آموزش

React Router

ساگار گناترا

ویرایش نخست

انتشارات پندار پارس

سید منصور عمرانی

سرشناسه	:	گناترا، ساگار Ganatra, Sagar
عنوان و نام پدیدآور	:	آموزش React Router/ ساگار گناترا؛ ترجمه سیدمنصور عمرانی.
مشخصات نشر	:	تهران : پندار پارس ، ۱۳۹۹.
مشخصات ظاهری	:	۱۲۶ ص.
شابک	:	978-600-8201-88-5
وضعیت فهرست نویسی	:	فیپا
یادداشت	:	عنوان اصلی: React Router quick start guide : routing in React applications made easy, 2018.
موضوع	:	نرم‌افزار کاربردی — طراحی و توسعه
موضوع	:	Application software -- Development
موضوع	:	برنامه‌های کاربردی وب
موضوع	:	Web applications
موضوع	:	جاوا اسکریپت (زبان برنامه‌نویسی کامپیوتر)
موضوع	:	JavaScript (Computer program language)
شناسه افزوده	:	عمرانی، سیدمنصور، ۱۳۵۶ -، مترجم
رده بندی کنگره	:	QA۷۶/۷۶
رده بندی دیویی	:	۳/۰۰۵
شماره کتابشناسی ملی	:	۶۱۶۴۷۰۲

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ **www.pendarepars.com**

تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸همراه: ۰۹۱۲۲۴۵۲۳۴۸ **info@pendarepars.com**

نام کتاب	:	**آموزش React Router**
ناشر	:	**انتشارات پندار پارس**
تالیف	:	**ساگار گاناترا (Sagar Ganatra)**
ترجمه	:	**سید منصور عمرانی**
چاپ نخست	:	**خرداد ۹۹**
شمارگان	:	**۲۰۰ نسخه**
طرح جلد	:	**رامین شکرالهی**
چاپ، صحافی	:	**روز**
قیمت	:	**۴۲۰۰۰ تومان**

شابک : ۵-۸۸-۸۲۰۱-۶۰۰-۹۷۸

فهرست

درباره‌ی نویسنده ۷
درباره‌ی ویراستار فنی ۷
پیشگفتار ۸
این کتاب برای چه کسی است؟ ۸
این کتاب چه چیزهایی را پوشش می‌دهد؟ ۸
پیش فرض‌های کتاب ۹
دانلود کُدهای پیوست کتاب ۹
دانلود تصاویر رنگی کتاب ۱۰
تماس با ما ۱۰
بازبینی ۱۰
فصل ۱. معرفی React Router و ایجاد نخستین مسیر برنامه ۱۱
مرور سریعی بر React ۱۲
معماری کمپوننت-محور در React ۱۳
ایجاد یک کمپونت React ۱۵
معرفی React-Router ۱۶
شروع کار با React-Router ۱۷
افزودن کتابخانه‌ی React-Router ۲۰
تعریف مسیر برای برنامه ۲۰
خلاصه ۲۳
فصل ۲. پیکربندی مسیرهای برنامه با استفاده از کمپونت Route ۲۵
خصوصیت‌های کمپونت Route ۲۵
خصوصیت exact ۲۶
خصوصیت strict ۲۷
خصوصیت sensitive ۲۷
رندر کردن درون خطی با استفاده از خصوصیت render ۲۸
رندر کردن درون خطی با استفاده از خصوصیت children ۲۸
مشخصات مسیر جاری ۲۹
شی تاریخچه یا History ۲۹
شی location ۳۰
شی match ۳۱
مسیرهای پارامتری ۳۲
اجباری/اختیاری بودن پارامترها ۳۳
پارامترهای با نام متغیر ۳۴
مسیرهای تو در تو و مسیریابی داینامیک ۳۴

تعریف داینامیک مسیرها به صورت JSON ۳۸
خلاصه ۳۹
فصل ۳. استفاده از کمپوننت‌های Link و NavLink برای ایجاد لینک به مسیرهای برنامه ۴۱
کمپوننت <Link> ۴۱
خصوصیت replace ۴۲
خصوصیت innerRef ۴۳
خصوصیت to ۴۳
کمپوننت <NavLink> ۴۵
خصوصیت activeClassName ۴۵
خصوصیت activeStyle ۴۶
خصوصیت exact ۴۶
خصوصیت strict ۴۶
خصوصیت isActive ۴۷
خصوصیت location ۴۷
پرش به مسیرهای تو در تو ۴۸
هدایت کاربر به یک مسیر از طریق برنامه‌نویسی با استفاده از history ۴۹
استفاده از کمپوننت رتبه بالای withRouter ۵۰
جلوگیری از خروج کاربر از مسیر جاری با استفاده از <Prompt> ۵۱
خلاصه ۵۳
فصل ۴. کمپوننت Redirect و Switch ۵۵
کمپوننت <Redirect> ۵۵
ویژگی to ۵۶
ویژگی push ۵۸
محافظت از مسیرها و تصدیق هویت ۵۸
برگرداندن کاربر به مسیر قبلی پس از ورود ۶۰
مسیریابی شرطی با استفاده از <Switch> ۶۱
ترتیب <Route> ها در <Switch> ۶۲
استفاده از <Route> با مسیر '/' به عنوان نخستین مسیر ۶۲
<Route> عمومی و <Route> پارامتری ۶۳
تعریف صفحه‌ی 404 – Page Not Found ۶۳
استفاده از <Redirect> در <Switch> برای هدایت کاربر به صفحه‌ی پیش فرض ۶۴
هدایت کاربر از مسیر قدیمی به مسیر جدید ۶۴
خلاصه ۶۵
فصل ۵. کمپوننت Router و پیکربندی BrowserRouter و HashRouter ۶۷
کمپوننت <Router> ۶۷
نحوه‌ی استفاده از <Router> ۶۸
پکیج react-router ۶۹
پکیج react-router-dom ۷۰
کمپوننت <BrowserRouter> ۷۱

