What is Life cycle method

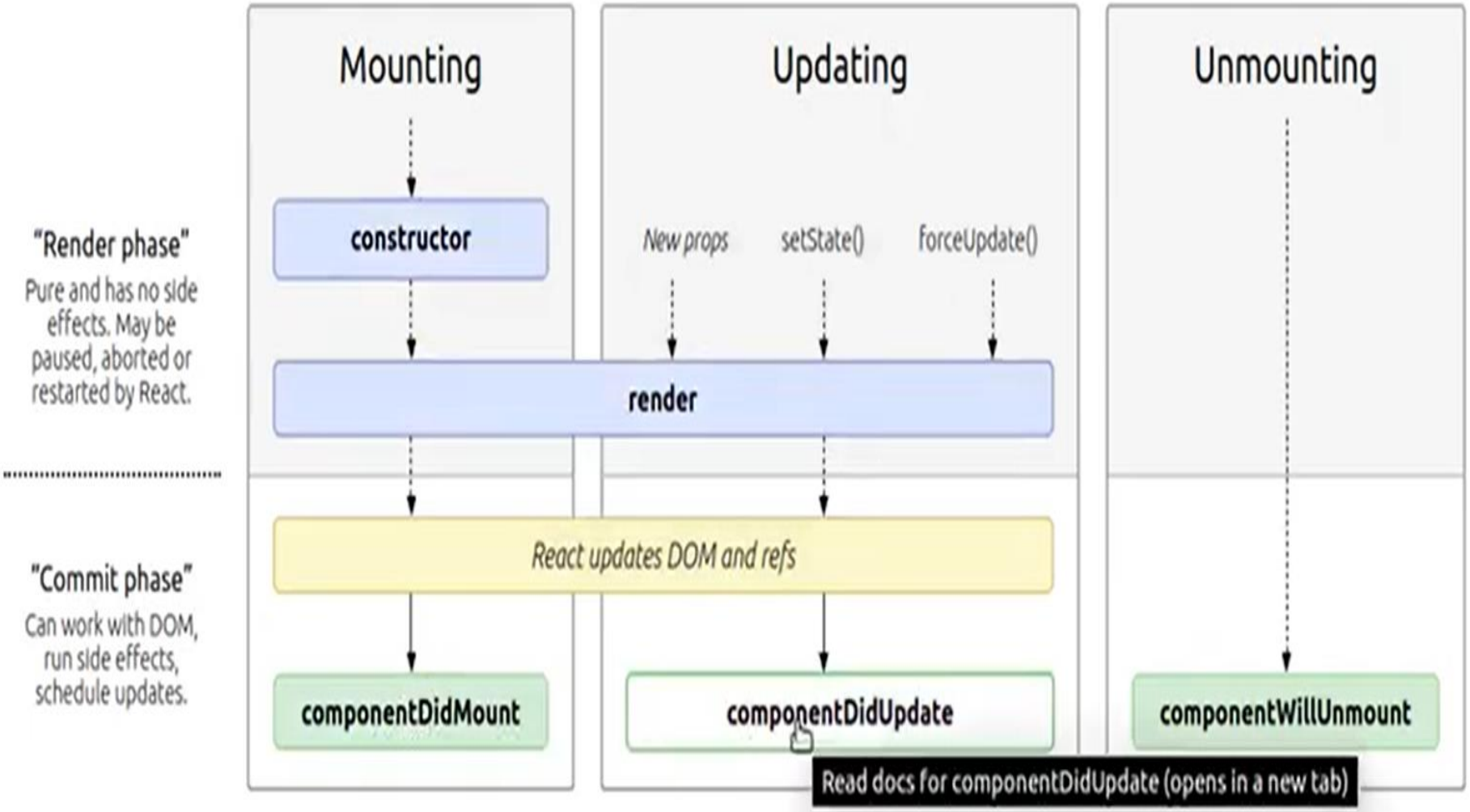
Phase of life cycle

Example

Importants of life cycle

Show less common lifecycles

React version 16.3 Language en-US



Phase:-

- Mounting:--Component is making
- Updating:-- Component is updating
- UnMounting:- Component is dead.

