7/02/24

Namaste React

Episode - 01

→ Keywords like innerHTML, document.createElement, appendchild.

Que How does this keywords are understand by our browser ?

Ans : Brower have a Java Script engine init that executes this JS.

Que : Do our browser understand react also ?

Ans : NO. Brower doesnot understand react.

we need to get react into our project.] → 1st Step

↳ 1st way : With help of cdn.

Que : What is cdn ?

Ans : cdn Stands for content delivery networks (CDN)

There are the ~~website~~ servers/websites where react has been hosted and we are just pulling react into our project.

* Look into document to know more!!!

Que : What is cross origin in Js Script tag ??

Ans It basically allows us to access resources or information or codes from a different website or domain where it is loaded.

→ When we inject react into our project after that the browser will be able to understand our react code.

→ writing Namaste Duniya in react.

→ 1.) create a Script tag.

→ 2.) create a h1 tag in react.

we can give any kind of attribute to our element — object — used to give attributes to any element like class, id

```
const heading = React.createElement("h1", {}, "Namaste Duniya");
```

React.createElement: as any tag is just a part of core react.

"h1": Name of tag

"Namaste Duniya": what to write inside that tag

→ 3) Where we want to render our code ?

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

root: as root is part of ReactDOM

"root": in HTML body where we want to render our code

→ 4) Do Rendering with code

```
root.render(heading);
```

root: place where we want to render

render: keyword

heading: code which we want to render.

→ React comes with a simple philosophy :-

Do DOM manipulation only using JavaScript (JS).

```
React.createElement("h1", {}, "Hiee");
```

{}: used for attributes.

→ It will give us or create a react ~~object~~ component / element

A React component is nothing but a JS object

→ This JS object will be converted into HTML while it get render.

→ But to create elements we will not use createReact, we will use JSX. It will make our life more easier.

```
<body>
  <div id = "root">
    <h1> Nihal is here </h1>
  </div>
</body>
```

HTML code

```
Some react code
.
.
.
const head = ReactDOM.createRoot
(document.getElementById
("root"));

head.render(element);
```

React code

→ Now when react call render its component (element) into root div then everything which is already written inside that div will be ~~replaced~~ by HTML version of react element.

↳ Replaced

→ In render( ) we can only pass one Single attribute of a time.

→ If you pass 2 attributes then it will take only 1st attribute & ignore 2nd attribute.

→ React can be apphed to only one Single tag in an HTML. Rest HTML code will be Same, there will not be any change in them. One of the reasons why react is called a Library.

→ React can work on a Single Part of the HTML also.

Understanding CND

Imagine you're at a restaurant with a huge menu. If the kitchen is far away, it takes longer to get your food. But if there's a small kitchen closer by with the most popular dishes ready, you get your food much faster!

A CDN (Content Delivery Network) is like that small kitchen. It's a network of servers around the world that stores copies of popular website content (like images, videos, and code) closer to users. When someone visits a website that uses a CDN, their request goes to the nearest server, not the main website, which can be far away. This makes the website load much faster!

Here's the gist:

- Main website: Like the main kitchen with all the dishes.
- CDN servers: Like small kitchens with popular dishes closer to customers.
- User: You, waiting for your website content (food).
- Faster delivery: Getting your content from the nearest CDN server instead of the main website.

So, CDNs help websites load faster, especially for users who are far away from the main server. This makes websites more enjoyable to use and keeps users happy!

React CDN Development Links

```
<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>
```

https://unpkg.com/react@18/umd/react.development.js

Above is the websites where plane js code of react is uploaded, as react is a js library, it is written into js.

https://unpkg.com/react-dom@18/umd/react-dom.development.js

Above is the react DOM library code in js which is useful for DOM (Document Object Model) manipulation.

What is ReactDOM ?

ReactDOM is a package in React that provides DOM-specific methods that can be used at the top level of a web app to enable an efficient way of managing DOM elements of the web page. ReactDOM provides the developers with an API containing the various methods to manipulate DOM.

React does not only works on browsers, it also work on phone as react native so there are different types of places where react are used so there are different methods or functions which are being used between different places where react are used such as on browser or on phone.

So that's the reason why we have 2 different links in our CDN, First one is the core react and second one is like a bridge between the react and the Dom of different places where react is being used such as browser or phone.

For using react on phone we have some different functions and same for browser.

What is crossorgin in script tag of react ??

The "crossorigin" attribute is a security measure that allows a script tag to access resources from a different domain than the one where it is loaded. This is useful when working with online APIs or external libraries, as it allows them to be used without having to be served from the same domain.

Rendering

Imagine you're building your dream house out of Legos. You have all the pieces and instructions (your React code), but you need to assemble them to see the actual house (the UI displayed on screen).

