses babel.

Package.json will take care of every dependency the app will need.

Packagelock.json will care about the versions of the dependencies.

If I have package.json and the packagelock.json then I can recreate my node modules.

We never push the node modules inside the git so we have to create a file name. gitignore and write inside like //node modules.

So, to re-create a node modules write this command npm install

Import React from “react” it means we are import react package from the node modules.

Parcel:

Dev Build

Create a local server

Perform hmr(hot module replacement) automatic refresh.

Parcel perform hmr because it uses the File Watching Algorithm which is written in c++.

Parcel give faster build because it uses caching which reduces the build time.

Image Optimization

When we make a production file parcel also performs the minification.

Bundling and Compressing

Consistent Hashing

Code Splitting

Differential Bundling: supports older browsers

Error Handling

Tree shaking algo: remove unused code.

Parcel.org

Dist. contains the production ready code. We don’t need to push dist. on GitHub.

For giving the browser support write a code inside the package.json

“browserlist”:[

“last 2 versions”

]

//again the ep 2 start

To add npm into your project write npm init.

By writing this command it will create a package.json file

Now install the parcel in the app by running the command npm install -D parcel.

We use -D because there are two types of dependencies the development dependencies and the Normal dependencies development dependencies are generally used in development and the normal dependencies can used in production also.

^ this will install the minor version automatically

~this will install the major version automatically

In our project there are many package.json every dependencies have their own package.json

By installing the parcel, the node\_modules will create and if we want to re install the node modules, we have to run npm install

To ignite our app, we have to generate the index.html that is a source file npx parcel index.html

On running this command npx parcel index.html two folders also created one was the dist and the second one was the parcel\_cache.

On running this command our app is start running on the <http://localhost:1234>

Now install the react and react-dom into the app by running the two commands

Npm I react and npm I react-dom

To start a server, write npx parcel index.html

While running the server our app starts but it does not print anything and our console have a warning

(index):16 Uncaught ReferenceError: React is not defined

So, for removing this error and making our app working perfect we have to import the react

Import React from ‘react’;

And this ‘react’ is referring to the react inside the node modules it means I am importing from node\_modules

## After importing the react and react dom we can see an error in the browser “ @parcel/transformer-js: Browser scripts cannot have imports or exports.

“

To make a production build npm parcel build index.html

To re generate the dist and the parcel\_cache we have to run thi command npx parcel index.html

To specify the browser specification we have to write this code inside the package.json

“browserList”: [

“Last 2 chrome versions”

] it will take care that the application will only support in the last two chrome versions only

Episode 3

To run our project, we have to write the npx parcel index.html but now we make a script so that we do not need to write this command every time

So we came inside the package.json and make a script

"scripts": {

    "start":"parcel index.html",

    "build":"parcel build index.html",

    "test": "jest"

  },

Now to run the application we can write the npm run start or npm start.

And to make a build I have to write this code npm run build.

Jsx is html like syntax it is not HTML. Before it reaches to the js engine and this trans piled is done by parcel

The jsx code is not understood by js engine so when we write the code so this jsx is trans piled.

Babel is js compiler it takes the jsx and convert into js so that react can understand and this babel is a package which is install by the parcel .

React.createElement is converted into React-Element-JS Object => Html elements(render).

Jsx =>React.createElement.

Babel is converting jsx to react.createElement .

Const heading = (

<h1>hello this is react element</h1>

);

Const Heading = () =>

{

<h1>This react component<h1/>

}

The return is not compulsory.

When we use a component inside another component then it is called component composition.

Const root = ReactDOM.createRoot(document.getElementById(“root”));

root.render(<Heading/>);

If we want to write js inside jsx then we use {} and write js inside these braces.

The babel converts this jsx into html.

We can call react element inside the component and the component inside the element.

Jsx prevents from injunctions attack it sanitize the code.

Components are the normal js functions we can call another js functions inside them.