Scaler Class Notes: Introduction to React

Introduction to React

React is a powerful JavaScript library used extensively for building user interfaces, specifically for web applications. Developed by Facebook, React is utilized by large companies like Netflix, Uber, and more due to its ability to build smooth, fast, and intuitive user interfaces .

Tools and Setup

- **Editor:** It is recommended to use VS Code.
- **Browser:** Chrome Browser is recommended.
- **Prerequisites:** A solid understanding of HTML, CSS, and JavaScript is essential before diving into React .

Building a Simple Hello World Program

Using Pure HTML and JavaScript

To begin understanding how React simplifies tasks, we start by writing a simple "Hello from JS" program using pure HTML and JavaScript. Here's how it works:

1. **Create an HTML element:** A `<h1>` element is created using `document.createElement` .
2. **Set Element Content:** The text "Hello from JS" is set as the content of this element.

4. **Append Element to DOM:** The `<h1>` is appended to the root element and displayed in the browser .

Using React

Loading React Using CDN Links

React and ReactDOM can be added to an HTML document using CDN links. These libraries allow us to create React elements and render them to the DOM .

```
<script src="https://unpkg.com/react@17/umd/react.developmen
<script src="https://unpkg.com/react-dom@17/umd/react-dom.de
```

Creating Elements with React

React simplifies element creation using `React.createElement` , which can create HTML elements in the DOM:

1. Create a React element, like an `<h1>` with "Hello from React" as content.
2. Use `ReactDOM.render` to mount the React element onto the DOM .

Here’s an example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initi
    <title>React App</title>
    <script src="https://unpkg.com/react@17/umd/react.develo
```

```
<body>
    <div id="root"></div>
    <script>
        const rootElement = document.getElementById('root');
        const headingElement = React.createElement('h1', nul
        ReactDOM.render(headingElement, rootElement);
    </script>
</body>
</html>
```

Understanding JSX

JSX is a syntax extension for JavaScript that looks similar to XML or HTML. It allows you to write HTML-like code directly in JavaScript, which is then transformed into JavaScript code by a transpiler such as Babel .

Rendering JSX

To use JSX in your code, Babel is required to transpile the JSX syntax into standard JavaScript. Here is how JSX is utilized:

1. **Write JSX within JavaScript:** Use JSX to create UI components with a familiar HTML-like syntax.
2. **Babel Transpilation:** Babel converts JSX into `React.createElement()` calls, making it compatible with React .

Example with JSX:

```
function App() {
    return <h1>Hello from JSX</h1>;
}

const rootElement = document.getElementById('root');
ReactDOM.render(<App />, rootElement);
```

- **Components:** Components are reusable code structures that encapsulate UI logic and rendering.
- **Props:** Short for properties, props are used to pass data from parent to child components, similar to parameters in functions .

Example of using Props:

```
function Greeting(props) {
    return <h1>Hello, {props.name}</h1>;
}

ReactDOM.render(<Greeting name="World" />, document.getEleme
```

Separation between React and ReactDOM

- **React:** Handles the creation and management of components.
- **ReactDOM:** Manages rendering to the DOM and provides the necessary functionalities for the web environment .

React's architecture allows it to adapt not just to web environments via ReactDOM but also to other platforms like mobile through libraries such as React Native .

This comprehensive exploration of React basics provides a structured understanding of React's core concepts, aiding in its adoption and application in web development.