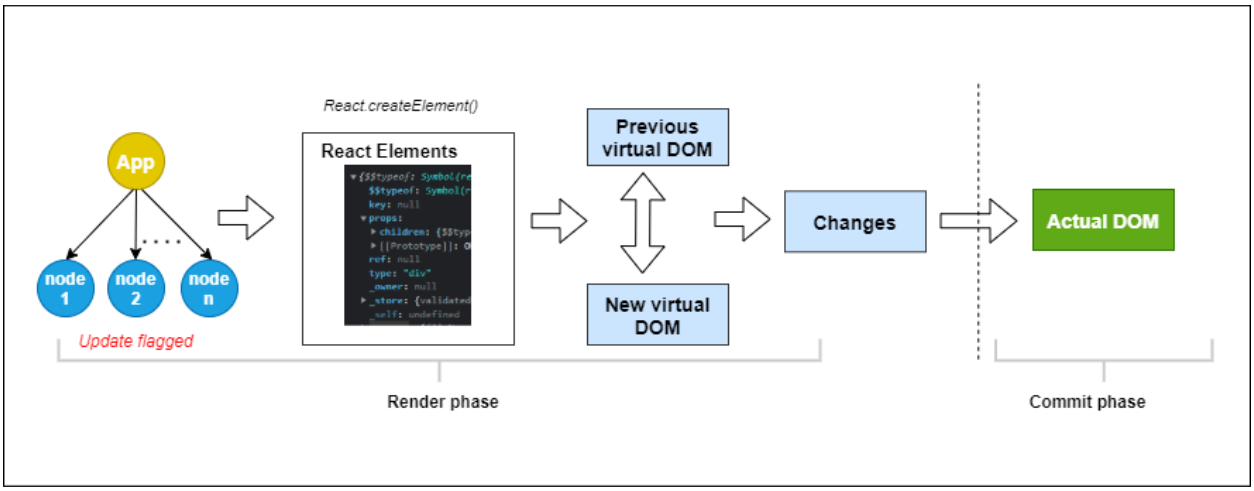
In React, "rendering" is like putting those Legos together according to the instructions. It takes your component code (like Lego pieces) and translates it into what users see on the screen (the assembled house).

Here's the basic process:

Building blocks: You write code for components, which are like individual Lego pieces, each with its own properties and data.

Putting them together: The render function in your component acts like the instructions, telling React how to assemble the pieces (components) into a complete UI structure.

Displaying the result: This structure is then translated into HTML code that your browser understands, so you see the final UI on the screen!



React library is used to create components. A component is a building block of an application. React library has classes and methods for this purpose.

On the other hand, React-DOM deals with placing the components on browser. It deals with shadow DOM and other efficient rendering techniques.

Now, if we consider React Native development, we again use React to build the app components. But we use React Native to build and publish the app for mobile devices.

The point I am trying to say is, React, as a component library is reused by several platforms like React Native, React 3D or React Art. That is why it is maintained as a separate project and package.

Production Links and development link in cdn

In the world of React and CDNs (Content Delivery Networks), imagine production links as special codes that make your website super fast for users!

Here's the scoop:

- Decvelopment links: These are like "uncooked ingredients" for your website, containing all the code needed (like React and React DOM). But they're bulky and take longer to load for users.

- Production links: These are like "ready-to-eat meals" made from those ingredients. They've been shrunk down and optimized (minified, combined, etc.) so they download and work much faster. Think of them as the "lite" versions of the code.
  Why use production links with a CDN?

- Speed boost: CDNs deliver content from servers closer to users, and production links make the content even smaller, resulting in lightning-fast loading times. Happy users, happy you!

- Data savings: Smaller files mean less data used, which is great for users on mobile networks or limited bandwidth.

- Security: Some techniques used in production links can make it harder for attackers to understand your code (but it's not foolproof!).

  Remember: Use production links only when deploying your website live, not during development (you need the full code for debugging then).

  It's like the difference between building a giant Lego castle and taking it apart to fit it in a travel-sized kit. Both have the same parts, but one travels and works better!

Using production links while coding your React website on your laptop isn't recommended for a few key reasons:

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

1. Debugging Difficulty: Minified and Uglified Code: Production links make the code more compact and harder to read because of minification and uglification. This makes it significantly challenging to debug issues as variable and function names become obscure. Imagine trying to decipher a recipe written in secret code – troubleshooting becomes tedious!