خصوصیت basename ۷۲
خصوصیت forceRefresh ۷۲
خصوصیت keyLength ۷۲
خصوصیت getUserConfirmation ۷۳
نمایش یک دیالوگ پیام شخصی با استفاده از getUserConfirmation ۷۵
کمپوننت <HashRouter> ۷۸
خصوصیت hashType ۷۹
خلاصه ۸۰
فصل ۶. استفاده از StaticRouter در برنامه‌های React سمت سرور ۸۱
انجام SSR در برنامه‌های React با استفاده از Node.js و Express.js ۸۱
نصب وابستگی‌ها ۸۱
پیکربندی Webpack ۸۲
برنامه‌ی سمت سرور ۸۴
رندر کردن برنامه‌ی React با استفاده از ReactDOMServer.renderToString() ۸۴
کمپوننت <StaticRouter> و ایجاد مسیر ۸۵
Redirect کردن سمت سرور با استفاده از <Redirect> و staticContext ۸۷
درخواست تطابق URL با matchPath ۸۸
خصوصیت context در StaticRouter ۹۱
ایجاد برنامه‌های ایزومورف یا هم‌شکل ۹۲
تنظیمات Webpack ۹۴
تنظیمات سمت سرور ۹۵
خلاصه ۹۶
فصل ۷. استفاده از NativeRouter در برنامه‌ی React Native ۹۷
استفاده از NativeRouter در یک برنامه‌ی React Native ۹۷
ایجاد یک پروژه‌ی جدید با استفاده از create-react-native-app ۹۸
افزودن کمپوننت <NativeRouter> ۱۰۰
کمپوننت <NativeRouter> ۱۰۳
خصوصیت initialEntries ۱۰۴
خصوصیت initialIndex ۱۰۵
کمپوننت <BackButton> ۱۰۵
ایجاد deeplink با استفاده از <DeepLinking> ۱۰۷
Eject کردن برنامه‌ی create-react-native-app ۱۰۷
افزودن <intent-filter> به مانیفست برنامه ۱۰۸
افزودن کمپوننت <DeepLinking> ۱۱۰
خلاصه ۱۱۱
فصل ۸. مقیدسازی به Redux با استفاده از connected-react-router ۱۱۳
مدیریت وضعیت با Redux ۱۱۳
اکشن‌ها ۱۱۳
Reducer ها ۱۱۴
انباره یا Store ۱۱۵

استفاده از Redux در React ۱۱۵
شروع کار با connected-react-router ۱۱۸
خواندن اطلاعات وضعیت از انباره‌ی Redux ۱۲۰
پیمایش با گسیل داشتن اکشن‌ها ۱۲۲
خلاصه ۱۲۲

درباره‌ی نویسنده

ساگار گاناترا یک برنامه‌نویس front-end و معمار هندی اهل بانگالور است که بیش از یک دهه تجربه‌ی برنامه‌نویسی وب و موبایل دارد. تخصص او معماری پروژه با استفاده از جاوااسکریپت و فریم‌ورک‌هایی مانند React، Angular و Node است. کتاب‌های پیشین او عبارت است از Kendo UI Cookbook و Kendo UI Mobile که هر دو توسط انتشارات Packt منتشر شده است. یکی از علاقمندی‌های او نوشتن بلاگ در سایت شخصی خود sagarantra.com درباره‌ی فناوری‌های front-end است.

درباره‌ی ویراستار فنی

تاداس سابونیس تقریبا از سیزده سالگی کُدنویسی را شروع کرد. از آن زمان تاکنون او با PHP، Javascript، Python، C++ و Java (زبانی که احتمالا بیشترین کُدنویسی را در آن انجام داده) برنامه‌نویسی کرده است. او کارشناسی ارشد هوش مصنوعی دارد، یک متخصص تایید شده و رهبر فنی است که پروژه‌های وبی پیچیده‌ی بیشماری را به مرحله‌ی deploy رسانده است. همچنین او یکی از مولفان کتاب Reactive Android Programming است.

ماریو کراجاکیک یک برنامه‌نویس جاوااسکریپت است. پیش از این او مدیر شبکه بود. اما هنگام تلاش برای خودکار کردن کارهای شبکه‌ای عاشق برنامه‌نویسی شد و پس از آن خودش جاوااسکریپت را یاد گرفت. او علاقه‌ی بسیار زیادی به فناوری و یادگیری دارد. بیشتر تجربه‌ی او مربوط به Node.js و React.js است. همچنین او بخش عمده‌ی آموخته‌های خود را از سایت Chingu.io یاد گرفته است که یک سکوی جهانی مشارکتی برای کسانی است که می‌خواهند فناوری یاد بگیرند.

مایلم از نویسنده‌ی کتاب بخاطر تالیف آن و از اعتمادش به من برای ویراستاری کتاب تشکر کنم. همچنین از چنس مکآلیستر برای توصیه به ویراستاری کتاب و بنیان‌گذاری و هدایت جامعه‌ی شگفت‌انگیز سایت Chingu تشکر می‌کنم.

پیشگفتار

React فریم‌ورکی است که توسط فیسبوک ساخته شده و نحوه‌ی ساخت برنامه‌های وب را دوباره تعریف نموده است. React Router کتابخانه‌ای مبتنی بر React است که عملا به فریم‌ورک مسیریابی برنامه‌های React تبدیل شده است. در جدیدترین نسخه‌ی React Router یعنی نسخه‌ی 4، این کتابخانه از نو بازنویسی شده و امکان می‌دهد بتوانید مسیریابی را به صورت اظهاری یا declarative مدیریت کنید. در این کتاب نحوه‌ی استفاده از react-router در انواع مختلف برنامه‌های React یعنی برنامه‌های وب (react) و موبایل (React Native) آموزش داده می‌شود. همچنین این کتاب موضوعاتی مانند مسیریابی سمت سرور و یکپارچه‌سازی با Redux را هم پوشش می‌دهد.

این کتاب برای چه کسی است؟

این کتاب برای برنامه‌نویسان وب و موبایلی است که برای درست کردن برنامه‌های خود از React و React Router استفاده می‌کنند. برای درک مفاهیم کتاب کمی آشنایی قبلی با React و زبان جاوااسکریپت مفید خواهد بود.

این کتاب چه چیزهایی را پوشش می‌دهد؟

در فصل ۱ «معرفی React Router و ایجاد نخستین مسیر برنامه» معماری کمپوننت–محور React معرفی شده و نحوه‌ی ایجاد مسیر برای برنامه با استفاده از کمپوننت Route نشان داده می‌شود.

