High-Level Program Design

React Development Accelerator

GENERAL ASSEMBLY

Scale Engineering Capabilities

React Accelerator

Instructor-Led | Onsite or Remote | ~35 Hours

Overview:

Upskill developers to use React — the popular and versatile JavaScript library — to add speed, flexibility, and simplicity to your front-end stack. Participants will learn how to create and deploy React applications that incorporate custom components, UI routing, and can plug into a back-end API.

Prerequisites:

Students must be comfortable with JavaScript.

Business Outcomes:.

- Expand the skillsets of your existing front-end engineers.
- Speed up development times for front-end projects.
- Modernize your tech stack with this simple-yet-powerful library.

~35 Hours of React Training

React Fundamentals	React State	Functional Components	APIs, Heroku & Routing	Redux/Hooks
Apply React fundamentals to solve common user interface problems.	**Understand the concept of state** in React, and how to manage it.	**Implement functional components** and define the component lifecycle.	**Learn to make API calls,** deploy an app on Heroku, and use React Router to link components.	**Become familiar with the Redux** state management library and hooks.

Why React Development Accelerator?

- **Validated tools and approaches** for programming in React, developed in partnership with top organizations.
- **Built with subject matter experts** with experience in JavaScript and React development.
- 35 hours of expert-led, hands-on learning, including:
  - **Projects, labs, and assignments** that mimic real-world tasks and workflows.
  - **Case studies and examples** that demonstrate how companies leverage the technology.
  - **Regular feedback and touchpoints** with instructors and peers to ensure students meeting learning goals.

Learner Personas for React Development Accelerator

This product is specifically designed for the following audience:

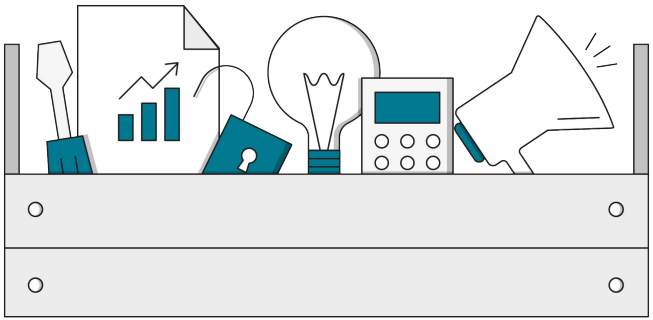
- **Junior- to mid-level developers** with a working knowledge of HTML, the Document Object Model (DOM), and JavaScript programming.

Anatomy of React Development Accelerator

Unit	What's Covered	Unit Project
Unit 1: React Key Components (7 hours) Introduce the fundamental concepts of React	• Props • Nest Components	Film Project Part 1
Unit 2: React State (7 hours) How to use and manipulate State in components	• State • Functional Components	Film Project Part 2
Unit 3: Underlying Concepts (3 hours) A closer look at the behind-the-scenes action of React	• Component Life-cycle • Unidirectional Data Flow	Film Project Part 3
Unit 4: APIs and Heroku (3 hours) How React components can interact with data sources	• Fetch • Heroku Deployment	Film Project Part 4
Unit 5: React Router (4 hours) Adding URL routing to React apps	• React router	Dentist Website
Unit 6: Redux/Hooks and Applied Practice (11 hours) Briefly introduce the Redux state management library and Hooks	• Redux • Hooks	Final Project

Tools Used in the Course

- Slack for communication
- Github for course materials
- Zoom for class sessions (if remote)



React Development Accelerator

What You'll Learn

Module Overview

React Key Concepts

7 hours 

Why?

In this hands-on unit, we'll explore the concept of nested components, which will enable us to create more complex user interfaces, and apply it directly to a blog application in a lab exercise.

Learning Objectives

- Apply React fundamentals to solve common user interface (UI) problems.
- Render components within another component.
- Pass props to a nested component.

In Your Blog...

Next, we'll put comments inside an individual `Post` component. To do this, we can reference a comment using `<Comment />` inside of `Post`'s `render()` method.

- Starting from the blog post code, let's create a new file for a `Comment` component, `src/Comment.js`:

```
import React, {Component} from 'react';

class Comment extends Component {
  render () {
    return (
      <div>
        <p>{this.props.body}</p>
      </div>
    )
  }
}

export default Comment
```

Discussion: What have we done?

Teaching Tips:

- Elicit answers before discussing the following talking points.

Talking Points:

- We've defined our component class, which inherits from `React.Component`.
- We're exporting this `Comment` class by default for anything importing this file.
- We are returning JSX that contains a paragraph displaying a `body` prop (which will be passed in).

