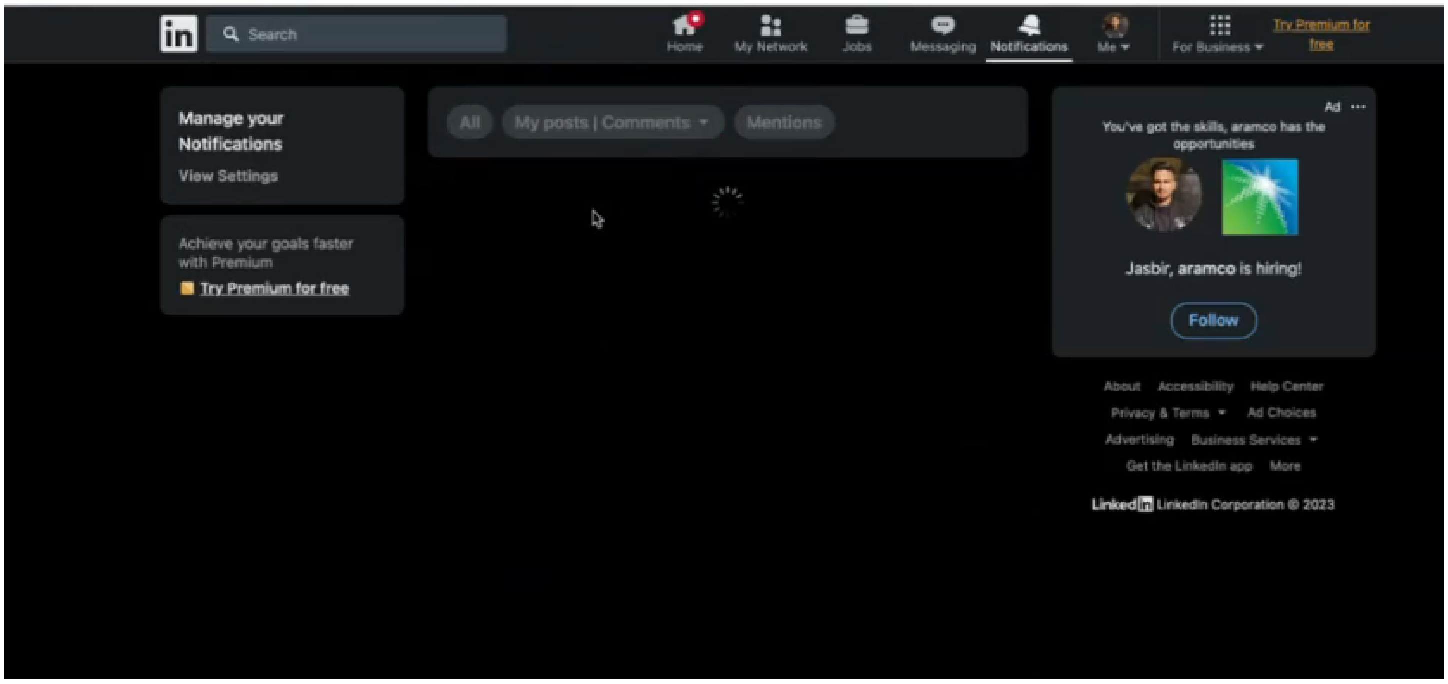
React-1

Full Stack LLD: React-1: Introduction to React: JSX, Props, lists

Challenges in Modern Frontend

Explore the single-page layout of LinkedIn and its diverse UI interactions. Identify various functionalities such as Home, My Network, Jobs, Messaging, and other `



Manage your Notifications

View Settings

Achieve your goals faster with Premium

Try Premium for free

You've got the skills, aramco has the opportunities

Jasbir, aramco is hiring!

Linkedin Corporation © 2023

Que What are the Challenges faced in the modern frontend and build a webapp like linkedin

These are the major challenges:

Single Page Application (SPA) Challenges: Websites are now behaving more like applications, where they don't reload the entire page but only specific parts of the UI.

Code Maintainability: As the web app becomes more complex, keeping the code organized and easy to understand becomes challenging. Collaboration among developers becomes important to ensure everyone is on the same page. Writing clear and readable code, breaking down the code into smaller parts, and adding comments help in maintaining the code. Additionally, testing the code regularly ensures that new changes don't break existing features, making it easier to fix issues and add new features in the future..

