1. **Explain what a React Native component does. What are the main advantages of the component abstraction?**
2. **Explain what a React Native component does.**

1) Components includes such things as text, list, input areas, buttons, images, videos and so on. A user interface component is associated with two important aspects namely its presentation and its behavior.

2). Components can be composed together in a hierarchical manner to create one or more complex applications but importantly components can be developed and tested independently of the application which supports scalable development and of potentially complex applications across a large team of contributors.

3). The component is the key abstraction used for provide all the other user interface network storage and application navigation logic in react native.

1. Advantages:

1. the component abstraction does offer the opportunity and the capability to comply with good design principles.

2. The component abstraction allows custom elements to be developed specifically designed to solve a particular problem for any given app.

1. **Describe what a stateless component is and how to create one. List three example use cases for stateless components.**
2. **What a stateless component is.**

React native components stateless means constructing compilers do not hold any state. Stateless components usually just render user interface elements and delegate any state management, if required, to other components.

1. **How to create a stateless component.**

In react native a stateless component can be constructed using a function.

Eg：

1. const Title = () => (  
    <Text style={styles.welcome}>  
    Welcome to React Native!  
    </Text>  
    )

2. const Instructions = ({text}) => (  
<View>  
<Text style={styles.instructions}>  
{text}  
</Text>  
<Text style={styles.instructions}>  
{instructions}  
</Text>  
</View>  
)

3. const App = () => (  
<View style={styles.container}>  
<Title />  
<Instructions text={'To get started, edit App.js'} />  
</View>  
);

1. **Explain what is meant by state in a React Native App. Why do we need to deal with state in applications?**

1. **Explain what is meant by state in a React Native App.**  Application state deals with the reality that even simple applications need to deal with data and that that data is likely to change over time as the user interacts with the app. For example how do we deal with user input to a text box, where do we store that user input. This user input is an example of what we call mutable state.

2. **Why do we need to deal with state in applications?**

1) Now the key problem with stateful functions is how they introduce relative complexity in your app. As we will explore in future videos stateful components can support complex life cycles which if not properly managed can be a source of error and bugs in your code.

2) Another problem with using stateful components for state management is that it breaks the component abstraction in many cases and violates the single responsibility principle. This coupling of components can make the code brittle which in turn makes it hard to write, hard to debug, hard to extend or even maintain.

1. **Describe how React Native components can handle state. Explain how to delegate state handling on behalf of stateless components in React Native.**

1. **Describe how React Native components can handle state.**

**Initializing State** : We need to create a state data structure. And to do this we include a constructor method on our class component. JavaScript class constructor is a method by the constructor keyword. The constructor arguments will receive any properties passed to this component when it's instantiated in the JSX syntax.

And we must call the superclass constructor with these optional properties before we can do anything else. Then we create a state object on our instance state via this keyword.

Accessing State: The state object is available via the keyword in our class methods. For example if we add a name attribute which is a string to the component in the state constructor later we can access this named value from the 'this.state'. So accessing state is simply a matter of reading the state object using normal JSX syntax.

Eg:Class MyComponent extends React.Component{  
 constructor(){

this.state = { name : “some name ”}

..

Render(){  
 console.log(this.state.value)}

}

}

**Mutating Component State:** We must use the set state method on our state to make this mutation. Now our component inherits this set state function by virtue of extending and built in react component class. Mutating state via the set state function allows react native to check to see if any render user interface content needs to be refreshed. In its first form the said state function takes an object argument and merges the attribute values of that object arguments into the existing state. The set state function takes a second form wherein you can pass a function rather than an object as its argument.

1. **Explain how to delegate state handling on behalf of stateless components in React Native.**

Now one of the most important uses for stateful components is to act as state management containers for a stateless component. In other words the stateless components which are nested inside a stateful component. In this pattern which works for simple apps there are relatively small there are a relatively small number of stateful components which wrap the stateless components.

1. **Explain how the React Native stateful component lifecycle works. Describe three lifecycle hooks and what they can be used for.**
2. **Explain how the React Native stateful component lifecycle works.**

In react native we consider stateful class components to have a lifecycle. A component lifecycle defines and describes what events can occur to the component during its existence and by existence. We mean the time during which the component is instantiated and rendered. As previously discussed stateful four components exist to allow the app to manage changes to application state such as input from a user or responses to network requests.

1. **Describe three lifecycle hooks and what they can be used for.**

React native events and hooks fall broadly into two categories which are the mounted events and the updating events.

1) componentDidMount()(Mounting)

ComponentDidMount() is invoked immediately after a component is mounted. Initialization requires that the user output change you should go here.

If you need to load data from a remote endpoint this is a good place to instantiate that network request. Setting state in this method will trigger a re-rendering of the user interface

Eg:

componentDidMount(){  
//Called after the component is mounted

fetch(‘https://api.example.com’)

then( response=>{

// Do something here…

})

}

2) componentWillMount ()(Mounting)

ComponentWillMount() is invoked immediately before mounting occurs. It is called before the render so setting state asynchronously in this method will not trigger a re-rendering.

