Developing Front-end Apps with React

Table of Contents

[MODULE 1 Building Rich Front-End Applications with React and ES6 2](#_Toc180777953)

[Introduction to TypeScript 2](#_Toc180777954)

[ES6 versus Typescript versus JSX 3](#_Toc180777955)

[First React Applicationº 3](#_Toc180777956)

[Module 1 Summary: 3](#_Toc180777957)

[Module 1 Glossary: 3](#_Toc180777958)

[Cheatsheet - Building a Rich Front-End Application using REACT & ES6 5](#_Toc180777959)

[MODULE 2 React Components 5](#_Toc180777960)

[Module 2 Summary 5](#_Toc180777961)

[Module 2 : Glossary - React Components 5](#_Toc180777962)

[Cheatsheet - React Components 6](#_Toc180777963)

[MODULE 3 Advanced React 7](#_Toc180777964)

[Hooks versus Redux 7](#_Toc180777965)

[React useEffect Hooks 7](#_Toc180777966)

[Module 3 Summary 7](#_Toc180777967)

[Glossary - Advanced React 7](#_Toc180777968)

[Cheatsheet - Advanced React 8](#_Toc180777969)

[Module 3 Lab 8](#_Toc180777970)

[MODULE 4 Developing Front-End Apps with React 9](#_Toc180777971)

[Module 4 Lab 1 9](#_Toc180777972)

[Module 4 Lab 2 9](#_Toc180777973)

## MODULE 1 Building Rich Front-End Applications with React and ES6

## Introduction to TypeScript

**Learning Objectives**

After reading this article, you will be able to:

* Differentiate between JavaScript, JSX, and TypeScript
* Describe the advantages of using TypeScript for React applications rather than JavaScript

Now that you have been introduced to JSX, let’s discuss TypeScript. TypeScript is a superset of JavaScript, meaning all JavaScript is TypeScript, but not all TypeScript is JavaScript. React allows you to choose whether to use either JavaScript or TypeScript. Before we discuss why you may want to choose TypeScript over JavaScript, let’s learn a little more about TypeScript.

TypeScript is a compiled language that supports type-checking. TypeScript is statically typed. This means that variables are static. Once they are defined, a variable’s type, such as “num” or “string,” cannot be changed. A variable that is declared a number cannot later take a string value. Using TypeScript can save a lot of headaches later to help avoid “run-time errors” when the code is being run or avoid hard-to-identify bugs during testing. With TypeScript many of these errors are identified as “type errors” during compilation rather than “undefined” bugs at run-time. During compilation of the Typescript code, this is called “type checking,” where the compiler ensures that once a variable is declared, it is not reassigned to another data type.

Now let’s explore the relationship between JSX and TypeScript. Recall that JSX provides additional syntax to JavaScript, allowing you to write HTML-like code for JavaScript. JSX requires a compiler to translate the JSX into JavaScript. Babel is a popular compiler for JSX.

TypeScript supports embedding. This means you can embed HTML directly into TypeScript, and the compiler will translate the HTML and the TypeScript into JavaScript at compile time, similarly to how the Babel compiler translates JSX into JavaScript.

There are a couple of compilation differences, though, if you choose to compile your JSX or JavaScript with Babel vs. the TypeScript compiler. First of all, Babel does not support type-checking. Secondly, the TypeScript compiler compiles the entire project simultaneously rather than one file at a time. This means that type errors are caught amongst different files rather than only within a single file.

There are several advantages to using TypeScript rather than JavaScript in your React application and possibly a few drawbacks. Regarding advantages:

* TypeScript makes it easy to define Prop types in React. This makes writing code with an IDE that supports code completion a breeze since the IDE automatically populates the Prop type.
* Most common third-party libraries support TypeScript definitions.
* As mentioned, TypeScript has static type-checking, which enables you to help identify errors earlier in the development process.
* Refactoring code becomes easier since types are known.
* There will be fewer “undefined” errors at run-time due to type-checking at compile time.
* Code is easier to read and maintain than JavaScript due to typed variables.
* You can still write JavaScript and use the TypeScript compiler.

