React: Class Components and Styling

Session 1

May 22nd 2021

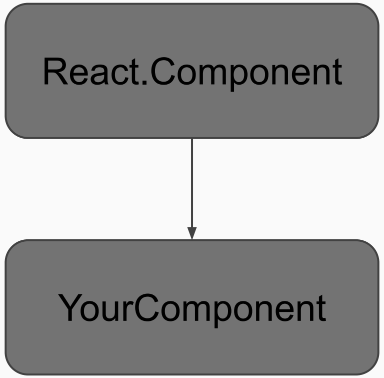
React.Component

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML via a render() function

Class syntax is one of the most common ways to define a React component. While more verbose than the functional syntax, it offers more control in the form of lifecycle hooks

A **subclass** is a class that derives from another class
A **subclass** inherits state and behavior from all of its ancestors
The term superclass refers to a class's direct ancestor as well as all of its ascendant classes

https://reactjs.org/docs/react-component.html
https://www.digitalocean.com/community/tutorials/react-class-components



Props

- Props are arguments passed into React components
- Props are passed to components via HTML attributes
- React Props are like function arguments in JavaScript and attributes in HTML
- Props are also how you pass data or variables or objects from one component to another, as parameters

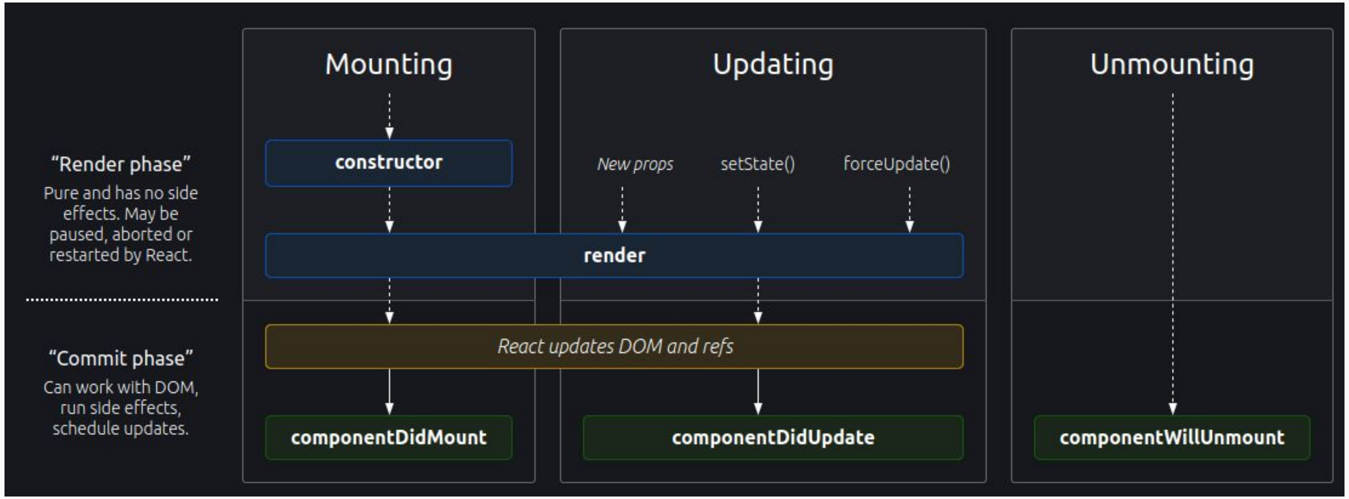
Note: React Props are read-only! You will get an error if you try to change their value.

https://www.w3schools.com/react/react_props.asp

React Component methods

render()	Only required method in a class component When called, it should examine this.props and this.state and return one of the following types: ● **React elements** ● **Arrays and fragments** ● **Portals** ● **String and numbers** ● **Booleans or null**
constructor()	● Initializing local state by assigning an object to this.state ● Binding event handler methods to an instance
componentDidMount()	For introducing any side-effects or subscriptions

React Component Lifecycle



https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

Mounting:

There are four built-in lifecycle methods that are called in the following order when a component is mounted:

- **constructor( )** - This method is used to set the initial state and bind methods to the component
- **getDerivedStateFromProps( )** - In this method, we can set the state of the component based on the props we received. This method is used very rarely where the state depends on changes in props over time
- **render( )** - This is the only required method in the class component. This method returns the HTML elements which are going to be rendered inside the DOM.
- **componentDidMount(**) - It is called right after the component is rendered inside the DOM. All the statements (side-effects or subscriptions) which require the DOM nodes can be executed in this method. Network requests from a remote end-point can also be instantiated in this method.

