React Component life cycle

Lesson 06

Capgemini

Lesson Objectives

At the end of this module on React fundamentals you will be able to:

- Explain and demonstrate

- Updating life cycle hooks
- PureComponents
- React’s DOM Updating Strategy
- Returning adjacent elements
- Fragments

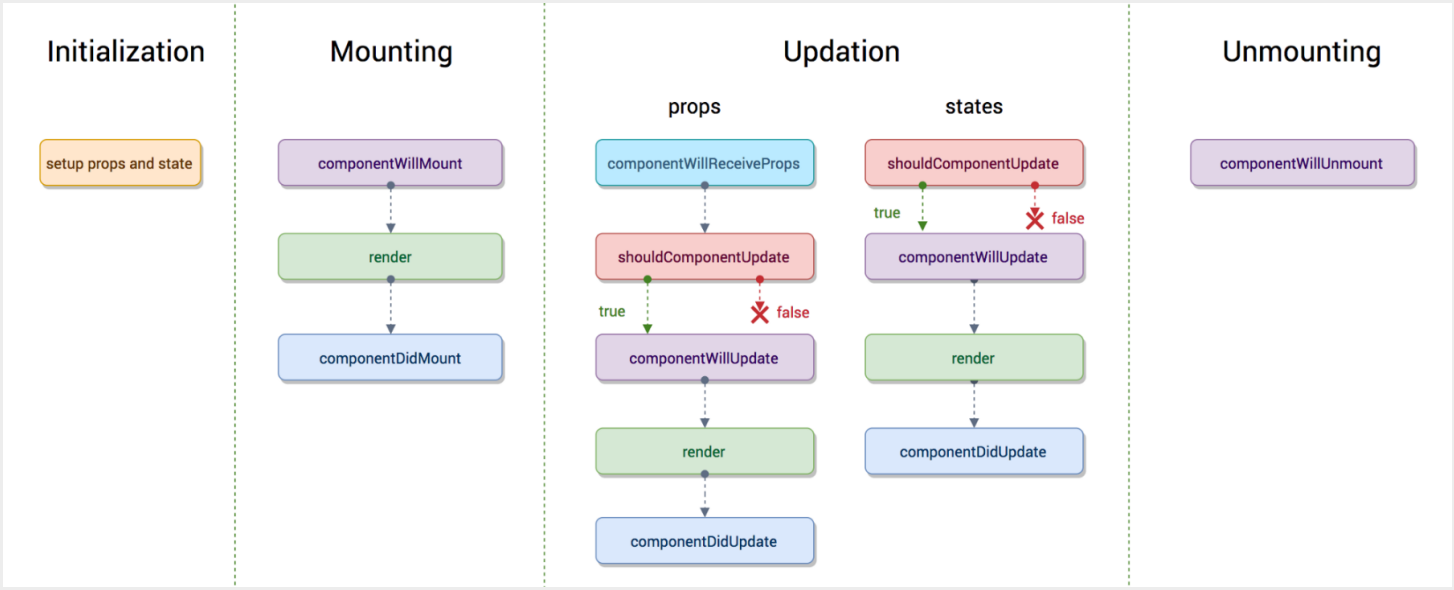
React Component - Life Cycle Phases

Changes undergone by a component from its creation till its termination.

Lifecycle methods are various methods which are invoked at different phases of the lifecycle of a component.

Four phases of a React component

Initialization
Mounting
Update
Unmounting



1. Initialization

React component is constructed by setting up the initial states and default props, if any.

State can be changed later by using setState method).

```
class Mp3Player extends
React.Component
constructor(props) {
  super(props);
  this.state = {
    volume: 70,
    status: 'stopped'
  }
}
Mp3Player.defaultProps =
{
  theme: 'dark'
};
```

2. Mounting

Mounting is the process that occurs when a component is being inserted into the DOM.

There are 2 methods in this phase

1. ***componentWillMount()***
2. ***componentDidMount()***

The ***componentWillMount()*** method is the first called in this phase. It's **invoked once** and **immediately before the initial rendering occurs**, hence before React inserts the component into the DOM.

componentDidMount() method is invoked right after the component is mounted on the DOM.