There are also possibly a few challenges with using TypeScript rather than JavaScript for React applications:

* There is a learning curve
* Compilation takes a little longer than using Babel with JSX
* Third-party libraries could be missing Type definitions, though this is rare because TypeScript has grown very popular

**Next Steps**

You can find more information about TypeScript in this [link](https://www.typescriptlang.org/).

## ES6 versus Typescript versus JSX

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ0QwMjEwRU4tU2tpbGxzTmV0d29yay9sYWJzL1JlYWRpbmcvRVM2X3ZlcnN1c19UeXBlc2NyaXB0X3ZlcnN1c19KU1gubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImFkbWluIjpmYWxzZSwiaWF0IjoxNzA3NzI3OTIxfQ.T8u7vd95KCujCK2dpTXWFj2xV6lohmtsFumRDbUJCLM>

## First React Applicationº

* cd /home/project
* npx create-react-app myfirstapp
* cd myfirstapp
* rm src/App.css src/App.test.js src/index.css src/logo.svg src/reportWebVitals.js src/setupTests.js
* npm start

## Module 1 Summary:

* React is an efficient, flexible JavaScript library for building user interfaces.
* New features introduced in JavaScript as a part of ES6 are let, const, arrow functions, promise, and class.
* The main benefits of using JSX are that you can leverage the full power of JavaScript in HTML and avoid learning or using a templating language. It allows React to show useful error and warning messages.
* The four types of React components are Functional, Class, Pure, and High-order component.
* Functional components are most useful when the lifecycle of the component doesn’t have to be managed.
* Class components are more versatile.

## Module 1 Glossary:

|  |  |
| --- | --- |
| Term | Definition |
| Angular | Angular is a platform for building mobile and desktop web applications. |
| App.js | Contains App, which is the root React component that you will add into your HTML page. |
| Arrow functions | Allow you to declare functions the same way you declare variables. |
| Babel | A special in-memory tool that compiles JSX and interprets it as JavaScript. |
| Class | A template or blueprint for creating objects. |
| Class component | Use JavaScript ES6 classes to create class-based components in React. |
| Component | Core building blocks of React. |
| const | Allows you to declare constants whose values cannot be changed. |
| CSS | Cascading style sheets. |
| Create React App | Simplifies the process of creating React applications. |
| DOM | Document object model. |
| ES | ES is short for ECMAScript, which is a standards organization that creates a wide range of global information and communications technology standards. |
| ES6 | JavaScript adheres to ECMA's specification EcmaScript6. |
| ECMAScript | A standard used for client-side scripting. |
| event | Enable the component to manage document object model (DOM) actions as a result of user interaction on the system. |
| Front-end frameworks | Are used to create a dynamic client that can connect to the server. |
| Functional component | Created by writing a JavaScript function. |
| High-order component | Are used to share logic with other components |
| HTML | Hypertext markup language. |
| index.js | Where you will add App to the HTML. |
| JSX | JavaScript XML which resembles HTML. |
| let | Allows you to restrict the scope of variables within the block where they were declared. |
| Local scope | Is the limited scope. |
| Promise | Represents the eventual completion of an asynchronous operation and its return value. |
| Props | Used to pass data from a parent component to a child component. |
| Pure component | Do not depend on or modify the state of variables outside their scope. |
| ReactDOM package | This is the glue between React and the DOM. |
| React package | Holds the React source for components and their states and properties. |
| React elements | Used to render the component to DOM. |
| src folder | The main folder in which you will make changes. |
| state | An object that describes how the component will behave and render currently. |
| subclass | The class that is inheriting one other class. |
| superclass | The class being inherited by the subclass. |
| this | Keyword that refers to the current object. |

## Cheatsheet - Building a Rich Front-End Application using REACT & ES6

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTS1DRDAyMTBFTi1Ta2lsbHNOZXR3b3JrL0NoZWF0U2hlZXRzL00xX0NoZWF0U2hlZXQubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImFkbWluIjpmYWxzZSwiaWF0IjoxNzAwNjcwMTc0fQ.Y5M4ZP09oTwvOeD9pCQeHZVxNsjCV78Y48-kJq0m8_g>

