hillel IT school

Lesson 31

План заняття

- Ref 2.0;
- Functional component;
- useState;
- useEffect;
- useLayoutEffect;

Function component

Функціональний компонент – це функція, яка приймає на вхід пропси та повертає дерево React-елментів. За фактом - це майже те саме, що метод render у класових компонентах.

Хуки - це функції, які дозволяють використовувати стан та життєвий цикл усередині функціональних компонентів. Вони можуть бути вбудованими в React та користувальницькими.

Назва хука починається зі слова use: useState, useEffect.

Не можна змінювати порядок виклику хуків. Їх не можна викликати всередині умов, циклів та вкладених функцій. Інакше React не розумітиме, який виклик відповідає якомусь із хуків.

Хуки можна викликати лише з:

- тіла самої функції функціонального компонента
- з інших хуків

useState

Викличте useState на верхньому рівні вашого компонента, щоб оголосити змінну стану.

Синтаксис:

const [example, setExample] = useState(initialState);

initialState: початкове значення, яким має бути стан. Це може бути значення будь-якого типу, але для функцій існує особлива поведінка. Цей аргумент ігнорується після початкового відтворення.

Якщо ви передаєте функцію як initialState, вона буде розглядатися як функція ініціалізації. Він має бути чистим, не приймати аргументів і повертати значення будь-якого типу. React викличе вашу функцію ініціалізації під час ініціалізації компонента та збереже її повернуте значення як початковий стан.

useState повертає масив рівно з двома значеннями:

- Стан. Під час першого відтворення він відповідатиме початковому стану, який ви передали.
- Функція set, яка дозволяє оновлювати стан до іншого значення та запускати повторне відтворення.

useState

Важливо!

- useState — це хук, тому ви можете викликати його лише на верхньому рівні вашого компонента або ваших власних хуків. Ви не можете викликати внутрішні цикли або умови. Якщо вам це потрібно, витягніть новий компонент і перемістіть у нього стан.
- У строгому режимі React двічі викличе вашу функцію ініціалізації, щоб допомогти вам знайти sideEffects. Це поведінка лише для розробки і не впливає на production. Якщо ваша функція ініціалізації чиста (як і має бути), це не повинно впливати на поведінку. Результат одного з викликів буде проігноровано.

setFunction in useState

Функція set, яку повертає useState, дозволяє оновити стан до іншого значення та запустити повторне відтворення. Ви можете безпосередньо передати наступний стан або функцію, яка обчислює його з попереднього стану.

Синтаксис:

const [state, setState] = useState({})

setState(nextState)

nextState: значення, яким має бути стан. Це може бути значення будь-якого типу, але для функцій існує особлива поведінка.

Якщо ви передаєте функцію як nextState, вона розглядатиметься як функція оновлення. Віна має бути чистою, повинена приймати стан очікування як єдиний аргумент і повинна повертати наступний стан. React поставить вашу функцію оновлення в чергу та повторно візуалізує ваш компонент. Під час наступного рендерингу React обчислить наступний стан, застосовуючи всі програми оновлення в черзі до попереднього стану.

setFunction in useState

Важливо:

- Функція set лише оновлює змінну стану для наступного відтворення. Якщо ви прочитаєте змінну стану після виклику функції set, ви все одно отримаєте старе значення, яке було на екрані до вашого виклику.
- Якщо нове значення, яке ви надаєте, ідентичне поточному стану, як визначено порівнянням Object.is, React пропустить повторне рендеринг компонента та його дітей. Це оптимізація. Хоча в деяких випадках React все одно може знадобитися викликати ваш компонент, перш ніж пропускати дочірні елементи, це не повинно вплинути на ваш код.
- React пакетно оновлює стан. Він оновлює екран після того, як усі обробники подій запущені та викликають свої встановлені функції. Це запобігає багаторазовому повторному рендерингу під час однієї події. У рідкісних випадках, коли вам потрібно змусити React оновити екран раніше, наприклад, щоб отримати доступ до DOM, ви можете використовувати flushSync.
- Виклик функції set під час візуалізації дозволений лише з компонента, який зараз рендерить. React відхилить свій вихід і негайно спробує відобразити його знову з новим станом. Цей шаблон рідко потрібен, але ви можете використовувати його для зберігання інформації з попередніх візуалізацій.

Докладніше тут https://react.dev/reference/react/useState

useEffect

useEffect — це хук React, який дозволяє синхронізувати компонент із зовнішньою системою. Позволяє працювати з сайд-ефектами при:

1. монтування (аналог компонентаDidMount)
2. оновлення стану або пропсов (аналог компонентаDidUpdate)
3. розмонтування (аналог компонентаWillUnmount)

Спрацьовує гарантовано після оновлення DOM для поточного компонента.