Constructor-render--componentdidmount

```
class App extends React.Component {

  constructor()
  {
    super();
    console.warn("constructor")
  }
  componentDidMount()
  {
    console.warn("constructor")
  }
  render() {
    console.warn("render")
    return (
      <div>
        <h1>Life Cycle Method</h1>
      </div>
    );
  }
```

localhost:3000

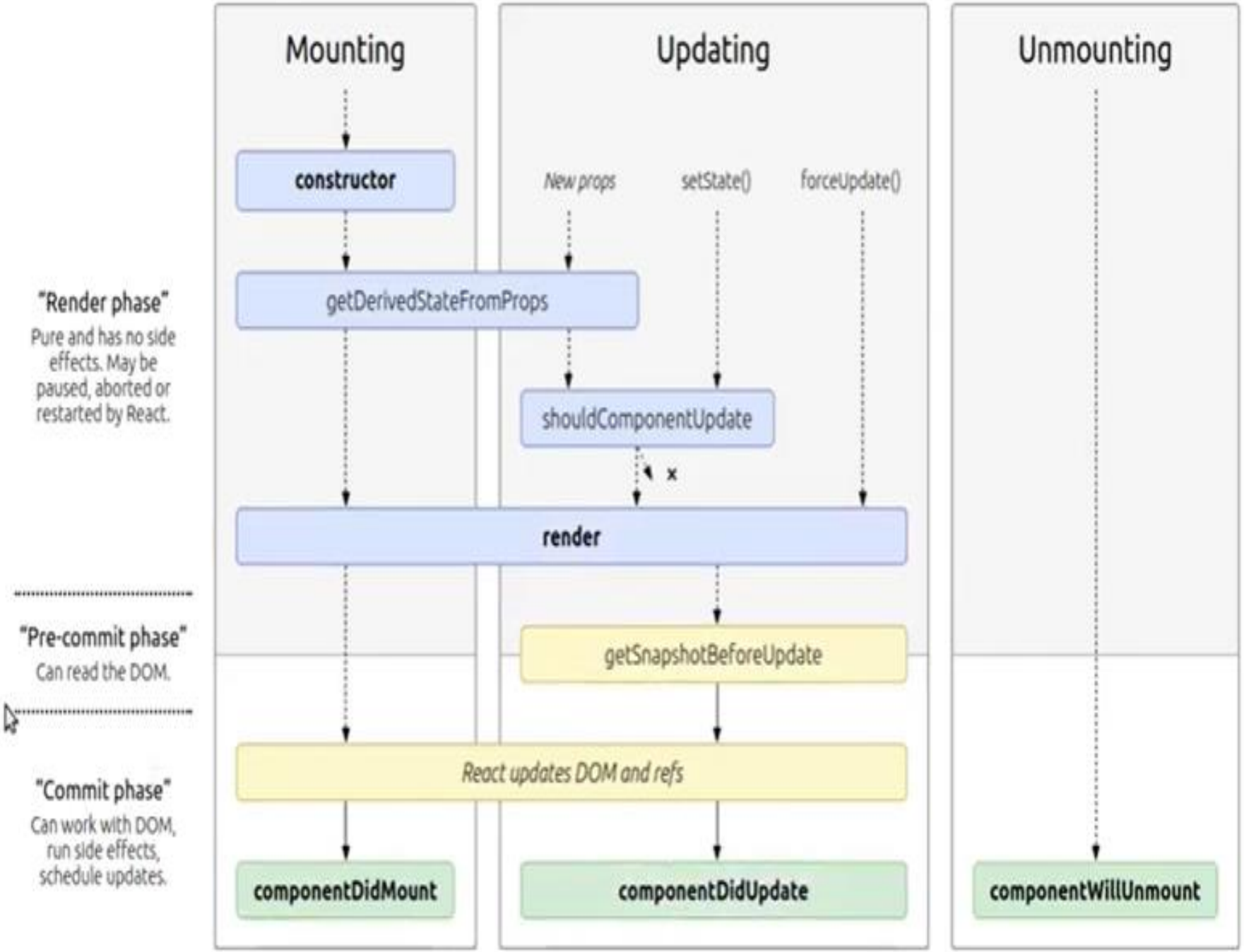
Life Cycle Method

Elements Console Sources Network Performance Memory Application Security Audits

top Filter Default levels

Download the React DevTools for a better development experience: https://fb.me/react-devtools

- constructor
- render
- componentDidMount


Mounting
Updating
Unmounting
constructor
New props
setState()
forceUpdate()
getDerivedStateFromProps
"Render phase"
Pure and has no side effects. May be paused, aborted or restarted by React.
shouldComponentUpdate
x
render
"Pre-commit phase"
Can read the DOM.
getSnapshotBeforeUpdate
React updates DOM and refs
"Commit phase"
Can work with DOM, run side effects, schedule updates.
componentDidMount
componentDidUpdate
componentWillUnmount

