upGrad

Course: Advanced Frontend Development Using React

Lecture On: React Components

Instructor: Mrigank Kaushik

In the last class, we discussed...

- Overview of Phone Directory Application
- How to Setup the Code?
- Folder Structure and Code Cleanup
- Introduction to JSX and how is it different from HTML
- Injecting Data Using Curly Braces {} in React
- React.createElement() method
- Rendering Components in Root Node
- Rendering Elements into DOM in React

Poll 1

Name the file which contains unit tests for the application.

A. index.js
B. App.js
C. App.test.js
D. manifest.json

upGrad

#RahoAmbitious

Full-Stack Software Development

Poll 1 (Answer)

Name the file which contains unit tests for the application.

A. index.js

B. App.js

C. App.test.js

D. manifest.json

Poll 2

Which of the following can be the JavaScript expression to be evaluated inside curly braces during compilation?

A. A variable
B. A function
C. An object
D. Any code snippet that returns some value

Poll 2 (Answer)

Which of the following can be the JavaScript expression to be evaluated inside curly braces during compilation?

A. A variable

B. A function

C. An object

D. Any code snippet that returns some value

Today's Agenda

1. Components, their Types and Conditions
2. Inline Styling in React
3. External Styling in React
4. Dynamic Rendering Using map() Method

Components

Can the code which you write in React be used again?

Does React provide reusability of code?

Yes, React supports code reuse

This is done using components

Components

upGrad

- React follows a **component-based approach** where a component refers to an independent and reusable piece of code that can be plugged anytime anywhere
- It also allows you to make changes to your code dynamically
- Conceptually, components are like JavaScript functions
- They accept arbitrary inputs (called ‘props’) and return React elements describing what should appear on the screen

upGrad

So, what are components?

Components are just the JavaScript way of writing independent, reusable, and dynamic code

Types of Components

There are two types of components:

- **Functional Components (Stateless):** Written in the form of functions
- **Class Components (Stateful):** Written in the form of classes

Note: You will learn about state in the upcoming sessions

Types of Components

- A functional component is used whenever you need to pass some data to a component and the component returns a React element
- On the other hand, class components provide additional features as compared to functional components
- A line of caution is that these class components, if not handled with care, can result in your application going into an inconsistent state

Note: Later in the course, you will discover that functional components can achieve whatever class components can and much more with help of hooks (a new addition in React 16.8).

Poll 3

Which of the following options is best suitable with respect to components in React?

A. Self-contained
B. Reusable
C. Customisable
D. All of the above

Poll 3 (Answer)

Which of the following options is best suitable with respect to components in React?

A. Self-contained
B. Reusable
C. Customisable
D. All of the above

Components

upGrad

You have seen a glimpse of what components are, but before building a component in the application, you should first identify what parts of your application can be moved out as separate entities

Components

upGrad

Phone Directory Application

- In our Phone Directory application, the header is one such component that is consistent and common across all pages, just that the heading it displays is different
- So, let's start with creating the header as a separate component, which can be utilised as a reusable entity across different pages in your application
- Deciding on which components to create is a personal choice and needs to be planned wisely with not segregating the code too much
- Also, a component need not be common across all pages. You can have a component that is being repeated multiple times on a single page
- You will look at how to change this heading for each page later. For the time being, let's simply display 'Phone Directory'

upGrad

Creating the Header Component

- So, let's start with creating the *Header.js* file inside the *src* folder
- Note that the name of a component in JSX should start with a capital letter. This is because you render both HTML elements as well as components in the same code
- The HTML elements are pre-defined in the language, whereas components are custom-generated by the user. So, JSX needs some mechanism to distinguish between these HTML elements and user components
- Therefore, HTML elements must start with a lowercase letter, whereas components must start with a capital letter

upGrad

Creating Header As the Class Component

```
import React, { Component } from 'react';
class Header extends Component {
    render() {
        return (
            <div className="header">
                Phone Directory
            </div>
        )
    }
}

export default Header;
```

