1) Make your own notes
2) use your laptop / Desktop
3) maintian your own github repo
   ↙ notes ↘ assignment

Hello world html → JS → React

H/w
CDN
be curious about these small dam things
cross orgin

html

```
<div><h1> Hello world from HTML </h1></div>
```

JS

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

<script>
// Create element in document object and update inner text with "Hello world..."
const heading = document.createElement("h1");
heading.innerHTML = "hello world from JS";
// select 'root' id element in document object and append its child with heading
const root = document.getElementById("root");
root.appendChild(heading);
</script>
```

React

```
<script crossorgin src="link..."> react. development
<script crossorgin src="link..."> react-dom development
```

we are using CDN links to get react in our Project

CDN – content Delivery network

there are two types of CDN links for React

development
heavy and useful for development purpose
/react.development.js
/react-dom.development.js

Production
minified & optimized
/react.production.min.js
/react-dom.production.min.js

so when want to push code to production use production CDN link

CDN - content delivery Network
- script codes files etc
- send & receive response request
- network of local servers based on region called as edge server / node

these edge server/node stores the copies of original files across it network and whenever the user request them it provides the file from the nearest servers

hosting V/S CDN

hosting - where your orginal code file are kept (main server) ex - hostinger, godaddy

CDN - copies the orginal files and stores them in local server / edge servers and

when request comes it delivers them from nearby servers

note – CDN in most cases only copies the static files only it doesn't include dynamic file (user info)
(html, css, js)

to make website faster

static files are stored (cached) in CDN with placeholder for dynamic components

as soon as the multiple request is generated by browser

static → CDN
dynamic → backend server

1) The page template (from CDN) is ready instantly
2) it has 'slots' or place holder for dynamic parts.
3) then using JS (React or Vue)
   - user specific data is fetched from backend API
   - fills in the dynamic part
   - this makes page interactive
4) and hence user sees a fully loaded dynamic, fast website

it still faster as if CDN was not there then whole website would be loaded from orginal server but in case of CDN only Dynamic component what only needed to be placed in Placeholder is

loaded

Cross orgin

whenever a HTML file load a script (JS file) from different orgin (i.e. from different website so use some component in the present file) this becomes cross orgin (cross /multiple website) request

Cross orgin attributes tell browers that whenever makes request to d that different website does you should include your current file credentials like cookies or HTTPS authentication

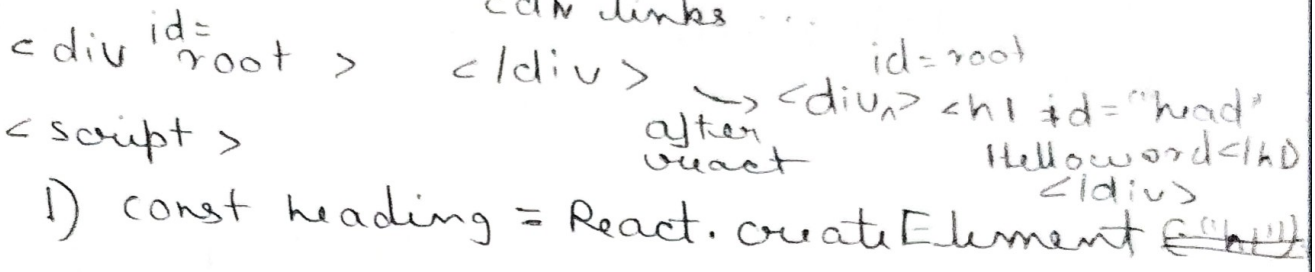
crossorgin = "anonymous" or no value
- No credentials are sent and requested script is get back without credentials sharing

"use-credentials"
credentials are shared

house = brower
your toys need a lego piece in another ~~home~~ friends house you requst brower to go ~~the~~ to another house and bring that piece while bringing dont share credential like homekey pass word security camera access etc

same in this case you tell hey browser fetch this React script from another domain (unpkg.com) but don't send any of my credentials (like cookies, tokens, session info) with the request for security purpose

React code for hello world



1) const heading = React.createElement("h1", {id:"head"}, "Hello world");

// this create an virtual element in React ~~de~~ and type of heading variable will be object

createElement is method in React object that takes 3 input (, { }, content

- the element to create
- object which stores attributes for the element
- content inside the element stored as [children]

in this step we create an virtual element using createElement method

2) const root = ReactDOM.createRoot(
document.getElementById("root"));

select the element where we want to make changes

Since we are doing modifying which is done through DOM, ∴ we will ReactDOM

~~inside~~ instead of React object
this create an object with properties & method that we can modify

3) root.render (heading);

// this render method will take this virtual element object and convert this in suitable element in this case h1 and put in the root div

usage of Render
React Element (Object) => HTML (Brower understand)

Syntax

root Object.render (child Object);
↓ React object of element to be added
React object in which I want to add element

Multiple Render

In react when you use multiple Render line then the last render statement replaces whatever was previously rendered in that root so. only the last render call will appear in the DOM

