به نام خدا

بوتکمپ مکتب شریف ۶۶

نام نام‌خانوادگی : ساتنیک شاهی جانی زرنه

- **تفاوت npm و npx**

(NPM node package manager)یک dependency package manager است، که هنگام نصب نود جی اس، به اصطلاح با آن می‌توانید از جعبه خارج شوید npm.راهی است برای توسعه‌دهندگان تا بتوانند هم پکیج محلی (locally)و هم پکیج همگانی (globally)را نصب و استفاده کنند.

بعضی‌اوقات ممکن است بخواهید، نگاهی به یک پکیج خاص بیندازید و بعضی از دستورات آن را امتحان کنید،اما شما بدون نصب dependency های آن در پوشه node_modules ، نمی‌توانید این کار را انجام دهید.و اینجا، جایی است که npx به میدان می‌آید!

NPM

NPM از موارد مختلفی تشکیل شده، که اولین و مهم‌ترین آن، یک repository (مخزن)آنلاین برای انتشار پروژه‌های اپن‌سورس نودجی‌اس است.

دومین مورد ، npmیک ابزار CLI (command-line-interface) است، که به شما کمک می‌کند بسته‌هایی را که می‌خواهید ، نصب کنید و ورژن و dependency های آن‌را مدیریت کنید.صدها هزار کتابخانه و برنامه نودجی‌اس، درnpm وجود دارد، و هر روزه، تعداد زیاد دیگری نیز به آن اضافه می‌شود.

Npmبه خودی خود هیچ پکیجی را اجرا نمی‌کند، اگر می‌خواهید با استفاده از npm پکیجی را اجرا کنید، باید آن پکیج را در فایل package.json خود مشخص کنید.

مسیر پکیج‌های نصب شده توسط: npm

- پکیج‌های محلی (locally)در دایرکتوری /node_modules/.bin/. ایجاد می‌شوند.
- پکیج‌های همگانی (globally)در دایرکتوری /bin ایجاد می‌شوند ــ به طورمثال در لینوکس در مسیر /usr/local/bin و در ویندوز در مسیر %AppData%/npm قرار دارند.

برای اجرای یک پکیج با npm باید در مسیر محلی (local)آن‌را تایپ کنید، مانند این:

```
$ ./node_modules/.bin/your-package
```

یا می‌توانید با اضافه کردن بسته‌ی محلی نصب شده به package.json در بخش scriptآن‌را اجرا کنید، مثل این:

```
{
 "name": "your-application",
  "version": "1.0.0",
  "scripts": {
    "your-package": "your-package"
  }
}
```

سپس می‌توانید اسکریپت را با استفاده از دستور npm run اجرا کنید.

```
npm run your-package
```

Npx (package runner)

از نسخه npx ،5.2.0 npmبه همراه npm عرضه شد. بنابراین،امروزه تقریباً یک استاندارد است.

Npm همچنین یک ابزار CLI هم هست،که هدف آن نصب و مدیریت آسانdependency های موجود در ریجیستری npm است. و اکنون، اجرای هر نوع پکیج مبتنی بر Node.js که معمولاً با npm نصب می‌کردید، بسیار آسان‌تر شده است.

دستور زیر را اجرا کنید، تا متوجه شوید که آیا **npx** از قبل برای شما روی ورژن جاری **npm** نصب شده یا نه.

```
$which npx
```

اگر نصب نشده بود،به این روش می‌توانید آن را نصب کنید:

```
$npm install -g npx
```

حالا که مطمئن شدید آن‌را نصب کرده‌اید، بگذارید تا چند مورد از موارد استفاده **npx** که بسیار مفید است را شرح دهم.

به‌سادگی یک پکیج محلی نصب شده را اجرا کنید

اگر می‌خواهید یک پکیج نصب شده محلی را اجرا کنید، تمام کاری که باید انجام دهید،تایپ این دستور است.

```
$npx your-package
```

npx بررسی خواهد کرد که آیا < **command**> یا < **package**> در این **PATH$** یا در باینری‌های پروژه‌ی **local** وجود دارد یا نه، اگر وجود داشت آن را اجرا می‌کند.

پکیج‌هایی که قبلاً نصب نکرده‌ایدرا اجرا کنید!

یکی دیگر از مزیت‌های اصلی **npx** ،امکان اجرای پکیج‌هایی است که قبلاً نصب نشده‌اند.

بعضی اوقات، شما فقط می‌خواهید از برخی ابزارهای **CLI** استفاده کنید، اما نمی‌خواهید آن‌ها را به صورت **globally** نصب کنید، تا تنها آن‌ها را آزمایش کنید.این بدان معناست که می‌توانید فضای هارد خود را ذخیره کنید و فقط در صورت نیاز آن‌ها را اجرا کنید. همچنین متغییرهای **global** شما نیز کم‌تر آلوده می‌شوند!

