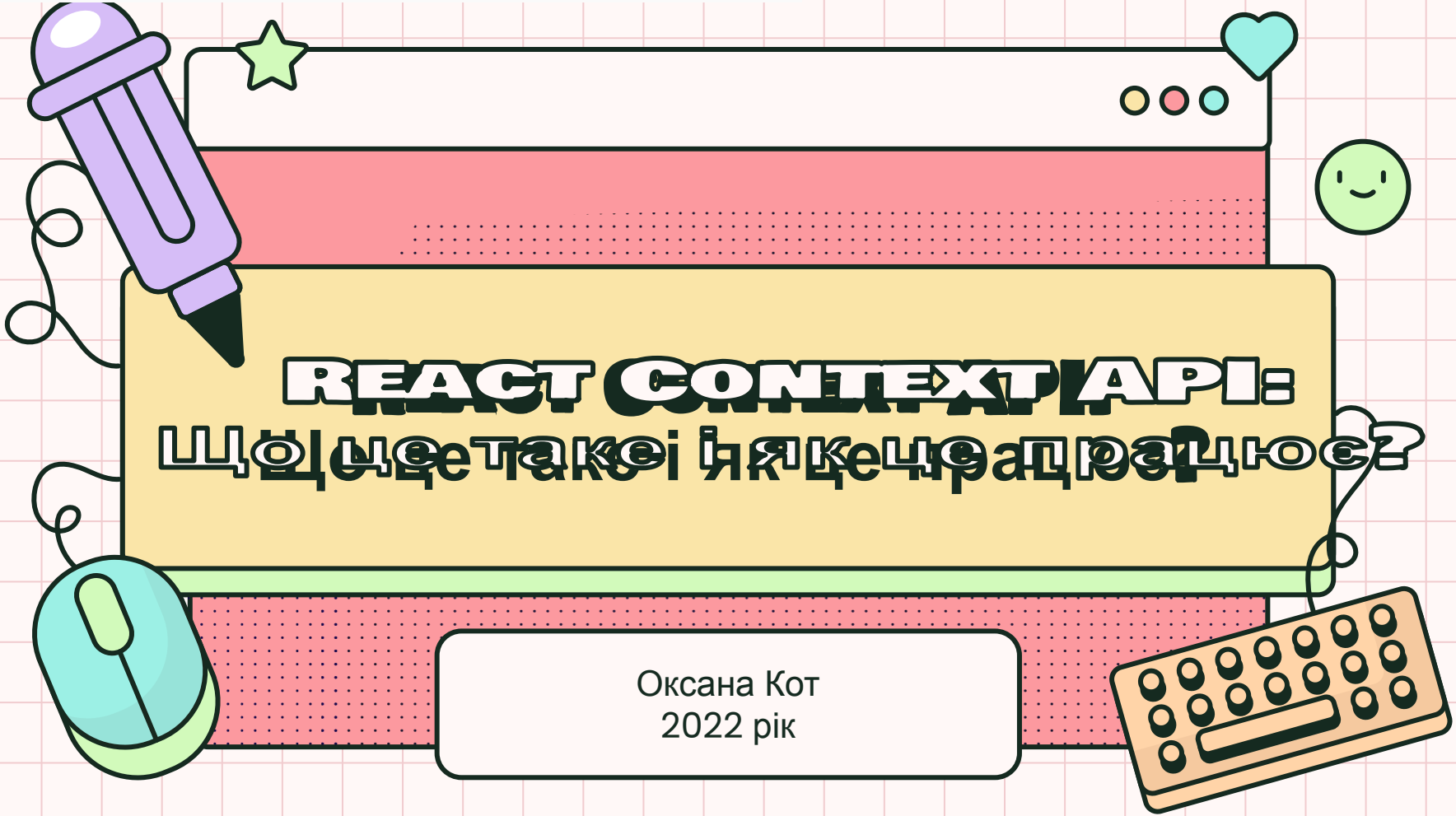


React Context API: Що це таке і як це працює?

Оксана Кот
2022 рік

ВСТУП

React Context API — це спосіб для програми React ефективно створювати глобальні змінні, які можна передавати. Це альтернатива «бурінню реквізитів» або переміщенню реквізитів від дідуся до дитини до батька тощо. Контекст також рекламується як простіший і легший підхід до управління станом за допомогою Redux.

