In React, functional components are a way to define components using JavaScript functions instead of ES6 classes. They are also known as stateless components or presentational components because they don't manage state or have access to lifecycle methods like class components do. Functional components receive data and return JSX, describing what should be rendered on the UI.

Here's a basic example of a functional component in React:

```jsx
import React from 'react';

const MyFunctionalComponent = (props) => {
  return (
    <div>
      <h1>Hello, {props.name}!</h1>
      <p>{props.description}</p>
    </div>
  );
};

export default MyFunctionalComponent;
```

In the above example:

- `MyFunctionalComponent` is a functional component.
- It accepts `props` as an argument, which contains data passed down to the component.
- It returns JSX, describing the UI that should be rendered.
- It doesn't have internal state or lifecycle methods.

With the introduction of React Hooks in React 16.8, functional components can now also manage state and perform side effects using built-in functions like `useState` and `useEffect`, which allows functional components to handle more complex logic previously reserved for class components. Here's an example of a functional component using hooks:

```jsx
import React, { useState, useEffect } from 'react';

const MyFunctionalComponentWithState = () => {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  });
```

```
  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
};

export default MyFunctionalComponentWithState;
```

In this example, `useState` hook is used to add state to the functional component, and `useEffect` hook is used to perform side effects, such as updating the document title, when the component renders.

Certainly! Here are two examples of functional components in React:

Example 1: Basic Functional Component

```jsx
import React from 'react';

const Greeting = (props) => {
  return (
    <div>
      <h1>Hello, {props.name}!</h1>
      <p>{props.message}</p>
    </div>
  );
};

export default Greeting;
```

In this example, `Greeting` is a simple functional component that takes in `name` and `message` as props and renders a greeting message with the provided name and message.

Example 2: Functional Component with Hooks

```jsx
import React, { useState, useEffect } from 'react';

const Counter = () => {
```

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

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
};

export default Counter;
```

In this example, `Counter` is a functional component that uses hooks to manage state and perform side effects. It maintains a count state and updates the document title to reflect the current count every time the component renders. Users can increment the count by clicking a button.