کد را مستقیماً از **GitHub** اجرا کنید

این یکی خیلی جذابه!!

شما می‌توانید از npx برای اجرای هر gist یا repository گیت‌هاب استفاده کنید.

بیایید روی اجرای یک GitHub gist تمرکز کنیم، چراکه ایجاد کردن یکی از آن‌ها بسیار ساده است.

ابتدایی‌ترین اسکریپت، شامل فایل main js و package.json است.بعد از تنظیم این فایل‌ها،تمام کاری که باید انجام دهید، این است که npx را توسط یک لینک به آن gist ، متصل کرده و آن را اجرا کنید.

قبل از اجرای هر اسکریپت، حتماً آن را بخوانید تا از بروز مشکلات جدی، که به دلیل کدهای مخرب ممکن است رخ دهد، جلوگیری کنید.

ورژن‌های مختلف پکیج را تست کنید!

Npx تست ورژن‌های مختلف پکیج و ماژول‌های node.js را بسیار آسان می‌کند. برای آزمایش این ویژگی فوق‌العاده، قصد داریم پکیج create-react-app را به صورت locally نصب کنیم و نسخه آینده آن را تست کنیم.

این دستور، dist-tag ها را برای شما در خروجی نمایش می‌دهد. Dist-tag ها ، (alias نام مستعار) را برای شماره های ورژن ها ارائه می‌دهد، که تایپ آن را بسیار ساده‌تر می‌کند.

```
$npm v create-react-app
```

بیایید از npx برای امتحان کردن next dist-tag، به وسیله‌ی creat-reat-app که برنامه را درون یک sandbox directory ، ایجاد می‌کند، استفاده کنیم.

```
$npx create-react-app@next sandbox
```

npx، ورژن بعدی creat-reat-app را نصب خواهد کرد و سپس آن‌را برای (scaffoldبه معنی داربست است، اما در اینجا به معنای چارچوبی است که به سرعت برای یک برنامه تنظیم می‌شود – چارچوب)، برنامه اجرا خواهد کرد، و dependency های آن را نیز نصب می‌کند.

بعد از نصب می‌توانیم به صورت زیر به برنامه برویم:

```
$cd sandbox
```

و سپس با این دستور آن را اجرا می‌کنیم:

```
$npm start
```

این به صورت خودکار اپ ری‌اکت شما را، در مرورگر پیش‌فرض سیستم شما باز خواهد کرد.

و حالا، برنامه‌ای داریم که روی ورژن بعدی پکیج create-react-app اجرا می‌شود! صفحه index برنامه ری‌اکت شما، باید اینگونه باشد.

نتیجه

npx به شما کمک می‌کند تا از ورژن‌سازی ، مشکلات dependency ها و نصب پکیج‌های غیرضروری، که فقط می‌خواهیم آن‌ها را امتحان کنیم، جلوگیری کنیم.

---

npm	npx
If you wish to run package through npm then you have to specify that package in your package.json and installed it locally.	A package can be executable without installing the package, it is an npm package runner so if any packages that aren't already installed it will installed automatically.
To use create-react-app in npm the commands are npm install create-react-app then create-react-app myApp(Installation required).	But in npx you can use that without installing like npx create-react-app myApp, this command is required in every app's life cycle only once.
Npm is a tool that use to install packages.	Npx is a tool that use to execute packages.
Packages used by npm are installed globally you have to care about pollution for the long term.	Packages used by npx are not installed globally so you have to carefree for the pollution for the long term.

- https://roocket.ir/articles/npm-vs-npx-whats-the-difference
- https://www.geeksforgeeks.org/what-are-the-differences-between-npm-and-npx/

- **مفهوم state و component در react**

State

State یک شیء جاوا اسکریپت است که داده‌های دینامیک کامپوننت را نگه‌داری می‌کند و آن را قادر می‌سازد که تغییرات بین رندرها را ردگیری کند. از آنجا که State دینامیک است، تنها به منظور ایجاد تعامل‌پذیری استفاده می‌شود و از این رو در مورد پروژه‌های استاتیک React کاربردی ندارد.

کامپوننت‌ها به صورت کلاس‌هایی تعریف می‌شوند که ویژگی‌های اضافی دارند. State محلی دقیقاً به معنای یک ویژگی است که تنها در کلاس‌ها وجود دارد. State صرفاً درون کلاس می‌تواند مورد استفاده قرار گیرد و معمولاً تنها جایی است که می‌توان this.state را به عنوان سازنده مطرح کرد.

امروزه می‌توان از State بدون سازنده و از طریق «مقداردهی اولیه مشخصه» (Property initializers)استفاده کرد که ویژگی جدیدی محسوب می‌شود.

