Q3

11.12.1400

—

Haniey Rastifar

تفاوت npm و npx

npm	npx
مدیریت پکیج پیش فرض node.js است و با نصب node نصب می شود و ابزاری است که برای نصب و مدیریت بسته ها استفاده می شود	ابزاری است که برای اجرا کردن پکیج ها استفاده می شود
می توانیم بسته ها را به صورت محلی یا سراسری نصب کنیم	نیازی به نصب بسته ها ندارد
با نصب Node.js نصب می شود	با نصب Node.js نصب می شود
هیچ بسته ای را اجرا نمی کند و اگر بخواهیم توسط ان بسته ای را اجرا کنیم باید ان بسته را داخل package.json در بخش script مشخص کنیم(البته قبل تر باید خود بسته نصب شده باشد)	برای npx هم می توان داخل package.json دستور در بخش script مشخص کرد و نیازی به نصب ان بسته نیست
اگر یک بسته را به صورت سراسری نصب کنید همیشه از همان ورژن بسته که نصب کرده اید استفاده خواهید کرد مگر انکه آن بسته را آپدیت کنید	از بسته ها در هر نسخه که بخواهید می توانید به صورت ازمایشی استفاده کنید بدون انکه نیازی به نصب انها داشته باشید

به عنوان مثال اگر ما دستور npx create-react-app my-app را بزنیم محیط مورد نیاز برای اجرای برنامه ریکت را ایجاد میکند بدون انکه نیاز داشته باشد که ریکت در سیستم ما نصب شده باشد ولی اگر این دستور را با npm بنویسیم کار نمی کند مگر انکه قبل تر ریکت را نصب

کرده باشیم ولی npx نیازی ندارد که create-react-app قبل تر نصب شده باشد تا از آن استفاده کند [1.2][1.1]

```
Npm i create-react-app -g
Npm create-react-app my-app
```

مفهوم state در ریکت

کامپوننت ها در ریکت یک شی پیش ساخته به نام State دارند و داخل این شی پراپرتی های مربوط به کامپوننت نگهداری می شود، اگر مقدار state تغییر کند، کامپوننت re-render می شود(یعنی خروجی بر اساس تغییر ایجاد شده تغییر می کند) . شی State داخل تابع سازنده مقدار دهی اولیه می شود و در طی برنامه به وسیله *this.state.propertyname* می توان به آن دسترسی داشت ولی برای تغییر مقدار ان باید از متد this.setState استفاده شود زیرا استفاده از این متد گارانتی می کند که کامپوننت از تغییر انجام شده مطلع است و متد render را فراخوانی می کند .[2.1]

به بیانی دیگر State شی ای از پراپرتی های قابل مشاهده است که در طی چرخه حیات کامپوننت ممکن است تغییر کنند

در زیر بررسی تفاوت state و prop آمده است

state	prop
شی ای متشکل از پراپرتی های قابل مشاهده است که داخل یک کامپوننت تعریف می شود و در طی حیات کامپوننت قابل تغییر است	غیر قابل تغییر هستند(بعد از مشخص شدن مقدار دیگر قادر به تغییر مقدار آن نیستیم)
برای استفاده از آن در functional کامپوننت ها باید از React Hooks و متدهای useState و ... استفاده شود	محدودیتی در استفاده از ان در انواع کامپوننت ها نداریم
توسط event Handler ها مدیریت می شود	به وسیله کامپوننت والد مقداردهی اولیه می شود

همواره تغییر مقدار State باید از طریق متد setState رخ دهد و نمی توان به صورت مستقیم مقدار ان را تغییر داد زیرا در این صورت تغییر رخ داده شده باعث فراخوانی متد Render و در نتیجه اعمال تغییر نمی شود

متد setState برای افزایش کارایی به صورت آسنکرون اجرا می شود، به همین دلیل گاهی مقدار فعلی state ممکن است خروجی مورد نظر را تولید نکند (در مواردی که مثلا مقدار State را بر اساس مقدار قبلی State اپدیت میکنیم که باید از prevState استفاده شود)

هر کدام از پراپرتی های داخل State می توانند به صورت مستقل اپدیت شوند و برای انها به صورت مستقل متد setState استفاده شود [2.2]

مفهوم component در ریکت

کامپوننت نماینده یک بخش از صفحه (UI) است و این بخش می توانند در مکان های مختلفی به دفعات استفاده شود(reusable= قابلیت استفاده مجدد کامپوننت ها به این معنا است که می

توانیم با پراپرتی ها مختلف کامپوننت خود را برای هر بخش سفارشی سازی کنیم .) و در نهایت تمام کامپوننت ها کنار هم قرار میگیرند تا برنامه اصلی را شکل دهند.