Code Reference

upGrad

Creating Header As the Functional Component

```
import React from 'react';
const Header = function() {
    return (
        <div className="header">
          Phone Directory
        </div>
    )
}

export default Header;
```

Note: Make sure that you remove the header from the App.js file

Code Reference

upGrad

Creating the Header Component

- In our application, the header displays the name of the app, Phone Directory, which is a static information and remains constant throughout the application
- As mentioned earlier, you should use stateless functional components, whenever the component does not play any role in maintaining application state
- So, the Header.js should be a **functional component**

Referencing a Component in Other File

Now, how to reference this component in another file?

- Add an import statement: Go to App.js and import Header.js file

```
import Header from './Header';
```

upGrad

Referencing a Component in Other File

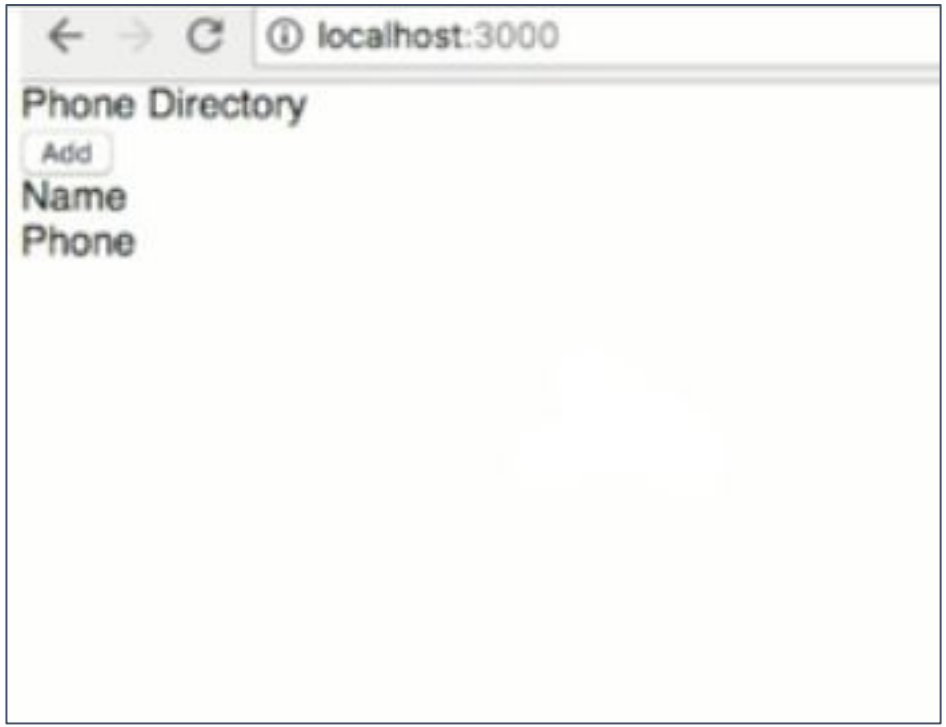
- Reference the component in JSX code

```
return (
    <div>
      <Header />
      <button>Add</button>
      <div>
        <span>Name</span><br />
        <span>Phone</span>
      </div>
    </div>
  );
```

Code Reference

upGrad

Output After Creating the Header Component



The output will look like this after building the header component

upGrad

Conditions of a Component

Some of the conditions to keep in mind while building the component are as follows:

- A component's name must start with a capital letter in order to distinguish between the predefined HTML elements and the custom elements (components) created by users

  An example can be **`MyComponent`**

- A class component must extend from a base class named Component.

