**React js Project Files and Course Documents**
You can access all script files belonging to the applications we will make during the course by going to this links;

https://github.com/OakAcademy/react-router/tree/master/react-router

https://github.com/OakAcademy/blog-posts
https://github.com/OakAcademy/hemisphere
https://github.com/OakAcademy/first-app
https://github.com/OakAcademy/image-list

**What is React?**
React is a front-end library in Javascript that was developed by Facebook. The simplest definition of React is that it is a user interface library used for building UI components for the web. But if that was all React did, it would be little more than a template library. Developers define components in React by using an HTML-like syntax called JSX. These encapsulated components manage their own state, making it simple to pass rich data to a component and keep the state of the application and its components out of the Document Object Model (DOM). These features give React components the ability to react dynamically and efficiently to changes in their state. A set of simple components in React can be composed into either simple single-page applications or large, complex web applications.

**What is React used for?**
React is an open-source JavaScript frontend library. Some developers consider it a frontend framework because it does more than standard libraries usually do. The most common use of React is to build dynamic user interfaces for single-page web applications. But the structure of React makes it useful for more than just creating websites. JSX, which is one feature of React, is a syntax extension for Javascript that provides a template for the HTML and manages the state of the element. But JSX can be used as an interface for things other than HTML. React Native uses React to build mobile applications. Here, JSX becomes an interface for mobile UIs instead of HTML. There are even a few libraries that use React and JSX to interact with and manage hardware like React Hardware.

**How does React work?**
React encourages engineers to write code using a Functional Programming approach. Engineers create components, which are normal Javascript functions. These functions return information to tell React what content should be displayed on the screen. The real power of React comes from the ability to nest or compose these functions inside of one another. This nesting ability encourages code reuse and allows an engineer to write a component a single time, but in many different places.

**Is React a framework or library?**
Frameworks provide an opinionated approach to building an entire application. Libraries, on the other hand, assist in building a single aspect of an application. With this in mind, React is a library. It assists engineers in presenting HTML in the browser. React has no opinions on how data is fetched, how styling is applied, or how the app is deployed or built.

**Is React worth learning?**
Yes, React is worth learning. There are a couple of reasons. The first one is that React is in high demand in the software development job market and has been for a few years. If you learn to code in React and can do it well, you will increase your chances of finding a job. This alone is another reason it is worth learning. Website users no longer will settle for old-fashioned sites that won't update without a page reload. React’s method of dynamically updating the HTML of a web page fits these users' needs and improves their experience. React has also been around a few years and has stood the test of time. A large portion of websites, both small and large, use React as a frontend framework.

**Is React hard to learn?**
Every technology or programming language has a learning curve, and React is no different. But it is easy to learn if you are dedicated and determined to create frontend web applications. To make learning React simpler, you should stick to the basics of React before you add any extra libraries, as that will only complicate your learning experience in the beginning. Since React is a Javascript framework, a solid grounding in Javascript will give you a head start. With this knowledge, the unique concept of JSX, React's templating language, and the way React uses state will be much easier to grasp.

**What is the difference between React Native and ReactJS, and which one should I learn?**
React, or ReactJS is a front-end Javascript library for building UI components for the web. If you are interested in web development, React is the perfect library to learn to create interactive, dynamic single-page apps, or even full-scale web applications. React Native is a framework for building native mobile applications for both the Android phone and Apple's iPhone. React Native is still React, which means the syntax and workflow for building applications are basically the same, but the generated components are different. In React, web components are generated. In React Native, the generated components interact with a phone's native APIs. If your focus is web development, then you should learn React. If you want to build mobile applications, it is best to learn React first and become familiar with the technology before you try React Native.

**Why is React so popular?**
There are many reasons why React is popular. One reason is that Facebook developed it. The social proof is that if it is good enough for Facebook, one of the most popular social networks on the web, it should be good enough for other applications. React also solved many of the past issues that developers had with developing single-page applications (SPAs). React came out when SPAs were becoming popular, and all the existing frameworks to build them made development complicated and prone to bugs. One feature that makes it better than past libraries is that React is relatively easy to use. Its components are reusable, plus React's use of the virtual DOM makes it very performant. React should remain popular in the future as each new release brings new features and performance improvements.

Learning Node js is a great way to get into backend web development, or expand your full-stack development practice. With Udemy’s hands-on Node.js courses, you can learn the concepts and applications of this wildly useful JavaScript runtime.

**Learn more about Node.Js**
Node.js is essential to developing real-time applications in JavaScript and has been instrumental in the development of websites like eBay and PayPal. Node

is designed around an event loop, which allows for easy management of asynchronous functions. This makes it a popular environment for modern developers working on chat and gaming apps.