```
root.render (heading);
root.render (parent); // Replace heading with parent
```

To combine both

 wrapper

wrap heading & parent object in comanon parent like

```
const contianer = React.createElement(
    "div", {}, heading, parent);
root.render (contianer);
```

downside is extra div/contianer is added to DOM structure which can alter CSS layout

React.Fragment

a special built in component in React that let you grouph multiple element with adding extra DOM nodes

```
const contianer
= React.createElement(
React.Fragment, {},
heading, parent);
root.render(contianer);
```

∴ To render two or more element together wrap them in common parent like or React.Fragment and render that common parent

Creating Nested React element

```
<div id="parent">
    <div id="child">
        <h1> hi I am h1 tag </h1>
        <h2> hi I am h2 tag </h2>
    </div>
</div>
```

the child div contian two children h1, h2

∴ to add both in child div we make array [ ] and add them inside array

```
const parent = React.createElement("div", {id:
"parent"}, React.createElement("div"
, {id: child}, [ React.createElement
("h1", {}, "hi I am h1 tag"),
React.createElement("h2, {},
"hi I am h2 tag") ]));
```

nested sblings using array

HIW

1) what is emmet

emmet are shorthand tool that saves our time by writing shorthand for a code snippets

type short expression ⟶ automatically expand into full code snippets

just press tab key after writing shorthand

important syntax

> → create child element

\+ → create sibling element

*n → multiply the element n times (repeat n times)

#idName → used to give Id name to element

.className → used to give class name to element

{text} → used to add inner text to the element

[custom attribute] → used to give custom attributes

ex 1) form > input : text + input : password + button {loggin}

```
<form>
    <input type="text" />
    <input type="password" />
    <button> login </button>
</form>
```

custom attribute ↑

2) a#link.red [href="https://google.com"]

```
<a id="link" class="red" href="google.com">
</a>
```

3) div.container > h1*3 {Hello world