## MODULE 2 React Components

## Module 2 Summary

* State is a plain JavaScript object used by React to represent information about the component’s current situation.
* Props is short for properties, and they are used to pass data between React components in a uni-directional flow from parent to child.
* You can pass data between components from parent to child using properties, from child to parent using callbacks, and between siblings.
* Components are created or mounted on the DOM; they grow by updating and then die or are unmounted on DOM. This is referred to as a component lifecycle.
* React components can be tested using Mocha, Chai, Sinon but preferred approaches are by using Jest and React Testing Library.

## Module 2 : Glossary - React Components

|  |  |
| --- | --- |
| Term | Definition |
| Actions | JSON objects that contain information about changes that need to be made to the state. For example, a GET action obtains information, a POST action sends information, and a DELETE action removes information. |
| Callback | A normal method called once a condition is met. |
| Chai | An object that can be used as the assertion library. |
| Component Lifecycle | A series of four phases make up the lifecycle of a component. These phases are: Initialization, Mounting, Updating, and Unmounting. |
| Component testing | When you render component trees in a simple test environment and assert their output. |
| Cross-origin requests | Requests that come from multiple sources. |
| Document Object Model (DOM) | It defines the logical structure of documents and the way a document is accessed and manipulated. |
| End-to-end testing | A multi-step test that combines multiple units and integrates all tests into one big test. It requires the running of a complete application in a realistic browser environment. |
| Enzyme | A library that can be added to your testing tools in order to render your React components |
| Initialization phase | The first component lifecycle phase. During this phase, the component is constructed with the given Props and default state. |
| Jest | A JavaScript test runner and assertion library used to test React components. It can be combined with Enzyme or the React Testing Library. |
| Local state | One of the two types of React states. Local state lives in a single component and is not used in other components. An example of local state is hiding and showing information. (See **Shared state**.) |
| Middleware | Library that connects applications and services. |
| Mocha | Object that can be used as the test runner. |
| Mounting Phase | The second component lifecycle phase. During this phase, the JSX is rendered. |
| Props | Short for 'properties.' Read-only objects that store attribute values of tags. Used to pass data from parent to child components. |
| Push | When a user or software automation moves data from one location to another. |
| React Testing Library | A set of helpers that let you test React components without depending on their implementation details. This library is a replacement for Enzyme. |
| Shared state | One of the two types of React states. Shared state is shared by multiple components and is complicated. An example of shared state is the list of all orders in an order application. (See **Local state**.) |
| Sinon | An object that can be used to test JavaScript logic with objects such as spies, stubs, and mocks. |
| Snapshot testing | Testing that helps check and verify the component rendering result. |
| State | A plain JavaScript object used by React to represent information about the component's current situation. States help track changes in a component. |
| Testing | A line-by-line review of how code executes. Testing replicates end user actions to ensure features work as intended. |
| Updating phase | The third component lifecycle phase. During this phase, the state of a component is updated, and the application is repainted. |
| Unmounting phase | The final component lifecycle phase. During this phase, the component is removed from the page. |

## Cheatsheet - React Components

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTS1DRDAyMTBFTi1Ta2lsbHNOZXR3b3JrL0NoZWF0U2hlZXRzL00yX0NoZWF0U2hlZXQubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImFkbWluIjpmYWxzZSwiaWF0IjoxNzAwNjcwMTc2fQ.Aeu7wzXCwJX1F4eyGwzfRq9ytRtux7djIIi3ffkUPnQ>+

Module 2 Lab

* cd /home/project
* git clone https://github.com/<your Github username>/uqwxd-react\_labs.git
* cd uqwxd-react\_labs/todo\_list/
* ls
* cd todo\_list
* export NODE\_OPTIONS=--openssl-legacy-provider
* npm install --save -s react react-dom react-scripts web-vitals
* npm start

