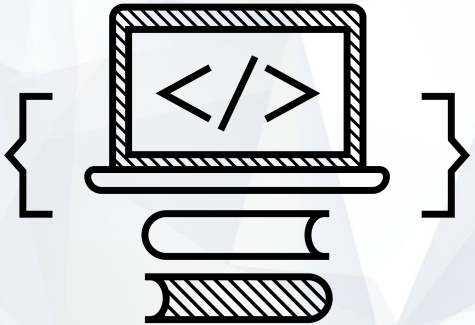
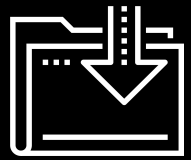


Forms, Conditional Rendering, and React Router

Skills Bootcamp in Front-End Web Development

Lesson 13.3



WELCOME

Learning Objectives

By the end of class, you will:

- Deepen your understanding of managing state with React components.
- Understand conditionally rendering React components.
- Understand the axios library.
- Understand routing with React Router.

Instructor Demonstration

Forms

In this example, we will demonstrate how to handle simple forms with React.

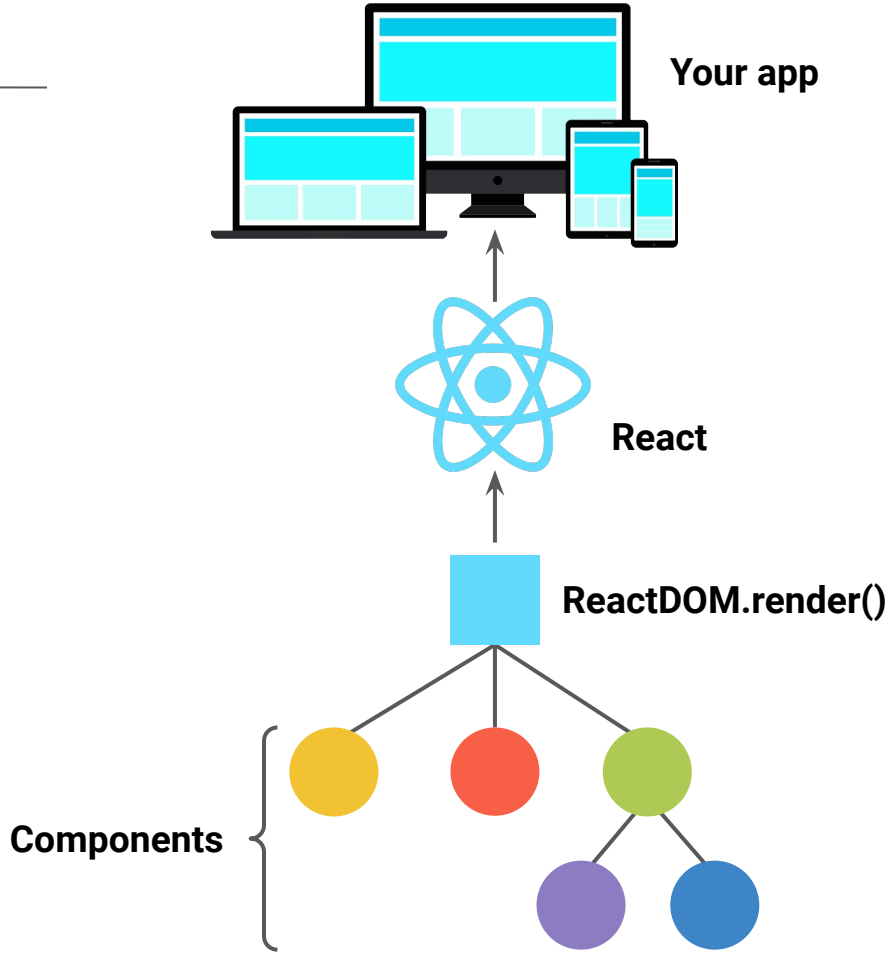
“State” is a property attached to a component and utilized for storing values that we want to associate with it.

A Component's "State"

This property is recognized by React and can be used to embed data inside of a component's UI, which we want to update over time.

Whenever a component's state is updated, it re-renders itself and all of its children.

This updates the application's UI to display the new data without having to refresh the browser.



Questions?

Activity: Fun with Forms

In this activity, you will add some new functionality to the previous form example.

Suggested Time:

15 Minutes

Time’s Up! Let’s Review.

What if we added one more input field under the control of this `handleInputChange` function?

