Welcome to Phase 2

(https://github.com/learn-co-curriculum/phase-2-welcome-to-phase-2) (https://github.com/learn-co-curriculum/phase-2-welcome-to-phas
2/issues/ne

SE Grading Scheme

SE Grading Scheme

Assignment Group	Percentage
Required Labs	60%
Required Quizzes	5%
Summative Assessment	35%
Optional Labs	0%
Optional Quizzes	0%

Phase 2: Required Labs

- Putting it All Together: Components and Props
- Putting it All Together: State and Events
- Putting it All Together: React Fetch CRUD Lab
- Basic Routes Lab
- Phase 2 Code Challenge

Phase 2: Required Quizzes

- Introduction to React Quiz
- Components and Props Quiz
- State and Events Quiz

- Side Effects and Data Fetching Quiz
- Client-Side Routing Quiz

Summative Assessments

- Phase 2 Project
- Phase 2 Blog

Learning Goals

- Understand how this phase is structured
- Learn the outcomes of this phase

Intro to Phase 2

Welcome to Phase 2! In Phase 1, you learned the fundamentals of JavaScript as a language, and how to use JavaScript to manipulate the DO respond to user events, and make network requests. In this phase, you'll be introduced to the most popular frontend library of the last several years: React.

React itself is written in JavaScript as a tool for JavaScript developers, so all the fundamentals you learned — working with functions; manipulating arrays; and dealing with complex objects — are going to be crucial to your success with React.

What you'll get in return is a library that does a lot of the heavy lifting in terms of handling the DOM manipulation side of your applications.

React comes with strong opinions about how your code is written, in the form of a *component-based* architecture. Your success in this phase w come down to how well you adapt to React's philosophy and pattern of writing code, and how well you can apply your JavaScript knowledge.

Phase 2 Structure

This phase is structured around first learning the React fundamentals, then adding complexity by learning a bit about the React ecosystem and other tools that you can use with React to make more advanced websites. Each module of content (a section of labs and readmes) ends with a quiz or "putting it all together" lab where you can check your understanding of the material. Here's what each module covers:

Introduction to React

Get an overview of what React is all about and its philosophy, as well as the tools needed to get a React application up and running with functionality provided by different JavaScript packages.

Components and Props

Components are the building blocks of every React application. Learn how to separate out your UI into small, reusable bits of code known as components. You'll also learn about JSX, which is React's declarative syntax for representing the DOM; and props, which is how data is sharec between different parts of your application.

By the end of this module, you'll be able to build a *static* website with React components.

State and Events

React's state system is where most of the "magic" happens in a React application. Learn how to use the `useState` hook, and how state can re render DOM elements in response to events. By the end of this module, you'll be able to build a *dynamic* website that changes in response to user events.

Side Effects and Data Fetching

A component's "main effect" is to produce some DOM elements; however, it can also be useful to handle some "side effects" as part of your component's rendering logic. In this module you'll learn how to use the `useEffect` hook with components to perform side effects, and interact with an API to fetch data that your components can use. By the end of this module, you'll have covered all the core React fundamentals and wi be able to make a single-page application in React.

[Optional] React Under the Hood

In this optional module, you'll learn a bit more about the tools that help React do its job, like Babel and webpack. You'll also learn about the process of "Reconciliation" in React, and how React is able to manage changes to the DOM in a highly performant manner. This module is less hands-on, but it'll help your mental model of how React actually works (and will definitely help with React interview questions down the road).

Client-Side Routing

For most applications, having different pages and some navigation is essential. Learn how to create multiple "pages" in a single-page applicati that each have their own unique URL, thanks to a popular tool in the React ecosystem, React Router.

[Optional] Advanced Hooks

You can get a lot of mileage out of the two hooks we've taught up to this point: `useState` and `useEffect` . However, there are a few other hooks that cover different cases that will be discussed in this module. We'll also cover how to create your own custom hooks and abstract som of the logic for managing state and effects outside of your components.

[Optional] Class Components

For many years, the only way to work with state and lifecycle methods in React was to use **class components** instead of function component Even though the React team recommends writing function components nowadays, it's very likely that you'll come across legacy code, blog pos and documentation that uses class components instead. In this module, we'll show how to refactor function components to class components t help learn the similarities and differences in syntax.

Additional Practice

This module is full of optional practice challenges. In general, these practice challenges are meant to be used after you've completed the **Side Effects and Data Fetching** section to practice with building features of single-page applications that interact with an API.

Phase 2 Objectives

By the end of this phase, you should be able to:

- Design a React component hierarchy based on a wireframe
- Use React to create components that interact with an API
- Incorporate client-side routing into a single-page application in React

We've got an exciting time ahead, so let's get started!