Creating a Newsfeed Component

Instructions

Estimated Time: 45 min

<iframe height='500' scrolling='no' title='React - Creating A NewFeed Component - Starter' src='https://codepen.io/jkeohan/embed/PBvxvY/?height=500&theme-id=0&default-tab=css,resultundefined&editable=true' frameborder='no' allowtransparency='true' allowfullscreen='true' style='width: 100%;'>See the Pen React - Creating A NewFeed Component - Starter by Joe (@jkeohan) on CodePen. </iframe>

Instructions

1. In the HTML, look for a section element with an ID of `main`. Nested within it, you will find three articles.
2. Using the sources array (defined below), create an app component that will recreate the section elements.

In the Creating a Newsfeed Component CodePen, you will find three article elements within the HTML section generating the newsfeed articles visible on the page.

Perform the following to complete the lab:

- Create an array called `newsFeedData` that will contain three objects with the following properties: `title`, `tags`, `image`, and `impressions`.
- Populate those objects based on those values assigned in the HTML.
- Create a `NewsFeed` component and pass it the `newsFeedData` array as `props`.
- The `NewsFeed` component will then render three `articles` based on properties of the objects in the `newsFeedData` array.

Solution: https://codepen.io/GAmarketing/pen/WYjmgx

Module Overview

React State

7 hours 

Why?

In React, data can be handled in two ways — with props or state. To manage dynamic data, state is the way to go!

In this unit, we'll use React's state to create edible blog entries for the blog application.

Learning Objectives

- Differentiate between props and state.
- Create and change state in a component.
- Describe the flow of methods in a component.
- Identify the triggers for the re-rendering of a component.
- Contrast class components with functional components.
- Define unidirectional flow.
- Diagram data in a component hierarchy

MoodTracker

```
// ...
import MoodTracker from "./MoodTracker";

class App extends Component {
  render() {
    // ...
    return (
      <div className="App">
        ...
        <MoodTracker />
      </div>
    );
  }
}

export default App;
```

Talking Point:

- After creating the file, let's make sure to import our new component into `App.js` and put the component in our `render()` method.

Teaching Tips:

- In this intro, make sure to mention that the concept of state is not React-specific. Any values that are stored and manipulated on the front-end can be considered the state of the application.
- It's worth spending extra time reiterating that props are **immutable** and cannot be changed.
- Use the existing React repo to show what happens if you directly manipulate props. Make sure the students do not code along and instead are focused on seeing what happens on your machine. Your error message should looks something like this:

Arrays Can Mutate!

```
const anArray = ['oneFish', 'two fish', 'red fish', 'blue fish']
anArray.pop() // new value of anArray is ['oneFish', 'two fish', 'red fish']
```

Talking Points:

- Arrays are also an object data type, meaning they can be mutated.
- Here is an example of mutating an array.

Other Array Methods That Can Mutate The Array That Calls Them

- `push()`
- `reverse()`
- `unshift()`
- `splice()`

Teaching Tip:

- Here is an additional resource you can share: https://medium.com/@fknussel/arrays-objects-and-mutations-6b23348b54aa

Module Overview

Underlying Concepts

3 hours

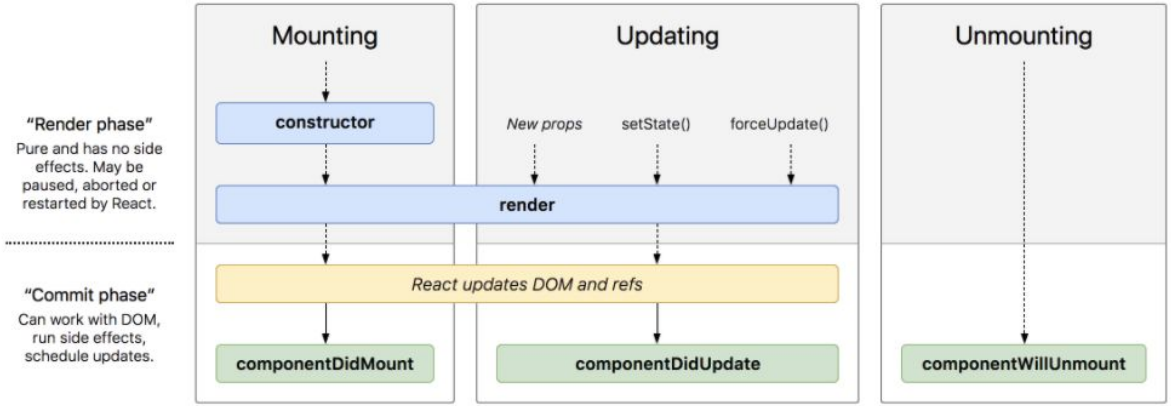
Why?

Did you know you can control your application based on the state of the UI? You can do so by using several life-cycle methods provided by the React class components. In a lab exercise, you'll also work on implementing functional components to an application.

Learning Objectives

- Rewrite class components into functional components.
- Define the main categories of the component life cycle.
- Identify general methods in each category of the component life cycle.
- Contrast the concepts imperative and declarative programming.