Review: Fun with Forms

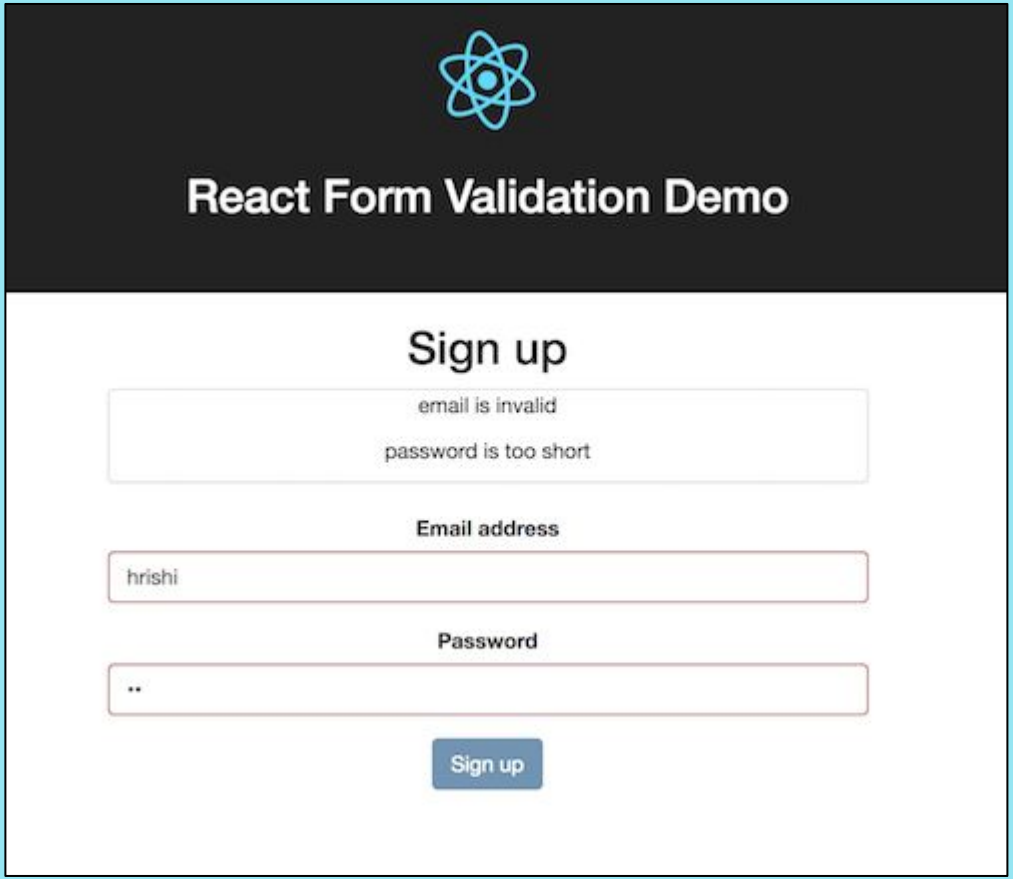
The current setup accounts for the possibility of adding new input fields:

```
const handleInputChange = event => {
let value = event.target.value;
const name = event.target.name;

  setFormData({
    …formData,
    [name]: value,
  });
};
```

Think back to forms you've created in the past.

Is there anything else we should do with our data after setting the application state?

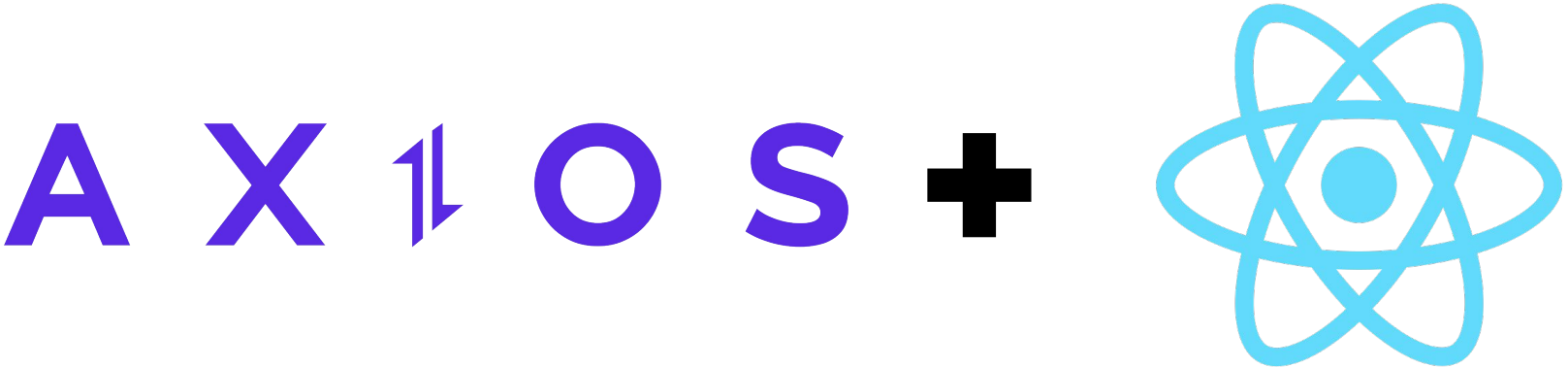


We could add a POST request to `handleFormSubmit` so that the user's input can be sent to a server and saved in a database.