کامپوننت ها شبیه به توابع جاوا اسکریپت هستند و به عنوان خروجی html بر میگردانند

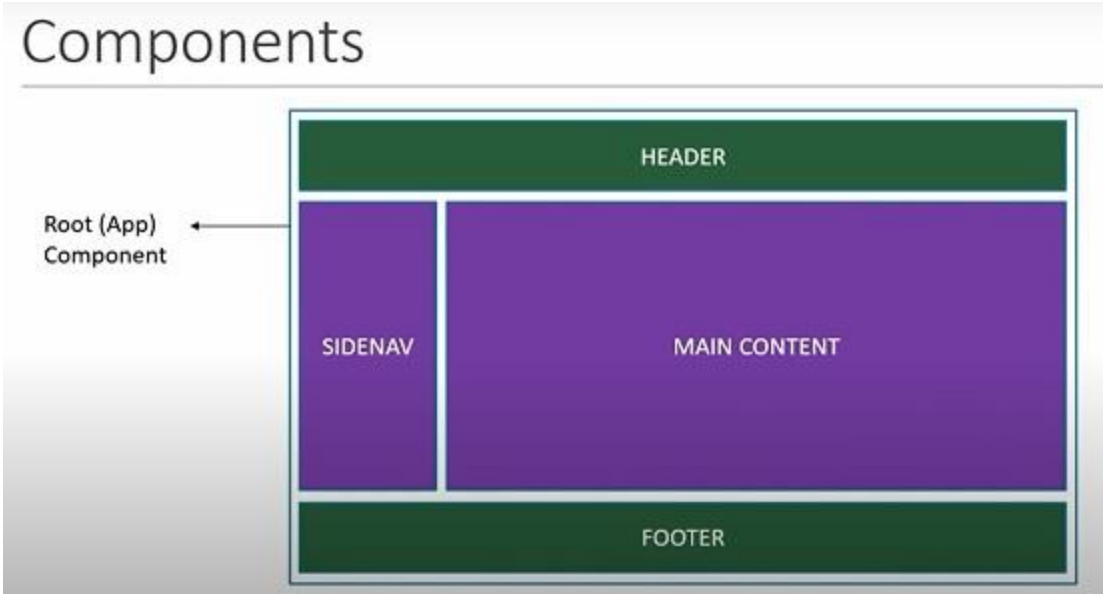
کامپوننت ها داخل فایل های جاوا اسکریپت با پسوند .js ایجاد می شوند .

2 نوع کامپوننت داریم :Functional component , class component

نام کامپوننت ها به صورت paskalcase نوشته می شود و دلیل این کار این است که انها را از تگ های html متمایز کنیم

در Functional تعداد خطوط کد کمتری داریم و مقادیر props توسط ارگومان های ورودی به داخل Functional کامپوننت ها ارسال می شود .

به عنوان مثال ، برنامه زیر 5 کامپوننت دارد :header,footer,sidenav,main و یک کامپوننت به عنوان container که ما معمولا ان را App(root component) نام گذاری میکنیم (= کامپوننت می تواند شامل سایر کامپوننت باشد)



بررسی نحوه اجرای قطعه کد زیر:

```
import React from 'react';
import ReactDOM from 'react-dom';

// This is a functional component
const Welcome=()=>
{
        return <h1>Hello World!</h1>
}

ReactDOM.render(
    <Welcome />,
    document.getElementById("root")
);
```

متد ReactDOM.render به عنوان اولین متد فراخوانی می شود سپس ریکت کامپوننت Welcome را فراخوانی میکند که یک h1 بر می گرداند و در نهایت به صورت کارا dom را اپدیت میکند تا h1 به ان اضافه شود و داخل تگ ای با id=root قرار بگیرد.[2.3][2.4]

چه مواردی باعث rerender شدن کامپوننت می شوند؟

1. آپدیت مقادیر State
2. آپدیت props (این کار باعث اپدیت در state می شود و در نهایت به rerender شدن می انجامد)
3. rerender شدن در کامپوننت والد باعث rerender شدن کامپوننت child هم می شود (در حالتی که کامپوننت ما در بر گیرنده کامپوننت دیگری است)[3]

منابع

[1.1] https://www.youtube.com/watch?v=SStviCTxFlw

[1.2]https://www.geeksforgeeks.org/what-are-the-differences-between-npm-and-npx/

[2.1] https://www.w3schools.com/react/react_state.asp

[2.2] https://www.geeksforgeeks.org/reactjs-state-react/

[2.3]https://www.w3schools.com/react/react_components.asp#:~:text=Components%20are%20independent%20and%20reusable,will%20concentrate%20on%20Function%20components.

[2.4] https://www.geeksforgeeks.org/reactjs-components/

[3] https://www.geeksforgeeks.org/re-rendering-components-in-reactjs/#:~:text=React%20components%20automatically%20re%2Drender,in%20their%20state%20or%20props.&text=A%20simple%20update%20of%20the,depends%20on%20some%20other%20data.