Handling Complexity and Performance Optimization: Websites like Facebook, Instagram, Twitch, and Flipkart have numerous dynamic components and micro-interactions, making the UI highly interactive. Managing this complexity while ensuring performance and responsiveness poses a significant challenge. This includes optimizing DOM manipulation, minimizing re-renders, and reducing the load on the browser.

Solution

- **Single Page Application** : to develop a single-page application (SPA), we utilize technologies like the Fetch API or AJAX requests. These technologies allow us to fetch data from the server without needing to reload the entire page. Additionally, we leverage the History API, which enables us to change routes within the application without triggering a full page reload. This helps in creating smoother and more seamless user experiences.
- **Code Maintainability** : Previously, tools like webpack, gulp, and grunt were used to address this issue. However, now tools like create-react-app or vite help us manage code maintainability more efficiently. These tools streamline the process of splitting the application into multiple parts, components, or files, and then combining them together seamlessly. This simplifies the development process and ensures that the codebase remains organized and easy to maintain]
- **Handling Complexity and Performance Optimization** : "Previously, we relied on the DOM API to manage user interface (UI) interactions. However, with the increasing complexity of modern web applications and the need for dynamic user experiences, using just the DOM API becomes insufficient. The level of interactivity demanded by today's applications often surpasses what can be efficiently managed solely through the DOM API, presenting challenges for developers to maintain and optimize performance.

Take away : There were no solutions to effciently update DOM -> **React solves this problem**

In this section we will understand what is the purpose of React
`Instructor cue : write this question and give the answer yourself`

What is React?

React: It has the algorithm which can efficiently tell you how to manipulate the UI .
This is more-or less only purpose of react
`Instructor cue : Now ask the question ->`

Why did i say User interfaces not DOM any idea ??

Ans : Because react is generic it explain you how to update the UI and UI can be anything like `Mobile, Desktop, VR` and there should be an associating library that will actual do the changes. Let's see some examples

Here are different User intefaces and the library we need to use with react to do the changes:

Mobile	React Native
Web	React DOM
VR	React VR

---

title: Hello World in react with CDN
description: Discussion of basic application in react in detail.
duration: 2100
card_type: cue_card

Hello World in React

`Instructor cue: Now emphasis that -> Let's start by understanding all the different parts involved in creating a React app. This will help us grasp the flow of the code better without diving into various technical jargons`

Steps

- create `hello.html` file

```
<!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>
</body>
</html>
```

- Add cdn link of react and reactDOM : you can choose to open the link and show that react is nothing but js file

```
<head>
    <!-- algorithm to efficiently manipulate your UI -->
    <script src = "https://unpkg.com/react@18/umd/react.development.js"
crossorigin></script>
    <!-- React DOM  -> uses react to update the DOM-->
    <script src = "https://unpkg.com/react-dom@18/umd/react-
dom.development.js" crossorigin></script>
</head>
```

- Write the Hello world component

  Root : it is the div in which the whole react application is rendered

  `script`: in script all the react code is written

  `HELLO component` *: it is the component and component starts with Uppercase and in the return it will return HTML*

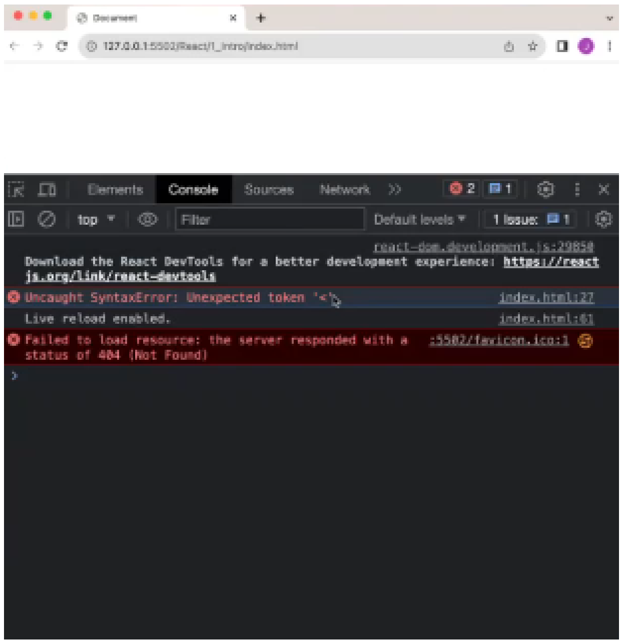
  `ReactDOM.render`: This function is responsible for generating the HTML code for the `Hello` component and inserting it into the root element.

