**What is React? A tutorial on how to get started -** Erin Doherty Apr 10, 2020

React is a JavaScript library for building user interfaces. React is a popular, easy to learn tool, that’s designed to work everywhere. With some basic web development skills – like JavaScript, HMTL, and CSS, you can build professional looking, slick UIs quickly and easily.

Today, we’ll help you get started with React by comparing native and hybrid apps and exploring some basic syntax.

**Here’s what will be covered in this article:**

* [Intro to Web Application Development](https://www.educative.io/blog/what-is-react-tutorial#intro)
* [Native apps](https://www.educative.io/blog/what-is-react-tutorial#native)
* [Hybrid apps](https://www.educative.io/blog/what-is-react-tutorial#hybrid)
* [React Tutorial - A step-by-step walkthrough](https://www.educative.io/blog/what-is-react-tutorial#tutorial)
* [React Syntax](https://www.educative.io/blog/what-is-react-tutorial#syntax)

**Intro to Web Application Development**

Web apps are software developed to display web pages in a browser. The most basic techstack is JavaScript, HTML, CSS. Here’s a quick breakdown of how these three technologies are used:

Web application basic tech stack is:

* HTML
  + DOM – Document Object Model
  + For structure of web page
* CSS
  + For styling
* JavaScript
  + For programming
  + Grabbing data from a server, processing inputs, generating outputs
  + Web pages are served up by a server, and then modified by the browser using JavaScript.

These tech stacks that developers use to build web applications are often called **Web Frameworks** or **Web Application Frameworks**. React.js is a modern framework for JavaScript. Adding React to the basic HTML/CSS/JavaScript stack makes it a lot easier to make more robust user interfaces.

In web app development, you -the developer - has to worry about:

* browser compatibility
* extensions
* Speed
* Security
* Web Protocols – HTTP
* Browser’s architecture rules

Within your web app, you’ll also have tiers which is explained below.

**Web Application Tiers**

For this quick overview, we’ll just assume a 3-tiered [architecture](https://www.educative.io/blog/how-to-design-a-web-application-software-architecture-101): browser, engine, database.

A screenshot of a computer

Description automatically generated with low confidence

What are tiers? Applications are segmented into logical chunks called tiers. The three-tiered application is the most common structure – that’s what’s diagrammed above. React is a library that deals with the web browser level, as React is a library for creating UIs. Here’s how the web application uses 3-tiered architecture to update a web page:

Diagram

Description automatically generated

**Native Apps**

The easiest example of native apps to understand when you’re new to software development are mobile apps for your cell phone. Mobile apps are the ones you use on your phone everyday – like Instagram or Spotify. Mobile apps come in two development styles:

* Native
  + Native is built for one specific Operating System (OS)
* Hybrid
  + Built to run anywhere

**What is native app development?**

To shortcut: native app development is usually building apps for iPhone or Android Phones.

Native app development is the creation of software programs that run on specific devices and platforms. You can build native apps for desktops, smart TVs, and all kinds of gadgets. Smartphones are the most popular native mobile app development platforms. Companies build operating systems for developers to build apps for their devices – this gives developers access to tools & features on that device – like a camera on a phone. Specifically, the two most popular mobile OS are:

* Google’s Android
* Apple’s iOS

Native mobile apps don’t run in the browser – unlike web applications which are built for browsers. As a user, you download native mobile apps from platform-specific stores like Apple’s App Store or Google Play.

This means as a developer building native apps, you don’t have to worry about browser compatibility, but you do have to worry about the compatibility of your native app’s Operating System. The huge benefit to Native Apps are getting free access to tools & features in that Operating System!

**Native App Development Advantages**

* Use Device Features & Tools
  + Example: Your phone’s camera
* Faster
  + Native apps are optimized for that specific Operating System, that specific device.
* Compliant
  + Native apps are built with the security & programming requirements of the native operating system in mind. So, native apps are less likely to break when the manufacturer of the OS puts out an update – because presumably, native developers have built to that Operating System’s specs & strengths.
* Secure
  + Can use built in security features to verify a user is logged in, for example Face ID on an iPhone
  + Native apps can often access the devices 2 Factor Authentication method, like getting a code texted to your phone, and having the code auto-populate in the app.

**Hybrid Apps**

Hybrid apps are built to run everywhere. Hybrid apps work as both web apps and native mobile apps. Hybrid apps are developed to be compatible with different browsers and operating systems.

You build hybrid apps using web technologies: HTML, CSS, and JavaScript, but the developer has to be very careful selecting a tech stack on top of that. Many features from different code libraries or tools only work with certain browsers or operating systems.

React’s platform versatility makes it a favorite for bypassing these problems.