  For this, you also need to include component as a named import from the 'react' library. If a component is not a named import then you could extend from **`React.Component`**

Conditions of a Component

- A class component must always have a render() function

```
import React, {Component} from 'react';
class MyComponent extends Component {
    render() {
        // code here
    }
}
```

Conditions of a Component

- A component must always return something

```
import React, {Component} from 'react';
class MyComponent extends Component {
   render() {
      return (
         // code here
      )
   }
}
```

  - This returned value is the content that is actually rendered into the DOM; it replaces the name of the component

Conditions of a Component

- In case you miss the return statement in a functional component, you will get an error saying, '*Nothing was returned from render. This usually means a return statement is missing*'
- On the other hand, if you miss a return statement inside the render() function in a class component, you get an error saying, '*Your render method should have return statement*'
- If you do not wish to return anything, you can return null from the component, as written below

```
return null;
```

- In any case, the return statement needs to be mandatorily written, no matter whether you wish to return anything or not

Conditions of a Component

- In order to reference a component written in a separate file, you need to first export the component from the file where it has been defined
  For example,
  ```
  export default Header;
  ```
- Then import the component in the required file where it needs to be used

Conditions of a Component

- A component can have a file extension as .js or .jsx
- In the case of components where the file extension is .js or .jsx, the extension is not required to be explicitly mentioned while writing the import statement

```
import Header from './Header';
```

- However, for all other files, the extension should also be mentioned along with the file name while writing the import statement

```
import logo from './logo.svg';
```

Poll 4

When do you need to convert a functional component into a class component?

A. When you need to pass props to the component

B. When you need to maintain state inside the component

Poll 4 (Answer)

When do you need to convert a functional component into a class component?

A. When you need to pass props to the component

B. When you need to maintain state inside the component

Poll 5

upGrad

Which of the following options correctly describes what happens when you run the given code snippet?

```
class App extends Component {
    render() {
      return (
        <div className="App">
          <p>UpGrad - Building Careers of Tomorrow!</p>
          <App/>
        </div>
      );
    }
}
```

A. The code snippet runs fine and creates a paragraph element inside a div container
B. The code snippet runs but throws a warning on the console
C. The code snippet runs but throws an error on the console
D. The code snippet does not run and makes the page unresponsive

Poll 5 (Answer)

upGrad

Which of the following options correctly describes what happens when you run the given code snippet?

```
class App extends Component {
    render() {
      return (
        <div className="App">
          <p>UpGrad - Building Careers of Tomorrow!</p>
          <App/>
        </div>
      );
    }
}
```

A. The code snippet runs fine and creates a paragraph element inside a div container
B. The code snippet runs but throws a warning on the console
C. The code snippet runs but throws an error on the console
D. The code snippet does not run and makes the page unresponsive

Styling in React

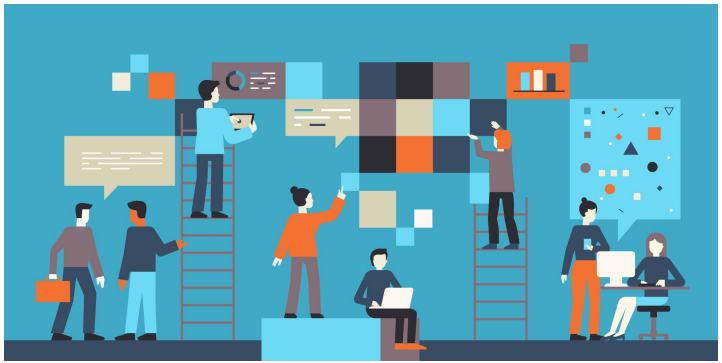
Styling in React

upGrad

Does the application not look bland to you?

Styling makes a website look better. So, let's look at styling in React

It is used to give a better appearance to the DOM elements. This is analogous to how painting the walls of your house makes them look better



Styling in React

upGrad

This is how Google's home page would look like without styling

Ways of Styling

React offers styling in two ways: **inline** and **external**.

