禪

با React یک اپلیکیشن کوچک کاربردی بسازید

برنامهنویسی با React برای افراد مبتدی

با وجود تمام امکاناتی که React در اختیار ما میگذارد، شروع کار با آن فرآیند چندان راحتی نیست. یادگیری این زبان برنامهنویسی یک منحنی پرپیچوخم از موانع بزرگ و کوچک است. در این مقاله سعی ما بر این است که ابتداییترین مراحل ساخت یک اپلیکیشن ساده React را به شما آموزش دهیم تا بتوانید علاوه بر تولید یک اپلیکیشن کوچک کاربردی این آمادگی را پیدا کنید که سراغ قابلیتهای پیشرفتهتر و نکات فنی عمیقتر در React بروید.

شروع کار با **JSX**

قبل از شروع ساخت اپلیکیشن توجه به این نکته مهم الزامی است: **React** شبیه بسیاری از کتابخانههای جاوا اسکریپتی که تابهحال از آن استفاده کردهاید، اینگونه نیست که شما بتوانید با استفاده از تگ Script به کدی که نوشتهاید، دسترسی پیدا کنید. **React** به شکل آزاردهندهای روش مخصوص به خود را دارد و انجام آن بر اساس نوع ساخت اپلیکیشنهای **React** تعیین میشود.

همانطور که میدانید، اپلیکیشنهای وب (و هر چیز دیگری که مرورگر نمایش میدهد) براساس زبانهای اسکریپتنویسی HTML, CSS و JavaScript ساخته میشوند.

LOREMIPSUM
=
HTML, CSS, and JavaScript
Totally sweet web app!

اینکه اپلیکیشن وب شما با استفاده از **React** یا کتابخانههای دیگری مانند Angular, Knockout یا JQuery نوشتهشده باشد، تفاوت زیادی ندارد، نکته اصلی این است که نتیجه کار باید ترکیبی از HTML, CSS و JavaScript باشد. در غیر این صورت، مرورگر نمیداند چهکاری باید انجام دهد .

اینجا همانجایی است که **React** ماهیت خاص خود را نشان میدهد. در کنار HTML, CSS و JavaScript عادی، مجموعه کدهای **React** در JSX نوشته خواهد شد. JSX زبانی است که به شما اجازه میدهد تا بهسادگی جاوا اسکریپت و تگهای HTML را برای مشخص کردن عناصر رابط کاربری و کاربرد آنها با یکدیگر ترکیب کنید

تا اینجا که همهچیز خوب به نظر میرسد؛ اما یک مشکل جزیی وجود دارد : مرورگر شما نمیداند با JSX چهکاری باید انجام دهد .

برای ساخت یک اپلیکیشن وب با استفاده از **React**، به روشی نیاز دارید تا JSX خود را به همان جاوا اسکریپت متنی قدیمی که مرورگر شما قادر به فهم آن است، تبدیل کنید .

LOREMIPSUM
?
React (JSX)
Web App

اگر این کار را انجام ندهید، اپلیکیشن **React** شما کار نخواهد کرد . خوشبختانه دو راهحل برای آن وجود دارد :

1. یک محیط توسعه حول Node و مجموعه ابزارهای کاربردی ساخت راهاندازی کنید : در این محیط هر زمان که به ساخت اقدام میکنید تمام JSX بهطور خودکار به جاوا اسکریپت تبدیلشده و برای ارجاع به آن در آینده مانند یک فایل جاوا اسکریپت متنی روی دیسک قرار میگیرد .

2. به مرورگر خود اجازه دهید در زمان اجرا بهطور خودکار JSX را به جاوا اسکریپت تبدیل کند : در این شیوه JSX خود را مستقیم به همان شیوه قدیمی انتخاب جاوا اسکریپت تعیین میکنید و باقی کار را به مرورگر خود میسپارید .

هر دو راهکار قابل انجام هستند، اما اجازه دهید تاثیر هر کدام را بررسی کنیم .

راهکار اول، ممکن است در ابتدا کمی پیچیده و زمانبر به نظر برسد، اما روشی است که این روزها توسعهدهندگان وب مدرن انجام میدهند . در کنار کامپایل JSX به جاوا اسکریپت، چنین کاری شما را قادر میسازد تا از مزایای ماژولها، ابزارهای ساخت بهتر و مجموعه دیگری از قابلیتهایی که امکان ساخت اپلیکیشنهای پیچیده وب را فراهم میکند، بهرهمند شوید .

