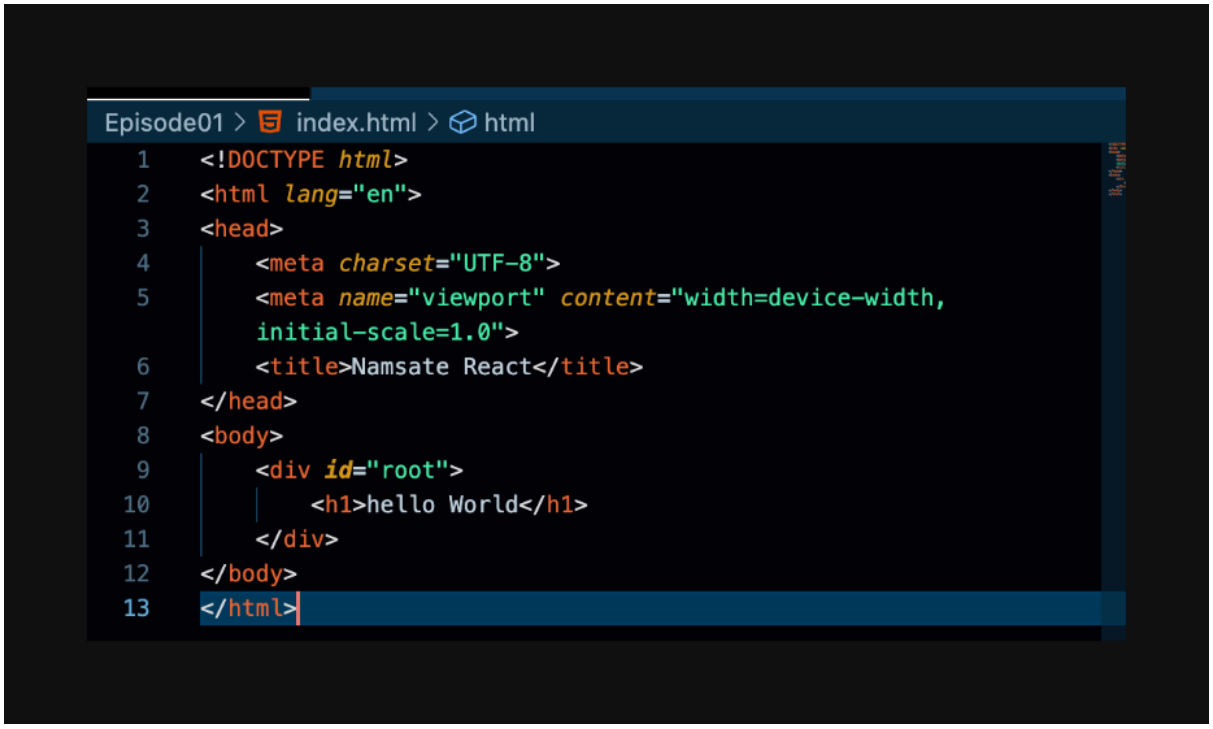
React.js Basics – Inception

Basic HTML Hello World



Basic JavaScript Hello World

Introduction to React.js

React.js is a JavaScript library developed by Facebook. It helps in building interactive UI components efficiently using a component-based architecture. React enables developers to build scalable and maintainable front-end applications.

Why Use React.js?

- **Efficient UI updates**: React uses a virtual DOM to optimize UI rendering.
- **Component-based architecture**: Code is modular and reusable.
- **Fast performance**: React efficiently updates and renders only the necessary components.
- **Large ecosystem**: A rich ecosystem of libraries and tools enhances development.
- **Unidirectional data flow**: Makes debugging easier.

Understanding React Elements

React elements are the smallest building blocks in React applications. They are used to create UI components programmatically.

Creating an `<h1>` Element

```
const heading = React.createElement(
  "h1",
  { id: "heading" },
  "Hello World from React!"
);
```

Here, `React.createElement` is used to create an `<h1>` element with an ID `heading` and text content `Hello World from React!`.

Adding Attributes to Tags

Attributes like `id`, `className`, and event handlers can be added as an object in `React.createElement`.

Nested Structure

A **nested structure** involves a parent-child hierarchy in React elements.

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

```
);
```

- The `parent` div contains a `child` div.
- The `child` div contains an `h1` tag.

Sibling Structure

A **sibling structure** consists of multiple child elements inside a parent.

```
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"),
  ])
);
```

- Here, `h1` and `h2` are siblings inside the `child` div.

Complex Nested Structure

This example demonstrates multiple nested children.

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

- There are two child divs (`child` and `child2`), each containing `h1` and `h2` elements.

Rendering React Elements

React elements must be rendered inside the root container using ReactDOM.

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

- `ReactDOM.createRoot` creates a root container.
- `root.render(parent);` renders the `parent` React element inside the root container.

Extra Notes

- **Internal vs. External JavaScript**: JavaScript can be written inline within HTML or externally as `.js` files.
- **React consists of two files**:
  - **`react`** (Core library for UI creation)
  - **`react-dom`** (Handles rendering in browsers)
- **Tailwind CSS**: Helps in writing CSS faster; order of styles in Tailwind does not matter.

Important Concepts

- **What is React doing?** It updates only the necessary parts of the UI without reloading the entire page.
- **First element replacement**: The first React element replaces the parent with a new element from the virtual DOM.

Homework (Detailed Explanation)

1. **What is CDN?**
   - A **Content Delivery Network (CDN)** is a system of distributed servers that deliver content to users based on their geographic location.
2. **Why use CDN?**
   - Faster content delivery.
   - Reduces server load.
   - Improves website performance.
3. **What is Cross-Origin?**
   - **Cross-Origin Resource Sharing (CORS)** allows web applications to request resources from different origins (domains).
   - Example: Fetching data from an API hosted on a different domain.
4. **How can I add a script in an external file?**
   - You can create a separate JavaScript file (e.g., `script.js`) and link it in HTML:

```
<script src="script.js"></script>
```

6. **How can I add a script in head tags?**
   - Add the script tag inside the `<head>`:

```
<head>
    <script src="script.js"></script>
</head>
```

10. **What are React production and development versions?**
    - `react.production.min.js`: Optimized for performance, removes debugging tools.
    - `react.development.js`: Used during development with additional debugging features.
11. **What is Virtual DOM?**
    - A lightweight copy of the real DOM used by React for efficient updates.
12. **How does React render elements?**
    - React **compares** the Virtual DOM with the real DOM and updates only the changed elements.
13. **Key Concept**

- When rendering lists in React, each element should have a unique `key` to help React optimize rendering.

14. `React.createElement("li", { key: 1 }, "Item 1")`

Conclusion

This guide covers the fundamentals of React.js, focusing on element creation, rendering, and core concepts. React's power lies in its efficient UI updates, component-based approach, and virtual DOM handling.

Next Episode: Understanding JSX (JavaScript XML) and how it simplifies React code!