Let’s use an example that shows two approaches of using inline styling to style the Header component inside the Phone Directory app to understand inline styling deeply

1. Directly writing style alongside JSX
2. Moving style as constant object

upGrad

Inline Styling: Approach 1: Directly Writing Style Alongside JSX

In the *Header.js* file, the inline styling can be added directly alongside JSX as shown below

```
<div style={{textAlign: 'center', padding: 20, background: '#000',
color: '#fff', textTransform: 'uppercase'}}>
    Phone Directory
</div>
```

Code Reference

upGrad

Inline Styling: Approach 2: Moving Style As Constant Object

```
const headerStyle = {
    textAlign: 'center',
    padding: 20,
    background: '#000',
    color: '#fff',
    textTransform: 'uppercase'
    };

<div style={headerStyle}>
    Phone Directory
</div>
```

To move *style* as a constant object, follow these steps:

1. Move the JavaScript object out of the style property of the element
2. Assign this object to a variable
3. Assign this variable back to the style property of the element

Note: This approach makes the code more readable when you have a lot of styles to be applied to an element or component

Code Reference

Characteristics of Inline Styling

- It allows you to write styles in the same line with the element or component. This is why it is called ‘inline’ styling
- The *style* property is used with the element or component to be rendered into the DOM. It accepts a JavaScript object enclosed in curly braces
- Two curly braces are used with the style property; one to evaluate the expression inside the JSX code and the other to represent a JavaScript object, which is taken as input by the *style* property

```
<div style={obj}>
   Phone Directory
</div>
```

Characteristics of Inline Styling

- The property names look like CSS property names, but they are not exactly like them. These names are actually JavaScript identifiers; they can be considered as the keys (or properties) of a JavaScript object
- The property names must be written in camelCase. Unlike CSS, hyphens are not allowed in JSX because the JSX code gets converted to JavaScript code, and hyphens are not allowed in JavaScript identifiers

```
<div style={{textTransform: 'uppercase'}}>
   Phone Directory
</div>
```

This is the reason why textTransform is written in camelCase in JSX unlike text-transform in CSS. In case you fail to follow this, you will get an error saying '*Uncaught SyntaxError: Inline Babel script: Unexpected token*'

Characteristics of Inline Styling

- The property values look like CSS property values, but they are not exactly like them. These values can be considered the values corresponding to the keys (or properties) in a JavaScript object
- Since all the values in JavaScript must be of a valid datatype, care must be taken regarding each value correctly mapping to a valid datatype in JavaScript

```
<div style={{background: '#000'}}>
   Phone Directory
</div>
```

This is the reason why '#000' is written inside quotes, because it corresponds to a string value. In CSS, you must write it without quotes in order to make it work

upGrad

Characteristics of Inline Styling

- All property-value pairs are separated using the *comma* operator. The reason is that the *style* property accepts a JavaScript object where a comma should be used in contrast to a CSS style, where a semicolon is used instead
- When a component or element is styled by moving out style as a constant object, only one pair of curly braces is used in the style property

External Styling

- Inline styling is not usually preferred because making a CSS class is generally better for performance
- So, let’s see how to style the header using an external stylesheet

Characteristics of External Styling

- Write all the styles in an external stylesheet
- This is similar to writing external CSS
- Give your element or component a selector, id or class. Skip this in case you want to use a tag selector
- Create a .css file and write your CSS code in it based on the element's selector
- Import this stylesheet into the file where the component or element is defined on which you want to apply the given style

Characteristics of External Styling

```
.header {
    text-align: center;
    padding: 20px;
    background: #000;
    color: #fff;
    text-transform: uppercase;
}
```

In the Phone Directory application, create a *Header.css* file and write your CSS code in this file for the header class

Characteristics of External Styling

```
import React from 'react';
import './Header.css';
const Header = function() {
    return (
        <div className="header">
          Phone Directory
        </div>
    )
}

export default Header;
```

Import the *Header.css* stylesheet in the *Header.js* file where the Header component is defined

