**What are components?**

Components are the building blocks of any React app and a typical React app will have many of these. Simply put, a component is a JavaScript class or function that optionally accepts inputs i.e. properties(props) and returns a React element that describes how a section of the UI (User Interface) should appear

1. Functional components
2. Class Components

**Functional components**

These components are purely presentational and are simply represented by a function that optionally takes props and returns a React element to be rendered to the page.

Generally, it is preferred to use functional components whenever possible because of their predictability and conciseness. Since, they are purely presentational, their output is always the same given the same props.

You may find functional components referred to as stateless, dumb or presentational in other literature. All these names are derived from the simple nature that functional components take on.

=> **Functional** because they are basically functions

=> **Stateless** because they do not hold and/or manage state

=> **Presentational** because all they do is output UI elements

A functional component in it’s simplest form looks something like this:

const Greeting = () => <h1>Hi, I’m a dumb component!</h1>;

const Greeting = props => <h1>Hello {props.name}</h1>;

**Class Components**

These components are created using ES6’s class syntax. They have some additional features such as the ability to contain logic (for example methods that handle onClick events), local state (more on this in the next chapter) and other capabilities to be explored in later sections of the book.

As you explore other resources, you might find class components referred to as smart, container or stateful components.

=> **Class** because they are basically classes

=> **Smart** because they can contain logic

=> **Stateful** because they can hold and/or manage local state

=> **Container** because they usually hold/contain numerous other (mostly functional) components

Class components have a considerably larger amount of markup. Using them excessively and unnecessarily can negatively affect performance as well as code readability, maintainability and testability.

A class component in its simplest form:

class Greeting extends React.Component {

render(){

return <h1>Hi, I’m a smart component!</h1>;

}

}

class Greeting extends React.Component {

render(){

return <h1>Hello {this.props.name}</h1>;

}

}

**Installing the React Developer tools**

1. [Mozilla Firefox Add-ons](https://addons.mozilla.org/en-US/firefox/addon/react-devtools/)
2. [Chrome Web Store](https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi)

**State**

A state is a variable which exists inside a component, that cannot be accessed and modified outside the component and can only be used inside the component. Works very similarly to a variable that is declared inside a function that cannot be accessed outside the scope of the function in normal JavaScript. State Can be modified using this.setState. State can be asynchronous. Whenever this.setState is used to change the state class is render itself.

**State in class componets**

import React from 'react'

class MyComponent extends React.Component {

  constructor(props){

    super(props);

    this.state = { date: new Date(), name: 'Kofi'};

  }

  render(){

    return(

      <div>

            <p> Hello {this.state.name} , it is {this.state.toLocaleTimeString()

            <p>Date: {this.state.date.toLocaleDateString()}

      </div>

    )

  }

}

**State in Functional Components (Hooks)**

import React, {useState} from 'react';

function MyComponent(){

  const [date, setDate] = useState(new Date())

  const [name, setName] = useState("Kofi");

  return(

      <div>

            <p> Hello {date.name} , it is {date.toLocaleTimeString()

            <p>Date: {date.toLocaleDateString()}

            <button onClick={setDate(new Date())}></button>

      </div>

  )

}