Hybrid Apps are:

* Portable
* Simple Development
* Run on different browsers, Operating Systems

**What are the downsides to hybrid apps?**

* Loss of speed
  + Native apps are optimized to that specific Operating System. Hybrid apps have to be designed to work everywhere.
* Limited tech tooling
  + There’s a lot of languages, code libraries, and packages for specific situations that won’t work everywhere. As a developer, you’ve got to be very careful picking a techstack when designing hybrid.
* Code duplication/complication
  + Some things in your app have to be written to handle different cases on browsers vs. native Operating Systems
* Loss of powerful features
  + There’s great features on your smartphone – like your camera. My iPhone has live photos, filters, and panorama. Coding a hybrid app you’ve got to dumb down the design a little bit to ensure it’s compatible everywhere.

**React Tutorial - A step-by-step walkthrough**

Here we’ll demonstrate a simple React component to get you started learning React.

**Note about Developer ToolChains:**

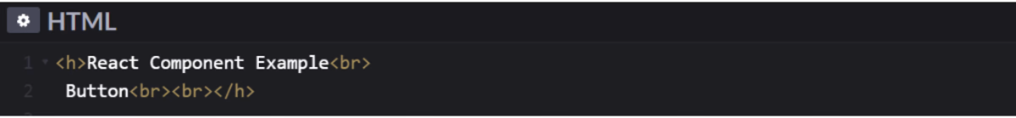
Tool Chains are the set of software used to implement development into production software. For the most part, when coding for a company you’ll be working with a long existing toolchain.

For this 101, we’re going to add React as a plain script tag on an HTML page. React as a plain scriptag skips a lot of complicated toolchain info.

**React as a Script Tag**

Here’s a 101 to build a button using React.

1. Add a title in HTML
   * Readability is a huge part of good code so put in a title!



1. Add a DOM container to the HTML web page inside the body
   * Add <body></body> tags
   * Inside the body, add <div id="click\_button\_container"></div>
   * Note: There can be multiple DOM containers in the HTML.
     + DOM containers act independently from other containers, but usually for React centric apps, there’s only one DOM container
     + Containers are empty: React replaces existing content inside DOM containers

Text

Description automatically generated

1. Script Tags
   * Add 3 script tags after the DOM container, right before </body> end tag
   * These tags Load React, then loads component code
2. Create a React Component in JavaScript
   * JavaScript code defines the React Component
   * In the JavaScript section of our codebase, enter in the following code:

Text

Description automatically generated

1. The code you entered in:
   * Creates a new React element
   * Creates a class called ClickButton which extends React.Component
   * Click button initializes a state variable called “clicked” to false
2. Now we’ve got the structure defined in HTML, the empty click\_button\_container HTML root node ready for something to populate it. We’ve also got the ClickButton class in JavaScript defining the logic of our button component. The next step, is to render() the ClickButton logic into the click\_button\_container.
   * Add this code to the bottom of the JavaScript file, which will render the ClickButton logic into the click\_button\_container.

Text

Description automatically generated

1. Now you can see “Click Button” on the right hand side. go ahead and click that and see what happens.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

You’ve created a React component! Congratulations! There’s a lot more to learn about React. Below we’ve got some syntax background explanation about the code you created.

**React Syntax**

There’s a lot more about React syntax on the React page, but let’s go over the very basic basics of the button we just built:

* React is Component Based
* Elements
* Render() method
  + Stateful Component

**Component Based**

React is component based. What are components? Components are small, isolated pieces of code that the developer uses to build UIs. The best thing about components, are they are reusable.

Examples of React Components:

* Button we just built

What do React components do?

* Takes in props
  + Properties, parameters passed into the component
* Returns display
  + Hierarchy of views
  + Via render() method

**Elements**

Elements are a lightweight description of what to render. Elements are our first building block of learning React syntax – elements describe what you see on the screen.

From our code example, here’s how React interacts from create an Element in JavaScript, creating a root DOM node in HTML, and using the render() method to put the logic from the element to display on the webpage.

Diagram

Description automatically generated

**What is render?**

The render() method returns a description of what to display on the screen. Render() returns a React element – remember a React element is a lightweight description of what to render().

**Stateful Component**

How does the browser know when to re-render a page? When the state of the component changes. A component maintains internal state data (this.state), as seen in our ClickButton example class where this.state = { clicked: false }.

Graphical user interface, text, application

Description automatically generated

When this.state changes, the render() method is re-invoked, and the rendered markup is updated with the latest information. In our example, when this.state.clicked changes to True, the button disappears, and a “You clicked this” method shows instead.

Text

Description automatically generated

**Next Steps**

Congratulations! You’ve made it through a React intro! There’s more to learn. The next steps include:

* JSX
* Toolchains
* State vs. Props
* Props are immutable
* Component API
* Forms
* Events