**Module 1: Anatomy of React**

In this introductory module, you'll learn about what React is and where it is used. You'll also learn how to set up your coding environment so that you have as productive a learning experience as possible. So, the purpose of this module is to understand the what and the why, and to get set up for the modules that follow.

Components are one of the foundations of React. In React, everything revolves around components. You'll learn how to build components, how to structure and customize your React projects, and how to compose layouts by importing components into other components.

You'll learn about passing data from one component to another. You'll learn about JSX syntax in React and how to use it to structure and style your components.

B​y the end of this module you will be able to:

* Explain the concepts behind React and component architecture.
* Describe how to use assets within an app to apply styling and functional components.
* C​reate a component to service a specific purpose.
* C​reate a folder and demonstrate how to create and import files within that folder.
* U​se and manipulate props and components to effect visual results.

**Module 2: Data and State**

The second module of this course deals with working with events and errors in React. You'll learn how events work and how you can handle them in React. Handling events can sometimes get a bit tricky, so you'll also learn about dealing with errors related to events in React.

B​y the end of this module you will be able to:

* Use common methods to manage state in React.
* Detail the concept and nature of state and state change.
* Describe the hierarchical flow of data in React.
* Describe how data flows in both stateful and stateless components.
* Use an event to dynamically change content on a web page.
* D​escribe some common errors associated with events and the syntax required to handle them.

**Module 3: Navigation Updating and Assets in React**

In this module, you'll learn about routing and navigation in React. You'll learn how to render partial views and how to update routes in your React apps. You'll understand how assets are used, bundled and embedded.

B​y the end of this module you will be able to:

* Use media assets, such as audio and video, with React.
* Demonstrate how to manipulate image assets using reference paths.
* Explain the folder structure of a React project in terms of embedded or referenced assets.
* Demonstrate the conditional implementation and rendering of multiple components.
* Create and implement a route in the form of a navbar.
* Describe navigation design in React, with a focus on single and multi-page navigation.

**Module 4: Portfolio Mini-Project (Calculator App)**

This module is focused on a practical mini project of building a calculator app in React. Upon completing this module, you'll have coded your own mini project in React, as a starting point for building your React portfolio.

You have now learned about the scope of things you will cover in this course.

B​y the end of this module you will be able to:

* Synthesize the skills from this course to create and style a React component.
* Reflect on this course's content and on the learning path that lies ahead.

Before you learn React

Do you know the fundamentals of HTML, CSS and JavaScript? Perhaps you learned about these technologies from another course. Either way, a quick summary will be useful so let's explore some fundamental HTML, CSS and JavaScript principles and practices.

In this reading, let’s take a practical approach, and revisit some of the development techniques you'll need to be comfortable with before learning React.

To get the most out of this course on React basics, you should first understand the fundamental methods and concepts of JavaScript. Otherwise, you may feel like you’re a child learning to run before you can walk. React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”.

React apps are built using modern JavaScript features, which are commonly known as ES6. Developers use React to develop Singe Page Applications. And you can also develop mobile applications with React Native.

As an aspiring developer, you may opt for a ‘learn as you go approach’ regarding JavaScript and React. But this may not help your productivity and even at times frustrate you. This is because you may confuse code or functionality that is plain JavaScript, or code that is React.

For example, with a solid foundational knowledge of JavaScript, you can quickly identify code that is JavaScript ES6 and code that is React. And throughout this course, there will be help for you along the way with some friendly reminders.

Also, keep in mind that you are using React to build user interfaces which also include HTML and CSS code.

So let’s begin with the fundamental HTML knowledge needed to learn React.

**HTML**

Recall that HTML is used to describe the structure of Web pages. Developers use HTML elements with their opening and closing tags to “mark up” an HTML document.

These elements form the structure of a web page and describe what to display to the web browser. When it comes to HTML it's important to know about:

1. The purpose of HTML in the web browser,
2. the use of HTML tags and correct syntax,
3. and how HTML elements are used in a web document.

Another important concept to know about when you're talking about HTML is the Document Object Model, or DOM.

Users need to be able to interact with elements on a web page. This means that an HTML document must be represented in a way that JavaScript code can query and update it. And that's the function of the DOM. It's a model of the objects in your HTML file.

And web developers interact with the DOM through JavaScript to update content, set up events and animate HTML elements.

Before you learn React, it’s advisable that you are comfortable with the following HTML tags and concepts

**Layout & Style**

* **<html>**
* **<head>**
* **<body>**
* **<div>**

**Text formatting & lists**

* **<h1>…<h6>**
* **<p>**
* **<ul><li>**
* **<b><i>**

**Images and links**

* **<img src="">**
* **<a href="">**

**Linking and Meta**

* **<link>**
* **<title>**
* **<meta>**

**Semantic**

* **<header>**

**CSS**

CSS (Cascading Style Sheets) is the code that you use to style HTML. You need to be familiar with basic CSS concepts before you start learning React. This is because you will need to style your React components as well, and basic CSS knowledge will help your learning journey.

