(List and Hooks)

• Explain Life cycle in Class Component and functional component with Hooks.

React lifecycle method explained.

First, let’s take a look at how it’s been done traditionally. In order to do that, we’re going to take a closer look at React components.

As you probably know, each React component instance has a lifecycle. The component’s lifecycle consists of three phases:

* **Mounting lifecycle methods**, that is inserting elements into the DOM.
* **Updating**, which involves methods for updating components in the DOM.
* **Unmounting**, that is removing a component from the DOM.

## Mounting in the React component lifecycle

## As we mentioned, during the mounting phase of the lifecycle, the class component is inserted into the DOM. A good example would be *componentDidMount()* – a lifecycle method that runs after the component is mounted and rendered to the DOM. It is great when you want to do an interval function or an asynchronous request. Example:

|  |  |  |
| --- | --- | --- |
|  | | componentDidMount() { |
|  | | fetch(url).then(res => { |
|  | | // Handle response in the way you want. |
|  | | // Most often with editing state values. |
|  | | }) |
|  | } Updating in the React component lifecycle The *componentDidUpdate()* render method is called right after the updating happens. This one is called always except for the initial render. That’s a good place to interact with a non-reactive environment. It’s a good idea to make http requests here.  You can call *setState()* in this method to enqueues changes to the component state. but it is very important to wrap that in some condition to avoid an infinite loop (doesn’t matter if state has the same values or not). If there is no condition, the process goes as follows:   1. You call *setState()* in the *componentDidUpdate()* method. 2. The component is updated. 3. *componentDidUpdate()* is invoked. 4. *setState()* is called again …  Unmounting in the React component lifecycle*componentWillUnmount()* is invoked just before the component is removed from the DOM. You should use that to remove event listeners, clear intervals and cancel requests. In other words: all the needed cleanup. | |