Stateدرون کامپوننت مدیریت می‌شود، همان طور که متغیرها درون یک تابع اعلان می‌شوند .

Stateدر کامپوننت React در واقع حالت محلی خودش است، یعنی حالت نمی‌تواند خارج از کامپوننت مورد دسترسی یا تغییر قرار گیرد و تنها درون آن کاربرد دارد. این وضعیت شبیه تابعی است که حیطه محلی خود را دارد.

آبجکت state شبیه props است با این تفاوت که کاملا در کامپوننت خود ایزوله شده و محفوظ است و در حالت عادی نمی توان از کامپوننت های دیگر به state های کامپوننت دسترسی داشت. مگر اینکه state را داخل props بریزیم و props را به کامپوننت دوم پاس دهیم.

setStateچه نقشی دارد؟

setState()یک به‌روزرسانی برای شیء state کامپوننت را زمان‌بندی می‌کند. زمانی که حالت تغییر یابد، کامپوننت از طریق رندرگیری مجدد پاسخ می‌دهد.

استفاده صحیح از State

- **Stateرا به صورت مستقیم دستکاری نکنید.**
- **به‌روزرسانی‌های State می‌توانند ناهمگام باشند.**
- **به‌روزرسانی‌های State ادغام می‌شوند.**

- https://react.sayjeyhi.com/#1e3642c6b1712d47bc959fae41b2ce87
- https://blog.faradars.org/what-is-state-in-react/

Component

کامپوننت componentها در React قطعات UI مستقل و قابل استفاده مجدد هستند. یک صفحه وب معمولی ممکن است از یکnavbar، قسمت محتوا و footerتشکیل شده باشد. در React ، ما این قسمت ها را با components ها ایجاد می کنیم. این باعث صرفه جویی در تکثیر کد می شود و همانطور که خواهیم دید ، امکان انعطاف پذیری بی نظیری را فراهم می کند..

روش دیگر برای فکر کردن به کامپوننت components ها اینست که آنها مانند توابع JavaScript هستند. به جای دریافت argument ، آنها "props" را دریافت می کنند ، و سپس عناصر React را برمی گردانند تا آنچه را که روی صفحه می بینیم بسازند.!

در واقع ، در React همه چیز کامپوننت component ها هستند ! حتی تگ هایHTML کامپوننت ها قطعاتی هستند که میتوان از آن ها هرجایی در پروژه استفاده کرد و در اصطلاح reusable هستند.

در ری اکت دو نوع کامپوننت میتوان نوشت ، یکی به شکل تابع و دیگری به شکل کلاس (class components , functional components)

Component می تواند به روش های مختلفی تعریف شود. می تواند یک class با متد render باشد. یا در حالات ساده یک کامپوننت می تواند بصورت یک تابع تعریف شود.

در هر دو حالت، کامپوننت می تواند props را بعنوان ورودی بگیرد و بعنوان خروجی یک درخت JSX برگرداند. JSX در مرحله بعد به تابع createElement ترنسپایل می شود.

فرق بین Element و Component چیست؟

یک المنت یک آبجکت ساده (plain object)است که بیانگر چیزی است که قرار است بعنوان یک گره (nodeدر DOM نمایش یابد. یک element می تواند شامل المنت های دیگر بعنوان props خود باشند. تابع createElement مطابق ساختار زیر تعریف می شود.

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.
Components come in two types, Class components and Function components

ReactJS Components

A **Component** is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier. You can see a UI broken down into multiple individual pieces called components and work on them independently and merge them all in a parent component which will be your final UI.

Google's custom search at the top can be seen as an individual component, the navigation bar can be seen as an individual component, the sidebar is an individual component, the list of articles or post is also an individual component and finally, we can merge all of these individual components to make a parent component which will be the final UI for the homepage.
Components in React basically return a piece of JSX code that tells what should be rendered on the screen. In React, we mainly have two types of components:

1. **Functional Components**: Functional components are simply javascript functions. We can create a functional component in React by writing a javascript function. These functions may or may not receive data as parameters, we will discuss this later in the tutorial. Below example shows a valid functional component in React:

```
const Democomponent=()=>
{
    return <h1>Welcome Message!</h1>;
}
```

5. **Class Components**: The class components are a little more complex than the functional components. The functional components are not aware of the other components in your program whereas the class components can work with each other. We can pass data from one class component to other class components. We can use JavaScript ES6 classes to create class-based components in React. Below example shows a valid class-based component in React:

```
class Democomponent extends React.Component
{
```

```
    render(){
          return <h1>Welcome Message!</h1>;
    }
}
```

The components we created in the above two examples are equivalent, and we also have stated the basic difference between a functional component and class component. We will learn about more properties of class-based components in further tutorials. For now, keep in mind that we will use functional component only when we are sure that our component does not require interacting or work with any other component. That is, these components do not require data from other components however we can compose multiple functional components under a single functional component. We can also use class-based components for this purpose but it is not recommended as using class-based components without need will make your application in-efficient.
In this post, we will mainly write functional components to make things easier to understand. We will discuss class-based components in detail later in the tutorial.