## MODULE 3 Advanced React

## Hooks versus Redux

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ0QwMjEwRU4tU2tpbGxzTmV0d29yay9sYWJzL1JlYWRpbmcvSG9va3NfdnNfUmVkdXgubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImFkbWluIjpmYWxzZSwiaWF0IjoxNzA3NzI5ODkyfQ.FLgxugeEhTB_JsRqQsequaLRbQ3xo1zfW0Zooccei10>

## React useEffect Hooks

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ0QwMjEwRU4tU2tpbGxzTmV0d29yay9sYWJzL1JlYWRpbmcvUmVhY3RfdXNlRWZmZWN0X0hvb2tzLm1kIiwidG9vbF90eXBlIjoiaW5zdHJ1Y3Rpb25hbC1sYWIiLCJhZG1pbiI6ZmFsc2UsImlhdCI6MTcwNzk0MjQwOX0.wxpuoW8g_rw6KX80jkRI6qwP7AHzbiBw_iyzzCwk4-I>

## Module 3 Summary

* Hooks provide a way to use functionalities such as context or state, without classes.
* Inputs in React can be one of two types: controlled or uncontrolled.
* Redux is a state management library that is often used with React to handle the state of your application.
* The Redux elements that are involved in updating the component properties are action, store, and reducer.
* You can interact with asynchronous data in your React Redux app using middleware.
* The data flow in the React-Redux application is unidirectional.

## Glossary - Advanced React

|  |  |
| --- | --- |
| Term | Definition |
| Actions | JSON objects that contain information about changes that need to be made to the state. |
| Action creators | Functions that create actions. |
| Asynchronous or async code | Runs in parallel, and an operation can occur while another one is still being processed. |
| Central Store | Holds the entire application list in the form of the 'state tree'. |
| Custom Hook | A new composition of one or multiple Hooks. |
| Controlled input | Use React to fully control the element by setting and updating the input value directly. |
| onChange | Attribute that controls changes in the form. |
| Hooks | Enable functional components to attach local state with it, and this state will be preserved by React when the component re-renders. |
| Middleware | Used to interact with asynchronous data in your React Redux app. |
| NPM | Package manager for the Node JavaScript platform. |
| Payload property | An optional property that contains some data that is required to perform a task. |
| React Hook form | A form state management and validation library for React web-based applications and React Native mobile applications. |
| Reducer | Pure functions that receive the current state and an Action object and return a new state with the actions performed. |
| Redux | A state management library that is often used with React to handle the state of your application. |
| Ref function | Used to get the form values from DOM. |
| Redux Thunk | Middleware that allows to pass functions within action creators to create async Redux. |
| Redux Saga | Middleware that uses an ES6 feature called Generators to enable async operations. |
| Store | Contains the Redux application's current state, other functions, and objects. |
| setState | Updates state of the form input elements. |
| Synchronous or sync code | Runs in sequence from top to bottom, and each operation must wait for the previous one to complete before executing. |
| Subscription | Is triggered in the components whenever the state is updated in the store. |
| Type property | A string that identifies the action. |
| useState | A hook that allows you to use state in your function. It adds state to a function component. |
| useEffect | Manages side effects such as document changes, HTTP, and so on. |
| useContext | Manages context changes and provides the component with access to a context. |
| useReducer | Manages Redux state changes. |
| Uncontrolled input | Allows the browser to handle most of the form elements and collect data through React's change events. |

s

## Cheatsheet - Advanced React

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTS1DRDAyMTBFTi1Ta2lsbHNOZXR3b3JrL0NoZWF0U2hlZXRzL00zX0NoZWF0U2hlZXQubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImFkbWluIjpmYWxzZSwiaWF0IjoxNzAwNjcwMTc4fQ.zr048CjzoFftzpDMIiEkXSrJ3f6oYtwDxx80C-Pl76I>

## Module 3 Lab

## MODULE 4 Developing Front-End Apps with React

## Module 4 Lab 1

* cd ejtos-react\_budget\_app && ls
* npm install -s

## Module 4 Lab 2