راهکار دوم، مسیر سرراست و سریعی در اختیارتان قرار میدهد تا بتوانید بهجای درگیر شدن با محیط توسعه بیشتر وقت خود را صرف نوشتن کدها کنید. برای استفاده از این راهکار، باید به یک فایل اسکریپت مراجعه کنید . این فایل اسکریپت وظیفه تبدیل JSX به جاوا اسکریپت را در صفحه بارگیری

انجام میدهد و اپلیکیشن **React** شما بدون نیاز به انجام کار خاصی در محیط توسعه اجرا میشود . در آموزش مقدمات **React** ما از راهکار دوم استفاده خواهیم کرد . شاید برای شما جای سوال باشد که چرا همیشه نباید از این راهکار استفاده کنید . دلیل آن این است که مرورگرتان هر زمان که میخواهد JSX را به جاوا اسکریپت تبدیل کند، زمان زیادی را برای اجرای آن صرف میکند . چنین مسئلهای در زمان یادگیری استفاده از **React** قابلقبول است، اما برای پیادهسازی یک اپلیکیشن در کاربردهای واقعی توجیهپذیر نخواهد بود .

شروع کار با **React**

در ابتدای کار شما به یک صفحه HTML خالی برای شروع کار نیاز دارید؛ بنابراین یک سند HTML با محتوای زیر ایجاد کنید :

```
<!DOCTYPE html>
<html>

<head>
 <meta charset="utf-8">
 <title>React! React! React!</title>
</head>

<body>
 <script>

 </script>
</body>

</html>
```

در این مرحله هیچچیز فراوانی در این صفحه وجود ندارد . حالا اجازه دهید یک ارجاع به کتابخانه **React** به این صفحه اضافه کنیم . درست در زیر تگ title این دو خط را اضافه کنید :

```
<script src="https://unpkg.com/react@16/umd/react.development.js">
</script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>
```

این دو خط هم کتابخانه اصلی **React** و هم چیزهای مختلف دیگری را که **React** برای کار با DOM به آنها نیاز دارد، فراخوانی میکند . بدون آنها شما اصلا نمیتوانید یک اپلیکیشن **React** بسازید .
در مرحله بعد باید یک کتابخانه دیگر را فراخوانی کنید و دو تگ Script قبلی این خط را هم اضافه کنید :

```
<script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js">
</script>
```

در اینجا شما یک ارجاع به کامپایلر Babel JavaScript را به صفحه افزودید .
Babel کارهای بسیار جالبی انجام میدهد، اما کاربردی که برای ما اهمیت دارد توانایی تبدیل JSX به جاوا اسکریپت توسط این کامپایلر است .
در این مرحله صفحه HTML شما باید شبیه به این شده باشد :

```
<!DOCTYPE html> <html>
<head>
 <meta charset="utf-8">
 <title>React! React! React!</title>
```

```
 <script src=”https://unpkg.com/react@16/umd/react.development.js”>
 </script>
 <script src=”https://unpkg.com/react-dom@16/umd/react-dom.development.js”>
 </script>
 <script src=”https://unpkg.com/babel-standalone@6.15.0/babel.min.js”>
 </script>
</head>

<body>
 <script>

 </script>
</body>

</html>
```

اگر الان از این صفحه پیشنمایش بگیرید، متوجه خواهید شد که این صفحه همچنان خالی است و چیزی در آن قابلمشاهده نیست و هنوز کارهای زیاد دیگری برای انجام دادن دارید .

نمایش نام شما با استفاده از React

در این مرحله قصد داریم یک نام را با استفاده از **React** روی صفحهنمایش دهیم. این کار را با استفاده از یک متد به نام Render انجام میدهید . داخل تگ خالی Script خود که در تگ Body قرار دارد، کد زیر را اضافه کنید :

```
ReactDOM.render(
 <h1>Sherlock Holmes</h1>,
  document.body
);
```

اگر این تکه از کدهای نوشتهشده برایتان نامفهوم است جای نگرانی نیست . ابتدا هدف این است که چیزی روی نمایشگر نمایش داده شود، اما قبل از اینکه چیزی را روی صفحه مشاهده کنید، باید مشخص کنید این تکه از Script توسط Babel قابلپردازش باشد . این کار را توسط مشخصه Type در تگ Script و با مقدار text/babel انجام دهید :

```
<script type=”text/babel”>
 ReactDOM.render(
  <h1>Sherlock Holmes</h1>,
  document.body
 );
</script>
```

بعد از اعمال این تغییرات فایل را روی مرورگر خود باز کنید . همانطور که در شکل 3 مشخص است شما نام Sherlock Holmes را مشاهده خواهید کرد .
حالا شما موفق به ساخت یک اپلیکیشن بسیار ساده با استفاده از **React** شدهاید. هر چند این اپلیکیشن در عمل کار خاصی انجام نمیدهد، اما شما را با یکی از پرکاربردترین متدهای موجود در **React** یعنی ReactDOM.render آشنا کرده است. متد Render دو پارامتر را بهعنوان ورودی میپذیرد :

- عناصر HTML که شما بهعنوان خروجی به آنها نیاز دارید .
- موقعیت مکانی در DOM که **React** محتوای JSX را در آن قرار میدهد .