  root element

```
<body>
    <!-- the whole application lives inside-->
    <div id = "root"></div>
    <!-- single Page applcation : JS -->
    <script>
        function HELLO() {
            return <h1>Hello React </h1>;
        }
        ReactDOM.render(<HELLO></HELLO>, document.getElementById("root"));
    </script>
</body>
```

`Instructor cue: run the code in browser. It will give error , now ask the students why are you getting the error`.

This just prints nothing on the browser as shown below:



```
                                            react-dom.development.js:29850
Download the React DevTools for a better development experience: https://react
js.org/link/react-devtools
Uncaught SyntaxError: Unexpected token '<'                   index.html:27
Live reload enabled.                                         index.html:61
Failed to load resource: the server responded with a    :5502/favicon.ico:1
status of 404 (Not Found)
>
```

`Que`: What are the languages that are understood by Browser ?
Ans : HTML , CSS , JS cool

Ask the learner : is the above code a valid JS ??

Answer : no, if that is the case now we will need a translator that can convert this code to valid JS code solution: As in the above function Hello which is returning something which is neither html and not the javascript. Hence we use the `Bable` transpiler to convert the code of the one language to another. Bable is a javascript transpiler to one language to another here it is JSX to JS

`Instructor cue: here for further understanding you can take the learners to babel playground and paste the above component that will convert this code into js code`

- Adding babel

```
<!DOCTYPE html>
<html lang = "en">
<head>

     <!-- transpiler -> JSX -> into javascript -->
    <script src = "https://unpkg.com/babel-standalone@6/babel.min.js">
</script>
</head>



<body>
    <!-- the whole application lives inside-->
    <div id = "root"></div>
    <!-- we are telling that it is to be converted by babel from JSX to js
-->
    <script type = "text/babel">
    -------
    react code
    -------
    </script>
</body>

</html>
```

complete code (for instructors reference)

```
<!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>
    <!-- algorithm to efficiently manipulate your UI -->
    <script src = "https://unpkg.com/react@18/umd/react.development.js"
crossorigin></script>
    <!-- React DOM  -> uses react to update the DOM-->
    <script src = "https://unpkg.com/react-dom@18/umd/react-
dom.development.js" crossorigin></script>
     <!-- transpiler -> JSX -> into javascript -->
    <script src = "https://unpkg.com/babel-standalone@6/babel.min.js">
</script>
</head>
<body>
    <!-- the whole application lives inside-->
    <div id = "root"></div>

    <!-- single Page applcation : JS -->
    <script type = "text/babel">

        function HELLO(obj) {
            return <h1>Hello React </h1>;
        }

        ReactDOM.render(<HELLO></HELLO>, document.getElementById("root"));
    </script>
</body>

</html>
```

What is JSX

JSX: JSX, short for JavaScript XML, is like HTML on steroids. It allows you to write JavaScript logic directly inside HTML code.

Explain these three Take Aways

- **Components:** In React, components are like special functions written in JavaScript. They return HTML-like code, which is what you see on a webpage.
- **JSX**: JSX is a cool feature in React that's a lot like using template strings. You can mix JavaScript logic with HTML-like code, and it all gets turned into regular HTML.
- **Starting the Code**: When you run your React app, it all begins with ReactDOM.render(). This function puts your main component onto the actual webpage for everyone to see. In other words, it's like placing your "Hello" component onto the screen and the content of Hello component is decided by it's reruned HTML

