Day 1

APCO
Date: / /

Namaste React

Ques⇒ 1.) Difference B/w Library & framework.

It takes minimum effort to just put it inside our code - Library

⇒ Tools :-

1.) Google chrome
2.) VS Code
3.) VS Code Extensions - Better Comments (Aaron Bond)
- Bracket Pair colorization (Dzhavat)
- ES7+ React/Redux ... (dsznajder)
- Gitlens - Git Supercharged (GitKraken)
- Prettier - Code formatter (prettier)
- VSCode-icons (VSCode icons Team)

⇒ Homework - Emmet

- Created a new element <h1> with the help of Javascript and append it to <div id="root">
document.createElement() stuff comes from Browser APIs / JS Engine. Browser Knows what is document

React ⇒ A JS library used for building user-interfaces & we can just inject React into our code with bare minimum things/efforts.

⇒ Homework - CDN (Content Delivery Networks)

Date: / /

- injected React CDN links in our HTML files same as we do to use any other library.

⇒ Homework - crossorigin

- Shortest Program of JS - An Empty file
- Shortest Program of React -

```
<body>
    <div id="root"></div>
</body>

<script
    crossorigin
    src="  ________        "(react.development.js)
></script>
<script
    crossorigin
    src="  ____________     "(react-dom.develpement.js)
></script>
```

React is just not limited to Browsers. React also works on App Dev. etc.

first file is core library of React.
Second file is web version of React.
↓
like for rendering & updating the DOM stuff

React can exist without Typescript, JSX, Redux

- We can achieve the same thing as we did using JS.

⇛ React.createElement() can take 3 Arguments.

(1) Tag we are going to create – Type
(2) {} – props
(3) what you need to put inside the Tag – children

→ Const heading = React.createElement("h1", {}, "Hello")

↙

This is not a <h1> Tag. It's a React Element (object)

→ Const root = ReactDOM.createRoot(document.getElementById("root"));

↙

Creating a HTML Div as a React Root Element.

→ root.render(heading)

render() Method Takes an React Element into Root and modify DOM.

☆ Beauty of React is you can add React to your existing projects Also. Suppose if you a big project and you ~~have~~ want to use React in search Bar. you can do that just make search Bar as the root. you can just make header, footer or anything as root and use React inside it.

We can write null also instead of { }

Date: / /

We generally have one root element in React.

React.createElement(`h1', { }, 'Hello')

Global variable

children are optional and we can pass as many as needed

→ Properties of the object like className, id, Eventhandlers and style etc. (All Tag Attributes)

```
React.createElement(

'h1',
{className: 'greeting'},
`Hello',
createElement('i', null, 'Mr.'),
`.welcome',
);
```

3 children

```
<h1 className="greeting">

        Hello <i>"Mr."</i>. welcome
</h1>
```

⇒ root.render()

React will override everything what is already there inside your Root. React will Replace existing with whatever you give inside render.

Date: / /

⇒ <div id = "root"> Not Rendred </div>

In generally practice we write Not Rendered inside root Bcz if any time in your app if you see Not Rendered that means root is not Configured properly and React is not able to modify stuff inside Root.

Suppose if I mistype the name of Id wrong

~~Const root = React.creat~~
ReactDOM.createRoot(document.getElementById('roott'));

then root element will not be created/updated. & we see Not Rendered

– If we put our React Code Above its CDN Links then obviously well get errors.

⇒ Homework ⇒ Async & DEFER

writing the script Tags in Different Orders matter.

☆ Const container = React.createElement('div',

```
    {id: 'container'},
    [heading1, heading2]
);
```

when we have pass multiple children we pass it as an array.

Date: / /

React came up with writing HTML/CSS inside Javascript so that you don't have to go to HTML file again.

But React.createElement() is so complex for big projects & not user friendly. That's why JSX came in picture.

☆ A better way is to split our ~~files~~ Code in Different files.

Cut the whole JS code and paste it to App.js and link it to index.html.
We can also make index.css and link it to Index.html in <head>

We saw Diff. CDN Links in Reactjs.org.
2 for Development
2 for production

☆ productions files contains similar code as Development files. But These are minified files and much more optimized for production use. (less file size)
But we never inject our code like this using CDN for production.

Async & Defer Attributes are boolean Attributes which are used along with script Tag to load the external scripts efficiently into our webpage.

When you load a webpage there are two major things happening in browser

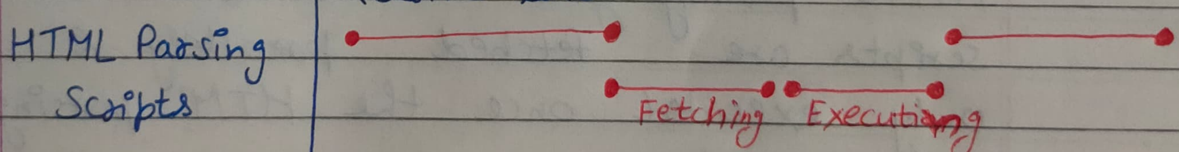
1) HTML Parsing
2) loading of Scripts

HTML files are just text files. Browser convert or parse it into DOM Tree to display it in the browser.

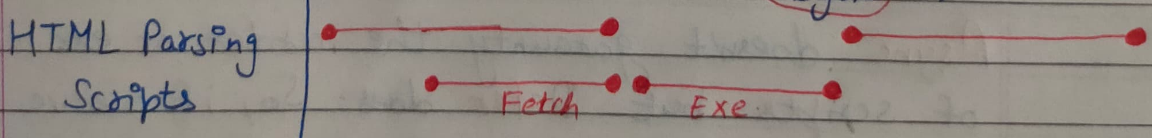
Loading of Scripts contain two parts

1) Fetching the Scripts from Network
2) Executing the scripts

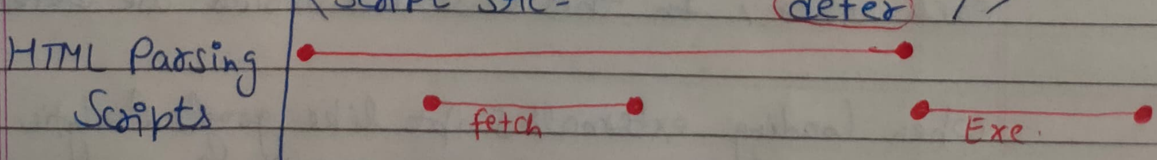
☆ <Script src=" " />



☆ <script src=" " async />



☆ <Script src=" " defer />



Date: / /

⇒ without using Async/Defer

HTML parsing goes on but as ~~y~~ soon as script tag is encountered. The scripts are fetched from the ~~Browser~~ Network and there & then they are executed. After that HTML parsing continuous.

⇒ with async ~~tag~~.

HTML parsing goes on and the scripts are fetched in async manner/parallely. After the scripts are fetched. The scripts get executed and after that HTML parsing Continuous.

⇒ with Defer

HTML parsing goes on and the scripts are fetched parallely & only executed once the HTML parsing is completed.

Async doesn't garaunty the order of exe. of scripts but Defer does. So, if we've multiple scripts ~~or~~ that are dependant to each other, In this case we should use Defer.

When loading externals script like google Analytic scripts which are independent of other scripts. Async is prefer.