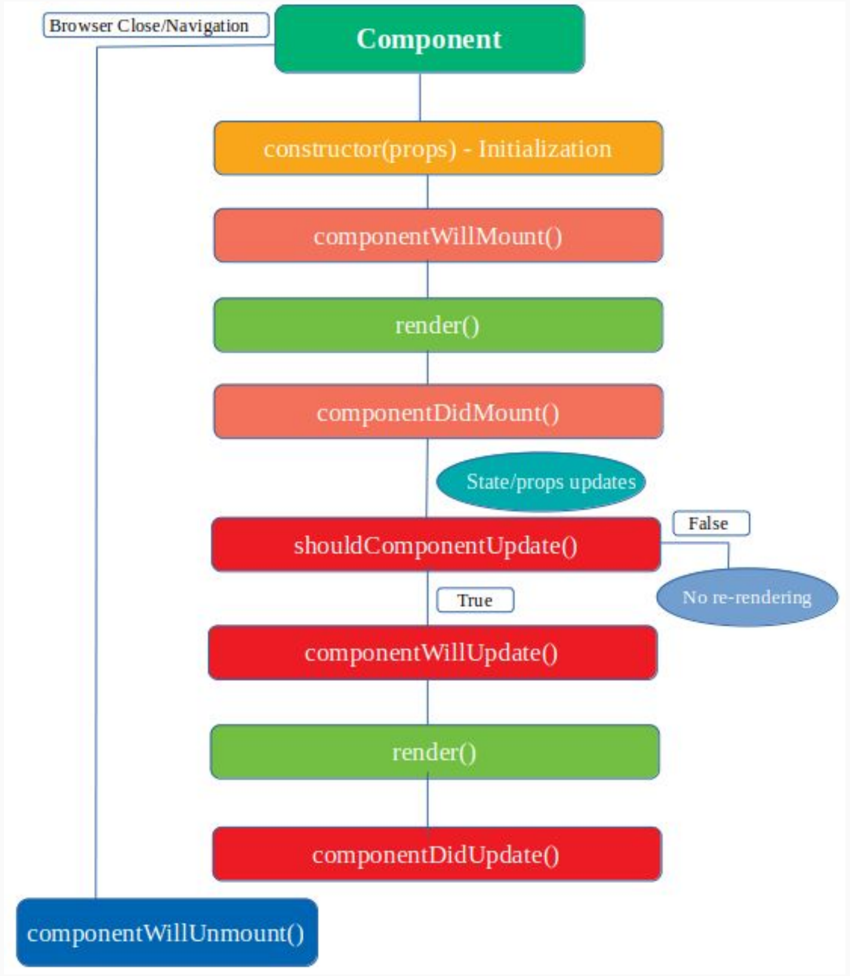
Updating:

Updates in React are caused by changes in state or props. Update leads to re-rendering of the component. The following methods are called when a component is re-rendered:

- **getDerivedStateFromProps( ) -** This method is called again when a component is being re-rendered.
- **shouldComponentUpdate( )** - This method is called before rendering the component when new props are received. It lets React know if the component's output is affected by the newly received props or by the state change. By default, it returns true.
- **render( )** - To re-render the HTML inside the DOM, this method gets called again.
- **getSnapshotBeforeUpdate( )** - This method is called just before the newly rendered HTML gets committed to the DOM. It stores the previous state of the component so that React has an idea of what parts of the DOM needs to be updated.
- **componentDidUpdate( )** - It is called after the component gets re-rendered. This method works just like the componentDidMount( ) method, the difference is that this method does not get called on initial render.

Unmounting:

componentWillUnmount( ) - This method is called just before the component gets destroyed. Any clean up statements should be executed inside this method.



https://www.freecodecamp.org/news/how-to-understand-a-components-lifecycle-methods-in-reactjs-e1a609840630/