**Learning React**

**Section 1: Introduction to the World of React**

React has taken the world of web development by storm. It is used for everything from web apps, to native iOS and Android apps. In this section we’ll learn core React concepts and methodologies to get up and running building interactive web apps.

* The Course Overview
* Thinking in React
* Start Using React Right away
* Local React Dev Environment Setup with create-react-app
* Building Our First Custom React Component
* Using State and Conditional Rendering in JSX
* Using React Dev Tools to Help Build and Debug Our Apps

**1.1 The Course Overview**

In this video, we will get a gist of what the course is all about.

**1.2 Thinking in React**

There are differences in the way React and DOM scripting modify web pages. We identify and explain the differences in this video.

* Break down the major differences into a few bulletpoints
* View diagrams of the different flows
* Understand flow differences between React and DOM Scripting

**1.3 Start Using React Right away**

Getting start with a new platform can sometimes be challenging. Here we identify two browser-based tools to quickly start using React.

* Review different methods of starting a React project
* Demonstrate two powerful web tools to write React
* Feel comfortable experimenting with React at any moment

**1.4 Local React Dev Environment Setup with create-react-app**

There are several steps to take when setting up a local dev environment. We install and run various tools to kick off a React project.

* Install required tools like Git, Node, and create-react-app
* Generate a new project with create-react-app
* Open and edit app in real time with hot-reloading

**1.5 Building Our First Custom React Component**

Getting started with a new tool can sometimes be challenging. In this video, we build a simple component to display user information.

* Start from scratch to build a simple React component
* Hard-code user info and style our first component
* Repeat to seamlessly structure and style future components

**1.6 Using State and Conditional Rendering in JSX**

Adding component state and controlling rendering logic is vital to dynamic web apps. In this video we learn how to populate our component with state and JSX rendering logic.

* Learn how to store info in a component’s state
* Pull and loop through different data types in JSX
* Parse data and populate React components dynamically

**1.7 Using React Dev Tools to Help Build and Debug Our Apps**

The complexity of building web apps often calls for tools to aid the process. Here we learn about React Dev Tools, the official plugin to help on any React project.

* Install and run React Dev Tools in Chrome or Firefox
* Explore features and functionality to help debug our React apps
* Solve state easily and rendering issues during development

**Section 2: Dive into Interactive React Apps**

JavaScript allows us to create highly interactive and dynamic UIs. Using React, we’ll learn how to set state, handle various user and browser events. This will lay the groundwork for more robust components and web apps.

* Handle Mouse, Keyboard, Form, and Browser Events
* Basic Form Validation
* Create a Shopping List Component
* Persist Shopping List State with Local Storage

**2.1 Handle Mouse, Keyboard, Form, and Browser Events**

Handling input, form, and browser events are essential for good web experiences. We will learn how to handle various events in our React components.

* Introduce various event types individually
* Implement event handling logic
* Utilize event handling behavior throughout React components

**2.2 Basic Form Validation**

Validating form data before submission is important. We will learn how to check field values and show error messages when invalid.

* Demo the desired behavior of our form validation
* Implement validation solution to handle form submission
* Make sure empty or malformed info isn’t submitted

**2.3 Create a Shopping List Component**

Interactive apps and components demonstrate what makes React powerful. Building a dynamic shopping list will show us how to piece together our knowledge so far.

* Understand the desired behavior of our shopping list
* Review and implement state modification logic for list items
* Add, delete, and toggle items in our shopping list state

**2.4 Persist Shopping List State with Local Storage**

Refreshing the browser will reset state and erase any state modification. We need to save the state to prevent losing changes on refresh.

* Implement logic state into Local Storage
* Retrieve saved Local Storage data and update the component state
* Use Local Storage to persist state in React when needed

**Section 3: Modular React Components and Shared State**

We take our skills to the next level by nesting React components, and sharing state between components. We will create reusable components, learn about file size and performance optimization techniques.

* Overview:Mini Address Book App
* Use Stateful and Stateless Components to Build UIs
* Pass Props Down to Nested Components
* Update Parent Component State from a Child Component
* Persist Address Book State with Local Storage
* Intro to Hooks
* Implement the useState and useEffect

**3.1 Overview:Mini Address Book App**

Modular code is essential to building complex React apps. We will learn about the app we’re going to build that uses this pattern.

* Learn about the goal of this section.
* Get Visual overview of the completed Address Book app.
* Introduce first step of creating modular components.

**3.2 Use Stateful and Stateless Components to Build UIs**

Stateful and Stateless Components play important roles in React apps. We will learn about their differences, and how to implement them.

* Know the essential differences between Stateful and Stateless Components
* List out responsibilities and differences between Stateless and Stateful Components
* Implement simple examples of Stateful and Stateless Components

**3.3 Pass Props Down to Nested Components**

Passing down props to nested components is essential to render content correctly. We will learn how to pass down props as we map over our contacts.

* Identify the JSX to extract into a separate component
* Cut and paste JSX into separate component
* Make sure app content is still rendering as expected

**3.4 Update Parent Component State from a Child Component**

Nested components need to access and update the parent component’s state. We will learn how to update the parent state from within a nested component.

* Know that Nested component needs to update parent component when clicked
* Create click handler and pass as prop to nested component
* Interact with nested Card component to update parent state

**3.5 Persist Address Book State with Local Storage**

When reloading the page, the state of our React app is lost. We will learn how to save data in the browser’s local storage to persist the state after reloading.