```
<div class="container">
    <h1> Hello world </h1>
    <h1> Hello world </h1>
    <h1> Hello world </h1>
</div>
```

2) difference between library & framework

library	framework
→ can be just applied to small part of code	→ framework calls your code it dyines structure and decide execution of your code
→ you call the library (function, components etc) in your code whenever wherever you want	→ It has more rules to follow
→ React is library	→ Next.js is a framework for React

library is a like a toolbox eg wood & tool provided to make your custom table

| it a pre built system like Ikea kit that just needed to be assembled

so library gives you more freedom - you pick and choose where and how to use it whereas framework provide pre built system / template that makes you code faster and efficient but less flexible

3) why is React called React

because before React developers had to

- manually track changes in the data
- manually find the right DOM elements to update
- make sure other parts are not broken will doing this also JS DOM reloads whole DOM for even a smallest change which became inefficient

with React

React used Reactive rendering i.e. whenever data changes you define the UI after the data changes the react recognise the data changes and changes the UI or re render only the UI that is needed

this is done through React virtual dom
the React compares its virtual dom with
real dom and changes / re rendered only
the nessary part in webpage

create React element ← when data changes
↓
change in virtual dom
↓
compare virtual dom with real dom
↓
make only nessary change i.e. re-rendered
only the required element

4) what is difference b/w React and ReactDOM

React: is the core library that contian
tools for building React components
and managing their state (data changes)
props (data from parent), lifecycle
(mount (born), update, unmount (die)
a react component based on state)
rendering logic (code that contian
exactly what to change

ReactDOM: it is used for rendering
React components into actual DOM in
web brower i.e. connecting React with
the brower's DOM

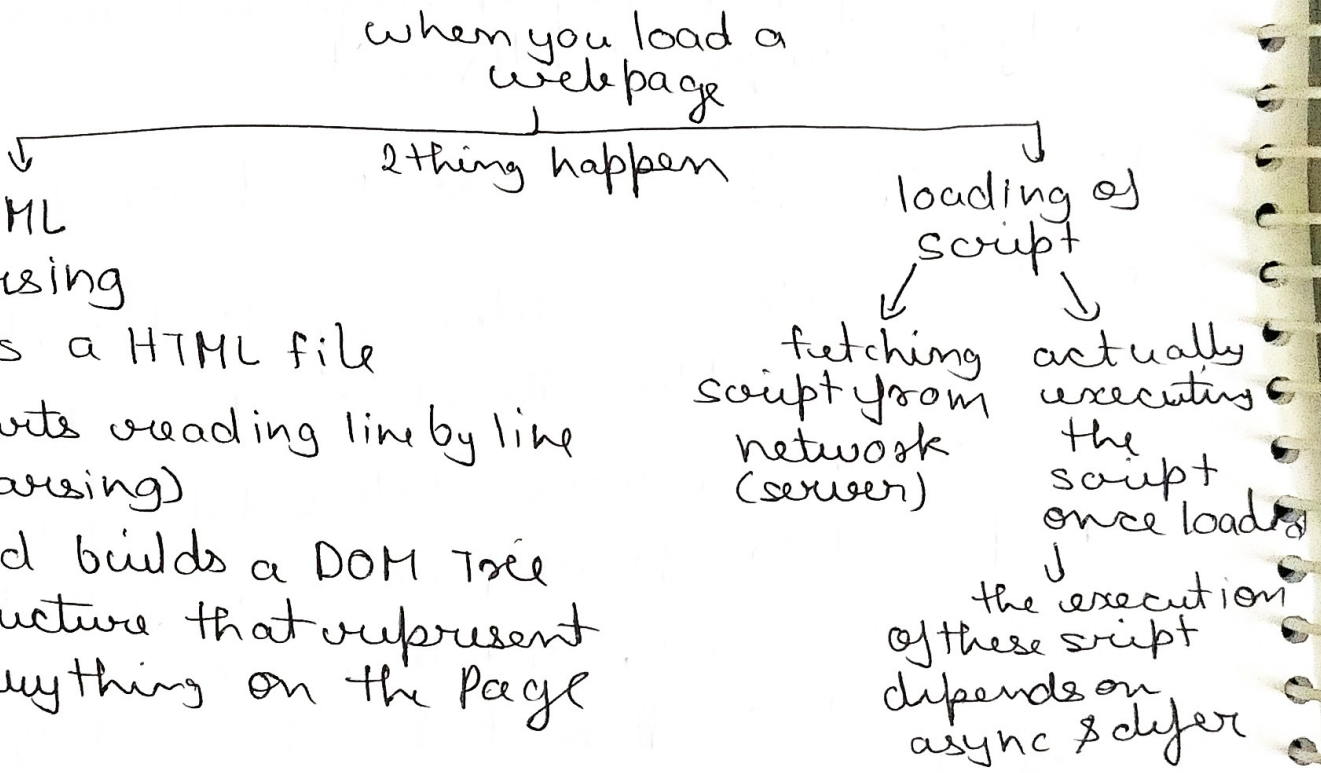
In and

React - blueprint & logic

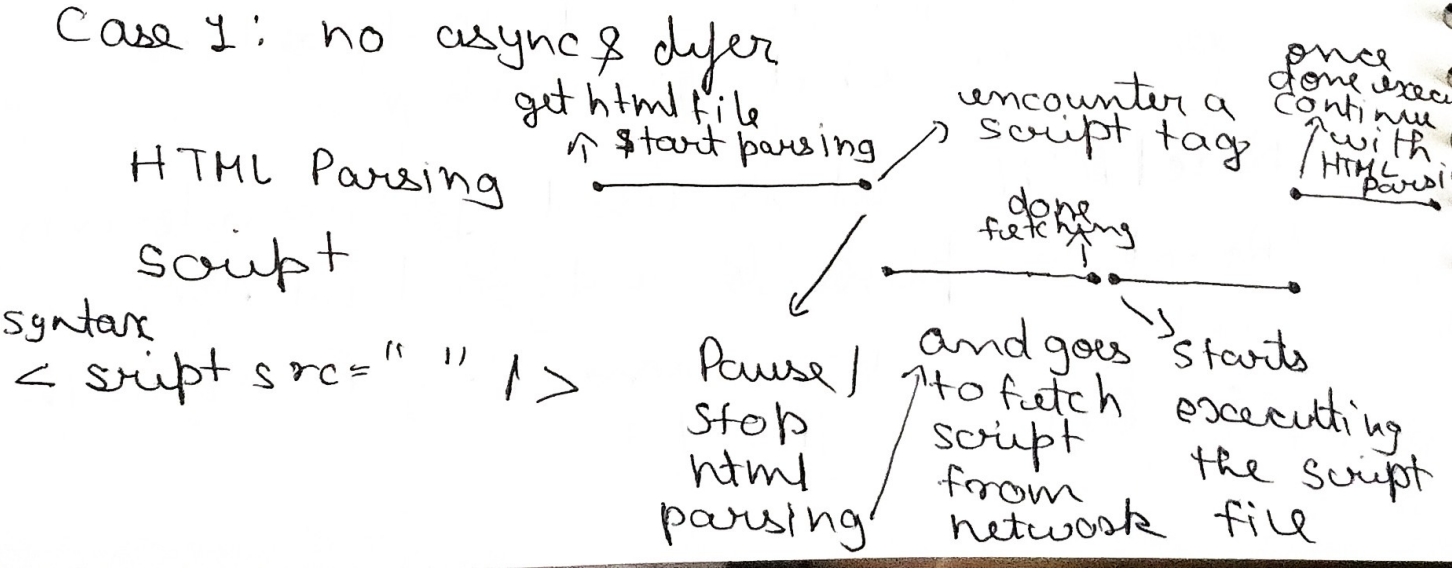
ReactDOM - use that blueprint & logic to build it on webpages

5) what is async and dufer

async & dufer are boolean attributes that are used in script tag to load the external script efficiently to our webpage



Case 1: no async & dufer

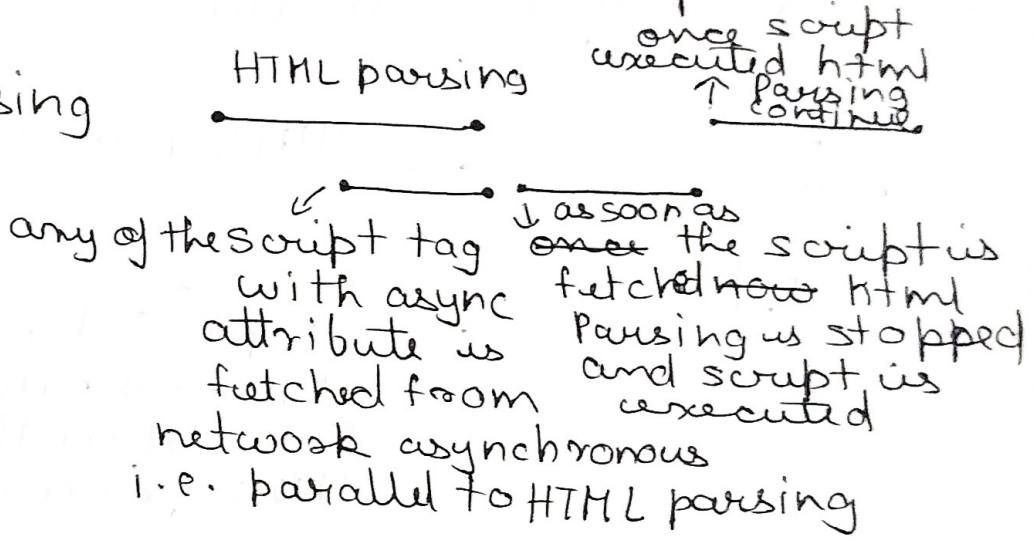


syntax