Instructor Demonstration

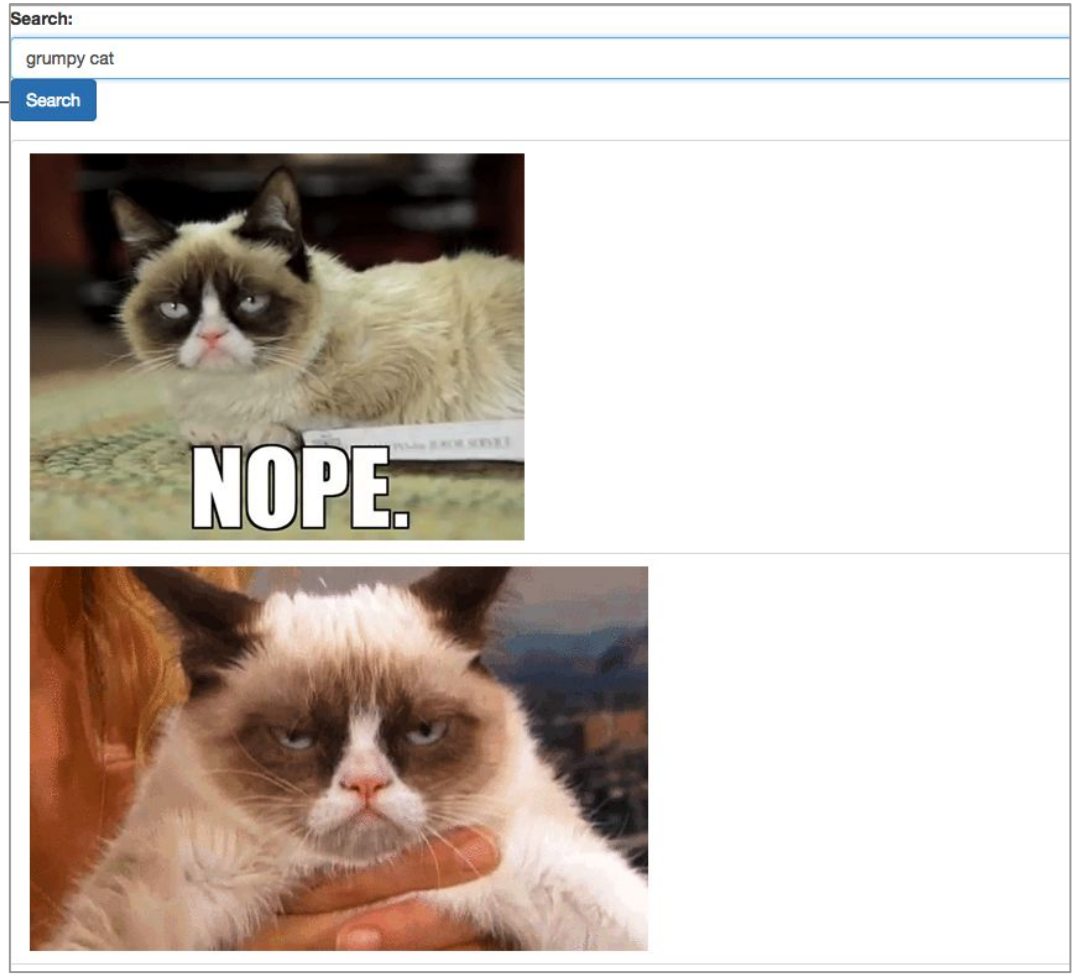
Axios Demo

In this example, we will demonstrate
Axios requests with React.



