REACT - COMPONENTS, STATE, PROPS

ASSIGNMENT

Components

Components are the building blocks of a React application. They are reusable and self-contained pieces of code that encapsulate the logic and user interface of a specific part of the application. Components can be divided into two types: functional components and class components.

1.Functional Components:

These are simple JavaScript functions that receive props as an argument and return JSX (JavaScript XML) to describe the component's UI. Functional components are stateless, meaning they don't have their own internal state.

Example of a functional component:

```
import React from 'react';

const MyComponent = (props) => {

  return <div>{props.text}</div>;

};

export default MyComponent;
```

2.Class Components:

Class components are ES6 classes that extend the `React.Component` class. They have their own internal state and can define lifecycle methods for more complex logic.

Example of a class component:

```
import React from 'react';

class MyComponent extends React.Component {

  constructor(props) {

    super(props);

    this.state = { count: 0 };

  }

  render() {

    return <div>{this.state.count}</div>;

  }

}

export default MyComponent;
```

State

State represents the internal data of a component. It allows components to manage and track changes over time. State is mutable and can be updated using the `setState()` method. However, it should be used carefully to ensure immutability and proper rendering.

To initialize state in a class component, you define a constructor and set the initial state using `this.state = { /* initial state */ };`.

To update state, you use the `setState()` method, which triggers a re-rendering of the component.

Props

Props (short for properties) are a way to pass data from a parent component to its child components. Props are read-only and cannot be modified within the child component. They are used to customize and configure child components based on the parent's data or behavior.

Props are passed as attributes to child components when they are used in the parent component's JSX.

Example of passing props to a child component:

```
import React from 'react';

import MyComponent from './MyComponent';

const ParentComponent = () => {

  return <MyComponent text="Hello" />;

};

export default ParentComponent;
```

In the above example, the `text` prop is passed to the `MyComponent` child component. Inside the `MyComponent`, it can be accessed as `props.text`.

Props can also be passed to functional components as function arguments.

These are the basic concepts of components, state, and props in React. They are essential for building dynamic and interactive user interfaces.