


ReactJs

Steps to start reactjs

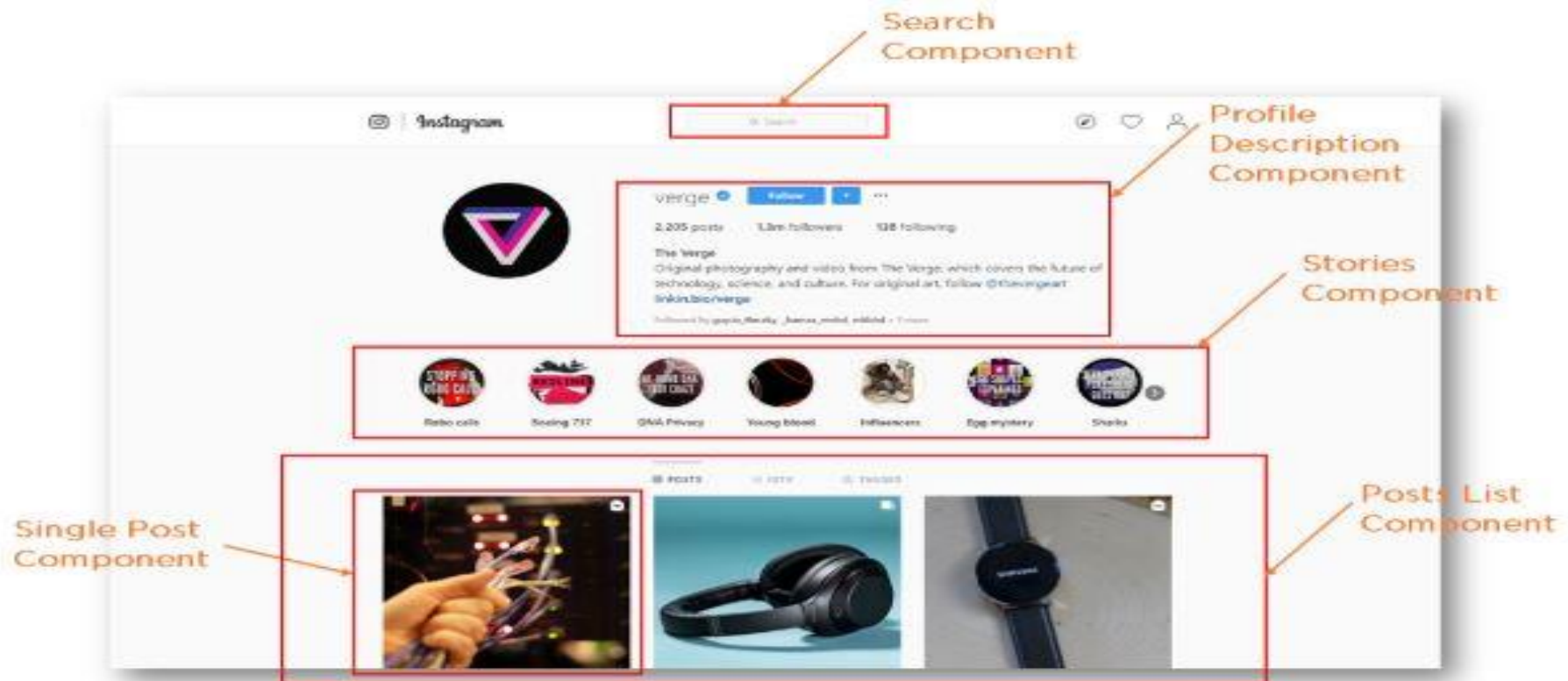
What is ReactJs

- React is a JavaScript library created for building fast and interactive user interfaces for web and mobile applications.
- It is an open-source, component-based, front-end library responsible only for the application's view layer.
- In Model View Controller (MVC) architecture, the view layer is responsible for how the app looks and feels.
- React was created by Jordan Walke, a software engineer at Facebook.
- model-view
- Fig: MVC architecture

Instagram webpage example

- Instagram webpage, entirely built using React,
- To get a better understanding of how React works.
- As the illustration shows, React divides the UI into multiple components, which makes the code easier to debug. This way, each component has its property and function.
- instagram-components
- Fig: Instagram Components
- Now that we know what React is let's move on and see why React is the most popular front-end library for web application development.