- https://errorweb.ir/%DA%A9%D8%A7%D9%85%D9%BE%D9%88%D9%86%D9%86%D8%AA-component-%D9%87%D8%A7-%D9%88-%D9%BE%D8%B1%D8%A7%D9%BE%D8%B3-props-%D8%AF%D8%B1-%D8%B1%DB%8C-%D8%A7%DA%A9%D8%AA-react/
- https://www.geeksforgeeks.org/reactjs-components/
- https://www.w3schools.com/react/react_components.asp
- https://virgool.io/iran-react-community/%DA%A9%D8%A7%D9%85%D9%BE%D9%88%D9%86%D9%86%D8%AA-%D9%87%D8%A7-%D8%AF%D8%B1-%D8%B1%DB%8C-%D8%A7%DA%A9%D8%AA-gxatvwucn1em
- https://blog.faradars.org/a-complete-beginners-guide-to-react/#%DA%A9%D8%A7%D9%85%D9%BE%D9%88%D9%86%D9%86%D8%AA_%D8%B1%DB%8C_%D8%A7%DA%A9%D8%AA
- https://sabzlearn.ir/component-in-react-js/
- https://react.sayjeyhi.com/#1e3642c6b1712d47bc959fae41b2ce87

- **چه مواردی باعث re-render شدن کامپوننت میشود؟**

وقتی از ری‌اکت استفاده میکنیم، به جایی میرسیم که متد render رو یه تابع میبینیم که درخت المنت های ری‌اکت رو میسازه. با آپدیت state یا prop تابع render یه درخت متفاوت از المنت ها بر میگردونه. ری‌اکت باید بفهمه بهینه ترین راه برای آپدیت رابط کاربری چیه تا با درخت جدید همخوانی داشته باشه.

الگوریتم های عمومی برای حل مشکل تبدیل یک درخت، به درخت دیگه وجود دارن، اما اردرشون $O(3n)$، که n تعداد المنت های درخته.

اگه ری‌اکت از این الگوریتم ها استفاده میکرد، نشان دادن ۱۰۰۰ المنت، یک میلیارد مقایسه لازم بود، که خیلی هزینه زیادی میشد. بجاش ری‌اکت از الکوریتمی با اردر $O(n)$ استفاده میکنه که بر پایه دو فرضه: اول اینکه دو المنت از دو نوع متفاوت، درخت متفاوتی تولید میکنن. دوم دولوپر میتونه با *key* بگه کدوم المنت فرزند ممکنه ثابت بمونه.

Re-rendering Components in ReactJS

As we know React JS is an open source JavaScript library that is used for building user interfaces specifically for single-page applications. It is also known for providing fast user experience by only updating the parts of the UI that have changed. Rendering components is not in user hands, it is a part of React Component Lifecycle and is called by React at various app stages, generally when the React Component is first instantiated. A second or subsequent render to update the state is called as **re-rendering**. React components automatically re-render whenever there is a change in their state or props.
Prerequisites: Introduction to React JS, React JS | Lifecyle of Components
A simple update of the state, from anywhere in the code, causes all the User Interface (UI) elements to be re-rendered automatically. However, there may be cases where the render() method depends on some other data. Re-render can be caused due to any of the three reasons listed:

1. Update in State
2. Update in prop
3. Re-rendering of the parent component

Unnecessary re-renders affect the app performance and cause loss of users' battery which surely no user would want. Let's see in detail why components get re-rendered and how to prevent unwanted re-rendering to optimize app components.

Why do components get re-rendered?
Let's have a deeper look at the three causes of re-rendering mentioned before.

1. **Update in state:** The state change can be from a *prop* or *setState* change to update a variable(say). The component gets the updated state and React re-renders the component to reflect the change on the app.

2. **Update in prop:** Likewise the change in *prop* leads to state change and state change leads to re-rendering of the component by React.
3. **Re-rendering of parent component:** Whenever the components render function is called, all its subsequent child components will re-render, regardless of whether their *props* have changed or not.

React schedules a render every time state changes (scheduling a render doesn't mean this happens immediately, this might take time and be done at the best moment). Changing a state means React triggers an update when we call the useState function (useState is a Hook that allows you to have state variables in functional components).

- **https://www.geeksforgeeks.org/re-rendering-components-in-reactjs/**
- **https://virgool.io/@bamdad/react-rerender-ls4fwh1oe5ws**