From this method we can access DOM

3. Update

This Phase starts after the component got displayed.

The component can be updated by two ways, sending new props or updating the state.

Methods when the current state is updated (setState)

1. **shouldComponentUpdate**
2. **componentWillUpdate**
3. **Render**
4. **componentDidUpdate**

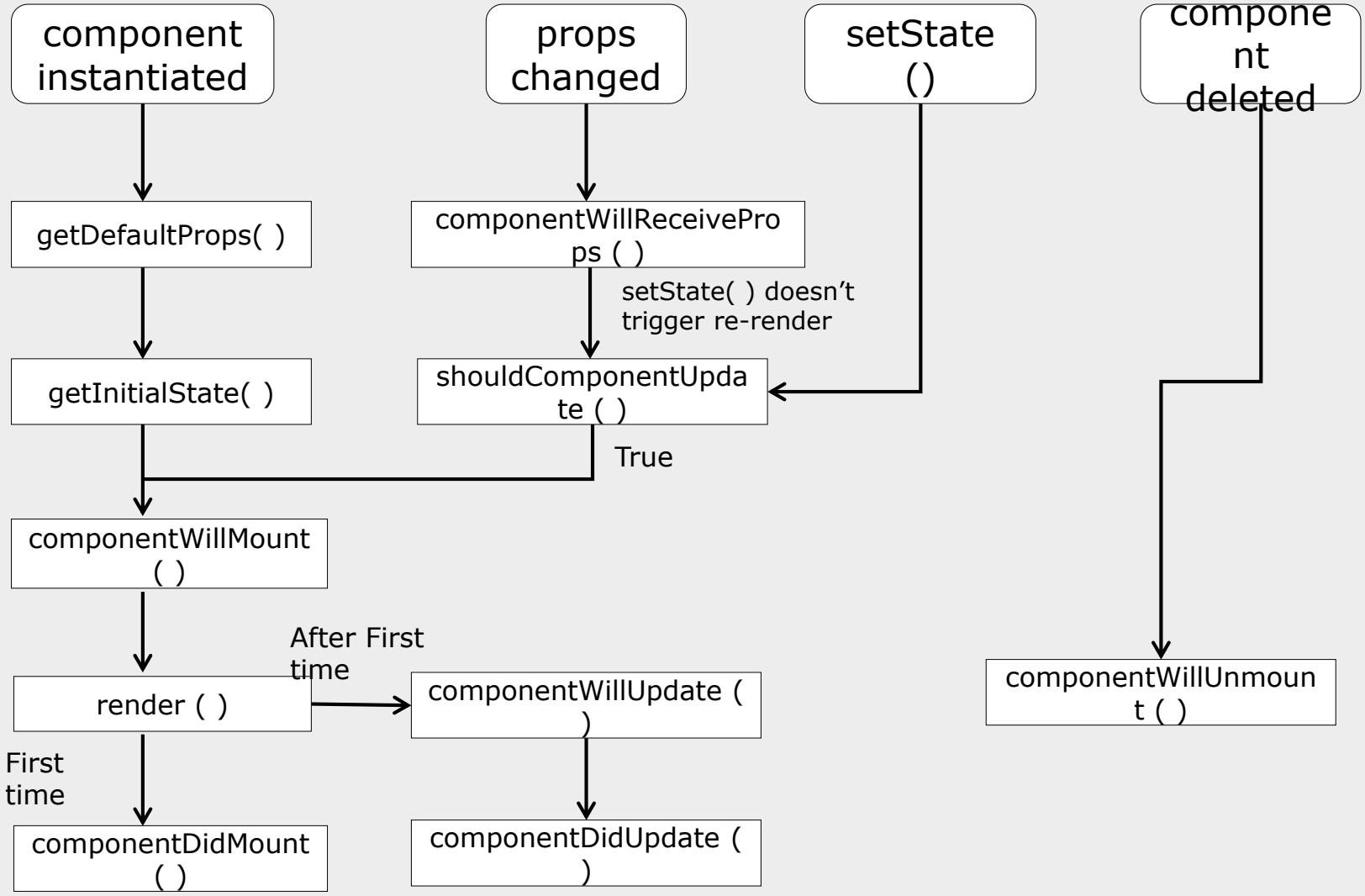
4. unmounting:

In this phase, the component is not needed and the component will get unmounted from the DOM.

componentWillUnmount :

```
componentWillUnmount() {
  this.chart.destroy();
  this.resetLocalStorage();
  this.clearSession();
}
```

React Component-Life cycle methods execution sequence



Pure Components

June 29, 2016 React 15.3 was released a new PureComponent class.

One of the most significant ways to optimize React applications.

Increase in performance by reduces number of rendering process.

Idea behind Pure Components:

React.Component life cycle hook has shouldComponentUpdate method which is set to always return true. This may be good but React might trigger unnecessary re-renders.

One way to deal with these extra re-renders is to change the shouldComponentUpdate function to check when your component needs to update.

Another way to stop extra re-renders is to use a PureComponent.

React.PureComponent is similar to React.Component. The difference between them is that React.Component doesn't implement shouldComponentUpdate(), but React.PureComponent implements it with a shallow prop and state comparison.

Demo

React-Component-Life-Cycle-Methods-Execution

react-create-componentlifecycle
React-pure_component

React’s DOM Updating Strategy

Returning adjacent elements

Until React 16 it was required that a render method returns a single React element. With React 16 it is possible now to return an Array of elements.

Below(refer code 1 in note section) you can see an app rendering a list of Employee Assets. In React 16, it's now possible to extend this list with a component containing multiple new entries.

Generally when we render elements , we cannot render them as separate elements, we will get an error stating ***'Parsing error: Adjacent JSX elements must be wrapped in an enclosing tag'*** .We have to wrap them Using a div or li tag or we have to make use of **React.Fragments**. [refer code 2 in note section]

Returning adjacent elements contd.

Another important thing is we cannot render array of elements also,

If we have an array of data and on rendering them we have to ensure that they have unique keys to separate them from other elements else we will get the below error.[refer code 3 in note section]

```
▶Warning: Encountered two children with the same key,      index.js:1446
`35`. Keys should be unique so that components maintain their identity
across updates. Non-unique keys may cause children to be duplicated
and/or omitted — the behavior is unsupported and could change in a future
version.
    in div (at App.js:28)
    in App (at src/index.js:7)
```

React Fragments

React@16.2 introduces the Fragment component. Until React@16 a component could only return a single element. If returning more than one and then we can see a message like "***Adjacent JSX elements must be wrapped in an enclosing tag***".

If in case need more than one, then it should be wrapped with another container tag.(say a ul or span or dic etc,. or array or Fragments).

```
import React, { Component } from 'react'
class BlogPostExcerpt extends Component
 {
render() {
return (
<React.Fragment>
        <h1>{this.props.title}</h1>
        <p>{this.props.description}</p>
</React.Fragment>
        )
        }
}
```

Shorthand syntax + supported attributes : <></> -> A set of empty tags. Be aware though, the Fragment syntax is only supported as of Babel v7.0.0-beta.31.

```
import React, { Component } from 'react'
class Blogs extends Component
 {
render() {
return (
<>
        <h1>{this.props.title}</h1>
        <p>{this.props.description}</p>
</>
        )
        }
}
```

Demo

Fragments

react-fragments-2019

Summary

- After this you should be clear with

- Updating life cycle hooks
- PureComponents
- React’s DOM Updating Strategy
- Returning adjacent elements
- Fragments

Review

What are the 3 React component's Life Cycle?

1. Mount,insert,demount,
2. Mounting, merge, unmounting
3. Mounting, merge, demounting
4. Mounting, update, unmounting

In React , Strong Type-checking is Provided by PropTypes? True or False.

_____ are used to share the code between multiple components.Which is an array of objects.

1. Prop
2. State
3. Mixins
4. components