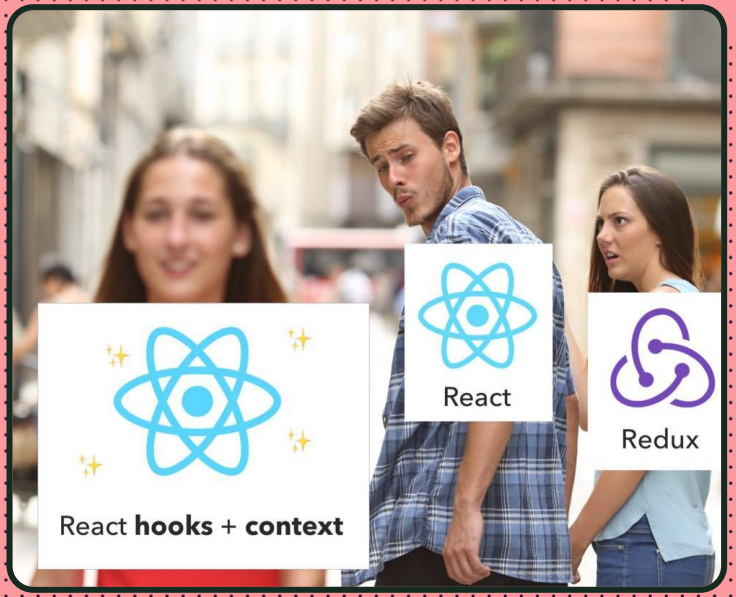
ЯК ЦЕ ПРАЦЮЄ?

`React.createContext()`це все, що вам потрібно. Він повертає споживача та постачальника.**Провайдер**є компонентом що, як випливає з назви, забезпечує стан його діти. Він буде зберігати «магазин» і бути батьком усіх компонентів, які можуть знадобитися в цьому магазині. **Споживач**як це буває, є компонентом, який споживає і використовує державу.

CONTEXT API ЗАМІНИТЬ **REDUX?**

Ну, не зовсім.

Redux чудовий і ідеально відповідає потребам управління станом. Насправді, він настільки добре задовольнив цю потребу, що стало відомо, що ви не можете бути «справжнім» розробником React, якщо не знаєте, як працювати з Redux.



CONTEXT API ЗАМІНИТЬ **REDUX?**

Однак Redux має свої недоліки, і тому важливо знати, що Context API дає нам, а Redux — ні:

Простота
Під час використання Redux люди, як правило, керують майже всім своїм станом у Redux, і це викликає 2 проблеми:

1. Накладні витрати. Навіщо мені створювати/оновлювати 3 файли лише для того, щоб додати одну маленьку функцію?
2. Однією з значних переваг одностороннього зв'язування даних React є те, що його легко зрозуміти – компонент передає стан своїм нащадкам. Використання Redux позбавляє нас цього.

Використовуючи Context API, ми можемо визначити кілька непов'язаних контекстів (сховищ) і використовувати кожен у відпов

02

Як використовувати контекст

Використання контексту в React вимагає 3 простих кроків: створення контексту, надання контексту та використання контексту.

Створення контексту

Вбудована заводська функція`createContext (за замовчуванням)` створює екземпляр контексту:

```
імпорт{createContext}від "реагувати";
конст Контекст=createContext('Значення за замовчуванням');
```

Фабрична функція приймає один необов'язковий аргумент: значення за замовчуванням.

Надання контексту

Компонент Context.Provider, доступний у екземплярі контексту, використовується для надання контексту його дочірнім компонентам, незалежно від того, наскільки вони глибокі.

Щоб встановити значення контексту, використовуйте властивість значення, доступну в <Context.Provider value={value} />:

Знову ж таки, тут важливо те, що всі компоненти, які пізніше захочуть використовувати контекст, повинні бути загорнуті всередину компонента провайдера.

Якщо ви хочете змінити значення контексту, просто оновіть властивість значення.

```
функція Головна() {
 конст значення="ЗНАЧЕННЯ КОНТЕКСТУ"
 повернення(
 <Context.Provider значення={значення}>
 <MyComponent />
 </Context.Provider>
);
}
```

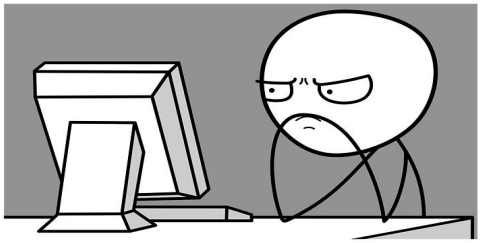
Споживання контексту

Споживання контексту можна виконати двома способами.

Перший спосіб полягає у використанні хука useContext(Context) React:

Хук повертає значення контексту: value = useContext(Context). Хук також забезпечує повторну візуалізацію компонента, коли значення контексту змінюється.