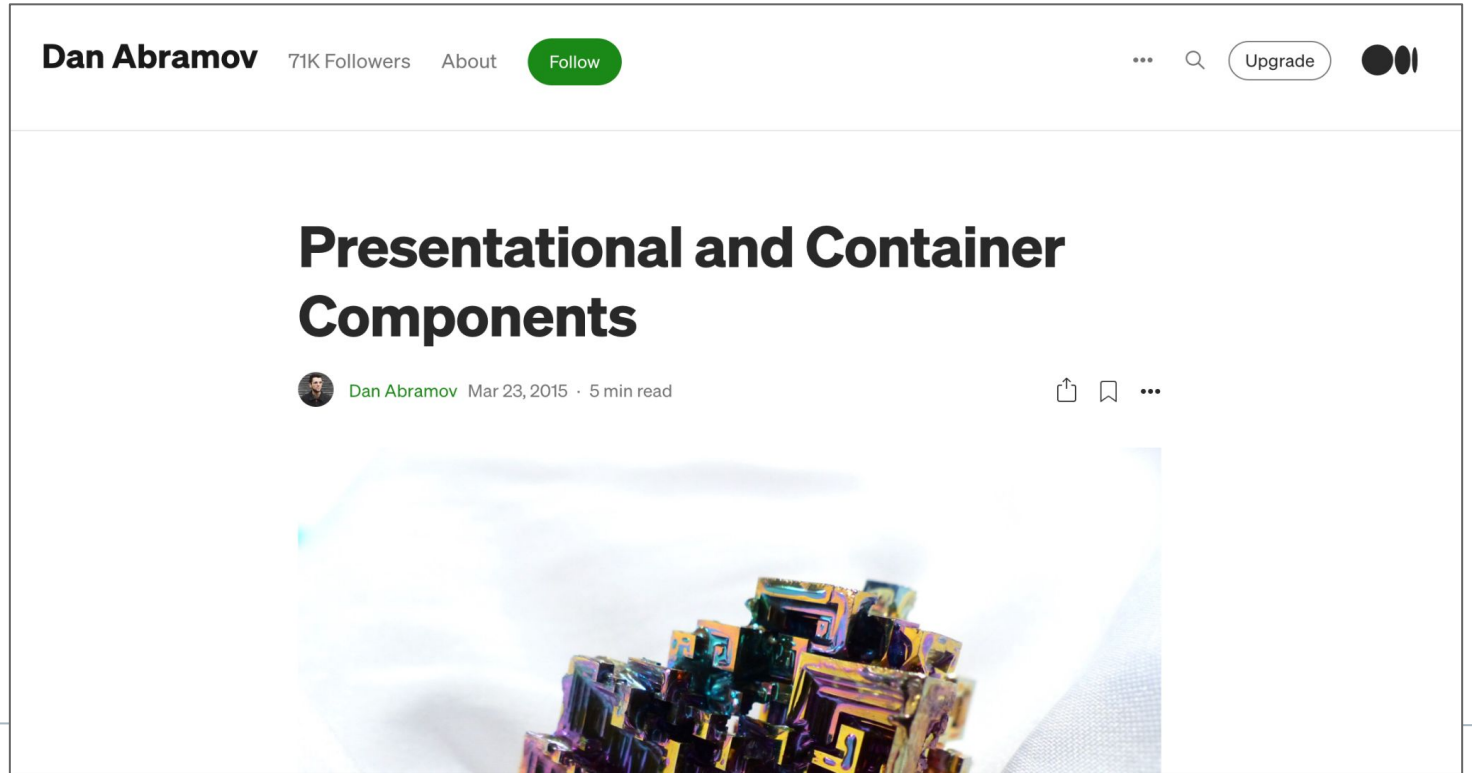
Axios Demo

This app searches the Giphy API for whatever is typed into the input field and then displays the results below.



Axios Demo

On your own time, read this [article written by Dan Abramov]() (Redux Author, React Core Contributor).



Axios Demo

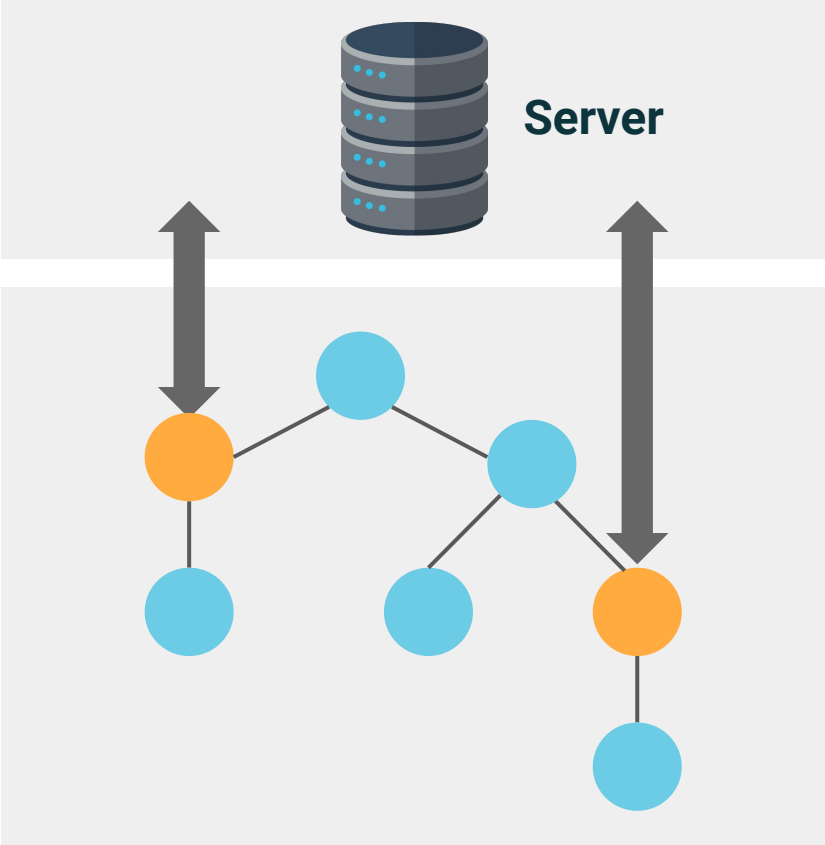
Dan Abramov describes the pattern of separating components into “container” and “presentational” components.

