All About Components in React 🚀 (Core Concept)

React **components** are the **building blocks** of a React application. They allow developers to create **reusable**, **independen**t and **modular UI element**.

1. **What is a Component?**
   A **component** is a JavaScript function or class that **returns JSX** (UI elements). Components help break down a complex UI into **smaller, reusable** parts.

   Example of React Component

```
function Greeting () {
      return <h1>Hello, Welcome to React </h1>;
}
```

♦ **Components are like JavaScript functions:**

- They take input (props)
- They return JSX (UI elements).
- They can be reused multiple time.

2. **Types of Components in React:**
   React has two types of components:

I. Functional Components (Modern)

A functional component is a JavaScript function that return JSX.

Example:

```
Imoport React from “react”;
    Function Welcome () {
        return <h1>Welcome to React! </h1>;
```

```
}
```

Export default Welcome;

- Why use functional Components?
  Simple and easy to read
  Uses Hooks for state management
  Better performance than class components

II. Class Components (Older)

A class component is a JavaScript class that extends React.Component and render () method.

Example:

```
Import React, { Component } from "react";
class Welcome extends Component {
      render () {
          return <h1>Welcome to react! </h1>;
   }
}

Export default Welcome;
```

Note: **Not recommended** because functional components with Hooks can achieve the same functionality.

3. **Component Structure**

Every React component has three main parts:

I. **Import statements** (optional) – import React or other components.
II. **Component Definition** – Function or class that returns JSX.
III. **Export Statement** – So the component can be used elsewhere.

Example JSX:

```
import React from “react”;
function MyComponent () {
   Return <h1>Hello, React! </h1>;
}
Export default MyComponent.
```

4. **How to Use a Component?**
   once a component is created, we can use it inside other components.
   Example: Using a Component inside APP.js

```
Import React from “react”;
Import Welcome from “./Welcome”;  // Importing the component

Function App() {
   Return (
      <div>
         <Welcome />  // Using the component
       </div>
 );
}

Export default App;
```

- Component names must start with an uppercase letter (e.g.: Welcome not welcome).

- Self-closing tag <Welcome /> is used instead of <Welcome></Welcome>.

5. **Props (Passing Data to Components)**

   Props (short for **properties**) allow us to pass **data from a parent component to a child component.**

   Example: **Passing Props**

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

    function App(){
        return(
            <div>
                <Greeting name = “Alice” />
                <Greeting name = “Bob” />
            </div>
    );
}

export default App;
```

   Note: Props make components reusable and dynamic!

   - {props.name} dynamically displays different names.

6. **State (Managing Component Data)**

   Unlike props, state is used to store and manage data inside a component.

   Example: Using State in Functional Component

```
import React, { useState } from "react";

function Counter() {
```

```
  const [count, setCount] = useState(0);  // useState Hook

  return (
    <div>
      <h2>Count: {count}</h2>
      <button onClick={() => setCount(count +
1)}>Increase</button>
    </div>
  );
}

export default Counter;
```

Note: State updates trigger re-renders, updating the UI dynamically.

7. **Types of Components Based on Functionality**

   React components can be categorized based on their functionality:

I. **Presentational Components (UI Components)**

- Focus only on UI.
- Receive props but don't manage state.
- Example: A Button component.

```
function Button({ label }) {
  return <button>{label}</button>;
}
```

II. **Container Components (Logic Components)**

- Manage state & logic.
- Pass data to presentational components via props.

  Example:

```
import React, { useState } from "react";
import Button from "./Button";
```

```
function App() {
  const [text, setText] = useState("Click Me");

  return (
    <div>
      <Button label={text} />
    </div>
  );
}

export default App;
```

8. **Nested Components (Composing UI)**

Components can contain other components to build complex UIs.

Example:

```
function Header() {
  return <h1>My Website</h1>;
}

function Content() {
  return <p>Welcome to my website!</p>;
}

function App() {
  return (
    <div>
      <Header />
      <Content />
    </div>
  );
```

```
}
```

9. Component Lifecycle (class components)
   Only applies to class components.
   React components have a lifecycle with specific methods: