优达学城（Udacity）React 开发者纳米学位

掌握最前沿 React JS 框架，成为抢手高级前端工程师

前言

祝贺你考虑学习 React 开发者纳米学位！在开始之前，请确保你有稳定的学习时间付出，并再三确认好你已经满足先修条件： 你必须掌握 HTML、CSS 和 JavaScript，还需要能够熟练使用命令行（bash 或 terminal）、Git、GitHub, 和 NPM。

React 开发者纳米学位 由 3 部分课程和 3 个实战项目组成。你挑战的每个实战项目都是基于已学习的知识进行巩固。 完成好的实战项目将会丰富你的简历作品集，并将向你的潜在雇主充分展示你的专业实力。

第一部分: React 基础知识

从基础知识开始掌握 React，可能这会带有一点挑战性，因为 React 生态系统中的模块化对于构建应用来说非常强大。 所以你需要学习的知识还有很多。 我们会从头开始，一点一滴让你掌握好 React 入门知识，奠定好创建应用的基础。

这是一个基于实战项目的课程， 你需要自己完成实战项目。 通过学习 React 的组件模型，你将能够编写声明式、可组合的用户界面，从而构建可上线发布的应用。 你将在实战项目中巩固 React 知识。此外，你提交的项目作品将会由专家审阅，他们会提供给你个性化的反馈，帮助你取得进步。

课程标题	学习目标
为何要学习 React？	➔ 了解 React 创建的原因 ➔ 使用 composition 从简单着手，构建复杂的功能 ➔ 利用声明式代码来表示没有控制流的逻辑 ➔ 理解 React 就是 JavaScript
使用 React 渲染 UI	➔ 使用 *create-react-app* 来创建新的 React 应用 ➔ 使用 composition 来创建可重复使用、集中的类组件 ➔ 利用 *JSX* 来描述 UI
状态管理	➔ 管理应用的状态 ➔ 使用 *props* 将数据传递到组件 ➔ 创建专注于 UI 而不是行为的功能组件 ➔ 将状态添加到组件 来代表可变的内部数据 ➔ 使用关键词 *this* 来访问组件数据和属性 ➔ 用 *setState()* 更新状态 ➔ 使用 *PropTypes* 来检查和调试组件 ➔ 使用受控组件来管理输入表单元素
使用外部数据渲染 UI	➔ 将组件的生命周期概念化 ➔ 在 React 的 *componentDidMount* 生命周期方法中发起 HTTP 请求
使用 React Router 管理应用位置信息	➔ 使用 React Router 将不同的路由添加到应用程序 ➔ 使用状态动态地呈现不同的“页面” ➔ 使用 React 路由器的 *Route* 组件 ➔ 使用 React 路由器的 *Link* 组件

实战项目1: 图书跟踪应用

在 MyReads 项目中，你将创建一个书架应用，使你能够选择和归类你阅读过的图书、正在阅读的图书以及想要阅读的图书。该项目重点讲解如何使用 React 构建该应用并提供一个 API 服务器和客户端库，使你在与应用互动时能够保存信息。最后，你将使用 React 的 setState 来实现 图书从一个书架到另一个书架 的功能。

第二部分: React 和 Redux

Redux 擅长状态管理，在本课程中，你将学习 Redux 和 React 如何协同工作，使你的应用程序状态更加简洁。

和之前的课程一样，这是需要动手实践的课程，实战项目是核心。 在这里，你将利用 React 与 Redux 一起构建 “Udacimeals”，一个膳食追踪应用程序。

课程标题	学习目标
为何要使用 Redux	➔ 建立何时使用 Redux 的认知 ➔ 了解 Redux 如何提高应用程序的可预测性 ➔ 确定何时使用 组件状态 与 Redux状态 ➔ 识别 Redux 用来影响状态的 JavaScript 技术
Redux 的核心	➔ 创建 action 和 action creator 来描述状态的改变 ➔ 创建 *Reducer* 返回状态 ➔ 使用 *createStore()* 来创建 Redux *store* 保持全局状态 ➔ 利用 store API: *getState(), dispatch(action), subscribe(cb)* ➔ 使用*React DevTools* 来调试 Redux 状态
React 与 Redux	➔ 利用 react-redux 绑定来扩展应用功能 ➔ 使用 Provider 将store传递给组件树 ➔ 使用 *currying* 提供部分输入的功能 ➔ 使用 *connect()* 来访问提供者设置的 store 上下文
构建 Redux Store	➔ 通过 *reducer合成 以及 状态标准化* 来优化应用程序 ➔ 使用 *combineReducers() 构建一个* Redux store ➔ 构建 Redux store 来维护 “单一来源”
Redux 中间件	➔ 确定在应用程序中实现中间件的好处 ➔ 在 Redux 循环中识别中间件的角色 ➔ 将中间件应用于 Redux 应用程序 ➔ 利用 *Thunk* 中间件来支持异步请求 ➔ 认识到构建应用程序目录的有效方法

实战项目2: 真心话应用

利用 Redux 构建一款“真心话”应用。你将从头构建这款动态应用，并结合使用 Redux 的状态管理功能和 React 的声明式组件模型。完成后，你将能够提交新的问题，回答现有的问题并查看结果。

第三部分: React Native

在课程中，你将学习如何开发在 iOS 和 Android 设备上运行的 React 应用程序。 我们将探索一切，包括建立适当的开发环境、构建和设计跨平台的移动应用程序。 你将整合本地 API（如地理位置和本地通知），甚至学习如何让你的应用程序准备好在 Google Play 商店和 App Store 上线！

课程标题	学习目标
使用 React Native 构建应用	➔ 确定 React Native 背后的意识形态 ➔ 建立一个理想的开发环境 ➔ 检查和调试应用程序
React 与 React Native	➔ 识别 Web应用 和 原生应用 之间的根本区别 ➔ 识别 Android 和 iOS 平台之间的差异 ➔ 利用普通的 React Native 组件 ➔ 在 React Native 应用程序中创建表单 ➔ 利用 AsyncStorage 来保存全局应用程序数据 ➔ 合并 Redux 来管理共享的应用程序状态
样式与布局	➔ 在 JS 中使用 CSS 来设置应用程序的样式 ➔ 识别样式与内联样式，对象变量和 Stylesheet API 之间的差异和用例 ➔ 了解 CSS flexbox 的核心理念和技术 ➔ 学习 Web 上的 Flexbox 与React Native 的 Flexbox 实现之间的主要区别 ➔ 掌握专业人员如何处理 styling 的最佳做法
导航	➔ 通过 React Native 应用程序管理导航 ➔ 使用 StackNavigator 向堆栈中添加和移除新屏幕 ➔ 实现 TabNavigator 通过使用选项卡切换屏幕 ➔ 使用 DrawerNavigator 在抽屉菜单中切换屏幕
原生功能	➔ 利用原生 API 来扩展应用功能 ➔ 为应用程序添加 地理位置、动画、通知和 ImagePicker 功能 ➔ 准备将应用程序上传到 Google Play 商店 和 App Store

实战项目 3:手机单词卡应用

你将开发一款手机应用（支持Android或iOS系统 - 或二者均支持），支持用户学习单词。通过本应用，用户能够创建不同类别的单词卡集合，即“卡片集”；在卡片集中添加单词卡；随后针对卡片集的内容进行测试。