Introduction to React

React.js is a an open-source JavaScript library for creating user interfaces

React adopts a component based approach in developing applications.

Every react app can be considered a component or a collection of components which can be reusable.

React.js is designed to make the process of building modular, reusable user interface components simple and intuitive. With React you can design simple views for each state of an application. You can also use React to efficiently update and render components when your data changes. I will walk you through how to create react components from scratch, review components properties, and component state, as well as the foundational parent/child relationships within React.js. I will also take you through the lifecycle of components . You will be transpiling JSX code using babel which is an in browser transpiler that converts JSX code into readable format.

Topics

#### Environment setup and Git - 2 days

#### HTML, CSS - 3 Days

#### Javascript - 4 Days

#### React - 6 Days

#### React advanced - 4 days

#### Redux and state management - 4 days

#### Sample Project - 14 days

Day 1 & 2 Environment setup and Git

## 1. Managing files and file systems

cat, cd, chmod, chown, compress, cp,find, gzip, ln, mkdir, mv, rm, rmdir, rpm, sort ,touch, umount, uncompress, uniq, unzip, zip,cat,tail,head,grep

1. [Reference 1](https://www3.ntu.edu.sg/home/ehchua/programming/howto/Unix_SurvivalGuide.html)
2. [Reference 2](http://www.howtogeek.com/107808/how-to-manage-files-from-the-linux-terminal-11-commands-you-need-to-know/)

## 2. Managing running programs

bg, fg, kill,, nice, ps, top, watch [Reference](http://www.howtogeek.com/107217/how-to-manage-processes-from-the-linux-terminal-10-commands-you-need-to-know/)

## 3. Console text editors

nano, vim [Reference](http://www.informit.com/articles/article.aspx?p=1670957&seqNum=3)

## 5. Git setup and basic commands

Git setup remote repository, push, pull, PRs, [Reference](https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/)

HTML , CSS, Bootstrap

## Day 3 HTML, HTML5

1. [HTML Tutorial](http://www.w3schools.com/html/)
2. [HTML5 Basics](https://www.w3schools.com/html/html5_intro.asp)

## Day 4 & 5 CSS, Bootstrap

1. [CSS Tutorial](http://www.w3schools.com/css/)
2. [Bootstrap Basics](http://getbootstrap.com/getting-started/)
3. [Bootstrap Components](http://getbootstrap.com/components/)

Javascript

## Day 6 Javascript basics

1. [Javascript Tutorial](http://www.w3schools.com/js/)
2. Variable hoisting
3. Functional hoisting

## Day 7 Data types and ES6 features

1. [JS Data types](https://www.w3schools.com/js/js_datatypes.asp)
2. Var vs let vs const
3. ES6 Arrow functions
4. ES6 template strings
5. ES6 Classes

## Day 8 Operators

1. Operators - Loops(while, for), conditional statements, etc.
2. Arrays and Objects
3. Array manipulations (map, filter, sort, reduce etc.)

## Day 9 Advanced

1. Closures
2. Call stack and event loop
3. Promises
4. [JS examples](https://www.programiz.com/javascript/examples)

React

## Day 10 React setup

1. [Setup React](https://reactjs.org/tutorial/tutorial.html#setup-for-the-tutorial)
2. create-react-app
3. Babel & React
4. JSX

## Day 11 Main concepts

1. [React component lifecycle](https://reactjs.org/docs/state-and-lifecycle.html)
2. Virtual Component
3. State in React
4. [Props in React](https://reactjs.org/docs/components-and-props.html)

## Day 12 Main concepts [Cont..].

1. Updating state
2. Class components
3. Functional components

## Day 13 Main concepts [Cont..].

1. [Handling events](https://reactjs.org/docs/handling-events.html)
2. [Rendering elements](https://reactjs.org/docs/rendering-elements.html)
3. [Conditional rendering](https://reactjs.org/docs/conditional-rendering.html)

## Day 14 Main concepts [Cont..].

1. Static type checking
2. PropTypes

## Day 15 React Hooks

1. [React Hooks](https://reactjs.org/docs/hooks-intro.html)
2. UseState Hook
3. UseEffect Hook
4. Running side effects

React Advanced Concepts

## Day 16 Code reusability

1. Higher order components
2. [JSX in depth](https://reactjs.org/docs/jsx-in-depth.html)

## Day 17 Routing

1. [React Router](https://reactrouter.com/)
2. Setup app routes
3. History & location APIs
4. Browser router
5. Link

## Day 18 Error boundaries

1. [Error boundaries](https://reactjs.org/docs/error-boundaries.html)
2. Try catch

## Day 19 Code Splitting and Bundling

1. Webpack bundler
2. Lazy loading
3. Supsense

Redux and state management

## Day 20 State management

1. [Redux principles](https://redux.js.org/introduction/getting-started)
2. Setup react with redux
3. [Core concepts](https://redux.js.org/introduction/core-concepts)

## Day 21 Redux in action

1. Actions
2. Reducers
3. Store

## Day 22 Asynchronous Operations

1. Middlewares (Thunk / Saga)  
   [Reference 1](https://redux.js.org/introduction/learning-resources#middleware)  
   [Reference 1](https://blog.krawaller.se/posts/exploring-redux-middleware/)
2. Asynchronous API calls

## Day 23 Architecture your application

1. [Redux architecture](https://redux.js.org/introduction/learning-resources#redux-architecture)

Sample project (14 days)

## Day 23 to 37

## Each individual should identify a sample application and get it approved by the mentor and can complete that by this time period.

End result:

* Implement the features.
* Commit it to a Github repository on a frequent basis.
* Deploy it a server(Can get in touch with the mentor and DevOps engineer for this)
* Do a presentation on the project.

Note: For template use a freely available template online(eg, from: https://themeforest.net/.

**Here is an example:**

Todo application development

Develop a todo application with the following options,

1. User registration ( Handle with a local database , localstorage APIs)
2. User login (Handle with a local database, localstorage APIs)
3. User should have an email id, password
4. List all todos of the logged in user (with the email id)
5. Create a todo item
6. Update a todo item
7. Delete a todo item

Proper form validations should be applied throughout the form fields.

Use axios/fetch for the API requests.

Use <https://jsonplaceholder.typicode.com/guide/> as the backend service.