Eg:

componentWillMount(){  
/\*called after the constructor but before the render() method\*/

}

3)shouldComponentUpdate() (Updating)

Use shouldComponentUpdate() to let react know if the components output is not affected by the current change in state or props. The default behavior is to re-render on every state change.

Eg:

shouldComponentUpdate(nextProps,nextState){  
//Logic to decide if the component should update

return true //or false

}

**.Explain what a React Native component does. What are the main advantages of the component abstraction?**

(I) what a React Native component does:  
      We could view React Native components as user interface components. It refers to elements for displaying information to a user or capturing information from a user. Examples of user interface components includes such things as text, list, input areas, buttons, images, videos, and so on.  
(II) Main advantages of the component abstraction:  
      1.The component abstraction allows custom elements to be developed specifically designed to solve a particular problem for any given app.  
      2.Also, the component abstraction offers the opportunity and the capability to comply with good design principles such as separation of concerns.  
      3. Component abstraction allows us to encapsulate and hide implementation details of the components presentation and behavior at the individual component level if we consider the react native component to be the container of abstraction.  
      4. Component abstraction allows developer to realize overall higher order complexity and create custom components based on the primitive components. It is of vital importance for developing React-Native which are constructed as a composition of hierarchically nested components.

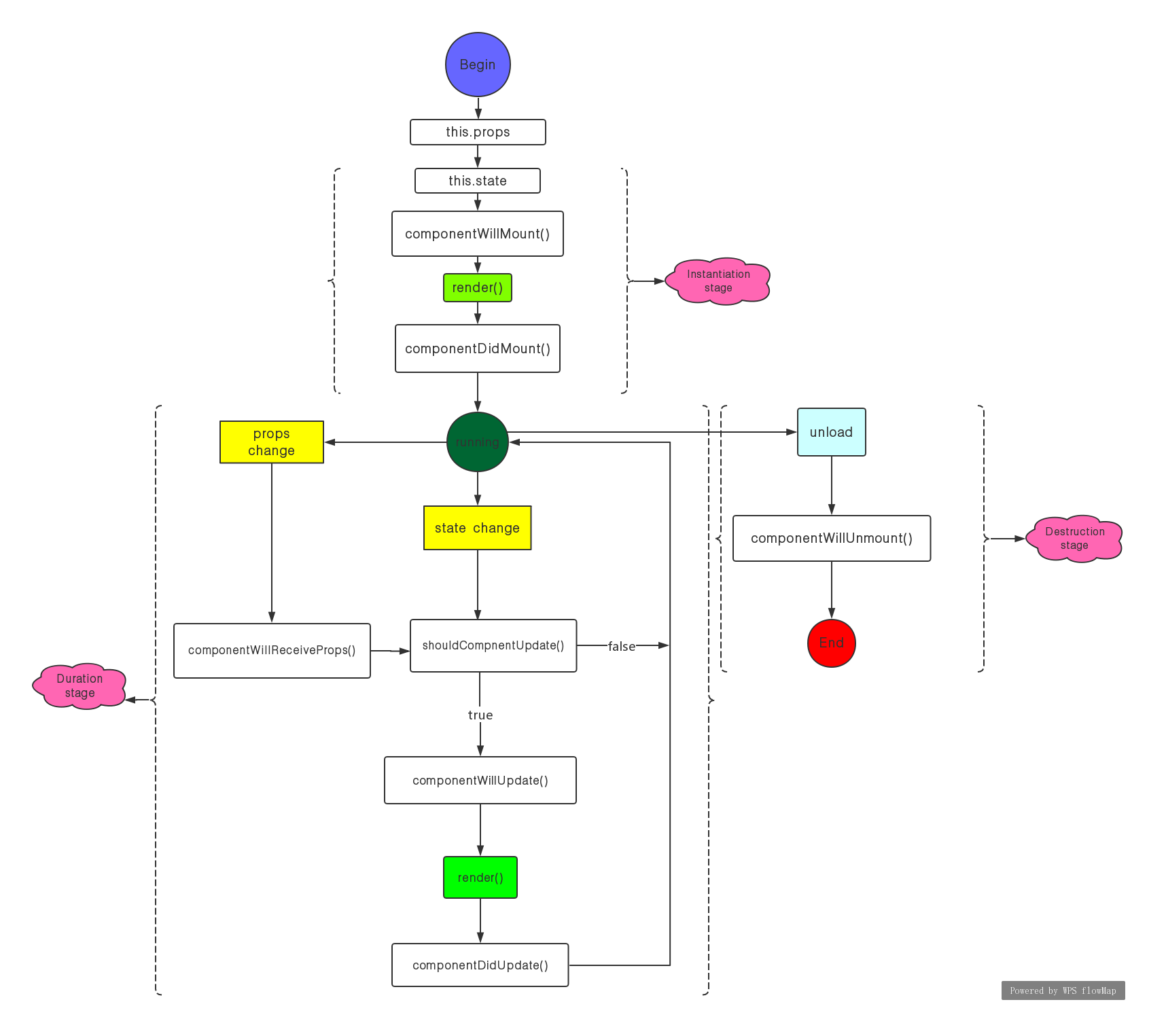
**b. Describe what a stateless component is and how to create one. List three examples use cases for stateless component.**

