ReactJS installation:

create-react-app reactproject at command prompt

ReactJS Components are 2 Types Those are:

1. Statefull Components.
2. Stateless Components.

ReactJS Component:

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.

function Welcome(props) {

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

}

This function is a valid React component because it accepts a single “props” (which stands for properties) object argument with data and returns a React element. We call such components “function components” because they are literally JavaScript functions.

1. We call ReactDOM.render() with the <Welcome name="Sara" /> element.
2. React calls the Welcome component with {name: 'Sara'} as the props.
3. Our Welcome component returns a <h1>Hello, Sara</h1> element as the result.
4. React DOM efficiently updates the DOM to match <h1>Hello, Sara</h1>.

**Properties**:

Props stand for Properties. They are the read-only components which work similar to the HTML attributes. Prop is a way of passing data from parent to child component. Since the props can only be passed from parent components, they cannot be changed. This makes them immutable and dumb.

## ****States:****

Generally, components take in props and render them. These are called stateless components. But they can also provide state which are used to store data or information about the component which can change over time. Such components are called stateful components. The change in state can happen as a response to user event or system event. In other words, **state** is the heart of every react component which determines how the component will behave and render. They are also responsible for making a component dynamic and interactive. Thus they must be kept as simple as possible.

**React Component Lifecycle**

React provides various methods which notify when a certain stage in the lifecycle of a component occurs. These methods are called the lifecycle methods. These lifecycle methods are not very complicated. You can think of these methods as specialized event handlers that are called at various points during a components life. You can even add your own code to these methods to perform various tasks. Talking about the lifecycle of the component, the lifecycle is divided into 4 phases. They are:

1. Initial Phase
2. Updating Phase
3. Props change Phase
4. Unmounting Phase

Each of these phases contains some lifecycle methods which are specific only to them. So let’s now find out what happens during each of these phases.

**a. Initial Phase –**The first phase of the lifecycle of a React component is the initial phase or initial rendering phase. In this phase, the component is about to start its journey and make its way to the DOM. This phase consists of the following methods which are invoked in a predefined order.

1. ***getDefaultProps():*This method is used to specify the default value of this.props. It gets called before your component is even created or any props from the parent are passed into it.**
2. **getInitialState():**This method is used to specify**the default value of this.state before your component is created.**
3. ***componentWillMount():*This is the last method that you can call before your component gets rendered into the DOM. But if you call setState() inside this method your component will not re-render.**
4. ***render():***Th**is method is responsible for returning a single root HTML node and must be defined in each and every component. You can return null or false**in case you don’t want to render anything.
5. ***componentDidMount():***Once the component is rendered and placed on the DOM, this method is called. Here you can perform any DOM querying operations.

**b. Updating Phase –**Once the component is added to the DOM, they can update and re-render only when a state change occurs. Each time the state changes, the component calls its **render()**again. Any component, that relies on the output of this component will also call its **render()** again. This is done, to ensure that our component is displaying the latest version of itself. Thus to successfully update the components state the following methods are invoked in the given order:

1. ***shouldComponentUpdate():***Using this method you can control your component’s behavior of updating itself. If you return a true from this method, the component will update. Else if this method returns a false, the component will skip the updating.
2. ***componentWillUpdate():***This method is calledjust before your component is about to update. In this method, you can’t change your component state by calling **this.setState**.
3. ***render():***If you are returning false via **shouldComponentUpdate()**, the code inside **render()**will be invoked again to ensure that your component displays itself properly.
4. ***componentDidUpdate():*** Once the component is updated and rendered, then this method is invoked. You can put any code inside this method, which you want to execute once the component is updated.

**c. Props Change Phase –**After **the component has been rendered into the DOM, the only other time the component will update, apart from the state change is when its prop value changes. Practically this phase works similar to the previous phase, but instead of the state, it deals with the props. Thus, this phase has only one additional method from the Updating Phase.**

1. ***componentWillReceiveProps():***This method returns one argument which contains the new prop value that is about to be assigned to the component.  
   *Rest of the lifecycle methods behave identically to the methods which we saw in the previous phase.*

1. ***shouldComponentUpdate()***
2. ***componentWillUpdate()***
3. ***render()***
4. ***componentDidUpdate()***

**d. The Unmounting Phase – This is the last phase of components life cycle in which the component is destroyed and removed from the DOM completely. It contains only one method:**

1. ***componentWillUnmount():***Once this method is invoked, your component is removed from the DOM permanently. In this method,you can perform any clean-up related tasks like removing event listeners, stopping timers, etc.