Container components

Are primarily concerned with how things work and render very little, if any, of their own markup. Instead, they mostly render other components and pass down the logic and data they need to work.

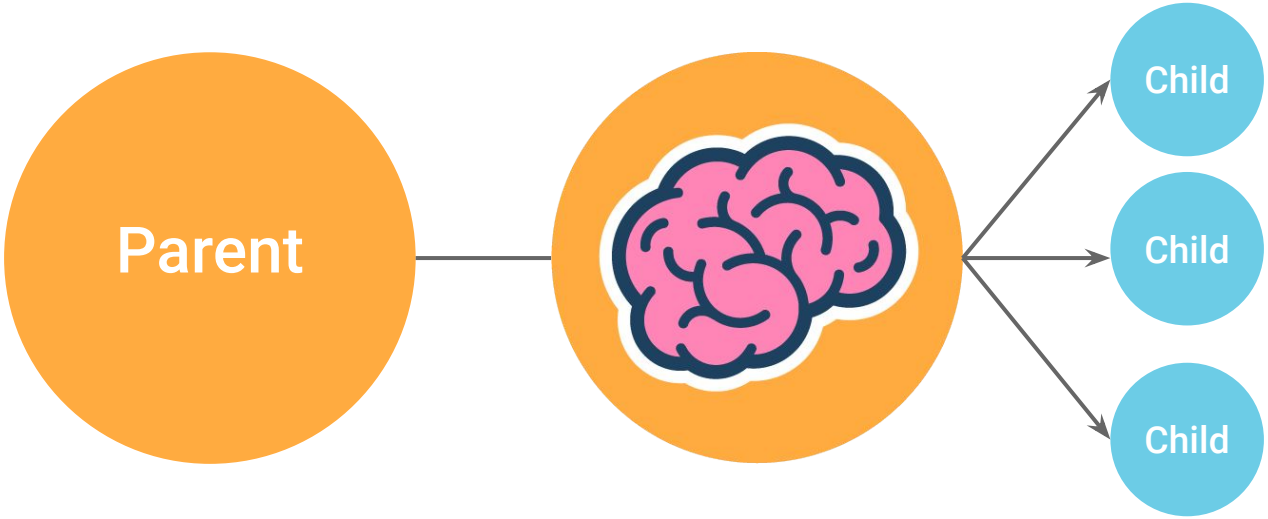
Presentational components

Are concerned with how things look and typically don’t contain any logic that doesn’t have to do with their own individual UI.



Takeaway

There should be a few of these “container” components that act as the “brain” for their children. In our case, this is `SearchResultContainer`.



Takeaways

- The majority of the components in our application should be stateless, focusing mainly on the presentation aspect of our application.
- A smaller number of components should be stateful components that contain part of our application's business logic and state.
- These components generally render the stateless presentational components and pass down data and functionality.
- The Axios library is used to perform our HTTP requests. React does not provide specific methods for creating requests.
- The API.js helper file contains the logic for creating Axios requests, which hides the specific implementation details of making requests to the Giphy API within the components that require them.

Questions?

Questions?

Activity: Axios

In this activity, you will create a simple React application with which users can query the OMDb API and display information about the movie that is searched for.