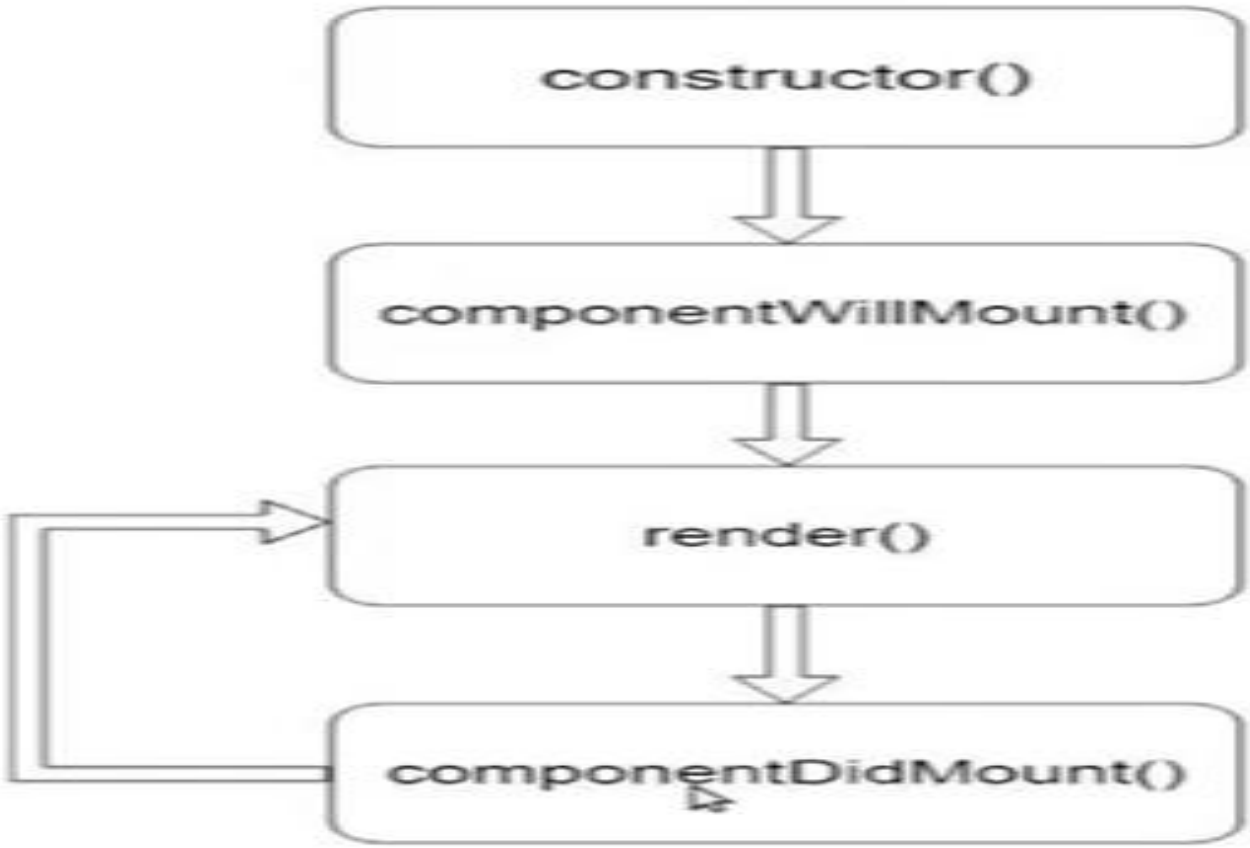
Why we use this life cycle:-

- Suppose we call the api if we call the api inside the constructor and if you attach in the html then it will generate the error.
- Constructor is ready but render is not ready means html is not making
- We can not put any logic inside the render
- So we call the api inside componentDidMount()
- Because all the html data ready then we call the api.
- componentDidMount() is called only all the html data is ready.

ComponentwillUnMount()

- Api call when component is dead

- shouldComponentUpdate():-
- Its asking we have to update the component or not


constructor()
componentWillMount()
render()
componentDidMount()

```
import React from 'react';

class App extends React.Component {
  constructor()
  {
    super()
    this.state={
      data:null
    }
    console.warn("constructor")
  }
  componentDidMount()
  {
    this.setState({data:"updated"});
    console.warn("componentDidMount")
  }
  render() {
    console.warn("render")
    return (
      <div>
        <h1>Life Cycle Method : componentDidMount</h1>
      </div>
    );
  }
}
export default App;
```

Render call two time

Life Cycle Method : componentDidMount

Elements Console Sources Network Performance Memory Application Security Audits

top Filter Default levels

```
Download the React DevTools for a better development experience: https://fb.me/react-devtools
▸ constructor
▸ render
▸ componentDidMount
▸ render
>
```

What is componentdidupdate
Understand somepoints from offical Wbsite
Make Example for it
Test it

```
constructor()
{
  super();
  this.state={
    active:null,
    who:null
  }
}
componentDidUpdate()
{
  console.warn("componentDidUpdate")
  this.setState({who:"anil sidhu"})
}
render() {
  return (
    <div>
      <h1>React componentDidUpdate {this.state.who}</h1>
      <button onClick={()=>{this.setState({active:"yes"})}}>A
    </div>
```

Its goes to the infinite loop

localhost:3000

Maximum update depth exceeded. This can happen when a component repeatedly calls setState inside componentWillUpdate or componentDidUpdate. React limits the number of nested updates to prevent infinite loops.