ما از متد Render به این شکل استفاده کردیم :

```
ReactDOM.render(
 <h1>Sherlock Holmes</h1>,
 document.body
);
```

اولین پارامتر متن Sherlock Holmes است که داخل یک تگ H1 قرارگرفته است . این بخش از فرمان HTML موجود در جاوا اسکریپت توسط JSX پردازش میشود. در JSX تنها کافی است محتوای HTML خود را به همان شکلی که است، در کد اصلی خود قرار دهید.

دومین پارامتر متن document.body است. این پارامتر مشخص میکند که کدهای پردازششده توسط JSX باید در کجای DOM قرار گیرد . در این مثال، وقتی متد Render اجرا میشود تگ H1 و هر چیزی که داخل آن قرار دارد، درون عنصر Body سند ما جاگذاری میشود .
در مثال هدف این بود که نام شما در صفحهنمایش ظاهر شود. برای این کار کافی است پارامتر اول متد Render را بانام موردنظر خود تغییر دهید :

```
ReactDOM.render(
 <h1>Batman</h1>,
 document.body
);
```

با فرض اینکه نام موردنظر شما Batman بوده، بعد از ذخیره و باز کردن دوباره این فایل در مرورگر مشاهده خواهید کرد که با نام Sherlock Holmes جایگزین شده است.
به لطف پردازش JSX کدنویسی جاوا اسکریپت به نظر ساده و قابلفهم میرسد . اما نباید این نکته را فراموش کنید که درنهایت مرورگر شما تنها HTML, CSS و JavaScript را مشاهده خواهد کرد . برای اینکه این موضوع بیشتر قابللمس باشد، اجازه دهید، وضعیت عملکرد و ظاهر این اپلیکیشن را کمی تغییر دهیم .

مطلب پیشنهادی

هفت جمله برای
عصبانی کردن
برنامه نویسان

برنامهنویسان از شنیدن چه جملاتی بیزار هستند؟

کدنویسان با شنیدن این هفت جمله به سرعت_عصبانی_میشوند

تغییر مقصد

ابتدا جایی را که قرار است خروجی JSX در آن قرار گیرد، تغییر میدهیم . استفاده از جاوا اسکریپت برای قرار دادن مستقیم محتوا در عنصر Body بههیچوجه ایده خوبی نیست. اشکالات زیادی ممکن است در این شیوه به وجود آید، بهویژه اینکه اگر قصد داشته باشید **React** را با سایر کتابخانهها و فریمورکهای جاوا اسکریپت ترکیب کنید . روش توصیهشده این است که یک عنصر جداگانه ایجاد کرده و با آن بهعنوان یک عنصر ریشه جدید رفتار کنید . این عنصر بهعنوان مقصدی که متد Render از آن استفاده خواهد کرد، عمل میکند . برای انجام این کار به فایل HTML خود برگردید و یک عنصر div با یک id با مقدار Container به آن اضافه کنید :

```
<body>
<div id=”container”></div>
```

```
 <script type=”text/babel”>
  ReactDOM.render(
   <h1>Batman</h1>,
   document.body
  );
 </script>
</body>
```

حالا باید متد Render را اصلاح کنید تا بهجای Document.body از این Div استفاده کند . یکی از روشهای انجام کار به این شکل است :

```
ReactDOM.render(
 <h1>Batman</h1>,
 document.querySelector(“#container”)
);
```

گزینه دیگر این است که کاری خارج از خود متد Render انجام دهید :

```
vardestination = document.querySelector(“#container”);

ReactDOM.render(
 <h1>Batman</h1>,
 destination
);
```

در اینجا متغیر Destination محتوای عنصر DOM با شناسه Container را در خود ذخیره میکند . کافی است، داخل متد Render این متغیر را فراخوانی کنید . هدف از انجام این کار این است که به شما نشان دهیم، همچنان از کدنویسی جاوا اسکریپت استفاده میکنید و متد Render نیز همچنان دو پارامتر بهعنوان ورودی دریافت میکند .