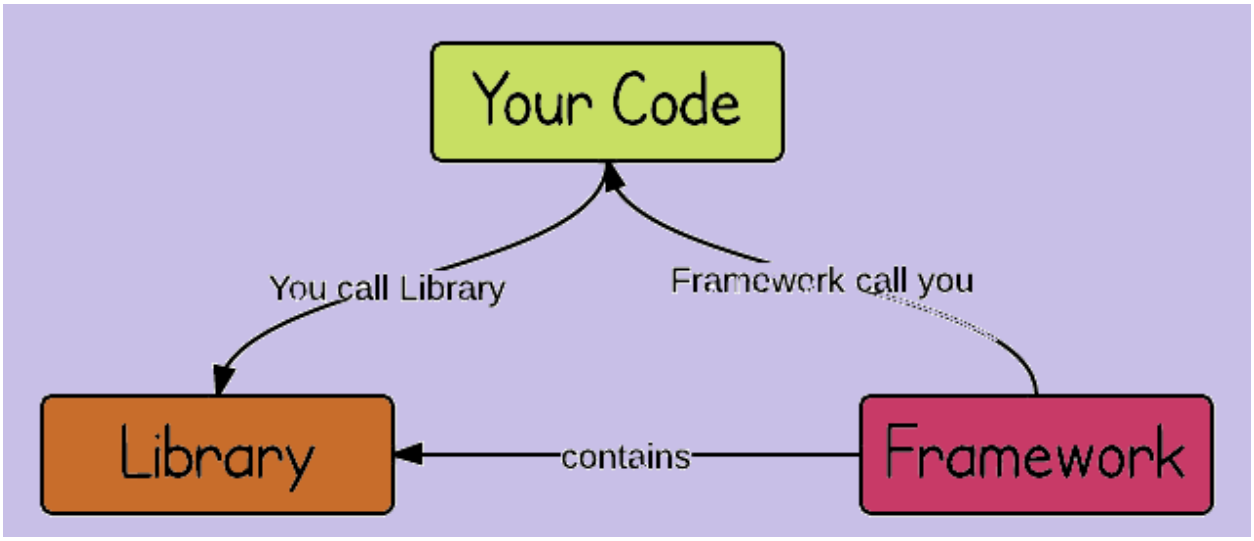
2. Source Map Overhead: Extra Files and Slowdown: While it's possible to use source maps with production links, these maps translate back to the original, readable code for debugging. However, they add additional files and can slow down development processes due to their size and complexity. It's like carrying around a thick dictionary to understand a simplified text – convenient for emergencies, but cumbersome for everyday use.

3. Development Server Benefits: Hot Reloading and Live Updates: Development servers typically offer features like hot reloading, which automatically updates your browser when you make code changes. This allows for faster iteration and instant feedback, something invaluable during development. Using production links wouldn't allow you to take advantage of these features. It's like having a self-updating cookbook – instant recipe adjustments for quicker learning!

4. Testing Environment Isolation: Simulating Real-World Scenarios: Development environments often provide tools to simulate real-world deployment scenarios like network throttling or different browser versions. This helps ensure your website performs well in various conditions. Production links are optimized for specific setups, so they might not reflect this broader range of situations accurately. It's like testing a recipe in a professional kitchen versus your home kitchen – different environments call for different approaches.

In summary, while production links are crucial for a fast and efficient live website, their optimized nature makes them unsuitable for the interactive and debugging-intensive needs of development. Stick to the "uncooked ingredients" (regular links) during development, and reserve the "ready-to-eat meals" (production links) for when your website is ready to serve the world!!!

Library VS Framework



Imagine you're building a house. Here's how libraries and frameworks differ in this analogy:

Library:

1. Like a toolbox with specialized tools (functions and classes) for specific tasks (e.g., hammer for nails, saw for wood).

2. You choose the tools you need and control the overall design and structure of your house.

3. More flexibility, but requires more planning and effort from you.

Framework:

1. Like a prefabricated house kit with walls, doors, windows already prepared, like a premade structure or architech.

2. You follow the instructions and customize the kit within its limitations.
3. Less flexibility, but easier and faster to build a basic house.

4. Remember:

- Choose a library for specific functionality or when you need more control over your project.
- Choose a framework for building complex applications where structure and consistency are important.

I hope above analogy helps!

- Both the framework and library are precoded support programs to develop complex software applications. However, libraries target a specific functionality, while a framework tries to provide everything like structure and architecture of the application required to develop a complete application.

- So when you develop a software application, you will need many libraries, but often one or two frameworks.

- Popular examples of frameworks are Ext JS, Angular, Django, Spring, and Rails, which offer a comprehensive set of tools and components for application development.

- On the other hand, popular examples of libraries are React and jQuery, which focus on specific tasks or functionalities and can be used in conjunction with frameworks. The choice between a framework vs library depends on the specific requirements and scope of the project at hand.

--→ What are async and differ attributes in <script> tag?

Async - The async attribute is a boolean attribute. The script is downloaded in parallel(in the background) to parsing the page, and executed as soon as it is available (do not block HTML DOM construction during downloading process) and don't wait for anything.

**Syntax** <script async src="demo_async.js"></script>

Defer - The defer attribute is a boolean attribute. The script is downloaded in parallel(in the background) to parsing the page, and executed after the page has finished parsing(when browser finished DOM construction). The defer attribute tells the browser not to wait for the script. Instead, the browser will continue to process the HTML, build DOM.

**Syntax** : <script defer src="demo_defer.js"></script>

--→ Why React is known as React ??

React is known as react because it reacts. It is developed by Facebook developers, it is mainly develop to show the changes effectively in UI to the user in a better way, means how page reacts to the changes being done on page while clicking on any button or etc. It "reacts" quickly to changes without reloading the whole page

-→ What is Emmet ??

It is used by developers for shortcuts, for example when we type ! and hit enter in vs code we will the basic HTML code, this is emmet. It basically allows us to type shortcuts which are converted into the actual code.