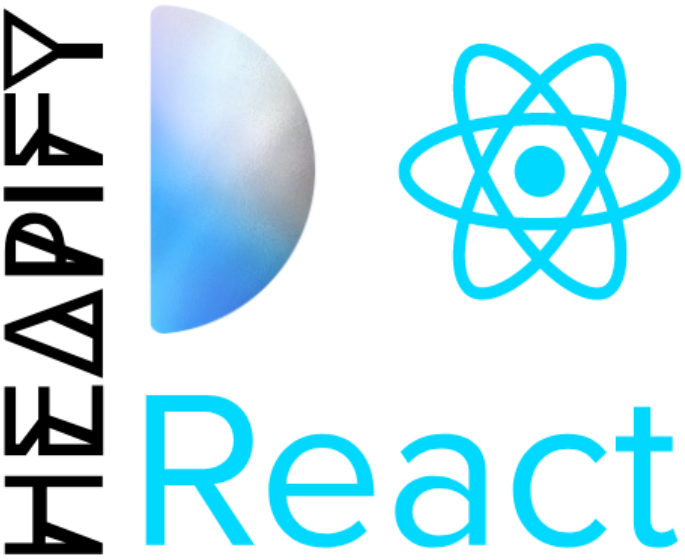


Heapify React-Able

September 2021 - November 2021

'Week 2'

Topics

1. Setting up the workspace with the IDE.
2. What is React
3. Benefits of React and why to use it
4. Installing React into the machine.
5. Start using React and getting to know the project structure.
6. Changing the title and the logo of the application.
7. Writing your first React code.
8. Components and elements in React
    8.1. Class and Functional Components

1. Setting up the workspace with the IDE.

In order to work with React, we'll be using the Visual Studio Code editor. This is an IDE that has been provided by Microsoft.

How to install VS Code

The complete installation process and the extensions to be used will be discussed in the class.

Refer to the video for installation of VS Code:
https://youtu.be/9aj_Mz5zgRM
Refer to the video for installing the required extensions:
https://youtu.be/sZYATPTYm_c

2. What is React

React is a free and open-source front-end JavaScript library for building user interfaces or UI components. It is maintained by Facebook(launched in 2013) and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.
React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple.
React could be run on online platforms like Codepen, CodeSandbox, etc.

3. Benefits of React

How do we compare React with other frameworks? They are not interchangeable and each one has pros and cons depending on what we are trying to achieve. So here, we'll get to learn about the benefits of React and will see why we should learn it. We'll only get to know about all the features in short, but will be covering all of them very deeply in the coming classes.

01. Reusable Components

Components are wonderful and React is based on them. We start with small things, which we use to build bigger things, which we use to build apps. Each component has its own logic and controls its own rendering, and can be reused wherever we need them. Code reuse helps make our apps easier to develop and easier to maintain. They also help us implement a consistent look and feel across the whole project.

02. The Virtual DOM

Normally, when we develop an app that has a lot of user interaction and data updates, we have to carefully consider how our app structure is going to impact

performance. Even with fast client platforms and JavaScript engines, extensive DOM manipulation can be a performance bottle-neck and even result in an annoying user experience. Worse, because the DOM is tree-structured, simple changes at the top level can cause huge ripples to the user interface.

React solves this by using a virtual DOM. This is, as the name implies, a virtual representation of the DOM. Any new view changes are first performed on the virtual DOM, which lives in memory and not on your screen. An efficient algorithm then determines the changes made to the virtual DOM to identify the changes that need to be made to the real DOM. It then determines the most effective way to make these changes and then applies only those changes to the real DOM.

03. Using JSX

JSX is really just a mix of HTML and JavaScript. It lets us add bits of HTML to our JavaScript. This lets us come up with a much simpler and cleaner code. JSX produces React “elements” and has a number of side benefits, including helping prevent injection attacks. To be fair, there are raging debates on whether JSX is a good thing because it makes coding easier, or a bad thing because it may or may not violate the separation of concerns.

04. React Native for Mobile App development

React can be boldly called a “**learn once – write anywhere**” library, since both web and mobile application development follows the same design patterns, facilitating the transition process. Using plain JavaScript and React we are able to build a rich UI for native apps, supported by both iOS and Android platforms.

Among other advantages of React js in mobile app development, React Native developers recite its portability and ability to reuse components, real-time reload, and open source. When it comes to the actual use of React Native, we can list such mobile apps as Skype, Tesla, Airbnb, and Walmart. And don't forget about Instagram and Facebook – the actual innovators and early adopters.

05. SEO Friendly

Another Reactjs benefit is its ability to deal with a common search engine failure to read JavaScript-heavy apps. As a solution, React can run on the server, rendering and returning the virtual DOM to the browser as a regular webpage.