در فصل ۲ «پیکربندی مسیرهای برنامه با استفاده از کمپوننت Route» به خصوصیت‌های مختلف کمپوننت Route پرداخته شده و نحوه‌ی تعریف مسیر بر اساس یک آدرس URL و تولید خروجی برای آن نشان داده می‌شود. همچنین این فصل بیان می‌کند چطور می‌توانید به طور پویا مسیرهای جدیدی به برنامه در حین پیمایش کاربر اضافه کنید.

فصل ۳ «کمپوننت Link و NavLink و هدایت کاربر به مسیرهای مختلف» نحوه‌ی استفاده از کمپوننت‌های Link و NavLink را برای هدایت کاربر به مسیرهای تعریف شده برای برنامه بیان می‌کند. همچنین این فصل کمپوننت رتبه بالای withRouter و جلوگیری از انتقال تصادفی کاربر با استفاده از کمپوننت Prompt را توضیح می‌دهد.

فصل ۴ «کمپوننت Redirect و Switch» به نحوه‌ی استفاده از کمپوننت Redirect برای هدایت کاربر به مسیری دیگر و کمپوننت Switch برای مسیریابی شرطی و هدایت کاربر به صفحه‌ی 404 page not found (در حالتی که آدرس جاری درخواست شده پیدا نشود) می‌پردازد.

در فصل ۵ «Router و تنظیمات کمپوننت‌های BrowserRouter و HashRouter» نحوه‌ی استفاده از واسط core router و به روزرسانی بخشی از صفحه و تاریخچه‌ی مرورگر به طور عمقی بیان می‌شود. همچنین در این فصل دو پیاده‌سازی از واسط انتزاعی مسیریابی یعنی BrowserRouter و HashRouter توضیح داده می‌شود.

فصل ۶ «استفاده از StaticRouter در برنامه‌های React سمت سرور» به نحوه‌ی استفاده از کمپوننت ReactRouter برای فراهم کردن قابلیت مسیریابی در برنامه‌ای که سمت سرور رندر می‌شود می‌پردازد. همچنین در این فصل توضیح داده می‌شود چطور می‌توانید از StaticRouter و BrowserRouter برای ساخت یک برنامه‌ی وبی ایزومورف استفاده کنید.

فصل ۷ «استفاده از NativeRouter در برنامه‌های React Native» نحوه‌ی استفاده از مسیریابی در برنامه‌های موبایلی React Native را با استفاده از کمپوننت NativeRouter بیان می‌کند. همچنین این فصل توضیح می‌دهد چطور می‌توانید با استفاده از کمپوننت BackButton و پشتیبانی از لینک عمیق با استفاده از کمپوننت DeepLinking برنامه را به قابلیت دکمه‌ی Back موبایل مجهز کنید.

فصل ۸ «مقیدسازی به Redux با استفاده از connected-react-router» نحوه‌ی استفاده از کتابخانه‌ی connected-react-router را توضیح می‌دهد که امکان Redux bindings را برای React Router فراهم می‌کند. این فصل نشان می‌دهد چطور می‌توانید اطلاعات مسیریابی را از وضعیت router در انبار Redux بخوانید و چطور با گسیل داشتن عمل‌های مختلف به سمت انبار، برنامه را پیمایش کنید.

پیش فرض‌های کتاب

React Router برای برنامه‌های وب و موبایلی مبتنی بر React طراحی شده است. در این کتاب فرض می‌شود درک خوبی نسبت به جاوااسکریپت و قابلیت‌های جدید ECMAScript 6 (مانند کلاس و عملگر spread) دارید. در ابتدای کتاب مرور مختصری به React و معماری کمپوننت–محور آن انجام می‌شود. برای آشنایی با مابقی اصول React به سایت رسمی این فریم‌ورک به آدرس https://reactjs.org مراجعه کنید. همچنین در این کتاب فرض می‌شود خواننده از Node.js و NPM برای نصب کتابخانه‌ها و پکیج‌های NPM استفاده نموده و با نحوه‌ی این کار آشنا است.

دانلود کُدهای پیوست کتاب

می‌توانید کُدهای پیوست کتاب را با استفاده از اکانت خود از سایت www.packt.com دانلود کنید. اگر کتاب را از جای دیگری خریده‌اید می‌توانید به آدرس www.packt.com/support بروید و ثبت نام کنید تا کُدهای پیوست کتاب برایتان ایمیل شود.

برای دانلود فایل‌های پیوست مراحل زیر را دنبال کنید:

۱. در سایت www.packt.com لاگین کنید.
۲. برگه‌ی SUPPORT را انتخاب کنید.
۳. دکمه‌ی Code Downloads & Errata را کلیک کنید.
۴. نام کتاب را در باکس جستجو وارد کرده و دستورالعمل نشان داده شده را دنبال کنید.

پس از دانلود فایل پیوست کتاب آن را با یکی از برنامه‌های زیر unzip کنید.

- WinRar/7-Zip در ویندوز
- Zipeg/iZip/UnRarX در مک
- 7-Zip/PeaZip در لینوکس

کُدهای کتاب در مخزن https://github.com/PacktPublishing/React-Router-Quick-Start-Guide در سایت GitHub نیز قرار داده شده است. اگر قرار باشد کُدهای کتاب به روز شود این کار در همین مخزن انجام می‌شود.

ما در کاتالوگ کتاب‌ها و ویدئوهای خود در https://github.com/PacktPublishing کُدهای دیگری هم داریم. اگر دوست داشتید نگاهی هم به آنها بیاندازید.

دانلود تصاویر رنگی کتاب

ما یک فایل PDF رنگی هم فراهم کرده‌ایم که حاوی تصاویر رنگی شکل‌ها و دیاگرام‌های کتاب است. می‌توانید آن را از آدرس زیر دانلود کنید:

https://www.packtpub.com/sites/default/files/downloads/9781789532555_ColorImages.pdf

تماس با ما

ما همیشه از بازخورد کاربران استقبال می‌کنیم.

بازخورد عمومی: به feedback@packt.com ایمیل بزنید و عنوان کتاب را در فیلد عنوان پیام خود ذکر کنید. اگر پرسشی درباره‌ی هر جنبه‌ای از کتاب دارید لطفا آن را به ایمیل questions@packt.com بفرستید.

اشتباه: با وجودی که حداکثر دقت و تلاش‌مان را در خصوص صحت محتوای کتاب به کار برده‌ایم ممکن است اشتباهاتی هم وجود داشته باشد. اگر در کتاب به خطا یا اشتباهی برخورد کردید خوشحال خواهیم شد آن را به ما گزارش کنید. لطفا به آدرس www.packt.com/submit-errata بروید، کتاب خود را انتخاب کنید، روی لینک Errata Submission Form کلیک کرده و در فرم اعلام خطا جزئیات آن را وارد کنید.

دزدی: اگر در اینترنت به هرگونه کپی غیر قانونی از کتاب به هر شکلی برخورد کردید خوشحال خواهیم شد آدرس یا سایت آن را به ما اعلام کنید. لطفا آدرس کپی غیر قانونی را به copyright@packt.com برای ما ایمیل کنید.

اگر مایلید به جمع مولفان ما بپیوندید و موضوعی وجود دارد که در آن تخصص دارید و مایلید برای آن کتاب بنویسید یا مایلید در نوشتن کتابی در آن موضوع مشارکت داشته باشید لطفا به آدرس authors.packt.com بروید.

بازبینی

لطفا پس از خواندن کتاب نظر خود را درباره‌ی آن بنویسید. وقتی کتاب را به پایان رساندید چه بهتر که نظر خود را به سایتی که کتاب را از آنجا خریداری کردید بگویید. پس از آن خوانندگان بعدی می‌توانند نظر شما را به عنوان فردی بی‌طرف ببینند و برای خرید کتاب تصمیم بگیرند. ما هم در Packt از نظر شما درباره‌ی محصولات‌مان آگاه می‌شویم و مولفان هم می‌توانند بازخورد شما را درباره‌ی کتاب‌شان ببینند. قبلا از شما برای نظرات‌تان ممنونیم.

برای کسب اطلاعات بیشتر درباره‌ی Packt لطفا به آدرس packtpub.com بروید.

فصل ۱. معرفی React Router و ایجاد نخستین مسیر برنامه

امروزه برنامه‌های تک صفحه‌ای SPA عملا به استاندارد ساخت برنامه‌های وب تبدیل شده‌اند. فریم‌ورک‌ها و کتابخانه‌های جاوااسکریپتی بسیار زیادی در این زمینه وجود دارد که برنامه‌نویسان front-end می‌توانند برنامه‌های SPA را با آنها درست کنند مانند React، Angular، Ember و Backbone. هر کدام این فریم‌ورک‌ها متدها، سرویس‌ها و کمپوننت‌هایی دارند که با آنها می‌توان برنامه‌ها را به سرعت ایجاد کرد.

از سوی دیگر برنامه‌های SPA برای فراهم کردن تجربه‌ای روان و هموار برای کاربران گزینه‌ی بسیار خوبی هستند. زیرا در حین این که کاربر در برنامه این سو و آن سو می‌رود بجای این که هر بار کل صفحه از سرور درخواست شده و صفحه رفرش شود تنها بخش‌های مورد نیاز صفحه به روز می‌شود.

React کتابخانه‌ی جاوااسکریپتی اُپن‌سورسی است که با آن می‌توانید واسط کاربر برنامه و لایه‌ی view را در برنامه‌های وب و موبایل ایجاد کنید. این کتابخانه برنامه‌نویسان را وا می‌دارد لایه‌ی view برنامه را به چشم کلکسیونی از کمپوننت‌های قابل استفاده‌ی مجدد ببینند که می‌توان آنها را در جاهای مختلف برنامه استفاده کرد.

بیشتر فریم‌ورک‌های front-end یک بسته‌ی مسیریابی یا routing هم دارند که با آن می‌توان هنگام کلیک کاربر روی لینک‌های برنامه بخش یا بخش‌هایی از صفحه را به روز کرد. در کتابخانه‌های routing کمپوننتی به نام مسیریاب یا router وجود دارد که تغییر آدرس URL صفحه را زیر نظر می‌گیرد و هنگام مشاهده‌ی تغییری در آدرس مرورگر با رندر کردن کمپوننتی متناسب با آدرس جدید، view برنامه را به روز می‌کند. به عبارت دیگر view را با آدرس جاری هماهنگ نگه می‌دارد.

برای نمونه هنگامی که کاربر به آدرس dashboard/ می‌رود کمپوننت‌های رایج داشبورد مانند نمودار و جداول گزارش نشان داده شده و وقتی کاربر مثلا به آدرس user/ می‌رود پروفایل و مشخصات کاربر نشان داده می‌شود. فریم‌ورک React خودش کمپوننت یا کتابخانه‌ای برای مسیریابی ندارد. از این رو در برنامه‌های React به کتابخانه‌ای برای مسیریابی نیاز است. React-Router کتابخانه‌ی بسیار محبوبی در این زمینه است که به طور ویژه برای React نوشته شده است.

این کتابخانه کمپوننت‌های مختلفی دارد که در حین پیمایش برنامه توسط کاربر می‌توان از آنها برای رندر کردن view برنامه استفاده کرد. افزون بر ردگیری تغییرات آدرس URL و رندر کردن کمپوننت‌ها، React-Router قابلیت‌های مختلفی دارد که با آنها می‌توانید مسیرهای برنامه را به سادگی تعریف و تنظیم کنید.

در این فصل به موضوع‌های زیر می‌پردازیم:

- مرور سریعی بر React: در این قسمت مفاهیم اصلی React مانند معماری کمپوننت–محور، نحوه‌ی کمپوننت‌سازی و این که دیتا چطور می‌تواند در درختواره‌ی کمپوننت‌ها در اختیار کمپوننت‌های فرزند قرار داده شود بیان می‌شود.

- معرفی React-Router: در این قسمت ابتدا با استفاده از ابزار خط فرمان create-react-app یک برنامه‌ی React درست می‌کنیم و سپس کتابخانه‌ی React-Router (پکیج react-router-dom) را به عنوان وابستگی در آن نصب می‌کنیم.
- ایجاد نخستین route: پس از افزودن React-Router به برنامه با استفاده از کمپوننت‌های <BrowserRouter> و <Route> نخستین مسیر برنامه را تعریف می‌کنیم.

مرور سریعی بر React

به طور خلاصه React یک کتابخانه‌ی جاوااسکریپتی است که کمپوننت‌ها و سرویس‌هایی فراهم می‌کند که با آنها می‌توانید به سرعت و سهولت واسط کاربری برنامه را درست کنید.

در اینجا جمله‌ی زیر را از زبان سازندگان React در سایت reactjs.org در توصیف این کتابخانه نقل قول می‌کنیم:

«React یک کتابخانه‌ی جاوااسکریپتی declarative، کارا و انعطاف‌پذیر برای ساخت واسطهای کاربر است».

React تحت مجوز MIT نوشته شده و نگهداری می‌شود. از این کتابخانه به طور گسترده در برنامه‌های شرکت فیسبوک مانند سایت فیسبوک و اینستاگرام استفاده شده است. React به شما امکان می‌دهد کمپوننت‌هایی بسازید که هنگام تغییر وضعیت برنامه شکل‌شان به طور خودکار به‌روز می‌شود. مقصود از وضعیت در اینجا دیتای داخلی برنامه یا محل جاری کاربر در برنامه است. React اطمینان می‌دهد شکل کمپوننت‌های صفحه همیشه با وضعیت برنامه هماهنگ باشد.

برخی از مهم‌ترین قابلیت‌های React بدین صورت است:

- **JSX:** برای درست کردن کمپوننت‌های React از قاعده‌ای شبیه XML/HTML استفاده می‌شود که به آن JSX گفته می‌شود. با JSX می‌توانید درون کُدهای جاوااسکریپتی برنامه مستقیما کُد HTML بنویسید. سپس کُدهای JSX توسط پیش‌پردازشگری مانند Babel پردازش شده و محتوای شبه HTML آن به اشیا جاوااسکریپتی‌ای تبدیل می‌شود که موتور جاوااسکریپت می‌تواند آنها را اجرا کند. به این فرآیند transpile گفته می‌شود. با استفاده از قاعده‌ی آشنای HTML داخل متدی به نام ()render که برای هر کمپوننت تعریف می‌شود و سپس نمونه‌سازی از کمپوننت‌ها با استفاده از تگ و ویژگی دیگر نیازی ندارید برای تعریف کمپوننت‌های برنامه یک زبان template اضافی هم یاد بگیرید.
- **مقیدسازی یک طرفه‌ی دیتا[1]:** برنامه‌های React به شکل سلسله مراتبی از کمپوننت‌های تو در تو ساخته می‌شوند. مقدار خصوصیت‌های کمپوننت‌ها از طریق ویژگی در تگ‌های HTML به آنها پاس داده می‌شود. به مجموعه‌ی این ویژگی‌ها یا خصوصیت‌ها props گفته می‌شود. React طوری ساخته شده که وقتی مقداری برای خصوصیت یا ویژگی یک کمپوننت مشخص می‌کنید دیگر نمی‌توانید آن را تغییر بدهید.

[1] One-way data binding

در واقع خود کمپوننت‌ها نمی‌توانند مقادیری را که برای خصوصیت‌هایشان مشخص شده (شی props را) تغییر بدهند. بجای آن باید از کمپوننت پدر خود بخواهند خصوصیت آنها را تغییر بدهد. کمپوننت پدر نیز این کار را با به روز کردن متغیرهای وضعیتی خود که از آنها برای مقداردهی خصوصیت‌های کمپوننت‌ها استفاده کرده انجام می‌دهد. پس از آن React به طور خودکار با رندر کردن دوباره‌ی کمپوننت، خصوصیت‌هایش را و همچنین نمای آن را به روز می‌کند.

- **DOM مجازی (Virtual DOM):** در React یک DOM مجازی وجود دارد که دقیقا کپی درخواره‌ی DOM مرورگر است. به ازای هر گره یا node در DOM مرورگر یک شئ در DOM مجازی ایجاد می‌شود که دقیقا همان خصوصیت‌های گره DOM واقعی را دارد. با این حال اشیا DOM واقعی و مجازی مستقیما با تغییراتی که کاربر به صفحه می‌دهد همراه هم به روز نمی‌شود. بجای آن وقتی React متوجه شود چیزی در وضعیت صفحه تغییر کرده المان‌های مرتبط با آن را دوباره رندر می‌کند. این رندر کردن باعث تغییر DOM مجازی می‌شود. سپس React درخواره‌ی DOM مجازی را با اسنپ‌شات قبلی مقایسه می‌کند تا ببیند چه اشیائی تغییر کرده است. سپس درخواره‌ی DOM واقعی را به روز می‌کند. اما تنها گره‌هایی را در DOM واقعی به روز می‌کند که تغییر کرده‌اند.

معماری کمپوننت-محور در React

از زمان انتشار نخستین نسخه‌ی React در سال ۲۰۱۳ این کتابخانه باعث شده روش ساخت برنامه‌های front-end از نو تعریف شود. همان گونه که بیان کردیم این کتابخانه معماری کمپوننت-محور را معرفی می‌کند که با آن می‌توانید برنامه‌ی خود را به گونه‌ای تجسم کنید که گویی بر اساس مجموعه‌ای از کمپوننت‌های کوچک و مستقل ساخته شده است. این کمپوننت‌ها قابلیت استفاده‌ی مجدد هم دارند. برای نمونه کارکرد کمپوننتی مانند CommentBox یا Footer داخل آنها کپسوله می‌شود و می‌توان آنها را هر جای دیگری از برنامه هم استفاده کرد.

در این دیدگاه خود صفحه هم کمپوننتی است که بر اساس تعدادی کمپوننت کوچک ساخته می‌شود. به مثال زیر توجه کنید:

```
<Dashboard>
    <Header>
        <Brand />
    </Header>
    <SideNav>
        <NavLink key=″1″>
        <NavLink key=″2″>
    </SideNav>
    <ContentArea>
        <Chart>
        <Grid data="stockPriceList">
    </ContentArea>
    <Footer />
</Dashboard>
```

در اینجا <Dashboard> یک کمپوننت است که چندین کمپوننت (Header، SideBar، ContentArea و Footer) درون خود دارد. خود آن کمپوننت‌ها بر اساس کمپوننت‌های دیگری (Brand، NavLink، Chart و Grid) ساخته شده‌اند.

معماری کمپونت–محور شما را تشویق می‌کند برای کارکردهای صفحه کمپونت‌هایی بسازید که مستقل بوده و به طور محکم به کمپونت‌های پدر یا خواهر و برادرشان وصل نیستند. این کمپونت‌ها امکانات و واسطی فراهم می‌کنند که از طریق آن می‌توان آنها را داخل صفحه قرار داد.

برای نمونه در کُد بالا کمپونت <Grid> قابلیت‌هایی مانند رندر کردن دیتا به شکل سطر و ستون، جستجو و همچنین مرتب‌سازی صعودی و نزولی دیتا بر حسب ستون‌های مختلف دارد. همه‌ی این قابلیت‌ها داخل کمپونت Grid پیاده‌سازی و به عبارت دیگر کپسوله شده است و این کمپونت واسطی فراهم می‌کند که با آن می‌توان به راحتی در صفحه گرید ایجاد کرد. این واسط همان تگ <Grid> و ویژگی‌ها این تگ است که از طریق آنها می‌توانید گرید را مطابق نیاز خود تنظیم کنید.

کمپونت <Grid> می‌تواند برای به دست آوردن دیتایی که می‌خواهد نمایش بدهد خودش با بخش back-end برنامه به طور مستقیم ارتباط برقرار کند. اما این طراحی مناسبی نیست. زیرا باعث می‌شود کمپونت Grid به آن واسط back-end وابسته شده و دیگر قابلیت استفاده‌ی مجدد نداشته باشد. به عبارت دیگر نمی‌توان گرید را جای دیگری در برنامه استفاده کرد. بجای این کار بهتر است کمپونت Grid دیتای مورد نمایش را از کمپونت پدر خود دریافت کند (کمپونت پدرش دیتا را به او بدهد).

```
<Grid data="stockPriceList" />
```

در اینجا کمپونت <Grid> از طریق ویژگی data دیتای قیمت سهام را دریافت کرده و به شکل جدول نمایش می‌دهد. به کمپونت پدری که <Grid> را استفاده می‌کند کمپونت ظرف یا container و به کمپونت Grid کمپونت فرزند یا child گفته می‌شود.

کمپونت container خودش یک کمپونت است. اما یکی از مسوولیت‌هایش فراهم کردن دیتای مورد نیاز کمپونت‌های فرزند است. برای این کار کمپونت container می‌تواند درخواست‌های HTTP برای بخش back-end برنامه بفرستد و دیتای مورد نیاز کمپونت‌های فرزند را به دست بیاورد. همچنین کمپونت container مسوولیت جای دادن کمپونت‌های فرزند در فضای نمایشی خودش را هم بر عهده دارد و او است که مشخص می‌کند کمپونت‌های داخل او چطور کنار هم قرار بگیرند.

ایجاد یک کمپوننت React

برای ساخت کمپونت در React باید کلاسی از کلاس پایه‌ی Component (که توسط React فراهم شده) مشتق کنید.

```
import React, { Component } from 'react';
import './button.css';

export class Button extends Component {
    render() {
        return (
            <button className={this.props.type}>
                {this.props.children}
            </button>
        );
    }
}
```

در اینجا کلاس Button از Component مشتق شده و متد ()render آن را override نموده است. مقدار برگشتی متد ()render نیز یک تکه JSX است که هنگام بارگذاری صفحه در DOM رندر شده و نمایش داده می‌شود. این کمپونت Button دو خصوصیت به نام type و children دارد که از طریق شئ this.props قابل دسترس هستند. هنگام استفاده از کمپونت Button به صورت تگ <Button> می‌توانیم مقدار این خصوصیت‌ها را از طریق ویژگی‌هایی به همین نام در تگ <Button> مشخص کنیم:

```
import React, { Component } from 'react';
import { Button } from './components/Button/button';
import './App.css';

export default class App extends Component {
    render() {
        return (
            <div className="App">
                <Button type="secondary">CANCEL</Button>
                <Button type="primary">OK</Button>
            </div>
        );
    }
}
```

در اینجا کمپونت Button را داخل کمپونت App استفاده کرده و دو دکمه ایجاد کرده‌ایم. کمپونت <App> یک کمپونت container است که مسئولیت رندر کردن دکمه‌های صفحه را بر عهده دارد. کمپونت Button داخل خودش مقدار ویژگی type را در تگ <button> به عنوان کلاس CSS (خصوصیت className) به کار می‌برد. همچنین متنی که کمپونت App بین تگ <Button> و </Button> قرار می‌دهد در کمپونت Button از طریق خصوصیت this.props.children قابل دسترس است و کمپونت Button در متد ()render خود آن را داخل تگ <button> به کار برده است.

مقدار this.props.children می‌تواند یک متن محض ساده مانند کلمات CANCEL و OK در مثال فعلی یا حتی یک یا چند کمپوننت باشد. خصوصیت children ارجاعی به همه‌ی کمپوننت‌های فرزندی که میان تگ شروع و خاتمه‌ی کمپوننت قرار گرفته فراهم می‌کند. اگر در گذشته از Angular استفاده کرده باشید مثال بالا برای شما شبیه نحوه‌ی استفاده از المان‌ها با استفاده از ng-transclude در AngularJS یا ng-content در Angular خواهد بود.

مرحله‌ی بعد این است که کمپونت <App> را در DOM رندر کنیم. کمپونت <App> به عنوان کمپوننت ریشه عمل می‌کند. یعنی گره ریشه‌ای درختواره‌ی کمپونت‌های برنامه. برای این کار از متد ReactDOM.render() باید استفاده کنیم.

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './index.css';

ReactDOM.render(<App />, document.getElementById('root'));
```

ابتدا کتابخانه‌های React و ReactDOM را import می‌کنیم. سپس متد render() را در شئ ReactDOM که در کتابخانه‌ی react-dom تعریف شده صدا می‌زنیم. این متد دو پارامتر دارد. نخستین پارامتر آن کمپوننتی است که قرار است رندر شود و دومین پارامتر آن ارجاعی به یک گره در DOM است که کمپوننت باید آنجا رندر شود.

اگر پس از اجرای برنامه DOM را ببینید خواهید دید محتوای داخل کمپونت <App> (کمپونت‌های Button) نیز رندر شده است.