▸ 5 stack frames were collapsed.

App.componentDidUpdate
src/App.js:14

```
  11 | componentDidUpdate()
  12 | {
  13 |   console.warn("componentDidUpdate")
> 14 |   this.setState({who:"anil sidhu"})
  15 | }
  16 | render() {
```

Elements Console Sources Network Performance Memory Application Security Audits

top Filter Default levels

```
at App.componentDidUpdate (http://localhost:3000/static/js/main.chunk.js:43:10)
at commitLifeCycles (http://localhost:3000/static/js/0.chunk.js:26201:26)
at commitLayoutEffects (http://localhost:3000/static/js/0.chunk.js:29182:11)
at HTMLUnknownElement.callCallback (http://localhost:3000/static/js/0.chunk.js:5407:18)
at Object.invokeGuardedCallbackDev (http://localhost:3000/static/js/0.chunk.js:5456:20)
at invokeGuardedCallback (http://localhost:3000/static/js/0.chunk.js:5510:35)
```

ComponentDidUpdate()

Use this as an opportunity to operate on the DOM when the component has been updated. This is also a good place to do network requests as long as you compare the current props to previous props (e.g. a network request may not be necessary if the props have not changed).

```
componentDidUpdate(prevProps) {
  // Typical usage (don't forget to compare props):
  if (this.props.userID !== prevProps.userID) {
    this.fetchData(this.props.userID);
  }
}
```

You **may call `setState()` immediately** in `componentDidUpdate()` but note that **it must be wrapped in a condition** like in the example above, or you'll cause an infinite loop. It would also cause an extra re-rendering which, while not visible to the user, can affect the component performance. If you're trying to "mirror" some state to a prop coming from above, consider using the prop directly instead. Read more about why copying props into state causes bugs.

```
componentDidUpdate()
{
  console.warn("componentDidUpdate")
  if(this.state.who==null)
  {
    this.setState({who:"anil sidhu"})

  }
}
```

ComponentWillUnmount:-

- If we delete the user and show the message user has delete.then use it

componentWillUnmount()

```
componentWillUnmount()
```

`componentWillUnmount()` is invoked immediately before a component is unmounted and destroyed. Perform any necessary cleanup in this method, such as invalidating timers, canceling network requests, or cleaning up any subscriptions that were created in `componentDidMount()`.

You **should not call `setState()`** in `componentWillUnmount()` because the component will never be re-rendered. Once a component instance is unmounted, it will never be mounted again.

Create a component user.js

```
JS App.js src                M
× JS User.js src             U
▲ BLOG
  logo512.png
  {} manifest.json
  robots.txt
▲ src
  ▸ cmp
  # App.css
  JS App.js                  M
  # index.css
  JS index.js
  JS serviceWorker.js
  JS User.js                 U
  .gitignore
  {} package-lock.json
```

```
import React, { Component } from 'react';

class User extends Component {
    componentWillUnmount()
    {
        alert("User has been Deleted")
    }
    render() {
        return (
            <div>
                <ul>
                    <li>Name:Anil</li>
                    <li>Email:webanilsidhu@test.com</li>
                    <li>Contact:00000000</li>

                </ul>
            </div>
        );
    }
```

Call user.js in App.js

```
import React from 'react';
import User from './User'
class App extends React.Component {
  constructor()
  {
    super()
    this.state={
      toggle:true
    }
  }
  render() {
    return (
      <div>
        <h1>React : component will unmount</h1>
        <button onClick={()=>{this.setState({toggle:!this.state.toggle})}} >Delete Us
      </div>
    );
  }
}
```

```
import User from './User'
class App extends React.Component {
  constructor()
  {
    super()
    this.state={
      toggle:true
    }
  }
  render() {
    return (
      <div>
        <h1>React : component will unmount</h1>
        {
          this.state.toggle?
          <User />
        }
        <button onClick={()=>{this.setState({toggle:!this.state.toggle})}} >Delete User</button>
      </div>
    );
  }
}
```

```
}
render() {
  return (
    <div>
      <h1>React : com
      {
        this.state.to
        <User />:null
      }
      <button onClick={()=>{this.setState({toggle:!this.state.toggle})}} >Delete User</button>
    </div>
  );
}
}
```

isNullableTypeAnnotation
isNullLiteral
isNullLiteralTypeAnnotation
isNullOrUndefined
tsNullKeyword
tsNonNullExpression
isTSNonNullExpression

React : component will unmount

- Name:Anil
- Email:webanilsidhu@test.com
- Contact:00000000

Delete User

Thanks