## Frequently asked questions

### What is Node.Js and what is it used for?

Node.Js is a server environment built for fast and easily scalable network applications. It was built on Chrome's JavaScript runtime and uses an event-driven, non-blocking model that makes it the best fit for applications that run on distributed devices and need to run in real-time. By using JavaScript, node.Js can be put to work by many software developers familiar with JavaScript. Because the code is open-source, you can also use it on any platform (Mac OS, Windows, or Linux). Node.js is the architecture for creating websites and real-time applications because it's easy for teams to learn, and it's fast. Examples of applications that use node.Js include video conferencing apps, chat servers, eCommerce software, and collaborative online gaming.

### What are the advantages of Node.Js?

Node.Js is open-source, meaning it's free code for all developers. On top of that, it also means that there is a thriving community of Node.Js users and programmers that all add to the knowledge base. Many are happy to share their flavor of the code with other developers, and collectively, the Node.Js environment continues to be enhanced. Because Node.Js uses JavaScript, there is a large pool of developers that understand and can code in the environment. It's also a relatively simple environment for new team members to learn, making it an efficient choice for development teams with people that need training. Node.Js was developed on Push technology instead of web sockets, which is why it's the preferred choice for real-time communication applications and programs that need to be highly scalable.

### What does it mean that Node.Js is a runtime system?

A runtime system is a platform where a software program runs. It's essentially an environment housing the collection of software and hardware that allows an application to execute (or run). Node.Js is a runtime system because it provides the environment necessary for applications to run within it, and no additional code or hardware is required. Because Node.Js makes use of

JavaScript, it's a runtime system that provides a framework where you can use JavaScript to create and run programs. The JavaScript programming language (which is quite popular) is then automatically translated into machine code for the hardware to run the program in real-time. It's an efficient system, making it the preferred choice for many software and tech companies.

**What is microservices architecture and how can Node.Js be used for it?**
Microservices architecture is a software development style or method where single-function modules originate. Each has a very well-defined operation and interface and can deploy on its own. In essence, it's a way of developing modules that you can repurpose from one program or application to another. When you create an application, it's a collection of modules that have been thoroughly tested and are well-maintained. The modules are typically built around different business-specific capabilities and are then loosely coupled to other modules when deployed as part of a program. You can use Node.Js in microservices architecture as the language of choice for one, some, or all of the microservices (or modules). The beauty of microservices is that you can use the best language for the specific microservice. But where highly scalable, fast programs are needed, Node.Js would be a

**All Web Development courses**
Each aspect of creating websites and applications entails a unique set of skills. Udemy offers a host of courses to bring you up to speed on modern front-end, back-end, and full stack web development practices and skills.

**From a Udemy Web Development student**
This course has taught me so much! Before, I hardly even knew how to run an HTML file. Now, I can create a fully functioning Node.js server responding to HTTP requests!

**Learn more about Web Development**
The world of web development is as wide as the internet itself. Much of our social and vocational lives play out on the internet, which prompts new industries aimed at creating, managing, and debugging the websites and applications that we increasingly rely on.

Web development is a broad description of the tasks and technologies that go into creating a website. It can be as simple as making a static text-based website or as elaborate as developing an interactive dynamic website. You can break web development into two different categories: frontend (client-side) and backend (server-side). Frontend code executes on the user's computer. This can include HTML, JavaScript, and CSS. Backend code runs on the server — this commonly includes communicating with a database and often involves languages like Python, Ruby, Java, or PHP. Web development does not necessarily include the design process — it focuses on code. A web designer builds wireframes to mock up their vision for a website and then shares that with a developer. The developer is responsible for writing the code that implements the design.

**What are the steps to becoming a web developer?**
Some web developers will obtain a degree or certification in the field. However, most jobs don't require a specific degree or level of education, just demonstrated knowledge of the field. So, it is more important that you know how to show off your skills. You could highlight your skills through relevant work experience or a portfolio of past projects. You might also share code you have developed using a platform like GitHub, or participate in solution-based forums like StackOverflow that reward you for helping others. HTML, CSS, and JavaScript are the first three coding languages you'll need to learn to break into web development. You need these three essential elements to create a modern website and work as a front-end web developer. HTML (Hyper-Text Markup Language) specifies the content of the website and builds the backbone. CSS (Cascading Style Sheets) styles the content. JavaScript controls the interactive elements of the website.

**How long does it take to become a web developer?**
The answer to this question will depend on you. The more time you spend developing your skills, the faster you can become a web developer. The good news is that web development generally uses lightweight code that is easier to learn than many other languages. If dedicated, you can learn the basics of web development in a couple of months. But good web developers never stop learning. A better question might be, "What can I do to become a better web developer faster?" The answer to this question is practice. Becoming familiar

with coding helps tremendously, but there is also a less obvious benefit of practicing. The more you code, the more you will run into problems or find bugs in your code. A significant aspect of web development is solving problems and debugging code. The better you get at solving problems and tracking down bugs, the faster you will get at coding.