Снтаксис:

useEffect(setup, dependencies?)

setup: функція з логікою вашого ефекту. Ваша функція налаштування також може повертати функцію очищення. Коли ваш компонент додано до DOM, React запустить вашу функцію налаштування. Після кожного повторного рендерингу зі зміненими залежностями React спочатку запустить функцію очищення (якщо ви її надали) зі старими значеннями, а потім запустить вашу функцію налаштування з новими значеннями. Після видалення компонента з DOM React запустить вашу функцію очищення.

dependencies: список усіх реактивних значень, на які посилається код налаштування. Реактивні значення включають властивості, стан і всі змінні та функції, оголошені безпосередньо в тілі компонента. Список залежностей повинен мати постійну кількість елементів і бути записаним у рядку, як [dep1, dep2, dep3]. React порівнює кожну залежність з її попереднім значенням за допомогою порівняння Object.is. Якщо ви пропустите цей аргумент, ваш ефект буде запускатися повторно після кожного повторного відтворення компонента.

useEffect

Важливо!

- useEffect — це хук, тому ви можете викликати його лише на верхньому рівні вашого компонента або ваших власних хуків. Ви не можете викликати внутрішні цикли або умови. Якщо вам це потрібно, витягніть новий компонент і перемістіть у нього стан.
- Якщо ви не намагаєтеся синхронізуватися з якоюсь зовнішньою системою, можливо, вам не потрібен ефект.
- Коли строгий режим увімкнено, React запустить один додатковий цикл налаштування+очищення лише для розробки перед першим справжнім налаштуванням. Це стрес-тест, який гарантує, що ваша логіка очищення "віддзеркалює" вашу логіку налаштування та зупиняє або скасовує будь-які дії налаштування. Якщо це викликає проблему, застосуйте функцію очищення.

useEffect

Важливо!

- Якщо деякі з ваших залежностей є об’єктами або функціями, визначеними всередині компонента, існує ризик, що вони спричинять повторний запуск Ефекту частіше, ніж потрібно. Щоб виправити це, видаліть непотрібні залежності об’єктів і функцій. Ви також можете отримати оновлення стану та нереактивну логіку за межами вашого ефекту.
- Якщо ваш ефект не був спричинений взаємодією (наприклад, клацанням), React, як правило, дозволить браузеру спочатку намалювати оновлений екран перед запуском вашого ефекту. Якщо ваш ефект виконує щось візуальне (наприклад, позиціонує спливаючу підказку), і затримка помітна (наприклад, він мерехтить), замініть useEffect на useLayoutEffect.
- Навіть якщо ваш ефект був спричинений взаємодією (наприклад, клацанням), браузер може перефарбувати екран перед обробкою оновлень стану всередині вашого ефекту. Зазвичай це те, що ви хочете. Однак, якщо вам потрібно заблокувати браузеру перемальовування екрана, вам потрібно замінити useEffect на useLayoutEffect.
- Ефекти запускаються лише на клієнті. Вони не запускаються під час рендерингу сервера.

useEffect

Ефект дозволяє синхронізувати ваш компонент із зовнішньою системою. Тут зовнішня система означає будь-який фрагмент коду, який не контролюється React, наприклад:

- Таймер, керований за допомогою setInterval() і clearInterval().
- Підписка на подію за допомогою window.addEventListener() і window.removeEventListener().
- Бібліотека анімації третьої сторони з такими API, як animation.start() і animation.reset().

Якщо ви не підключаєтесь до жодної зовнішньої системи, вам, ймовірно, не потрібен ефект.

useLayoutEffect

useLayoutEffect — це версія useEffect, яка запускається до того, як браузер перефарбує екран.

Синтаксис:

useLayoutEffect(setup, dependencies?)

Більшості компонентів не потрібно знати їх положення та розмір на екрані, щоб вирішити, що відображати. Вони повертають лише деякі JSX. Потім браузер розраховує їх розташування (положення та розмір) і перефарбовує екран. Але іноді нам потрібно знати розміри елементів перш ніж відобразити щось (наприклад підказку) на екрані.

Для цього потрібно виконати рендеринг у три проходи:

- Візуалізуйте спливаючу підказку будь-де (навіть у неправильному місці).
- Виміряйте його висоту та вирішіть, де розмістити підказку.
- Знову відобразіть підказку в потрібному місці.

useLayoutEffect

Важливо!

- useLayoutEffect — це хук, тому ви можете викликати його лише на верхньому рівні вашого компонента або ваших власних хуків. Ви не можете викликати внутрішні цикли або умови. Якщо вам це потрібно, витягніть компонент і перемістіть туди ефект.
- Коли строгий режим увімкнено, React запустить один додатковий цикл налаштування+очищення лише для розробки перед першим справжнім налаштуванням. Це стрес-тест, який гарантує, що ваша логіка очищення «віддзеркалює» вашу логіку налаштування та зупиняє або скасовує будь-які дії налаштування. Якщо це викликає проблему, застосуйте функцію очищення.
- Якщо деякі з ваших залежностей є об’єктами або функціями, визначеними всередині компонента, існує ризик, що вони спричинять повторний запуск Ефекту частіше, ніж потрібно. Щоб виправити це, видаліть непотрібні залежності об’єктів і функцій. Ви також можете отримати оновлення стану та нереактивну логіку за межами вашого ефекту.
- Ефекти запускаються лише на клієнті. Вони не запускаються під час рендерингу сервера.
- Код усередині useLayoutEffect і всі заплановані з нього оновлення стану блокують браузер від перефарбування екрана. При надмірному використанні це робить ваш додаток повільним. Якщо можливо, віддайте перевагу useEffect.

Дякую за увагу