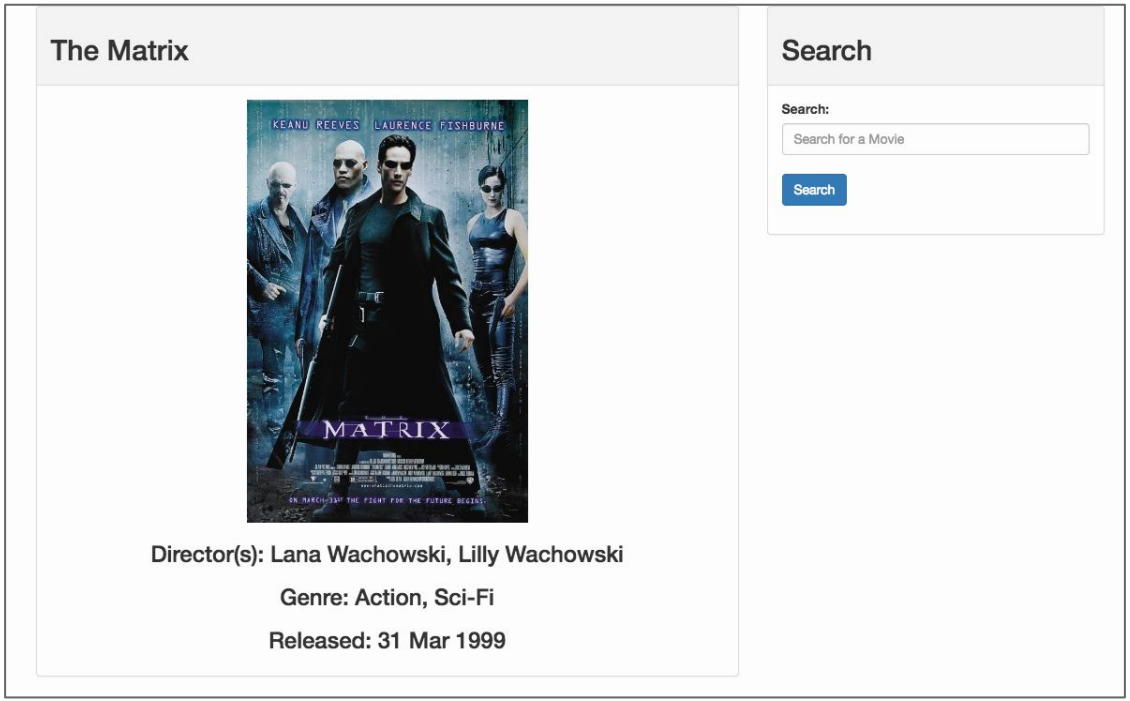
Suggested Time:

15 Minutes

Time’s Up! Let’s Review.

Review: Axios

When we search for a movie using the form on the right side, some information about the movie is displayed in the left card.



Since we definitely only have one input field under the control of this `handleInputChange` function, could we decrease the amount of code being used inside of this function?

Review: Axios

One input field:

```
const handleInputChange = event => {
  setSearchData({
    …searchData,
    search: event.target.value,
  });
};
```

Questions?

Questions?

Activity: Conditional Render

In this activity, you will render one of four different components based on a component's state.

Suggested Time:

15 Minutes

Time’s Up! Let’s Review.

Review: Conditional Render

When we click the different navigation items, a different component is rendered. The address bar doesn't *actually* change when we do this, but we are still rendering different content depending on our application state.

Questions?

