Using `useState` in React is straightforward. Here's a step-by-step guide:

1. Import `useState` from React

First, you need to import `useState` from the React library:

```javascript
import React, { useState } from 'react';
```

2. Initialize State

Within your functional component, call `useState` to initialize a state variable and a function to update that state. `useState` takes an initial state value as an argument and returns an array with two elements: the current state and the state updater function.

3. Use the State Variable and Update Function

You can use the state variable in your component's JSX and call the updater function to change the state.

Example

Here's a detailed example demonstrating how to use `useState`:

```javascript
import React, { useState } from 'react';

function Counter() {
  // Step 2: Initialize state
  const [count, setCount] = useState(0); // count is the state variable, setCount is the function to update it

  // Step 3: Use the state variable and update function
  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}

export default Counter;
```

Detailed Explanation:

1. **Importing `useState`**:
   ```javascript
   import React, { useState } from 'react';
   ```
   This line imports `useState` from the React library.

2. **Initializing State**:
   ```javascript
   const [count, setCount] = useState(0);
   ```
   - `count` is the state variable holding the current state value (initialized to `0`).
   - `setCount` is the function used to update the `count` state.

3. **Using State in JSX**:
   ```javascript
   <p>You clicked {count} times</p>
   ```
   This line displays the current value of `count` in the component.

4. **Updating State**:
   ```javascript
   <button onClick={() => setCount(count + 1)}>
     Click me
   </button>
   ```
   The button's `onClick` handler calls `setCount` with `count + 1` to increment the count value. This triggers a re-render of the component with the updated state.

Example with Input Field

Here's another example demonstrating how to use `useState` to handle form input:

```javascript
import React, { useState } from 'react';

function NameForm() {
  const [name, setName] = useState('');

  return (
    <div>
      <input
        type="text"
        value={name}
        onChange={(e) => setName(e.target.value)}
        placeholder="Enter your name"
      />
      <p>Your name is: {name}</p>
    </div>
  );
}

export default NameForm;
```

Detailed Explanation:

1. **Initializing State**:
   ```javascript
   const [name, setName] = useState('');
   ```
   - `name` is the state variable holding the current input value (initialized to an empty string).
   - `setName` is the function used to update the `name` state.

2. **Using State in Input Field**:
   ```javascript
   <input
     type="text"
     value={name}
     onChange={(e) => setName(e.target.value)}
     placeholder="Enter your name"
   />
   ```
   - The `value` attribute of the input field is set to `name`, making it a controlled component.

   - The `onChange` handler updates the `name` state whenever the user types in the input field.

3. **Displaying State**:
   ```javascript
   <p>Your name is: {name}</p>
   ```
   This line displays the current value of `name` in the component.

By following these steps, you can effectively use `useState` to manage state in your functional React components.