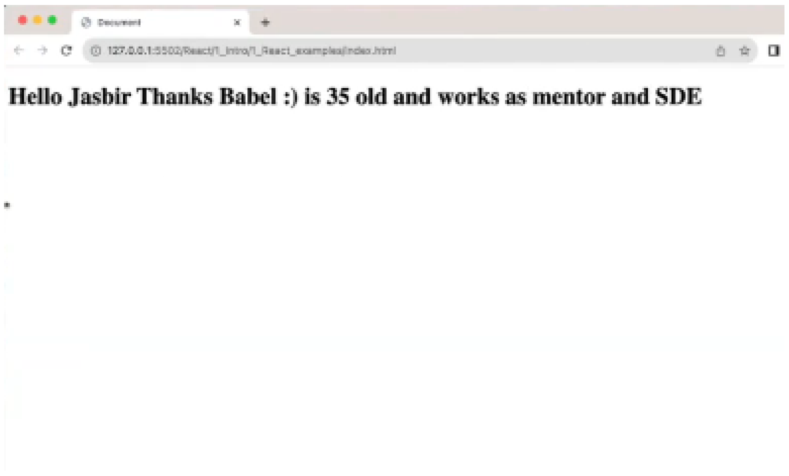
Props

In React, we use it to make websites interactive. Now, let's learn how we can make components act differently by giving them information.

Here's how it works:
We give components something called "props". Props are like properties that we give to components. We pass these props as objects, kind of like passing parameters

```
<script type = "text/babel">
        function HELLO(obj) {
            let {age,occupation} = obj;
            let name = "Jasbir";

            return <h1>Hello {name} Thanks Babel :) is {age} old and works
as {occupation} </h1>;
        }
        ReactDOM.render(<HELLO age = {35} occupation = {"mentor and SDE
"}></HELLO>, document.getElementById("root"));
</script>
```

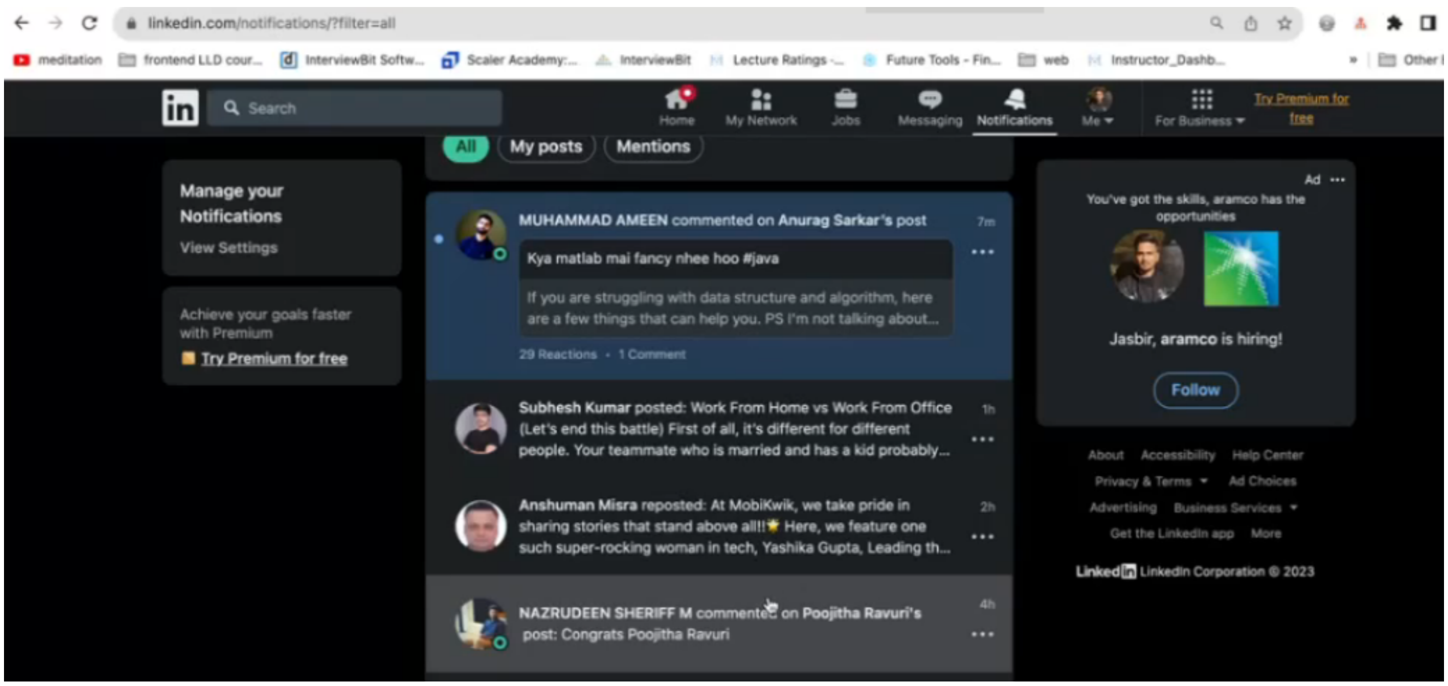

Hello Jasbir Thanks Babel :) is 35 old and works as mentor and SDE

Component Composition

In this section we will talk about concept of `component composition`.

