DESTRUCTURING PROPS & STATES IN REACT

Destructuring is a feature introduced in ES6 (ECMAScript 2015) that allows you to unpack values from arrays or properties from objects into distinct variables. In React, destructuring is commonly used with props and state, making the code more concise and readable.

1. DESTRUCTURING PROPS

When you pass props to a component, you can destructure them directly in the parameter of a functional component or in the render method of a class component.

Functional Component Example:

```
function Welcome({ name, age }) {
  return <h1>Hello, {name}! You are {age} years old.</
h1>;
}
```

Class Component Example:

```
class Welcome extends React.Component {
  render() {
    const { name, age } = this.props;
    return <h1>Hello, {name}! You are {age} years old.</
h1>;
  }
}
```

2. DESTRUCTURING STATE

Similarly, you can destructure the state of a component to access its values.

Example:

```
class Counter extends React.Component {
  state = {
    count: 0
  };

  render() {
    const { count } = this.state;
    return <h1>Count: {count}</h1>;
  }
}
```

BENEFITS OF DESTRUCTURING

Conciseness: Reduces the amount of code you have to write.

Readability: Makes it clear at the beginning of a function or render method which props or state variables are being used.

Avoids Repetition: No need to repeat `this.props` or `this.state` every time you access a value.

CONCLUSION

Destructuring is a powerful feature in JavaScript that can make your React code cleaner and more intuitive. By understanding and using destructuring effectively, you can improve the readability and maintainability of your React applications.