What is React?

(https://github.com/learn-co-curriculum/react-hooks-what-is-react) (https://github.com/learn-co-curriculum/react-hooks-what-is-react/issues/new)

Learning Goals

- Understand React at a high level
- Explain some of the benefits of using React when developing web applications

Introduction

In the world of web development, there are a number of hip names tossed around — Angular, Vue, React. These are some of the front end frameworks that have gained popularity over the last few years. These frameworks provide a standardized way of creating and deploying parts of web applications. This allows developers to put their energy and focus towards designing a good experience for end users of the application.

The following lessons are all about just one framework, React. Why learn React? In the last few years, **React has surpassed other frameworks in popularity and demand** (https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190) . For good reason, too! In this lesson, we will discuss what React is and why it is so awesome and useful.

A Quick Note on React

Before we move on, it is important to note one thing: React is technically a library, not a framework. This is a big topic of debate, with one side standing by the technical definitions of frameworks and libraries, and the other arguing that React does essentially the same things as the other two main JS frameworks (and arguably better). Most people just stick to calling it a framework, but regardless of which term you use, one fact is not up for debate: React allows developers to build large, scalable applications, quickly and painlessly.

React Overview

React is built entirely out of JavaScript, using a combination of dependencies. Using React might seem significantly different from what you're used to when writing vanilla JavaScript (or, in other words, just JavaScript). This is because React provides a specific way to organize and structure the design of a web application.

Using JSX, an extension of vanilla JavaScript with a specific syntax, we can write code that looks very similar to HTML. Snippets of this JSX get separated out into components, sort of like building blocks.

When combined, these components form a fully working web application. The use of components lets us separate code and functionality in a logical and easy to read way, producing highly reusable, independent, chunks.

Some of the Awesome Features of React

From the **React docs (https://reactjs.org/)**, here are three of the key features of React:

- **Declarative**
- **Component-Based**
- **Learn Once, Write Anywhere**

Let's dig into each of these features and talk a bit more about each one.

Declarative

You may have heard the expressions **imperative** and **declarative** programming before. Here are some quick definitions:

Imperative Programming:

- Explicitly describes the actions a program should take, each step of the way
- Describes *how* a program should go about doing those actions

Declarative Programming:

- Describes *what* a program should accomplish (or what the end result should be)
- Leaves the determination of *how* to get to the end result up to the program

Creating DOM elements without a framework like React tends to be a very **imperative** operation:

```
const header = document.createElement("h1");
header.textContent = "Hi!";
header.className = "main";

const container = document.querySelector("#container");
container.append(header);
```

React, however, encourages a **declarative** approach to working with the DOM. In our code below (which is a special JSX format that React uses — more on that later!), we don't describe *how* to update the browser (i.e. "remove that `<div>`, add this `<li>`, etc."). Instead, we provide React with a template of *what* the component should look like once it is finished being prepared, i.e.:

```
// JSX syntax
const header = <h1 className="main">Hello from React!</h1>;

ReactDOM.render(header, document.querySelector("#container"));
```

In the example above, we are saying: "There should be a `<h1>` element with the text of 'Hello from React' rendered to the page".

This is us interacting in a **declarative** programming manner with React! We neither told it what to explicitly add/delete from the DOM when we wanted something changed, nor did we tell it how to go about changing the DOM. This is perfectly acceptable for React! As we explore more going forward, we will see how this code acts as a template and React does the rest.

Component-Based

React encourages us to structure our applications as "components": building blocks of our web page that handle their own data and UI logic. Individual components can be "composed" together with other components to put together a complete app.

For instance, here's how we could design a site like Yelp by thinking in components:

And here's how it might translate to our code in React:

```
function App() {
  return (
    <div>
      <NavBar />
```

```
        <ResultList>
          <ResultItem result={result1} />
          <ResultItem result={result2} />
        </ResultList>
        <Map />
      </div>
    );
  }
```

By separating out our code into these *reusable, composable components*, we can more easily put together complex UIs by thinking about each piece in isolation.

Learn Once, Write Anywhere

We won't be taking advantage of this feature of React as much during your time in this course, but one great thing about React is that it works in a few different environments. Once you learn the key concepts of working with React to create client-side web applications, you can also more easily learn **React Native** (https://reactnative.dev/) and write native mobile apps using React, or use tools like **Next.js** (https://nextjs.org/) and **Gatsby** (https://www.gatsbyjs.com/docs/) to write server-rendered React.

Writing React Projects

React has a recommended tool, `create-react-app` (https://create-react-app.dev/), that makes it easy to create a new React project from scratch. The `create-react-app` tool sets up a preconfigured, default project, ready to launch with `npm start`. This package includes functionality built to be mobile friendly and progressive. That means apps will work on all modern browsers and mobile devices. `create-react-app` also gives us a couple of additional tools to make React development better:

- **Babel** (https://babeljs.io/): an included transpiler that converts modern JavaScript and custom code like JSX into more widely compatible JavaScript
- **webpack** (https://webpack.js.org/): a 'bundler' that takes all our work, along with any required dependency code, and packages it all up in a single, transferable bundle

- Built in linting and code analysis functionality using **ESLint** (https://eslint.org/) to help improve our code, reinforce best practices and catch common mistakes.

React is actively maintained by Facebook, and new features are added regularly! The **React docs** (https://reactjs.org/) are well-written and translated into many languages. Once you have a good grasp of React, you will have the ability to create cutting-edge web applications and sites. Knowing how to use React also opens doors to similar frameworks, such as React Native for building mobile apps.

> **Note**: While the link above to the current React docs is an excellent resource, the React team recently released a **beta version of their new documentation** (https://beta.reactjs.org/). The new docs are not yet complete, but they're another great resource to learn about the latest React features and best practices.

Conclusion

You can imagine a framework is like a car — while it is interesting and useful to know how the engine works or how to fix a transmission, that knowledge isn't necessary for using the car to get where you want to go. What is necessary is that you understand how to use the pedals, turn the wheel and change gears.

First, we will be covering what you need to know to use React. Later, we'll go into more detail on the critical parts: the engine under the hood. By the end, you will be able to quickly design and create your own React apps!

Resources

- **React Docs (current)** (https://reactjs.org/)
- **React Docs (beta)** (https://beta.reactjs.org/)