Let's Examine Each Category



Talking Points:

- Here they are in diagram form.
- Again, you don't need to write these methods — they happen automatically, just like constructors did before we explicitly wrote one. You only have to worry about these if you want to change something in them. But, if you do, it's important to understand them!
- Note that `getDerivedStateFromProps()`, `shouldComponentUpdate()`, and `getSnapshotBeforeUpdate()` are not included in this diagram because they are less commonly leveraged.

Imperative Programming

- Finally, we have them speak their name — or in our case, we'll just `console.log()` their name out as represented by the `person` variable.

```
const room = ['Superman', 'Black Panther', 'Wonder Woman', 'Iron Man'];

for (let i = 0; i < room.length; i++) {
  const person = room[i]
  console.log("Imperative way: " + person)
}
```

Teaching Tips:

- For convenience, slack this CodePen URL to students.

Talking Points:

- Run this in CodePen and check the Console tab. It works! We've built this function using the *imperative* way of thinking.
- We have explicitly listed out each step, paying attention to each and every detail of what we needed to achieve.
- Imperative programming focuses on the *why*, *how*, *where*, and *when* your code executes. It allows precise control over your code and allows for line-by-line code execution; you're writing every single thing that happens.

Module Overview

APIs and Heroku

3 hours 

Why?

API, which stands for “application programming interface,” can be used to define everything from URL endpoints to DOM methods. In your career as a developer, you’ll encounter API countless times. As practice, we’ll use an API to display the current weather on your blog application.

Learning Objectives

- Describe what an application programming interface (API) is and why we might use one.
- Using fetch() to make an API call and working with API keys.
- Describe Heroku.
- Deploy an app on Heroku.
- Set up a CORS proxy on Heroku.

API Exploration

What's the API?	A Sample URL You Can Call
Giphy's API: Request a list of all funny cats.	http://api.giphy.com/v1/gifs/search?q=funny+cat&api_key=dc6zaTOxFJmzC
The Star Wars API: Request R2-D2 info.	http://swapi.co/api/people/3
Markit Digital's API: Request current Apple stock info.	http://dev.markitondemand.com/Api/Quote/xml?symbol=AAPL

Talking Points:

- A variety of APIs are available on the internet. To call an API, send a request to a URL and that API will return data to your program. You can pull data from any web app that offers an API.
- You can make this request as specific as you'd like. Each web app offers different options for its API; you just have to find out what you can request from it.
- These are just a few examples of APIs you can use. Check it out — the left column is the common name you might know. The right column is the URL you would send a request to in your program.

Teaching Tips:

- Does the JSON look unreadable in the browser? If you're using Google Chrome, install the JSONView plugin.
- Click the URLs in the table to show students what each call would return.
- Make sure to also get rid of the API key query parameter when requesting Giphy so you can show what a response looks like when a key isn't provided.
- Unless the class is already versed in making API calls, it's likely that they don't use JSONView. Make sure to check that students have this plugin installed.
- Alternatively, have students download Postman and provide a brief overview.

Heroku Command Line Interface

Download and install the Heroku command line interface (CLI):

https://devcenter.heroku.com/articles/heroku-cli

Talking Points:

- Before we go any further, make sure you stop any currently running React apps.
- Let's download and install the Heroku CLI. You can download the installer from the link provided or run `brew install heroku` in your terminal application.
- This tool is designed to make your life easier as a developer by integrating Heroku application development and deployment directly into the command line.
- You may need to restart your terminal session before moving forward. Why? Because you've just installed software to modify your terminal's environment, and your environment variables may not have updated yet.

Log In To Heroku

```
Enter your Heroku credentials.
Email: adam@example.com
Password (typing will be hidden):
Authentication successful.
```

Talking Point:

- Log in to Heroku using the command line. Do so by typing in `heroku login`. You'll be prompted to enter your Heroku credentials, followed by an authentication message like the one shown here.

Module Overview

React Router

4 hours 

Why?

React Router is a third-party library that makes it easy to route URLs by dynamically loading different components on the same page as the user navigates to different URLs. It also manages browser history automatically. As practice, we'll configure React Router for a dentist website.

Learning Objectives

- Contrast historical and modern browser history mechanics.
- Define routing.
- Describe React Router's main features and history.
- Use React Router to map URLs to components.
- Use React Router to create links to different components.

Creating Our Homepage Component

7. Instead, we need to call our new component. Put `<Home></Home>` inside the `<div>` in the `App` component. This tells the `App` component to render the `Home` component right there inside the div.

> Note: We have been using `<Home />` to call components. `<Home></Home>` is just a different syntax we're showing you so that if you see it elsewhere, you're familiar with it.