```
< script src="" />
```

get an html file & starts parsing it when it encounter a script tag then and there stop parsing html and goes to fetch the script from the network once the script is fetched it executes it after execution it goes/continue to HTML parsing if another script is encountered then same process is repeated till we reach end of the Program

Case 2: async attribute in script



HTML parsing & fetching ~~from~~ the sript from the network is done parallel i.e. at the same time

once script is fetch the html parsing is stopped and script is ~~loade~~ executed after execution html parsing continue

Note: async does not gurrentte the order of execution of script file if one script get fetched faster than other then it is executed first. this is important because we dont want to use async in script

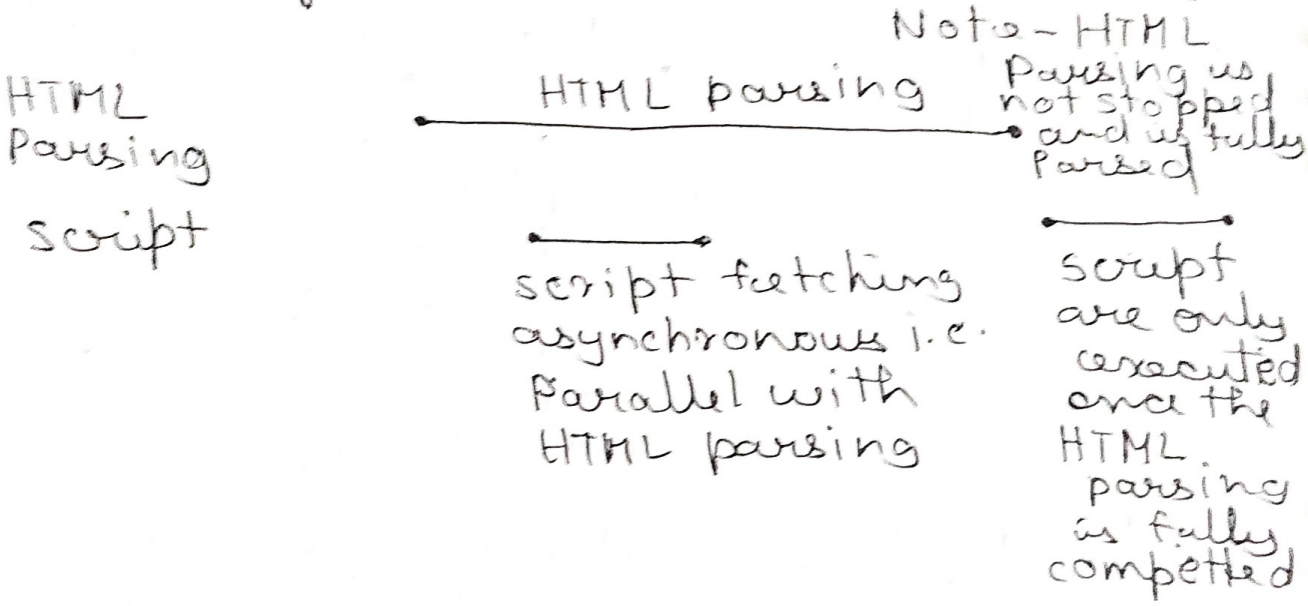
where one script is depend on the executing of another script

ex script src = " unpkg.com/react "
script src = " ./ app.js "

loading of react from CDN is important to execute react code in app.js ∴ async should not be used where order of script matter

syntax -> <script async src = " " />

Case 3 : defer attribute in script tag



HTML parsing & script fetching from network is done parallely
once the full html parsing is completed then only the execution of script starts
defer maintian the order of execution of script

syntax -> <script defer src = " " />