React Components Lifecycle

What is Component Lifecycle in React?

React components have three main stages in their lifecycle :

- Mounting or initialization.
- Updation in state or props.
- Unmounting.

When these stages occur a number of function are called. We can override these functions and perform some specific tasks before and after these stages. Let's take a look at all the function which are being called.

Initialization

Function called in order are :

- getDefaultProps
- getInitialState
- componentWillMount
- render
- componentDidMount

State changes

Functions called in order are :

- shouldComponentUpdate
- componentWillUpdate
- Render
- componentDidUpdate

Props Changes

Functions called in order are :

- componentWillReceiveProps
- shouldComponentUpdate
- componentWillUpdate
- Render
- componentDidUpdate

Unmounting

At the time of unmounting only one method is called :

- componentWillUnmount

getDefaultProps(es5)

```
//es5

var Greeting = React.createClass({
  propTypes: {
    name: React.PropTypes.string
  },

  getDefaultProps: function() {
    return {
      name: 'Mary'
    };
  },

  // ...

});
```

defaultProps(es6)

```
//es6

class Greeting extends React.Component {
  render() {

  }
}

Greeting.defaultProps = {
  name: 'Mary'
};

Greeting.propTypes = {
   name: React.PropTypes.string
},
```

defaultProps(es6)

```
//es6

class Greeting extends React.Component {
static defaultProps = {
  name: 'Mary'
};

static propTypes = {
   name: React.PropTypes.string
};

 render() {

  }
}
```

getInitialState(es5)

```
var SayHello = React.createClass({
  getInitialState: function() {
    return {message: 'Hello!'};
  },

  render: function() {
   //..
  }
});
```

getInitialState(es6)

```
class SayHello extends React.Component({
  constructor(props) {
    super(props);
    this.state = {
      message: 'Hello!',
    };
  }
  render: function() {
   //..
  }
});
```

componentWillMount

- Called just before the render() is called.
- We have the initial state and default props by this state.
- Is okay to call this.setState() here.

```
componentWillMount() {
  console.log('Component WILL MOUNT!')
}
```

render

- Available multiple lifecycle stages.
- Never call setState or try to access dom nodes inside render.
- Render returns a single react dom node which can contain multiple dom nodes.

```
render() {
    return <div> rendering component <div/> ;

}
```

componentDidMount

- Called after the render method.
- Dom is available to make some changes.

```
componentDidMount() {
    ReactDOM.findDOMNode(this.refs.myInput).focus();
}
```

componentWillReceiveProps

- Gets called when new props are passed to the component.
-

```
componentWillReceiveProps(newProps) {
  this.setState({ name: newprops.name })
}
```

shouldComponentUpdate

- Gets called when props or state changes
- Will stop updation of component if returns false.

```
shouldComponentUpdate(newProps, newState) {
  return true

}
```

componentWillUpdate

- Same as componentWillMount, just gets called at time state or props are updated.

```
componentWillUpdate(nextProps, nextState) {
   console.log('Component WILL UPDATE!');
}
```

componentDidUpdate

- Same as componentDidMount, just gets called at time state or props are updated.

```
componentDidUpdate(prevProps, prevState) {
  console.log('Component DID UPDATE!')
}
```

componentWillUnmount

- Gets called when component unmounts.
- Is best time when you want to clear any type of listener in your component.

```
componentWillUnmount() {
  console.log('Component WILL UNMOUNT!')
}
```