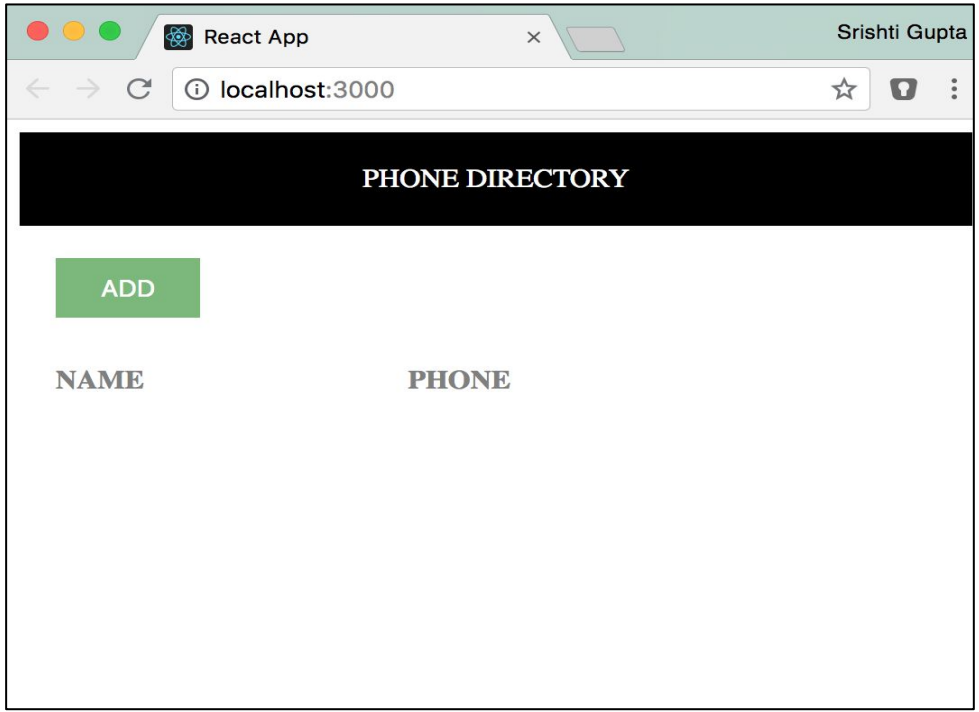
Note that since the extension of a stylesheet is .css (not equivalent to .js or .jsx), you need to specify the file extension while writing the import statement for a stylesheet

Code Reference

Hands-On Exercise

upGrad

Adding External Styling to the Home Page



Let's use the external stylesheet method to style the home page, which is *App.js* file. The final web page should look like this image

You just need to create a new *App.css* file and write all your CSS code there

Code Reference

Poll 6

How do you write an inline style directly alongside JSX to specify the text align as ‘center’ and color as ‘blue’?

A. `style={{textAlign: 'center', color: 'blue'}}`

B. `style={text-align: 'center', color: 'blue'}`

C. `style={{text-align: center, color: blue}}`

D. `style={{textAlign: center, color: blue}}`

Poll 6 (Answer)

How do you write an inline style directly alongside JSX to specify the text align as ‘center’ and color as ‘blue’?

A. `style={{textAlign: 'center', color: 'blue'}}`

B. `style={text-align: 'center', color: 'blue'}`

C. `style={{text-align: center, color: blue}}`

D. `style={{textAlign: center, color: blue}}`

Rendering Content Dynamically

Rendering Content Dynamically

upGrad

Now, it is time to see how you can dynamically render content inside components while using JavaScript's *map()* method

Characteristics of map() Method

- The entire map() method is written inside curly braces, since it is JavaScript code that needs to be evaluated
- map() is a JavaScript function and returns an array after applying the given function to each element of the array

```
{
    arrayNameToIterateOver.map(
        //code here
    )
}
```