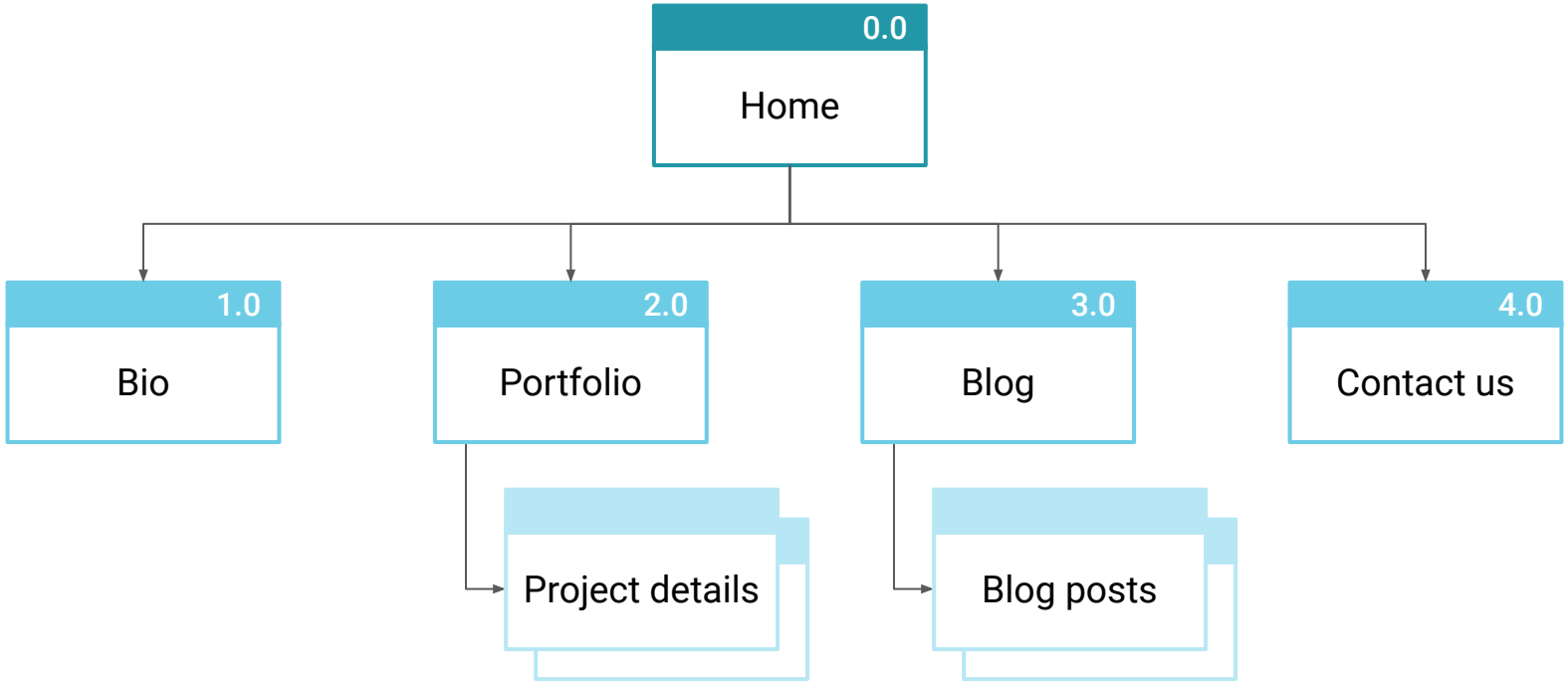
Questions?


Break

React Router

React Router

So far, we've been working with React applications with only one page of content, but in the real world, web applications have multiple—often complex—pages and routes.



What if we deployed the previous activity's portfolio website and we wanted to share a URL with someone that they could use to visit the About "page"?

React Router

Currently, we don't have a way to do that. The user would still have to navigate to the `About` "page" on their own from scratch every time since the URL in our address bar doesn't actually change as we click through the tabs.

This may seem trivial now, but:

- What if our application were as large as Amazon.com?
- What if we wanted to share the URL to a page containing one of millions of different products with someone?
- How would we get users where we intend for them to go?

React Router

Thankfully, we don’t have to code out our own solution to this problem.

One of the most popular React companion libraries out today is React Router.

Getting Started

- Feature Overview
- Tutorial
- Examples
- FAQs
- Main Concepts

Upgrading

- Migrating to RouterProvider
- Upgrading from v5
- Migrating from @reach/router

Routers

- Picking a Router
- createBrowserRouter
- createHashRouter
- createMemoryRouter
- createStaticHandler
- createStaticRouter

What's New in 6.4?

v6.4 is our most exciting release yet with new data abstractions for reads, writes, and navigation hooks to easily keep your UI in sync with your data. The new feature overview will catch you up.

I'm New

Start with the tutorial. It will quickly introduce you to the primary features of React Router: from configuring routes, to loading and mutating data, to pending and optimistic UI.

I'm on v5

The migration guide will help you migrate incrementally and keep shipping along the way. Or, do it all in one yolo commit! Either way, we've got you covered to start using the new features right away.

I'm Stuck!

Running into a problem? Chances are you're not the first! Explore common questions about React Router v6.

React Router is a **library** made up of special components for conditionally rendering other components based on the current URL path.

React Router

React Router has modules for routing React applications:

On the web

In native applications

In native applications

In our case, we're going to be working with React Router on the web.

React Router

While a little intimidating at first, the [React Router documentation]() is some of the best for any library we've covered so far, full of concise and helpful examples.

Getting Started

- Feature Overview
- Tutorial
- Examples
- FAQs
- Main Concepts

Upgrading

- Migrating to RouterProvider
- Upgrading from v5
- Migrating from @reach/router

Routers

- Picking a Router
- createBrowserRouter
- createHashRouter
- createMemoryRouter
- createStaticHandler
- createStaticRouter
- RouterProvider

Client Side Routing

React Router enables "client side routing".

In traditional websites, the browser requests a document from a web server, downloads and evaluates CSS and JavaScript assets, and renders the HTML sent from the server. When the user clicks a link, it starts the process all over again for a new page.