```
▼<div id="root">
  ▼<div class="App">
      <button class="secondary">CANCEL</button>
      <button class="primary">OK</button>
    </div>
  </div>
```

معرفی React-Router

همان طور که گفتیم React-Router کتابخانه‌ی مسیریابی برنامه‌های React است. نسخه‌ی چهارم این کتابخانه به طور کامل از نو بازنویسی شده و بر مبنای فلسفه‌ی معماری کمپونت–محور React طراحی شده است. این جمله در توضیح این کتابخانه از مستندات خود آن نقل می‌شود:

> «React Router مجموعه‌ای از کمپونت‌های پیمایشی است که به طور ساده و declarative کنار کمپونت‌های برنامه قرار گرفته و در برنامه استفاده می‌شوند. چه بخواهید در برنامه‌ی وب خود آدرس‌های URL قابل بوک‌مارک داشته باشید و چه بخواهید در برنامه‌ی موبایلی React Native خود قابلیت پیمایش فراهم کنید. React Router هر جا که React باشد کار می‌کند – فرقی نمی‌کند چه برنامه‌ای انتخاب کرده باشید.»

React Router هر جایی که React قابل استفاده باشد می‌تواند به کار برود. یعنی هم در مرورگر و هم در محیط بومی با React Native کار می‌کند. این کتابخانه از سه پکیج تشکیل می‌شود:

- react-router: حاوی کمپوننت‌های هسته‌ای مشترک برای استفاده در DOM و همچنین نسخه‌ی بومی آنها برای استفاده در برنامه‌های React Native
- react-router-dom: حاوی کمپوننت‌هایی اختصاصی برای استفاده در مرورگر و برنامه‌های وب
- react-router-native: حاوی کمپوننت‌هایی اختصاصی برای استفاده در برنامه‌های React Native

این کتابخانه کمپوننت‌های دارد که با آنها می‌توانید به صورت پویا به برنامه مسیر اضافه کنید. با استفاده از سیستم مسیریابی داینامیک React-Router v4 می‌توانید در حین پیمایش کاربر در برنامه برای آن مسیر جدید تعریف کنید. در فریم‌ورک‌هایی مانند AngularJS و Express مجبور هستید مسیرها را از ابتدا تعریف کنید و اطلاعات مسیریابی هنگام بالا آمدن برنامه استفاده شده و دیگر غیر قابل تغییر است. نسخه‌ی پیشین React-Router نیز به همین صورت نوشته شده بود و در آن لازم بود اطلاعات مسیریابی را از ابتدا تعریف می‌کردید. اما در نسخه‌ی 4 این کتابخانه قابلیت مسیریابی داینامیک هم فراهم شده است.

افزون بر قابلیت مسیریابی داینامیک و فراهم کردن پیمایش سیال (نرم و روان) برای برنامه‌های React کتابخانه‌ی React-Router قابلیت‌های متنوع دیگری هم دارد که معمولا در سایت‌های سنتی قابل دسترس است اما React Router امکان استفاده از آن را برای هر نوع برنامه‌ی React مانند برنامه‌های React Native فراهم می‌کند. این قابلیت‌ها شامل موارد زیر است:

- جلو و عقب رفتن در برنامه با حفظ تاریخچه و بازیابی وضعیت برنامه
- رندر کردن خودکار کمپوننت‌های متناسب در صفحه هنگام تغییر آدرس URL (لینک زنی عمیق یا deep-linking)
- هدایت یا redirect کردن کاربر از یک مسیر به مسیر دیگر
- رندر کردن صفحه‌ی 404 هنگامی که هیچ یک از مسیرها با آدرس URL جاری مطابقت ندارد
- پشتیبانی از مسیرهای مبتنی بر هش[1] و آدرس‌های URL زیبا با مُد HTML5

بسیاری تصور می‌کنند React-Router راه‌حل رسمی فیسبوک برای مسیریابی است. اما این تصور نادرستی است. React-Router یک کتابخانه‌ی شرکت ثالث است و تحت پروانه‌ی MIT منتشر شده است.

شروع کار با React-Router

بگذارید یک برنامه‌ی React درست کرده و React-Router را در آن نصب کنیم. برای ایجاد برنامه‌ی React از create-react-app استفاده می‌کنیم. ابزار خط فرمان create-react-app ساخت برنامه‌های React را بسیار ساده می‌کند. هنگام ایجاد برنامه‌ی React با این ابزار، سورس برنامه بر اساس یک داربست از پیش آماده ایجاد می‌شود تا

[1] Hash-based routes

بتوانید به سرعت و بی‌دردسر از React و جدیدترین قابلیت‌های جاوااسکریپت استفاده کنید. همچنین اسکریپت‌هایی برای برنامه تعریف می‌کند تا بتوانید برنامه را برای محیط production یا تولید build کنید.

البته بسته‌های starter مختلفی برای ایجاد برنامه‌ی React مبتنی بر React-Router وجود دارد. با این وجود استفاده از create-react-app کمک می‌کند نشان بدهیم چطور می‌توانید React-Router را به یک برنامه‌ی React از پیش موجود و خالی اضافه کنید.

ابتدا باید create-react-app را با استفاده npm توسط دستور زیر به صورت سراسری یا global نصب کنیم.

```
npm install –g create-react-app
```

برای استفاده از create-react-app باید از node نسخه‌ی 6 یا بالاتر و همچنین npm حداقل نسخه‌ی 5.2.0 استفاده کنید.

پس از نصب create-react-app می‌توانید به صورت زیر یک برنامه‌ی React جدید به نام react-router-demo-app ایجاد کنید. create-reacta-pp برنامه را در دایرکتوری react-router-demo-app ایجاد می‌کند (اگر این دایرکتوری از قبل وجود نداشته باشد خودش آن را ایجاد می‌کند).

