REACT.JS TIPS

An intro to
React Basics

@thecodecrumbs
www.codecrumbs.com

JSX

With JSX, we can create HTML elements directly in JavaScript and integrate them into the DOM, eliminating the need for **createElement()** and **appendChild()** methods.

JSX has the capability to change HTML tags into React elements.

```
const element = <h1>Hello World</h1>

ReactDOM.createRoot(
  document.getElementById('root')
).render(element)
```

React Components

Components in React are like self-contained, reusable pieces of code. There are two kinds of components, known as Class and Function components.

These components are like individual building blocks that can be used to construct an entire webpage or app.

```
function Component() {
  return <h2>Hello World</h2>
}

ReactDOM.createRoot(
  document.getElementById('root')
).render(<Component />)
```

React Props

Props in React are like details given to self-contained pieces of code, similar to how you'd set characteristics in website elements.

They're used in the same way you'd specify an attribute when creating a website element.

```
function Component(props) {
  return <h1>Hello {props.name}</h1>
}

ReactDOM.createRoot(
  document.getElementById('root')
).render(<Component name="World" />)
```

Handling Events

In React, actions can be performed based on user interactions like clicking or moving the mouse. These interactions are referred to as events.

These events are written slightly differently in React: they use a mix of lowercase and uppercase (camelCase), like **onClick** instead of **onclick**.

```
function Component() { return (
  <button onClick={() => alert("Codecrumbs")}>
    Instagram Name
  </button>
)}

ReactDOM.createRoot(
  document.getElementById('root')
).render(<Component />)
```

'if' Statement

You can choose to show different parts of your app based on certain conditions. One way to do this is using an 'if'.

```
function Component(props) {
  let message;
  if (props.isLoggedIn) {
    message = 'Welcome back!';
  } else {
    message = 'Please sign in.';
  }
  return <h1>{message}</h1>;
}

ReactDOM.createRoot(
  document.getElementById('root')
).render(<Component />);
```

Ternary Operator

You can also use a ternary operator for conditional rendering of elements.

```
function Component(props) { return (
  <h1>
    {props.isLoggedIn
      ? 'Welcome back!'
      : 'Please sign in.'
    }
  </h1>
)}

ReactDOM.createRoot(
  document.getElementById('root')
).render(<Component isLoggedIn={true} />);
```

Logical && Operator

You can also decide to show parts of your app based on certain conditions using something called the '&&' operator, similar to making a decision based on whether a condition is met.

```
function Component(props) { return (
    <h1>
      {props.showMessage && <p>Hello World!</p>}
    </h1>
)}

ReactDOM.createRoot(
  document.getElementById('root')
).render(<Component showMessage={true} />)
```

Lists

In React, lists are displayed using a process that repeats for each item, often using JavaScript's **map()**.

Each item has a unique **'key'** to track changes, allowing only the changed item to be updated, not the whole list. Keys must be unique in their list, but can repeat in separate lists.

```
function Component() {
  return <ul> {[1, 2, 3].map(num =>
    <li key={num}>{num}</li>
  )}
  </ul>
}

ReactDOM.createRoot(
  document.getElementById('root')
).render(<Component />)
```

TURN ON POST NOTIFICATIONS •••

Do you use React?

@thecodecrumbs

www.codecrumbs.com

DROP A COMMENT

SAVE FOR LATER