Characteristics of map() Method

- In React, you need to give a unique key to each element being rendered into the DOM. In case you fail to do this, you will get an warning saying, ‘Each child in an array or iterator should have unique ‘key’ prop’
- To overcome this, you need to first assign each array element with a unique value for a property

Rendering Content Dynamically

upGrad

map() method

- The *map()* method in JavaScript creates a new array after calling the given function on each array element in order (Note that it does not change the original array)
- JavaScript's *map()* method can be used to iterate over an array and inject data into the React components or elements dynamically
- You do not need to hard-code the data inside each component
- This is one of the major reasons why React refers to its components as 'reusable' entities
- Also, this is yet another application where curly braces are used in order to write some JavaScript code alongside JSX

upGrad

Characteristics of map() Method

Let's say the property is *id*, and the unique values are 101 and 102, corresponding to the individual array elements

```
let demoArray = [
    {
        id: 101, // unique
        prop1: "SomeValForProp1",
        prop2: "SomeValForProp2"
    },
    {
        id: 102, // unique
        prop1: "SomeOtherValForProp1",
        prop2: "SomeOtherValForProp2"
    }
];
```

Characteristics of map() Method

Secondly, you need to assign this property (*id* here) to the property *key* of the outermost element inside the *map* method

```
{
    demoArray.map(arrayElement => {
        return <div key={arrayElement.id}>
            {arrayElement.prop1}
        </div>
         })
}
```

Rendering Content Dynamically

upGrad

- The concept of unique keys helps in distinguishing between different elements that are rendered into the DOM in React
- In the Phone Directory application, use the map method over an array of subscribers to call the function, which renders the subscribers into the DOM after iterating over each subscriber in the array in the *App.js* file

Rendering Content Dynamically

upGrad

Array of Subscribers:

```
let subscribers = [
    {
      id: 1,
      name: "Shilpa Bhat",
      phone: "8888888888"
    },
    {
      id: 2,
      name: "Srishti Gupta",
      phone: "9999999999"
    }
];
```

Rendering Content Dynamically

upGrad

Using the map method over the array:

```
{
    subscribers.map(sub => {
      return <div key={sub.id} className="grid-container">
        <span className="grid-item">{sub.name}</span>
        <span className="grid-item">{sub.phone}</span>
      </div>
    })
}
```