 (I)Describe what a stateless component is and how to create one:  
      In the context of react native components stateless means constructing compilers do not hold any state. Stateless components usually just render user interface elements and delegate any state management, if required, to other components. In react native a stateless component can be constructed using a function.  
(II)Three examples use cases for stateless component:  
     1./\* const First = name =>(  
         <text style={styles.name}>{name}  
      )\*/  
      In this example, we create a function called First, and it takes a single parameter called name  
 which is a property of the component. The component returns a built-on <Text> component as its only value with th e value of the name property expanded within.    
    2. /\* const Second = (prop1,prop2) => (...)  
        <Second prop1="..."prop2="..."/> \*/  
      In this example, we create a function named Second, and it takes two string parameters called prop1 and prop2.Properties are captured as the function arguments.  
    3. // const Third = title =>( <Text style={styles.title}>{title}</Text> )  
      In this example, we create a function named Third which returns a JSX expression as its only return value.

**c. Explain what is meant by state in a React Native App. Why do we need to deal with state in applications?**  
(I)Explain what is meant by state in a React Native App:  
     State used to mean some arbitrary mutable data which can potentially be changed at some point after being set being associated with an instance of component.  
(II)Why do we need to deal with state in applications:  
     Because of the fact that even simple applications need to deal with data and that data is likely to change over time as the user interacts with app. Such as how we deal with user input.  
 And I think that dealing with state is important for creativity as well as maintenance of App. Which can add flexibility and diversity to React Native App.

**d. Describe how React Native components can handle state. Explain how to delegate state handling on behalf of stateless components in React Native.**(I)Describe how React Native components can handle state:  
     First Step : Extend the built-in React Component class , and the component must implement a single method called render() which must return a component or a scaler quantity .The we could create  a component called class component or stateful which is declared as Javascript class using ES6 class keyword to handle state preliminarily.  
     Second Step: Create a state data structure. Include a constructor method on class component, during which we must call the superclass constructor with properties first, then create a state object on instance state via this keyword.  
     Result: We could use normal Javascript syntax to access and handle state.  
(II)Explain how to delegate state handling on behalf of stateless components in React Native:  
     Since stateless component cannot handle state directly, the stateless components are nested inside a stateful component which is responsible for managing any state of its nested children. In this pattern, there are a relatively small number of stateful componnets which wrap the stateless components.

**e. Explain how the React Native stateful component lifestyle works. Describe three lifecycle hooks and what they can be used for.**  
(I)Explain how the React Native stateful component lifestyle works:  
    The lifestyle of React Native can be divided into three stages:  
    1. Instantiation stage:  
      In this stage, functions of constructor()、componentWillMount()、render() and componentDidMount() will be called.  
    2. Duration stage:  
      In this stage, functions of componentWillReciveProps()、shouldComponentUpdate()、componentWillUpdate()、render() and componentDidUpdate() will be called under certain circumstances.  
    3. Destruction stage:  
      In this stage, functions of componentWillUnmount() will be called.

Picture for translation:

(II)Describe three lifecycle hooks and what they can be used for:  
     1.render()  
       Of all of those the render() method is the only one that we must provide an implementation for.The render method must return either react component or a scalar quantity such as a string or a number or a boolean value or a null value to indicate that nothing should be rendered.The render() method will not be called by the react native framework if the should component update method returns false.  
     /\*render(){  
        return (  
          <View>  
            <Text>...</Text>  
          </View>  
        )  
      }\*/  
     2.constructor()  
       The constructor for our react component is called before it is mounted. When implementing the constructor for our react component subclass we should call the super(props) argument before any other statement.Otherwise this.props will be undefined in the constructor and this can lead to bugs later on. The constructor is the right place to initialize state which is optional in some situations. It's okay to initialize state based on props.This effectively"forks"the props and sets the state with the inital props.  
       /\*constructor(props){  
              super(props);  
              this.state = {color:props.initialColor};  
      }\*/  
      3.componentWillReciveProps()  
         React donnot call componentWillReciveProps() with initial props during mounting.It only calls this method if some of component's props may update.Calling this.setState generally doesn't trigger componentWillReciveProps().  
          /\*componentWillReciveProps(nextProps){  
                  if(this.props.name != nextProps.name){  
                       this.setState({  
                             name: "..."  
                                         })  
                 }  
          }