```
create-react-app react-router-demo-app
```

اگر بجای npm عادت دارید از yarn (https://yarnpkg.com/en) استفاده کنید در دستورهای قبل npm را با yarn جایگزین کنید.

وقتی create-react-app نصب پکیج‌های مورد نیاز برنامه را تمام کرد خروجی‌ای مانند زیر نشان داده می‌شود:

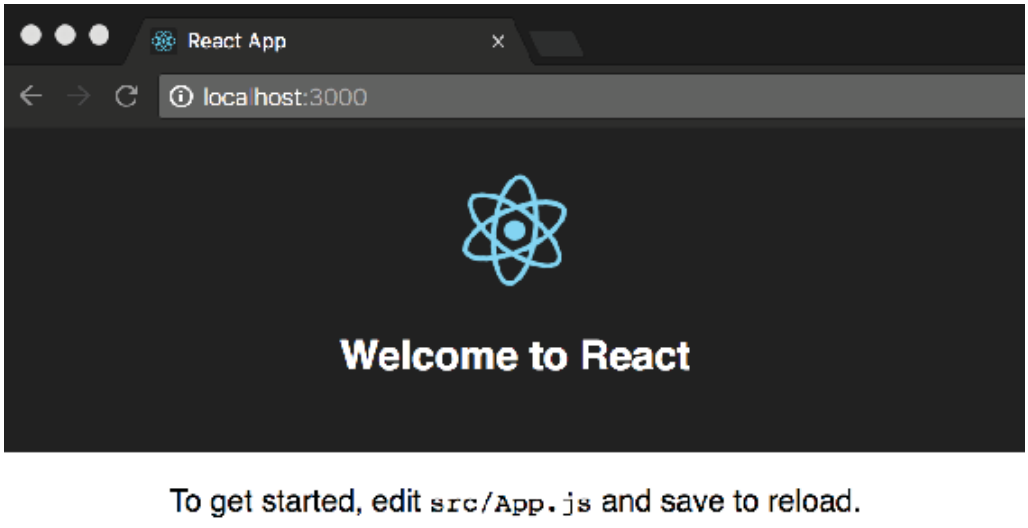
```
Inside that directory, you can run several commands:
       npm start
              Starts the development server.
       npm run build
              Bundles the app into static files for production.
       npm test
              Starts the test runner.
       npm run eject
              Removes this tool and copies build dependencies, configuration
files and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:
       cd react-router-demo-app
       npm start
```

دستورهای npm start، npm run build، npm test و npm run eject همان اسکریپت‌های از پیش آماده‌ای است که create-react-app هنگام ایجاد برنامه برای آن تعریف می‌کند و با آنها می‌توانید برنامه را اجرا کنید، build کنید، تست کنید یا اگر زمانی هم خواستید با eject فنوداسیون برنامه را از هم باز کنید تا تمام تنظیمات برنامه به طور کامل در اختیارتان قرار بگیرد.

پس از اتمام ایجاد برنامه داخل دایرکتوری آن ساختار فولدربندی زیر ایجاد خواهد شد:

```
/react-router-demo-app
|--node_modules
|--public
| |--favicon.ico
| |--index.html
| |--manifest.json
|--src
| |--App.css
| |--App.js
| |--App.test.js
| |--index.css
| |--index.js
| |--logo.svg
| |--registerServiceWorker.js
|--package-lock.json
|--package.json
|--README.md
```

ابزار create-react-app خودش از قبل همه‌ی پکیج‌های لازم را نصب می‌کند (مانند Babel برای transpile کردن کدهای ES6 به ES5 تا بتوانید از جدیدترین قابلیت‌های جاوااسکریپت بهره بگیرید). همچنین این ابزار یک build pipeline هم بر پایه‌ی webpack برای برنامه تنظیم می‌کند تا بتوانید سورس برنامه را به راحتی bundle کرده و اجرا کنید. پس از ایجاد برنامه برای اجرای آن به چیز دیگری نیاز نیست. همان طور که در خروجی نشان داده شده برای اجرای برنامه کافی است از دستور npm start استفاده کنید. با دستور npm build هم می‌توانید نسخه‌ی build و آماده‌ی انتشار برنامه را تولید کنید. وقتی دستور npm start را اجرا می‌کنید برنامه کامپایل می‌شود و همان گونه که در تصویر زیر نشان داده شده یک پنجره‌ی مرورگر حاوی پیام خوش‌آمدگویی React باز می‌شود.



اگر فایل index.js را از فولدر src باز کنید دستور زیر را در آن خواهید دید که برای رندر کردن کمپوننت ریشه‌ای برنامه به کار رفته است:

```
ReactDOM.render(<App />, document.getElementById('root'));
```

کمپوننت <App /> ریشه‌ی درختواره‌ی برنامه است و سورس آن در فایل App.js قرار دارد.

افزودن کتابخانه‌ی React-Router

پس از راه‌اندازی این برنامه‌ی نمونه بگذارید React-Router را توسط npm در آن نصب کنیم:

```
npm install --save react-router-dom
```

این دستور پکیج react-router-dom را دانلود کرده و در دایرکتوری node_modules/ نصب می‌کند. این پکیج مخصوص برنامه‌های وب است. اگر برنامه‌ی شما React Native باشد بجای آن باید پکیج react-routernative را نصب کنید. این را با جزییات بیشتر در فصل‌های بعد توضیح خواهیم داد. پس از نصب react-router-dom اگر فایل package.json را باز کنید نام پکیج react-router-dom را در قسمت dependencies مشاهده خواهید کرد:

```
"dependencies": {
   "react": "^16.4.0",
   "react-dom": "^16.4.0",
   "react-router-dom": "^4.3.0",
   "react-scripts": "1.1.4"
}
```

در زمان نوشتن کتاب آخرین نسخه‌ی react-router-dom برابر 4.3.0 بود. هنگام استفاده از npm می‌توانید با ذکر @next به صورت react-router-dom@next نسخه‌ی آلفا و بتای این کتابخانه را هم نصب کرده و امتحان کنید.

تعریف مسیر برای برنامه

داخل پکیج react-router-dom کمپوننتی به نام BrowserRouter وجود دارد که به عنوان wrapper به عنوان ظرف مسیرهای برنامه به کار می‌رود. کمپوننت <BrowserRouter> پیاده‌سازی‌ای از یک واسط انتزاعی است که از قابلیت history API در HTML5 برای مسیریابی استفاده می‌کند. برای استفاده از React Router در برنامه نخستین کاری که باید انجام بدهیم این است که کمپوننت ریشه‌ای برنامه یعنی <App /> را داخل <BrowserRouter> قرار بدهیم:

```
import { BrowserRouter } from 'react-router-dom';

ReactDOM.render(
    <BrowserRouter>
        <App />
    </BrowserRouter>,
    document.getElementById('root')
);
```