TW-010 STUDENT VERSION

TEAMWORK

CLARUSWAY
WAY TO REINVENT YOURSELF

Meeting Agenda

- Icebreaking
- Questions
- Interview Questions
- Coffee Break
- Coding Challenge
- Video of the week
- Retro meeting
- Case study / project

Teamwork Schedule

Ice-breaking 5m

- Personal Questions (Study Environment, Kids etc.)
- Any challenges (Classes, Coding, studying, etc.)
- Ask how they're studying, give personal advice.
- Remind that practice makes perfect.

Team work 5m

- Ask what exactly each student does for the team, if they know each other, if they care for each other, if they follow and talk with each other etc.

Ask Questions 15m

1.What is the purpose of the useState hook in React?

A. To manage state in functional components
B. To pass data from parent to child components
C. To handle asynchronous operations
D. To define lifecycle methods

Correct answer: A *The useState hook is used to manage state in functional components. It allows you to add stateful logic to functional components.*

2. How do you pass data from a parent component to a child component in React?

A. Using the useState hook
B. Using the useEffect hook
C. Using props
D. Using lifecycle methods

Correct answer: C *You can pass data from a parent component to a child component using props. Props are passed as attributes to the child component when it is rendered.*

3. What is the significance of the lifecycle methods in React?

A. They help manage state in functional components
B. They handle event handling in React
C. They provide a way to clean up resources in components
D. They control the rendering of components

Correct answer: D *Lifecycle methods provide hooks for executing code at specific stages of a component's lifecycle. They allow you to perform tasks such as initializing state, making API calls, and cleaning up resources.*

4. Which lifecycle method is called only once, immediately after a component is mounted in React?

A. componentDidMount
B. componentDidUpdate
C. componentWillUnmount
D. componentWillReceiveProps

Correct answer: A *The componentDidMount method is called only once, immediately after a component is mounted onto the DOM.*

5. How do you update the state of a component in React?

A. Using the setState function
B. Using the useState hook
C. Using props
D. Using the useEffect hook

Correct answer: A *You can update the state of a component using the setState function. This function allows you to modify the state and trigger a re-render of the component.*

6. What is the difference between props and state in React?

A. Props are used for state management, while state is used for data passing.
B. Props are used to pass data from child to parent components, while state is used for data within a component.
C. Props are immutable, while state can be changed.
D. Props are used in class components, while state is used in functional components.

Correct answer: B *Props are used to pass data from a parent component to a child component, while state is used to manage internal component data that can change over time.*

7. Which lifecycle method is called just before a component is updated in React?

A. componentWillUpdate
B. componentDidUpdate
C. componentWillMount
D. componentWillUnmount

Correct answer: A *The componentWillUpdate method is called just before a component is updated.*

8. Can you use multiple useState hooks in a single component?

A. Yes, you can use multiple useState hooks in a single component.
B. No, you can only use one useState hook in a component.
C. Yes, but only in class components, not in functional components.
D. No, useState can only be used in conjunction with useEffect.

Correct answer: A *Yes, you can use multiple useState hooks in a single component to manage different pieces of state independently.*

9. How can you fetch data from an API in React using the useEffect hook?

A. By using the useState hook
B. By using the useContext hook
C. By using the useFetch hook
D. By making an asynchronous request inside the useEffect callback function

Correct answer: D *By making an asynchronous request inside the useEffect callback function, you can fetch data from an API in React.*

10. How can you perform side effects only once in React using the useEffect hook?

A. By using the useCallback hook
B. By using the useMemo hook
C. By passing an empty dependency array as the second argument to useEffect
D. By using the useLayoutEffect hook

Correct answer: C *By passing an empty dependency array as the second argument to useEffect, you can ensure that the effect is only run once when the component is mounted.*

11. Can you pass props from a child component to its parent component in React?

A. Yes, props can be passed from a child component to its parent component.
B. No, props can only be passed from a parent component to its child component.
C. Yes, but it requires additional configuration.
D. No, React follows a one-way data flow for props.

Correct answer: B *No, in React, props flow in a one-way direction, from parent to child components. Child components cannot directly pass props back to the parent.*

12. Which of the following is not one of the lifecycle phases?

A. Mounting
B. Updating
C. Rendering
D. Unmounting

Correct answer: C *Rendering is not one of the distinct lifecycle phases in React. Rendering is part of the updating phase.*

13. What is the mounting phase in the React component lifecycle?

A. It is the initial phase when a component is created and inserted into the DOM.
B. It is the phase when a component receives new props and updates its state.
C. It is the phase when a component is removed from the DOM.
D. It is the phase when a component re-renders due to changes in its state or props.

Correct answer: A *The mounting phase is the initial phase when a component is created and inserted into the DOM.*

14. What is the unmounting phase in the React component lifecycle?

A. It is the phase when a component is created and inserted into the DOM.
B. It is the phase when a component receives new props and updates its state.
C. It is the phase when a component is removed from the DOM.
D. It is the phase when a component re-renders due to changes in its state or props.

Correct answer: C *The unmounting phase is the phase when a component is removed from the DOM.*

15. What is the updating phase in the React component lifecycle?

A. It is the initial phase when a component is created and inserted into the DOM.
B. It is the phase when a component receives new props and updates its state.
C. It is the phase when a component is removed from the DOM.
D. It is the phase when a component re-renders due to changes in its state or props.