Code Reference

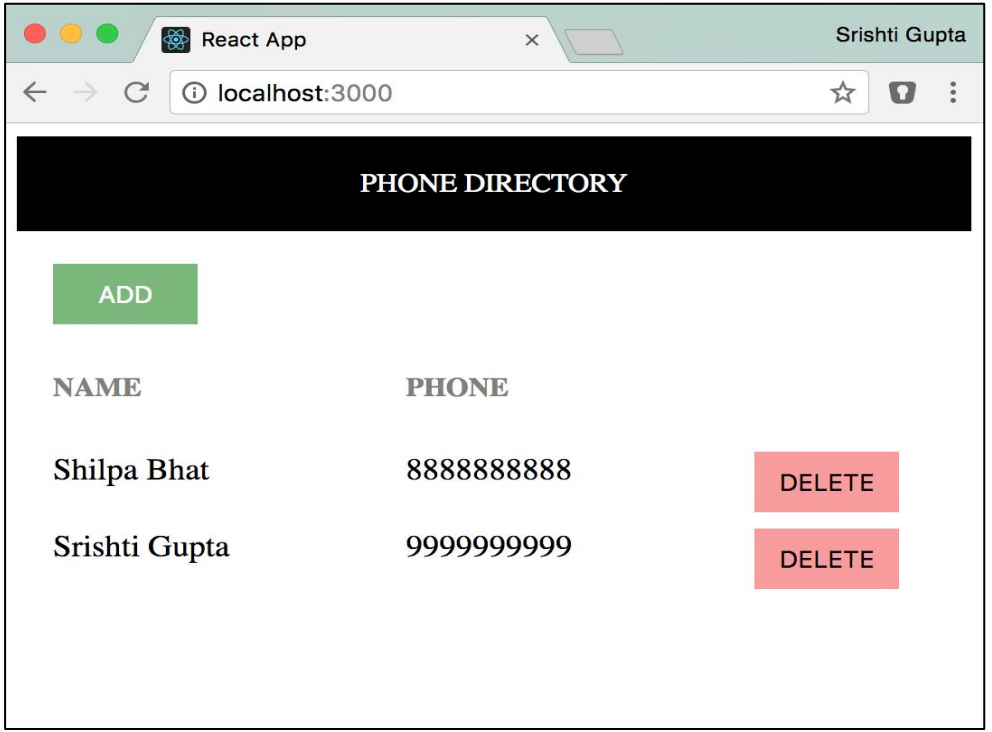
Hands-On Exercise

upGrad

Adding Delete Button With Each Subscriber on Home Page

Now, try to add a delete button that should be displayed with each subscriber details. It should look like this image

You do not need to add any functionality to it yet. You just need to render a JSX element and style it



Code Reference

Poll 7

Which of the following is true with respect to JavaScript's *map()* method?

(**Note:** More than one option may be correct.)

A. It does not change the original array.

B. It is used to iterate over an array and inject data into the React elements dynamically.

C. It is written inside curly braces.

D. It does not return anything.

Poll 7 (Answer)

Which of the following is true with respect to JavaScript's *map()* method?

(**Note:** More than one option may be correct.)

A. It does not change the original array.

B. It is used to iterate over an array and inject data into the React elements dynamically.

C. It is written inside curly braces.

D. It does not return anything.

Poll 8

What will happen if the following **render** method is executed?

```
render(){
  let gems = ["Ruby","Pearl","Sapphire"]
    return (<div>
      {gems.map(it => <p>{it}</p>)}
    </div>)
}
```

A. Displays nothing

B. Displays the names of gems in the array

C. Throws an error because of writing JavaScript code in JSX

D. Repetitive output appears on the screen

Poll 8 (Answer)

What will happen if the following **render** method is executed?

```
render(){
  let gems = ["Ruby","Pearl","Sapphire"]
    return (<div>
      {gems.map(it => <p>{it}</p>)}
    </div>)
}
```

A. Displays nothing

B. Displays the names of gems in the array

C. Throws an error because of writing JavaScript code in JSX

D. Repetitive output appears on the screen

All the code used in today's session can be found in the link provided below (branch session3-demo):

https://github.com/upgrad-edu/react-class-components/tree/session3-demo

Doubt Clearance (5 minutes)

The following tasks are to be completed after today's session:

MCQs
Coding Questions
Course Project (Part A) - Checkpoint 2

Key Takeaways

- Components are just the JavaScript way of writing independent, reusable, and dynamic code
- There are two types of components in React: functional components, which are written in the form of functions, and class components, which are written in the form of classes
- React offers styling in two ways: external and inline
- For external styling, you need to create an external stylesheet and define all the CSS styles inside this stylesheet. Then, you need to import this stylesheet inside the file where that component or element is defined on which this style is to be applied

Key Takeaways

- For internal styling, a style property with two curly braces is used; one to evaluate the expression inside the JSX code and the other to represent a JavaScript object, which is taken as input by the style property
    - The property names must be written in camelCase
    - All the property values must map to a valid data type and should be written within single/double quotes to make it a string value
    - All property-value pairs are separated using the comma operator
- JavaScript's map() method can be used to iterate over an array and inject data into the React components or elements dynamically
- You do not need to hard-code the data inside each component. This is one of the major reasons why React refers to its components as 'reusable' entities

upGrad

In the next class, we will discuss...

1. Props and how to use them inside a functional as well as a class component
2. Events and event handling
3. Developing functionality for adding a subscriber

upGrad

#RahoAmbitious

Thank You!