- React divides the UI into multiple components, which makes the code easier to debug. This way, each component has its property and function.



WhyReactJS

- Easy creation of dynamic applications:
- Improved performance:
- Reusable components:
- Unidirectional data flow:
- Small learning curve:
- It can be used for the development of both web and mobile apps:
- Dedicated tools for easy debugging:

Features of React

- JSX - JavaScript Syntax Extension
- JSX is a syntax extension to JavaScript. It is used with React to describe what the user interface should look like. By using JSX, we can write HTML structures in the same file that contains JavaScript code. This makes the code easier to understand and debug, as it avoids the usage of complex JavaScript DOM structures.
- example

```
const name = 'Klshori';
```

```
const message = <h1>Hello, {name}</h1>;
```

- The above code shows how JSX is implemented in React. It is neither a string nor HTML. Instead, it embeds HTML into JavaScript code.

Virtual DOM

- React keeps a lightweight representation of the “real” DOM in the memory, and that is known as the “virtual” DOM (VDOM).
- Manipulating real DOM is much slower than manipulating VDOM because nothing gets drawn on the screen. When the state of an object changes, VDOM changes only that object in the real DOM instead of updating all of the objects.
- How do Virtual DOM and React DOM interact with each other?
 - When the state of an object changes in a React application, VDOM gets updated.
 - It then compares its previous state and then updates only those objects in the real DOM instead of updating all of the objects.
 - This makes things move fast, especially when compared to other front-end technologies that have to update each object even if only a single object changes in the web application.

- Performance

- React uses VDOM, which makes the web applications run much faster than those developed with alternate front-end frameworks. React breaks a complex user interface into individual components, allowing multiple users to work on each component simultaneously, thereby speeding up the development time.

- Extensions

- React goes beyond simple UI design and has many extensions that offer complete application architecture support. It provides server-side rendering, which entails rendering a normally client-side only web application on the server, and then sends a fully rendered page to the client.
- It also uses Flux and Redux extensively in web application development.
- React Native, is a popular framework derived from React, used to create cross-compatible mobile applications.

- One-way Data Binding
- In a React app, child components are nested within parent components. It gives better control of the whole web application, helpful to find the errors.
- Debugging
 - React applications are easy to test due to a large developer community.
 - Facebook even provides a small browser extension that makes React debugging faster and easier.
 - example,
 - adds a React tab in the developer tools option within the Chrome web browser.
 - The tab makes it easy to inspect React components directly.

concepts.

- Components
- State
- Props

Components

- Components are the building blocks of any React application,
- single app usually consists of multiple components.
- A component is essentially a portion of the user interface.
- React splits the UI into independent, reusable parts that can be processed separately.
- There are two types of components in React:
 1. Functional Component (Stateless Component)
 2. Class Component (stateful component)

Stateless Component

- These components are simply javascript functions.
- We can create a functional component in React by writing a javascript function.
- These functions may or may not receive data as parameters

```
const ContactList=()=>>
{
  return <h1>Welcome Message!</h1>;
}
```

```
export default ContactList;
```

Stateful Component

- The class components are a little more complex than the functional components.
- The functional components are not aware of the other components in your program whereas the class components can work with each other.
- We can pass data from one class component to other class components.
- We can use JavaScript ES6 classes to create class-based components in React. Below example shows a valid class-based component in React:

```
class AddContactComponent extends React.Component
{
  state={}
  render(){
    return <h1>Welcome Message!</h1>;
  }
}
export default AddContactComponent
```

When would you use a stateless component??