Correct answer: B *The updating phase is the phase when a component receives new props and updates its state.*

Interview Questions 15m

1. How does the useEffect hook work in React?

The useEffect hook is used to perform side effects in functional components, such as data fetching, subscriptions, or DOM manipulations. It takes a callback function as its first parameter, which will be executed after every render. The hook also allows you to specify dependencies as the second parameter to control when the effect is re-run.

2. What are lifecycle methods in React and how are they used with useEffect?

Lifecycle methods in React are special methods that allow you to perform actions at specific stages of a component's lifecycle, such as when it is created, updated, or removed from the DOM. Prior to React 16.3, class components were commonly used to define and utilize lifecycle methods. However, with the introduction of React hooks, specifically the useEffect hook, the use of class components and traditional lifecycle methods has been supplemented.

The useEffect hook in React can be used as a replacement for several lifecycle methods. It combines the functionality of componentDidMount, componentDidUpdate, and componentWillUnmount into a single hook. The useEffect hook allows you to perform side effects, such as fetching data, subscribing to events, or manipulating the DOM, in functional components.

The useEffect hook takes two arguments: a callback function and an optional array of dependencies. The callback function is executed after the component renders, and it can contain the code for the desired side effect. The dependencies array is used to specify which variables or values the effect depends on.

When the component renders, React runs the effect after the initial render and after every subsequent render if any of the dependencies have changed. If the dependencies array is empty, the effect is only run once, similar to the componentDidMount lifecycle method. If you want to specifically handle updates, you can include conditional logic inside the effect to check for specific changes and mimic the behavior of componentDidUpdate.

To replicate the behavior of the componentWillUnmount lifecycle method, the useEffect hook can optionally return a cleanup function. This cleanup function will be executed when the component is unmounted or when the dependencies change before the next execution of the effect.

Overall, the useEffect hook provides a versatile and flexible way to incorporate lifecycle-like behavior in functional components, allowing you to handle side effects, manage component updates, and manage component lifecycle concerns.

3. What is Lifting State Up in React?

Lifting State Up in React refers to the process of moving the state from a child component to its parent component. Normally, each component in React manages its own state independently. However, there are cases where multiple components need to share the same state or when a state change in one component needs to affect another component.

In such scenarios, lifting state up allows you to elevate the state to a common ancestor component that is higher up in the component hierarchy. By doing so, the state becomes accessible to multiple child components, enabling them to share and synchronize the same state.

To implement lifting state up, you need to follow these steps:

- **Identify the shared state:** Determine the specific state that multiple components need to access or synchronize.
- **Find the common ancestor:** Identify the component higher up in the component hierarchy that is a common ancestor to the components requiring access to the shared state.
- **Define the state in the common ancestor:** Move the state from the child component(s) to the common ancestor component. This involves creating a state variable in the common ancestor and passing it down as props to the child components.
- **Pass down state and callbacks:** Pass the state values and any necessary callbacks as props from the common ancestor component to the child components that need access to the shared state.
- **Update the state in the common ancestor:** Whenever a child component needs to modify the shared state, it should call the callback function provided by the common ancestor component. This allows the common ancestor to update the state, triggering a re-render and propagating the updated state to all relevant child components.

By lifting state up, you achieve a single source of truth for the shared state, enabling synchronization and consistent behavior across the components. This approach promotes better data flow and makes it easier to manage and reason about the state in your React application.

4. Why fragments are better than container divs?

Below are the list of reasons to prefer fragments over container DOM elements,

- Fragments are a bit faster and use less memory by not creating an extra DOM node. This only has a real benefit on very large and deep trees.
- Some CSS mechanisms like Flexbox and CSS Grid have a special parent-child relationships, and adding divs in the middle makes it hard to keep the desired layout.
- The DOM Inspector is less cluttered.

5. What are inline conditional expressions?

Inline conditional expressions, also known as conditional rendering or ternary expressions, are a way to conditionally render content or apply logic within JSX (JavaScript XML) in React. They allow you to conditionally include or exclude elements or values based on a specific condition.

In JavaScript, the ternary operator (condition ? expression1 : expression2) is used to create inline conditional expressions. This operator evaluates the condition and returns expression1 if the condition is true, or expression2 if the condition is false. You can also embed any expressions in JSX by wrapping them in curly braces and then followed by JS logical operator &&.

```
<div>

  <h1>Products</h1>
  {
    products.length > 0 && !isLogin ? (
      <h2>You have {products.length} products.</h2>
    ) : (
      <h2>You don't have products.</h2>
    );
  }
  {
    showInput && <input type="text"/>
  }

</div>
```

Coding Challenge 15m

- RC-CC-02 Interview Q-A App

☕

Coffee Break 10m

☕

Video of the Week 10m

- How to fetch data from API using axios in React useEffect hook?
- What NOT to do in an Interview

Case study/Project 15m

1. RP-03 React ToDo App
2. **Optional** Checkout Page (RP-04) **Optional**

- ***You can solve either the mock API version or the mock Data version, depending on your preference.An additional project has been provided for those who are interested. You can solve it at any time you wish.***

Retro Meeting on a personal and team level 10m

Ask the questions below:

- What went well?
- What could be improved?
- What will we commit to do better in the next week?

Closing 5m

-Next week's plan

-QA Session