* Reload the page loses our favorites state
* Implement local storage to save favorites state
* Toggle favorites and reload page to view persisted data

**3.6 Intro to Hooks**

In this video, we will get introduced to a new concept of Hooks.

* Learn about in brief what hooks are all about

**3.7 Implement the useState and useEffect**

In this video, we will look into how to implement the useState and useEffect.

* Learn by example how to implement the useState and useEffect hooks using the card component

**Section 4: Multi-Page React Apps with React Router**

In this section, we build a multi-page React app using React Router.

* Overview: Multi-Page React App
* React Router 5 Setup and Configuration
* Handle Dynamic URL Values, and Error State
* Convert Render Props into Components
* Conditionally Render Components, Redirects

**4.1 Overview: Multi-Page React App**

Complex apps often demand a multi-page experience. We will learn about the multi-page React app we’re going to build.

* Learn about the goal of this section
* Visual overview of the completed multi-page app
* Introduce next step of configuring React Router

**4.2 React Router 5 Setup and Configuration**

Routing based on URLs is essential for multi-page web apps. We will learn how to use React Router to handle multiple pages.

* React Router provides components to configure multiple routes
* Utilize provided components to render URL-specific content
* Click links to change to different pages

**4.3 Handle Dynamic URL Values, and Error State**

URL values can be dynamic, but sometimes entered incorrectly. We will learn how to handle dynamic URL values, and error states.

* URLs need to support dynamic values
* Use React Router to match dynamic value, or display error
* Click to verify dynamic URL value and error handling

**4.4 Convert Render Props into Components**

When configuring routes, it may be practical to move render code into separate components. We will learn how to create and render external components for our routes.

* Route configuration can get bulky with render logic
* Extract render props into separate components and use them
* Verify that restructured rendering works equivalently

**4.5 Conditionally Render Components, Redirects**

In a multi-page app, URL structures may change, therefore requiring redirects. We will learn how to redirect from old to new URLs, and redirect based on a state condition.

* Know that when URL structures change, existing routes will no longer render
* Use provided Redirect component to handle old URLs
* Enter old URL, verify redirect to new URL pattern

**Section 5: Fetch Data from Third-Party APIs**

Interacting with APIs is essential to modern web applications. In this section we fetch data from third-party APIs and populate our React application state. These patterns can be adapted to consume most JSON APIs.

* Overview: React App + Star Wars API
* Fetch DATA from swapi.co
* Extract Logic into Separate Components
* Fetch Data Based on URL/Route
* Persist API Data in Local Storage

**5.1 Overview: React App + Star Wars API**

Interacting with third party APIs is essential web apps. We will learn how to pull data from the Star Wars API.

* Learn about the goal of this section
* Visual overview of the completed app pulling SWAPI.co data
* Introduce next step of fetching data from Star Wars API

**5.2 Fetch DATA from swapi.co**

The Star Wars API has the data we need to populate our component state. We will learn how to fetch and render a list of vehicles.

* Fetch all vehicle data from SWAPI.co
* Fetch and iterate through data to output vehicle links
* Use unique links to show specified content

**5.3 Extract Logic into Separate Components**

The main App.js file is overloaded with a lot code and logic. We will extract most of the logic into separate components.

* App.js now contains a mix of all the logic
* Extract logic into separate components for better organization and maintainability
* Code is now more manageable, and ready for more complexity

**5.4 Fetch Data Based on URL/Route**

Fetching specific vehicle details from the API is required when navigating. We will learn how to fetch single vehicle data.

* Fetch specific vehicle data when a link is clicked
* Fetch data by calling SWAPI with specific vehicle ID
* Update component state, and display specific vehicle info

**5.5 Persist API Data in Local Storage**

App state is lost when refreshing the page, and data is fetched every single time a link is clicked. We will learn how to save data and prevent unnecessary API requests.

* Know that our state is lost when the page is reloaded
* Save data into local storage, populate component state after reloading
* Unnecessary API calls are prevented, data displays instantly after click

**Section 6: Prepare and Deploy React Apps to Production**

After building interactive, responsive, multi-page apps, the next step is to show the world. In this section we prepare our app for production, and deploy to various hosts and servers.

* Build React App to Optimize for Production
* Deploy to Netlify: Drag and Drop
* Deploy to Netlify: Command Line
* Deploy to Netlify: Continuous Deployment
* What’s Next? Eject from CRA, More Hosting Options

**6.1 Build React App to Optimize for Production**

We will get to see how to build React App to optimize production.

* Get introduced to Netlify
* Get help from the deployment application documentation to run commands
* Proceed to build the App

**6.2 Deploy to Netlify: Drag and Drop**

We will deploy to Netlify using the drag and drop.

* Upload the build directory to Netlify
* Get to know the loop holes and learn to solve them

**6.3 Deploy to Netlify: Command Line**

We will deploy to Netlify using the command line.

* Install the Netlify CLI
* Login into Netlify using the login command on the terminal
* Follow the steps that will get us to deploy Netlify using the command line

**6.4 Deploy to Netlify: Continuous Deployment**

We will deploy to Netlify using continuous deployment.

* Get an overview of continuous deployment
* Configure the init command to initiate continuous deployment

**6.5 What’s Next? Eject from CRA, More Hosting Options**

We will see how to Eject from CRA and even look at more hosting options.

* Learn about the command used to Eject from CRA
* Learn about Webpack, Babel, Zeit and code sandbox
* Learn about cloud deployment options like AWS EC2, Digital Ocean’s droplets, Hostgators, and Cyberduck