React is a popular JavaScript library for building user interfaces, particularly single page applications where a seamless and dynamic user experience is essential. Developed by Facebook, React allows developers to create large web applications that can change data, without reloading the page.

JSX (Javascript XML) → Used with React to describe what the UI should look like.

```
eg, const element = <h1> Hello, world! </h1>;
    const element = React.createElement('h1', null, 'Hello, world!');
```

React.createElement is a function that creates a virtual DOM element, which React will later render into the actual DOM.

Components → Building blocks of a React application.
They are custom HTML elements that have their own structure, style & behaviour.

1. Function Component → Simple JS functions that return JSX.

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

2. Class component → ES6 classes that extend React.Component and must have a render method that returns JSX

```
class Welcome extends React.Component {
    render(){
        return <h1>Hello, {this.props.name} </h1>;
    }
}

function App() {
    return (<div> <Welcome name = "Bob" /> </div>);
}
```

State → It is a built in object that is used to contain data or information about the component. A component's state can change over time, & the component re-renders to reflect these changes.
State is managed within the component and can be updated using the setState method in class components or the useState hook in function components.

```
class Counter extends React.Component {
    constructor (props){
        super (props);
        this.state = { count: 0};
    }
    increment = () => {
        this.setState ({ count: this.state.count + 1});
    };
    render () {
        return (
            <div>
                <p> Count: {this.state.count} </p>
                <button onClick = {this.increment}>
            </div>  Increment </button>
        );
    }
}

function Counter (){
    const [count, setCount] = React.useState (0);
    return (
        <div>
            <p> Count: {count} </p>
            <button onClick = {() => setCount (count + 1)}>
                Increment </button>
        </div>
    );
}
```

Props → Read only attributes that are passed from parent to child components.

They allow data to flow from the parent component to the child component and help in rendering the UI dynamically based on the data received.

Passing & Using Props

```
function Welcome(props){
    return <h1>Hello, {props.name}</h1>;
}
const element = <Welcome name="Bob" />;
```