**INCEPTION**

***Hello world using HTML***

Simply type html:5 in vs code it will generate basic html code with header and body by making use of emmet

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Document</title>

</head>

<body>

    <div id="root">

        <h1>Hello world</h1>

    </div>

</body>

</html>

***Hello world from JavaScript***

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <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 = 'Hello world form JavaScript';

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

        root.appendChild(heading);

    </script>

</body>

</html>

***Injecting React superpowers using CDN***

* We have to add two CDN links one is for React core (react.development.js) another one is for DOM operations (react-dom.development.js).
* React have two CDN because the core react we will be able to use with other react based apps such as React Native, React 3D etc.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <link rel="stylesheet" href="./index.css">

    <title>Document</title>

</head>

<body>

    <div id="root"></div>

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

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

</body>

</html>

***Hello world from React***

<body>

    <div id="root"></div>

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

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

    <script>

        let heading = React.createElement('h1', {id: 'heading'}, 'Hello world from React');

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

        root.render(heading);

    </script>

</body>

* The craeteElement() method of React is having three arguments they are Tag, Attributes and children respectively
* createElement(<tag>, <attributes>, <children> or [<child1>, <child2>])
* CreateElement() will return an object which is having props, which is a combination of attributes and children
* Root is where all our react elements will get rendered
* All the contents we written inside the root elements will get replaced by the content which we will add with root.render() method
* We will be able to add React to an existing Jquery applications without affecting anything as it is a library
* Order in which the script imports do matter as well as the ‘root’ where we render the React, that can be small portion of the page as well

Complex structure implementation ex:

 \*  <div id="parent">

 \*      <div id="child">

 \*          <h1>I'm an h1 tag :) </h1>

 \*          <h2>I'm an h2 tag :) </h2>

 \*      </div>

 \*      <div id="child2">

 \*          <h1>I'm an h1 tag :) </h1>

 \*          <h2>I'm an h2 tag :) </h2>

 \*      </div>

 \*  </div>

 \*/

let parent = React.createElement('div', { id: 'parent'}, [

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

            React.createElement('h1', {}, "I'm an h1 tag"),

            React.createElement('h2', {}, "I'm an h2 tag")

        ]),

        React.createElement('div', { id: 'child2'},[

            React.createElement('h1', {}, "I'm an h1 tag"),

            React.createElement('h2', {}, "I'm an h2 tag")

        ])

    ]);

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

root.render(parent);

**IGNITING OUR APP**

* To make our code production ready code we should remove comments and console, also should do code minification (bundling), splitting, image optimization, chunking,
* React works so fast is not only because of its performance but, it’s also with the additional packages we add to work along with it
* NPM is not node package manager instead it will manage packages
* Create-react-app It automatically has NPM inside it
* To make our project use NPM, we can add NPM to our project by using

*npm init* command with relevant inputs which generate package.json

* Package.json – It is a configuration for NPM also helps on dependency management by its version
* Most important package: bunder
* Bundlers are used to bundle/packages our app so that it will be production ready

IE: minified, cached, compressed, cleaned

* Bundler ex: Webpack, Parcel, Wheat
* create-react-app uses Webpack as the bundler (Webpack and bebel)
* npm install -D parcel

There are two type of dependencies one is dev and the other is normal

Dev is used only in the development phase

-D is used to indicate that it is a dev dependency

* In package.json if we have versions with

~ as prefix which indicate that it will allow automatic patch updates

^ as prefix, it will allow patch and minor updates

[major. minor. patch]

* package.json will keep track of what version of package installed into our system and will have ~ or ^ to mark how it can upgrade
* package-lock.json will keep a track of exact version that is being installed
* integrity key is basically having a hash which will make sure the deployed version is same as in the local
* Transitive dependency – dependency chain of dependencies,

In our case we need parcel as dependency but parcel have its own dependency, that each dependency has their own dependencies and so on. Thus, it’s a transitive dependency

* In effect will be be having a lot of package.json files because each of our dependencies have its own
* npm <cmd> is used to install a package, npx <cmd> is used to execute a package
* npx parcel index.html -> used to ignite out app
* As of now we added React into our app using CDN, we will be able to add via NPM as well
* CDN links are not a right way to inject React into our app,
  + If there is a network delay it might affect it.
  + We have to keep on updating the links if there is version update
* Browser scripts cannot have imports or exports. -> We will get such an error if we try to import/export any files in normal JS files to resolve it we have to add type=”module” as an attribute in script tag where we import our JS file
* By using type=”module” -> It will considered as module
* npx parcel index.html

Parcel will go to our index.html and build a dev build of our app and host that build into localhost

* Commands to insall React using NPM

npm i react

npm I react-dom

* Just after this installation, if we remove our CDNs and try to execute our app using parcel will fail as we didn’t injected React to our app
* Importing React into our app

import React from ‘react’

import ReactDOM from ‘react-dom/client’

Now if we try to execute our app, it will fail as we added the JS file as normal script import <script src=”/App.js”></script

To make it work use <script type=”module” src=”/App.js”></script> so that browser will consider it as module

* Parcel

Dev Build

Local Server

HMR = Hot Module Replacement

File Watching Algorithm – written in C++

Caching – Faster build (.parcel-cache)

Image Optimization

Minification

Bundling

Compress

Consistent Hashing

Code Splitting

Differential Bundling – support older browsers

Diagnostics

Error Handling

HTTPs

Tree Shaking – Remove unused code for us

Different dev and prod builds

* HMR

We will be able to see the auto refresh of our app whenever we made any change and save the code, its done with the help of HMR which uses file watching algorithm

* Prod build

npx parcel build index.html // build will helps to make prod build

We have to remove ‘main: ”App.js”’ from package.json as we already give the entry point as index.html for parcel

* browserlist

We have to use browserlist to make our app support older browser

We will be adding it in package.json

Even if we add browser list configuration only for chrome but still it will work in other browsers but there is no guarantee it will not break

Refer : <https://browserslist.dev/?q=bGFzdCAyIHZlcnNpb25z>

By adding browserlist only for a specific browser, we can reduce the bundle size

It can be used for country specific bundling as well