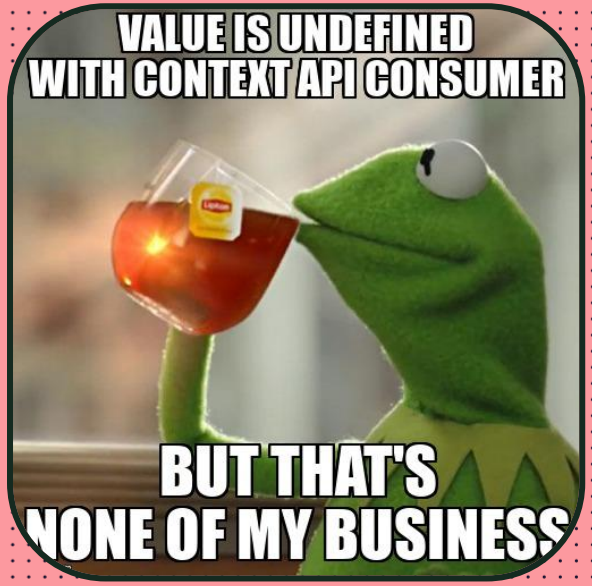
```
імпорт{useContext}від "реагувати";
функція MyComponent() {
 конст значення=useContext(Контекст);
 повернення <span>{значення}</span>;
}
```



Споживання контексту

Другий спосіб полягає у використанні функції рендерингу, наданої як дочірній компонент спеціального компонента Context.Consumer, доступного в екземплярі контексту:

```
функція MyComponent() {
 повернення(
 <Контекст. Споживач>
 {значення => <span>{значення}</span>}
 </Контекст. Споживач>
);
}
```