4. ReactJs vs React-Native

ReactJs	React Native
Used for developing Web applications.	Used for developing mobile applications.
Animations could be added using JavaScript and CSS.	Animations are added usibng built-in libraries.
It uses HTML tags.	It doesn't use HTML tags.
In ReactJs, Virtual DOM is responsible for rendering the code on the browser.	Uses it's native API to render the code on mobile application.
CSS is used for adding styles.	Uses stylesheet to add the styles.
ReactJs is platform i.e. it can be executed on all platforms.	It is not platform independent, a lot of efforts are needed to run it on several platforms.
Provides high security.	Less secure as compared to ReactJs.

5. Installing React into the machine

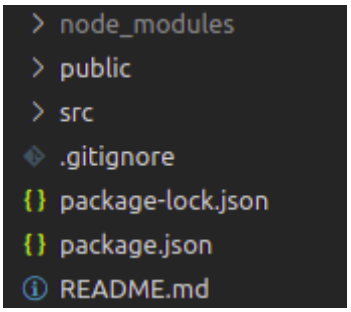
Till now we have seen what React is and why we should use it and what are its benefits. So, now let's proceed to the actual working and let's install React in our machine. To install the full React toolchain, we would be using **create-react-app**.
In order to install create-react-app, run the following command in your terminal:

npm install -g create-react-app

This command will install create-react-app in your system globally. Once this is done, we are ready to create our first react project. In order to do so, follow the below commands -

npx create-react-app {project_name}

The above command will generate a template react project for you to begin with. Simply replace the {project_name} with the name of your own project. The contents of the project that you will get after running the above command are shown below:



Then navigate to the project directory using

cd my-app

Now to start the react project, run the following command in the same terminal.

npm start

By running the above command, your react project will start running on localhost:3000 and will open automatically in your default browser.
Now you are all set. You have installed react and have also initialized your very first project.

6. Components and Elements in React

01. Elements

An element is a plain object describing a component instance or DOM node and its desired properties. It contains information only about the component type (eg: a button), its properties, and any child element inside it.
In simple words, an element is a way to tell React what you want to see on the screen. You cannot call any methods on the element. It's just an immutable description object.

```
<button class="btn btn-primary">
  <b>Click Me</b>
</button>;
```

02. Components

A React Component is a template, a blueprint, a global definition. This can be either a function or a class. If the component is a functional one, then it returns an element, but if the component is a Class one then it renders the element. In simple words, it is a function or class that accepts an input(props) and returns a React element. React components are independent and reusable blocks of code.
A React component can have methods and can also receive props as well. Components also have the state which they use to display dynamic content.
React components are of two types:

a. Class components
b. Functional components

Note: The name of a React Component must start with a capital letter.

1) **Class Components**

React class-based components are the bread and butter of most modern web apps built in ReactJS. These components are simple classes (made up of multiple functions that add functionality to the application). All class-based components are child classes for the Component class of ReactJS. So, the component has to include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives the component access to React.Component's functions. Class components also require a render() method, which returns the code to be displayed i.e. JSX.

The main feature of class-based components is that they have access to a state which dictates the current behavior and appearance of the component. This state can be modified by calling the setState() function. One or more variables, arrays, or objects defined as part of the state can be modified at a time with the setState() function. Class-based components also have access to Lifecycle methods.

```
import React from "react";

class App extends React.Component {
  render() {
    return <h1>Heapify React-Able</h1>;
  }
}

export default App;
```

2) **Functional Components**

Functional components are just simple JavaScript functions. We can create a functional component in React by writing a simple JavaScript function. These functions may or may not receive data as parameters(props). In the functional Components, the return value is the JSX code to render to the DOM tree.

Functional components lack a significant amount of features as compared to class-based components. So, in February 2019 React 16.8 was launched which introduced the concept of React Hooks. Hooks are special functions that added a lot of missing functionalities in the functional components. Functional components do not have access to dedicated state variables like

class-based components, this was one of the problems that were overcome by the introduction of hooks.

```
import React from "react";

const App = () => {
  return <h1>Heapify React-Able</h1>;
};

export default App;
```

Class-based components are slightly slower than their functional counterparts. The difference is very small and is almost negligible for smaller web apps – though the performance difference increases when the number of components in the app increases. Moreover, class-based components involve a lot more coding on the programmer's part, making them slightly more inefficient to use. And hence, functional components are some of the more common components that you'll come across while working in React.

Lets Rock!

HEAPIFY
REACT-ABLE

"Redefining your journey to Success"

✉ E-mail: heapify.reactable@gmail.com