8. Let's try it out! Look at the browser and see if the homepage appears. Unfortunately, if you've been following along, it won't. You'll see an error, which should look like this:

```
Failed to compile.

Error in ./src/App.js

/Users/moonmayor/Lessons/dentist-website-runthrough/src/App.js
  8:10  error  'Home' is not defined  no-undef

✖ 1 problem (1 error, 0 warnings)
```

Talking Points:

- We've been editing `App.js`, which defines one component for our entire application. So far our app manually shows just the homepage. Let's refactor this so the content of the homepage is moved into its own component called `Home`.
- It's not enough to simply create the `Home.js` file and create the `Home` component. We must also remember to import the component into the `App.js` file. Any components you want to use in a file need to first be imported into that file.

Creating Routes

React Router uses some of its own components to define how URLs are routed to your components and to create links to those routes. You must have one `<Router>` component that wraps itself around multiple `<Route>` components. Each `<Route>` component has two pieces:

- `path` - defining the URL path that leads to the component.
- `component` - defining what component users will see when they navigate to the path.

Delete what is currently returned in the `render` function of your `App.js`, and replace it with a Router component call with three routes, as shown below.

```
class App extends Component {
  render() {
    return (
      <Router>
        <div>
          <Route exact path="/" component={Home} />
          <Route path="/procedures" component={Procedures} />
          <Route path="/contact" component={Contact} />
        </div>
      </Router>
    );
  }
}
```

Talking Points:

- This is the general syntax for creating routes.

Module Overview

Redux/Hooks & Applied Practice

11 hours 

Why?

As your application becomes larger, it can get increasingly difficult and confusing to manage your app's state across multiple shared components. This is where Redux comes in! It provides a central store that holds the entire state of your application and keeps your code clean and predictable.

Learning Objectives

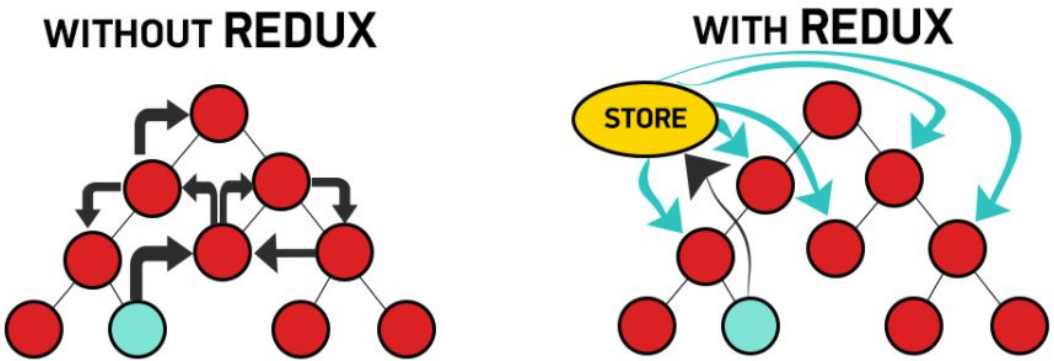
- Define what Redux is and why you would use it
- Explain what actions, reducers and the Redux store is.
- Name benefits of using Redux.
- Define React Hooks and why they were introduced
- Explain different use cases for Redux and React Hooks

Why Use Redux?

As we've learned, React is pretty good at state management without any additional tools. So why use Redux too?

- As your application becomes larger, it becomes increasingly difficult and confusing to manage state across multiple shared components.
- Keep in mind, components are reusable so you might be using these components across several pages or screens within your app.

Imagine multiple components that all need to talk to each other, but for different reasons. You might have a hard time understanding where state should live. In a perfect world, component data should be isolated to that particular component. Now consider sibling components and how you would share that data.



Problems Hooks Solve (Partial List)

- **Reduces Concepts** - Hooks could make your life in React more simple. In React, we often juggle between ideas and concepts. Hooks let you use functions instead of having to switch between functions, classes, higher-order components, and render props.
- **Potential Bundle Reduction** - As a React developer with growing complex applications, managing your bundle size will become a constant battle. Below is an example of the warning thrown when running `npm run build` on an application that bundles over 500kb.

```
Hollys-MacBook-Pro:DoingThingsMedia hollywhite$ npm run build

> doing-things-app@3.5.0 build /Users/hollywhite/Desktop/Freelance/DoingThingsMedia/DoingThingsMedia
> react-scripts build

Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

  557.39 KB  build/static/js/main.b3572a4b.js
  23.75 KB   build/static/css/main.6dfbf9bc.css

The bundle size is significantly larger than recommended.
Consider reducing it with code splitting: https://goo.gl/9VhYWB
You can also analyze the project dependencies: https://goo.gl/LeUzfb

The project was built assuming it is hosted at https://www.doingthingsmedia.com.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  yarn global add serve
  serve -s build

Find out more about deployment here:

  http://bit.ly/2vY88Kr
```


GA