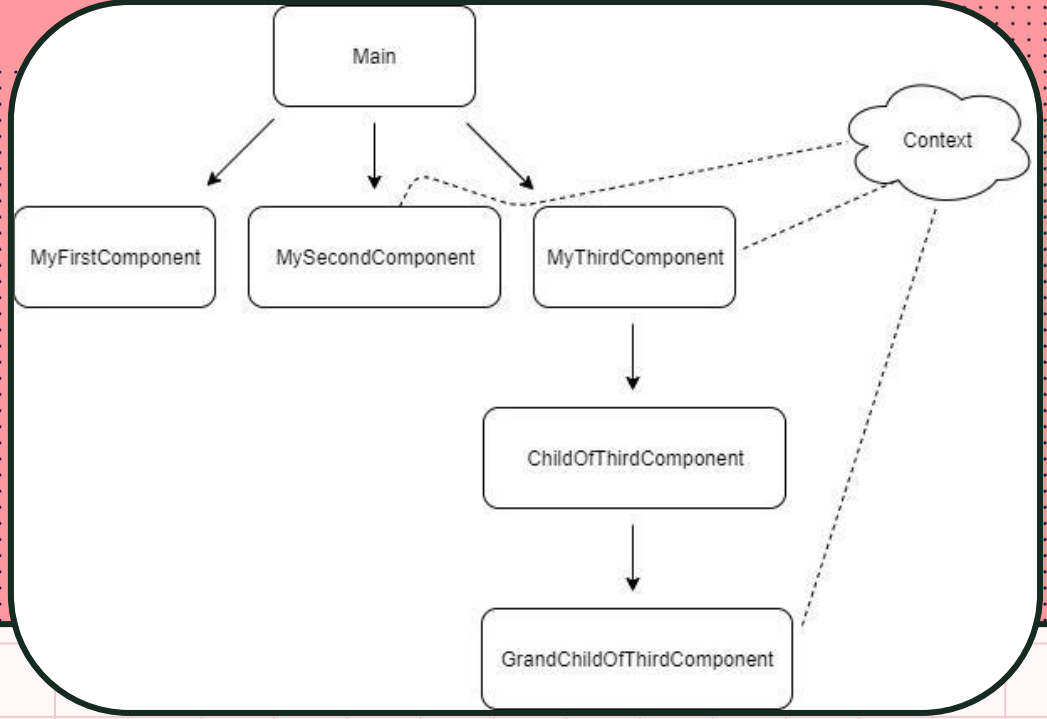


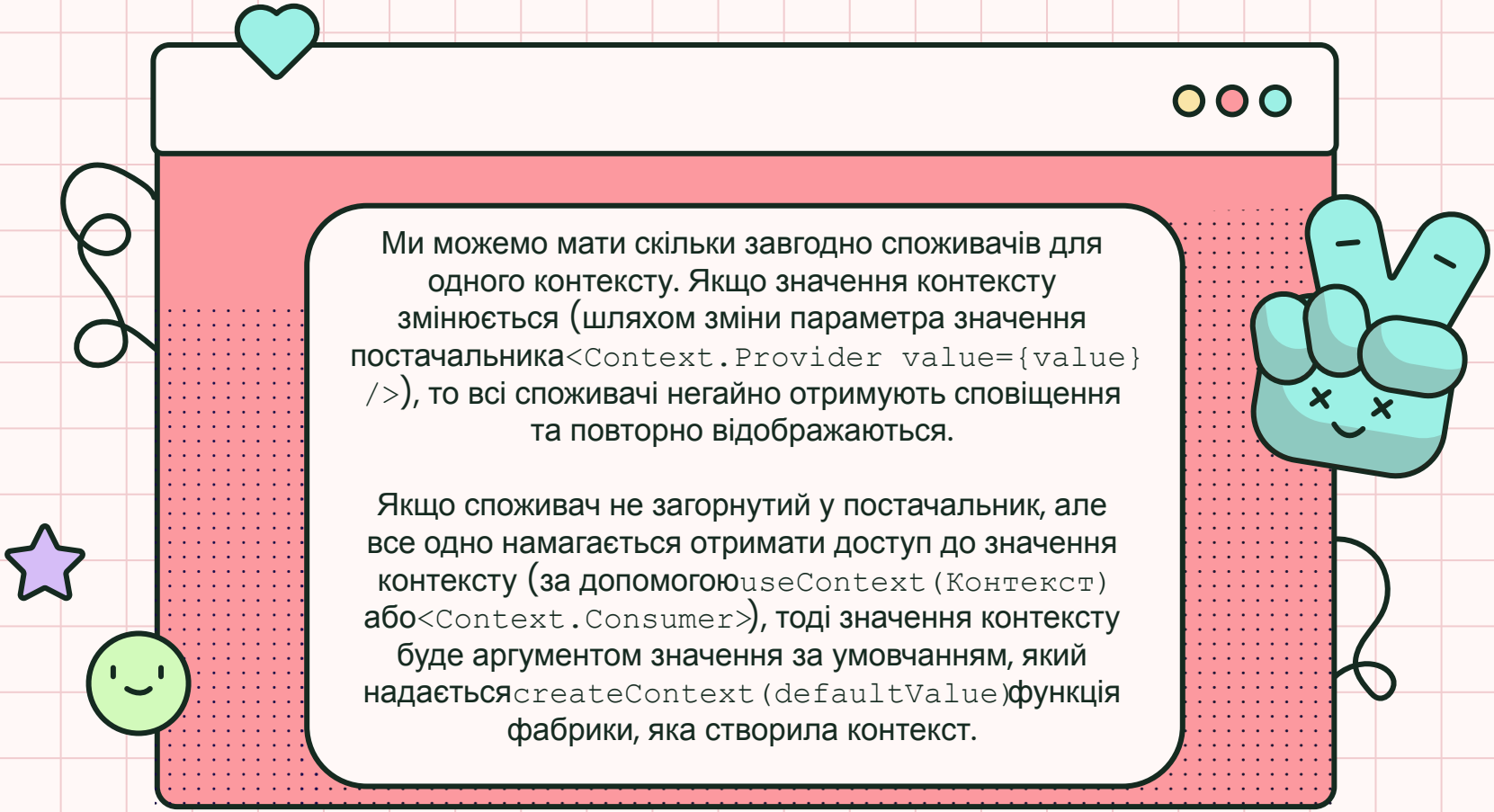
НАТИС

НАТИСНІТЬ

shorturl.at/CFQVZ

Реагувати на контекст





Ми можемо мати скільки завгодно споживачів для одного контексту. Якщо значення контексту змінюється (шляхом зміни параметра значення постачальника`<Context.Provider value={value} />`), то всі споживачі негайно отримують сповіщення та повторно відображаються.

Якщо споживач не загорнутий у постачальник, але все одно намагається отримати доступ до значення контексту (за допомогою`useContext(Контекст)` або`<Context.Consumer>`), тоді значення контексту буде аргументом значення за умовчанням, який надається`createContext(defaultValue)`функція фабрики, яка створила контекст.

Висновки

API контексту

Винахідливий і ідеальний для невеликих програм, де зміни стану мінімальні

Redux

Ідеально підходить для великих програм, де є високочастотні оновлення стану

Спасибі!

Оксана Кот
2022 рік