- When you just need to present the props
- When you don't need a state, or any internal variables
- When creating element does not need to be interactive
- When you want reusable code

When would you use a stateful component?

- When building element that accepts user input
- ..or element that is interactive on page
- When dependent on state for rendering, such as, fetching data before rendering
- When dependent on any data that cannot be passed down as props

Stateful Component Vs Stateless component

- The state is a built-in React object that is used to contain data or information about the component.
- A component's state can change over time; whenever it changes, the component re-renders.
- The change in state can happen as a response to user action or system-generated events, and these changes determine the behavior of the component and how it will render.

```
class AddContact extends  
React.Component {  
  state = {  
    id: "",  
    name:"",  
    email:"",  
    mobile:""  
  };  
  updateName() {  
    this.setState({ name: "Rajan" });  
  }  
}
```

```
render() {  
  return(  
    <div>  
      {this.state.name}  
    </div>  
  )  
}
```

Props

- Props are short for properties.
- It is a React built-in object which stores the value of a tag's attributes and works similar to the HTML attributes.
- It provides a way to pass data from one component to other components

Props Vs State

Props	State
Props are read-only.	State changes can be asynchronous.
Props are immutable.	State is mutable.
Props allow you to pass data from one component to other components as an argument.	State holds information about the components.
Props can be accessed by the child component.	State cannot be accessed by child components.
Props are used to communicate between components.	States can be used for rendering dynamic changes with the component.
Stateless component can have Props.	Stateless components cannot have State.
Props make components reusable.	State cannot make components reusable.
Props are external and controlled by whatever renders the component.	The State is internal and controlled by the React Component itself.


	Props	State
Can get initial value from parent Component?	Yes	Yes
Can be changed by parent Component?	Yes	No
Can set default values inside Component?	Yes	Yes
Can change inside Component?	No	Yes
Can set initial value for child Components?	Yes	Yes
Can change in child Components?	Yes	No

How to use map in react

```
function App() {  
  // let person={id:11,name:"Kishori",desg:"manager"};  
  let per_arr=[  
    {id:11,name:"John",desg:"designer"},  
    {id:12,name:"Kishori",desg:"manager"},  
    {id:13,name:"Jack",desg:"programmer"}  
  ];  
  return (  
    <div className="App">  
      <h1>Person Details</h1>  
      <div>  
        <table border="2">  
          <thead>  
            <tr><th>Person Id</th><th>Name</th><th>Designation</th></tr>  
          </thead>
```

```
<tbody>  
    {per_arr.map((p)=>{  
      return (<tr key={p.id}><td>{p.id}</td>  
        <td>{p.name}</td>  
        <td>{p.desg}</td>  
      </tr>);  
    })}  
  </tbody>  
</table>  
</div>  
</div>  
);  
}
```

What is key property while using map?

- Every React application displays an array list of some kind using the method map.
- React tells us that for each element of that list that we return for rendering, we must provide a unique key prop.
- But do you know why it's necessary? Why does React need this key prop? 
 - Keys help React identify which items have changed, are added, or are removed. Keys should be given to the elements inside the array to give the elements a stable identity.
- When the state of your component changes, the render function will return a new tree of React elements, different to the previous/current one.
- React needs to figure out what are the differences to efficiently update the UI with the most recent changes.
- This process of matching both element trees is called reconciliation.

React hooks

- useSate
 - In functional component if local variable value changes the component will not get rendered.
 - So to keep check on change of the value of the variable we need to consider those variables as state variables
 - So we use useState hook
 - It can be used directly inside functional component. It cannot be called outside functional component
 - Also cannot be called in functions which are inside functional component

- useState is a function of React library
- It creates a special variable and also returns a function

```
const Message= () => {
```

```
  const [messageState,setMessageSate] = useState( " "); // use state  
  assigns blank to messageState
```

```
  const listState = useState( [] );  
}
```

- The component in which we call the setMessageState function will schedule the change in messageState and it will also rerender the component in which it is called.