Anology :

In the LinkedIn example below, think of creating a comment list as similar to how LinkedIn organizes information on its comments page. You can break down the comment into smaller parts, just like LinkedIn has multiple components within a comment. For example, a comment could contain components for a profile picture, options menu (like three dots), and user information


MUHAMMAD AMEEN commented on Anurag Sarkar's post 7m

Kya matlab mai fancy nhee hoo #java

If you are struggling with data structure and algorithm, here are a few things that can help you. PS I'm not talking about...

29 Reactions · 1 Comment

Subhesh Kumar posted: Work From Home vs Work From Office (Let's end this battle) First of all, it's different for different people. Your teammate who is married and has a kid probably... 1h

Anshuman Misra reposted: At MobiKwik, we take pride in sharing stories that stand above all!! Here, we feature one such super-rocking woman in tech, Yashika Gupta, Leading th... 2h

NAZRUDEEN SHERIFF M commented on Poojitha Ravuri's post: Congrats Poojitha Ravuri 4h

linkedin.com/notifications/?filter=all
Search
Home
My Network
Jobs
Messaging
Notifications
Me
For Business
Try Premium for free
All
My posts
Mentions
Manage your Notifications
View Settings
Achieve your goals faster with Premium
Try Premium for free
You've got the skills, aramco has the opportunities
Jasbir, aramco is hiring!
Follow
LinkedIn Corporation © 2023

Question to learner: Imagine you want to display the "HELLO" component five times and the "BYE" component once.

Answer:
We will use `component composition`, which is a fancy way of saying we can combine smaller components to make bigger ones.

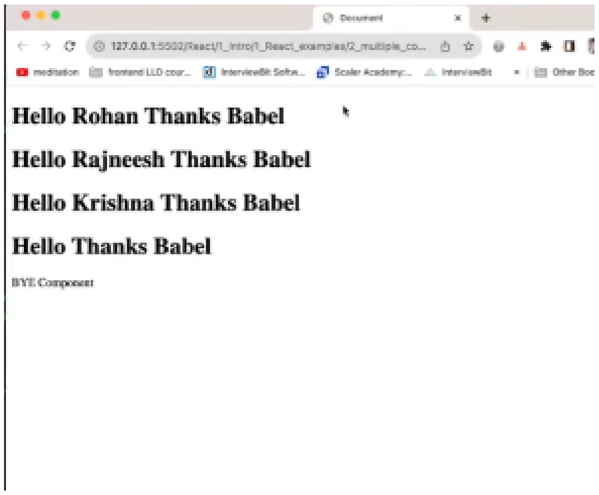
Imagine you have two parts of your app: a "HELLO" part and a "BYE" part. If you want to show "HELLO" five times and "BYE" once, you can combine them together to make a bigger component.

For example, we can create a new component called HELLO PARENT. This component will include five instances of the "HELLO" component and one instance of the "BYE" component. This idea of putting smaller components together to make a larger one is what we call component composition.

```
<body>
    <!-- the whole application lives inside-->
    <div id = "root"></div>

    <!-- single Page applcation : JS -->
    <script type = "text/babel">
        // -> component : a function that returns HTML
        function HELLO(prop) {
            let {name,age} = prop
            return <h1>Hello {name} Thanks Babel is {age} old </h1>;
        }
        function BYE(){
            return <p>BYE Component</p>
        }

        function HELLOPARENT() {
            return <div>
                <HELLO name = {"Rohan"} age = {10}></HELLO>
                <HELLO name = {"Rajneesh"} age = {30}></HELLO>
                <HELLO name = "Krishna" age = {40}></HELLO>
                <HELLO ></HELLO>
                <BYE></BYE>
                </div>
        }
        ReactDOM.render(<HELLOPARENT></HELLOPARENT>,
document.getElementById("root"));
    </script>
</body>
```



```
Hello Rohan Thanks Babel
Hello Rajneesh Thanks Babel
Hello Krishna Thanks Babel
Hello Thanks Babel
BYE Component
```