Namaste React

Page :
Date : 13/07/2025

Episode - 01 Inception

Hello world using plain HTML :

```
<html>
  <head>
    <title> Namaste React </title>
  </head>

  <body>
    <h1> Hello World! </h1>
    <div id = "root"></div>
  </body>
</html>
```

Page :
Date :

```
const heading = document.createElement("h1");
heading.innerHTML = "Hello World from Javascript!";
const root = document.getElementById("root");
root.appendChild(heading);
</script>

</body>
</html>
```

=> document.createElement creates an element h1 and in its inner HTML we write the code and then we get div container with the help of its id and append h1 in it as its inner child.

=> Browsers are able to understand the document.createElement, innerHTML etc because it has an API called DOM API and DOM API have all this inside. it and browsers also have Javascript engine inside it which help them understand these code.

=> Browser don't have react engine and we should configure our project as well to use react.

(dn → Content Delivery Network these are the website where several libraries like react has been hosted and pulling from there to use in our project.

Page:
Date:

=> One way to do that is go to cdn link and bring the react link and just above the closing tag of body paste that code and now your project is ready to use react.

```
<body>

<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>
```

=> The first script tag above basically brings react library to your document.
(framework algorithm)

=> The core react & fundamental is written inside the first file

=> The second file inside the script tag is used for the manipulation of the DOM file using React.

Page:
Date:

⇒ So, couldn't we has just one file and inject everything to it?
No, because react also works on Mobile phone as react-native, there is react-3D so react is not only the browser thing so here we include 2 files one is core react framework algorithm and other one is used for manipulating documents of the browser. It is like the bridge between the react and browsers.

Hello World! using React :-

```
<script>

const heading = React.createElement("h1", {},
                "Hello world from React!");

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

root.render(heading);

</script>
```

⇒ Inside script tag we just create an element using React.createElement and it takes 3 arguments and React.createElement is core React thing :-
(i) What element you want to create (h1 here)
(ii) an ~~empty~~ object (there objects are used to give attribute to those elements).
ex:- {id: "heading"}, xyz: "abc"}

(iii) What you need to put inside that heading. (just like inner HTML using JS).

⇒ Now what we have to do is to create a root and everything is going to render inside that root. All our code is going to be put inside that root.
for that we have used ReactDOM.createRoot(document.getElementById("root")) and here we used ReactDOM because it is the part of DOM where we create our root and do the manipulation.

⇒ Now in the third step we have put our heading (render our heading) inside root.

⇒ You have to put these code next to the React file. otherwise it throws ReferenceError: React is not defined.

⇒ One of the most costly operation on the web page is the manipulation of DOM. i.e. inserting elements, removing elements inside DOM. and these libraries and framework are the optimized way to do these.

⇒ React also comes with philosophy that whenever you have to do any manipulation on web page do it using JavaScript.

⇒ Now you can also put that JS code in separate file App.js and link it to your page using <script src = "./App.js"></script>

Page :
Date :

What is React.createElement ?

=> React.createElement returns an object. This is the react element at the end of the day and this heading is the react element which is a javascript object.

If we do :- console.log(heading);

O/P :- ▾ {$$typeof : Symbol(react.element), type: 'h1', key : null, ref : null, props : {...}, ...}

=> It has also something called props inside it all its attributes and children.

▾ props :
children : "Hello World from React!"
id : "heading"
xyz : "abc"

So, Hello World from React! is the children that will go inside h1.

=> The job of "root.render(heading);" is basically it takes the object (React element) and creates a h1 heading inside DOM and put it inside root.

Page:
Date:

Creating Nested element

```
<div id = "parent" >
    <div id = " child" >
        <h1> I'm an h1 tag </h1>
    </div>
</div>
```

code:-

```
const parent = React.createElement
    ("div", {id: "parent"},
    React.createElement ("div", {id: "child"},
    React.createElement ("h1", {}, "I'm an h1 tag"
    )));

const root = ReactDOM.createRoot (document.getElementBy
    ("root"));

root.render(parent);
```

=) So we can create nested structure by passing React.createElement as third ~~attribute~~ (argument) to it.

=) React.createElement is a React element which is a JS object and it gets converted to HTML tag which browser understands. by using root.render(parent).

Page :
Date :

=> If I had to create more than one children then I have to use the array of children over there.

ex:-
```
<div id="parent">
    <div id="child">
        <h1> I'm h1 tag </h1>
        <h2> I'm h2 tag </h2>
    </div>
</div>
```

code:-
```
const root parent = React.createElement
    ("div", {id: "parent"},
    React.createElement("div", {id: "child"},
    [React.createElement("h1", {}, "I'm h2 tag"),
    React.createElement("h2", {}, "I'm h2 tag")]
    ));

const root = ReactDOM.createRoot(document.getElement
                                 ById("root"));

root.render(parent);
```

=> <h1> and <h2> both are siblings because both of them are present inside tag.

Page :
Date :

=)

create :-

```
<div id="parent">
    <div id="child">
        <h1> I'm h1 tag </h1>
        <h2> I'm h2 tag </h2>
    </div>

    <div id="child2">
        <h1> I'm h1 tag </h1>
        <h2> I'm h2 tag </h2>
    </div>
</div>
```

code :-

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

  React.createElement("div", {id: "child2"},
    [ React.createElement("h1", {}, "I'm h1 tag"),
      React.createElement("h2", {}, "I'm h2 tag"),
    ])
  ]);
```

Since this is very complex to write therefore we have jsx and this way of writing code is the core part of react.

Page:
Date:

Suppose I have:

```
<div id="root">
    <h1> Prashant is here! </h1>
    <h2> Akshay is here! </h2>
</div>
```

No matter how much tags are inside this div tag all of them will get replaced by what we passed inside root.render

```
root.render(parent)
```

parent will replace all the tags of div.

⇒ What happens is browser started reading html file it sees h1 inside a div is fine that then loads React files then ReactDOM files and then it loads our JS code and then that code is getting executed line by line and when it encounters react.render it replaces our ~~div~~ h1 inside div with that parent. Hence React is taking control of our code.

⇒ Suppose we have two h1 element one on the top of div and one below so they won't be getting replaced. Everything that is going to render will render inside the div which has id = "root".

Page :
Date :

=> That is why React is called as Library. because it can work independently even in the smallest portion of our code app.

=> Framework is whole of a way to design code