Client side routing allows your app to update the URL from a link click without making another request for another document from the server. Instead, your app can immediately render some new UI and make data requests with `fetch` to update the page with new information.

This enables faster user experiences because the browser doesn't need to request an entirely new document or re-evaluate CSS and JavaScript assets for the next page. It also enables more dynamic user experiences with things like animation.

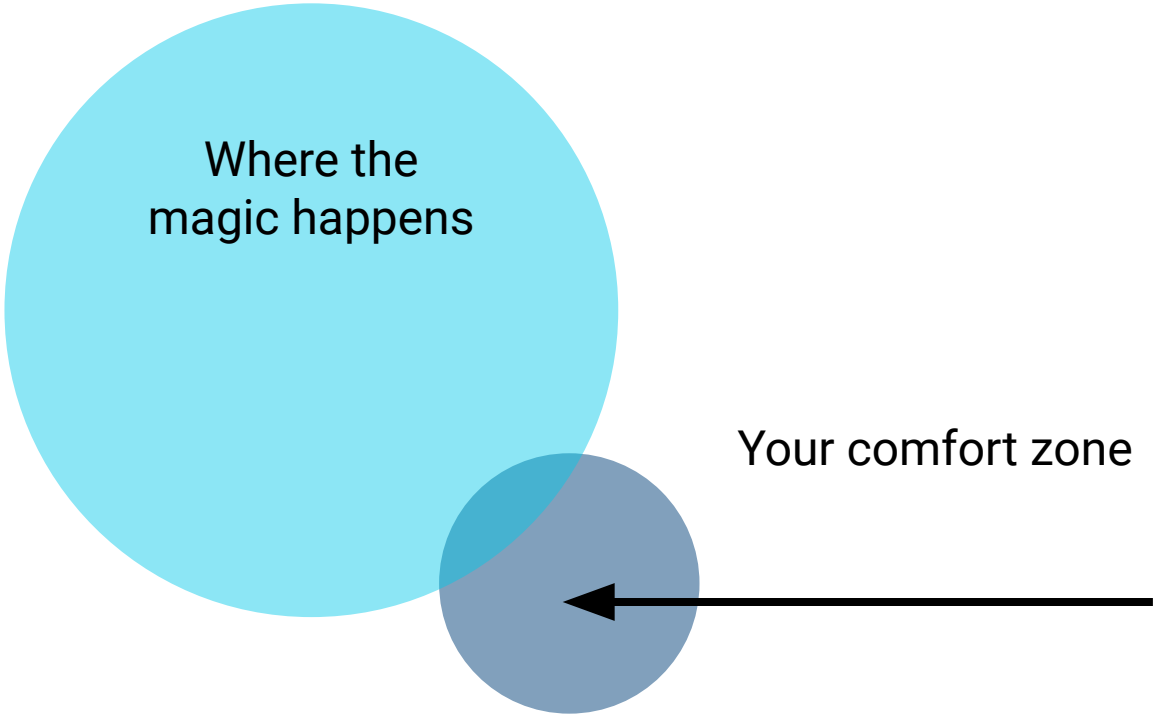
Client side routing is enabled by creating a `Router` and linking/submitting to pages with `Link` and `<Form>`:

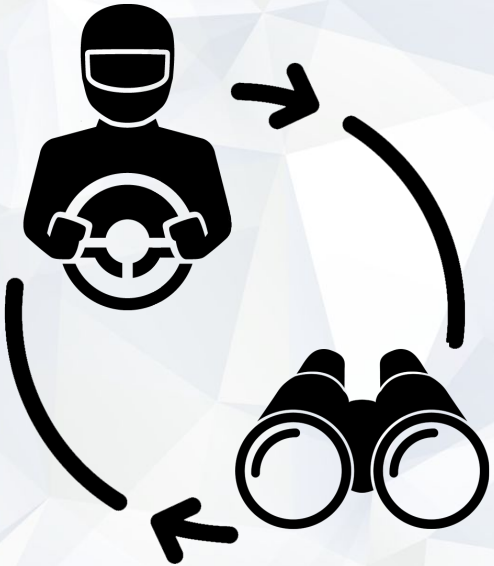
```
import * as React from "react";
import { createRoot } from "react-dom/client";
import {
  createBrowserRouter,
  RouterProvider,
  Route,
  Link,
} from "react-router-dom";
```

On this page

React Router

We'll be going over the fundamental 20% or so of syntax that you're likely going to be using 80% of the time.





Pair Programming Activity:

Pupster App

In this activity, you will work with partners to create a full React application from scratch, complete with routing and Axios requests to the [Dog Ceo API](#).

Suggested Time:

20 Minutes

Time’s Up! Let’s Review.

Questions?

Questions?

The End