طراحی ظاهری

زمان آن رسیده تا قبل از بارگذاری صفحه در مرورگر، کمی تغییرات ظاهری روی آن انجام دهیم . در حال حاضر، نام ما تنها به شکل یک هدر ساده نمایش داده میشود . برای افزایش جلوههای بصری به متن خود ما باید از CSS کمک بگیریم. داخل تگ Head یک بلوک Style به همراه CSS□□□ زیر را اضافه کنید :

```
<style>
 #container {
  padding: 50px;
  background-color: #EEE;
 }
 #container h1 {
  font-size: 144px;
  font-family: sans-serif;
  color: #0080A8;
 }
</style>
```

بعد از ذخیره فایل آن را روی مرورگر اجرا کنید . در اینجا مشاهده خواهید کرد، ظاهر نوشته شما با فرامین CSS تغییر پیدا کرده است .



Batman

فرآیند کار به این صورت است که بعد از اجراشدن تمام کدهای **React** محتوای Body در DOM شما شامل یک عنصر با شناسه Container است که داخل آن نیز یک تگ H1 قرار دارد . در اینجا مهم نیست که تگ H1 بهطور کامل داخل جاوا اسکریپت در این فرمان JSX تعریفشده باشد
یا اینکه CSS شما خارج از متد Render ساختهشده باشد. نتیجه اپلیکیشن **React** شما همچنان کاملا براساس ساختار HTML, CSS و JavaScript شکل میگیرد. در نهایت خروجی سند شما چیزی شبیه این خواهد بود :

```
<!DOCTYPE html>
<html>

<head>
 <meta charset=”utf-8”>
 <title>React! React! React!</title>
 <script src=”https://unpkg.com/react@16/umd/react.development.js”>
</script>
 <script src=”https://unpkg.com/react-dom@16/umd/react-dom.development.js”>
</script>
 <script src=”https://unpkg.com/babel-standalone@6.15.0/babel.min.js”>
</script>

 <style>
 #container {
  padding: 50px;
  background-color: #EEE;
 }
 #container h1 {
  font-size: 144px;
  font-family: sans-serif;
  color: #0080A8;
 }
 </style>
</head>

<body>
<div id=”container”></div>
 <script type=”text/babel”>
   var destination = document.querySelector(“#container”);

  ReactDOM.render(React.createElement(
   “h1”,
   null,
   “Batman”
  ), destination);
 </script>
</body>
```

`</html>`

مطلب پیشنهادی

دلایل و راهکارهای برونرفت از مشکل تاخیر

چرا برنامههای نوشته شده بسیار کند اجرا میشوند

نتیجهگیری

مهمترین نکتهای که در مورد **React** باید به آن توجه داشته باشید، اختلاف آن نسبت به سایر کتابخانهها است، زیرا کاملا از یکزبان جدید به نام JSX برای مشخص کردن ظواهر بصری استفاده میکند. تاثیرات JSX فراتر از تعیین عناصر رابط کاربری شما است . این زبان ساخت تمام اپلیکیشن شما را بر عهده دارد . ازآنجاکه مرورگر شما قادر به درک زبان JSX نیست، به یک مرحله میانی برای تبدیل JSX به JavaScript احتیاج دارید . یک راهکار این است که شما اپلیکیشن خود را به شکلی بسازید که یک خروجی جاوا اسکریپت مطابق با منبع JSX تولید کند . راهکار دوم (که ما در این مقاله از آن استفاده کردیم) استفاده از کتابخانه Babel برای تبدیل JSX به JavaScript در خود مرورگر است. ازآنجاکه در این شیوه عملکرد کلی تحت تاثیر قرار میگیرد، استفاده از آن برای اپلیکیشنهای پیچیده پیشنهاد نمیشود .

نویسنده :
محسن آقاجانی
دسته بندی:
کارگاه
برنامه نویسی
تاریخ انتشار :
07:40 - 24/10/1397
برچسب:
زبان های اسکریپت نویسی - React - برنامه نویسی React - شروع React - آموزش کار با REACT - آموزش برنامه نویسی - برنامه نویسی

نشانی
□□□□: https://www.shabakeh-mag.com/workshop/14343/%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87%E2%80%8C%D9%86%D9%88%DB%8C%D8%B3%DB%8C-%D8%A8%D8%A7-react-%D8%A8%D8%B1%D8%A7%DB%8C-%D8%A7%D9%81%D8%B1%D8%A7%D8%AF-%D9%85%D8%A8%D8%AA%D8%AF%DB%8C