Before you learn React, make sure you are comfortable with these CSS styling options:

* Font styling (font size, font color, etc.)
* Flex Box Layout (Layout of items using CSS Flex Box Layout)
* CSS Selectors
* Position, Padding, Margins and Display
* Colors, Background and Icons

Y​ou can refresh your knowledge of HTML and CSS in the Meta course titled: [Introduction to Front-End Development](https://www.coursera.org/learn/introduction-to-front-end-development)

**JavaScript fundamentals and ES6**

React is completely written in JavaScript and uses the more modern version of JavaScript which is ES6. While learning React, you should already know JavaScript fundamentals.

JavaScript is the programming language and React is a JavaScript UI library. This means the first step is to be proficient at JavaScript.

Here are some of the JavaScript topics that you need to be comfortable with before you begin your journey learning React.

* Data types
* Using var, let and const
* Conditionals and Loops
* Using objects, arrays and functions
* ES6 Arrow functions
* In-built functions such as map(), forEach() and promises.
* Destructuring Arrays and Objects
* Error Handling

**Package Manager (Node + npm)**

React is a UI library, and you will encounter that many times you will need to add other packages to your React application. A package in JavaScript contains all the files needed for a module. To install these packages effectively and manage their dependencies you can use a package manager like NPM (Node Package Manager).

You can install npm by installing Node.js, which will then automatically install npm.

You need to be comfortable with using npm as your package manager, since you will be using npm to install packages within your React application.  Make sure you are aware of how to do the following with npm before you get started on this course.

* Installation command to install npm modules in your project
* Installing a package as a dev dependency
* Start command
* Updating npm version
* Navigating around the package.json file

Once you have become confident with these skills, you’ll be in a better position to learn and apply React concepts and prepare yourself for development of React apps.

T​o refresh your knowledge of JavaScript and the basics of Node and npm, please visit Meta course titled: [Programming with JavaScript](https://www.coursera.org/learn/programming-with-javascript).

# JavaScript modules, imports - exports

import addTwo from "./addTwo";

// the rest of the mathOperations.js code goes here

Or

import { addTwo } from "./addTwo";

// the rest of the mathOperations.js code goes here

// addTwo.js module:

function addTwo(a, b) {

    console.log(a + b);

}

export default addTwo;

Why is React using the concept of components? It allows to build modular components

Below you will find links to helpful additional resources.

* [Components and props](https://reactjs.org/docs/components-and-props.html)
* [Intoducing JSX](https://reactjs.org/docs/introducing-jsx.html)
* [Styling and CSS in React](https://reactjs.org/docs/faq-styling.html)
* [Introducing expressions in JSX](https://reactjs.org/docs/introducing-jsx.html#embedding-expressions-in-jsx)

# Creating React components

Go to folder

Then

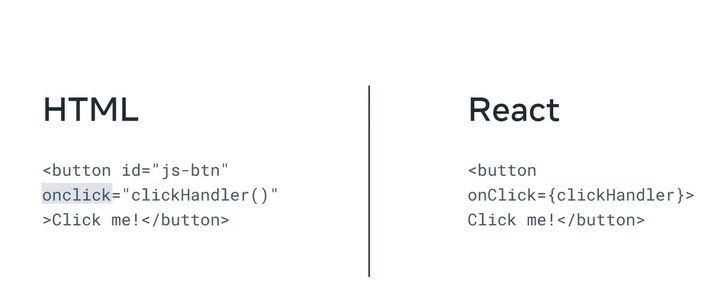
## Install react app

### JS Vanilla

npm init react-app .

### TypeScript

npm init react-app . --template typescript



## ****Handling events using inline anonymous ES5 functions****

<button onClick={function() {console.log('first example')}}>

    An inline anonymous ES5 function event handler

</button>

## ****Handling events using inline anonymous ES6 functions (arrow functions)****

<button onClick={() => console.log('second example')}>

    An inline anonymous ES6 function event handler

</button>

## ****Handling events using separate function declarations****

function App() {

    function thirdExample() {

        console.log('third example');

    };

    return (

        <div className="thirdExample">

            <button onClick={thirdExample}>

                using a separate function declaration

            </button>

        </div>

    );

};

export default App;

## ****Handling events using separate function expressions (react preferred)****

function App() {

    const fourthExample = () => console.log('fourth example');

    return (

        <div className="fourthExample">

            <button onClick={fourthExample}>

                using a separate function expression

            </button>

        </div>

  );

};

export default App;

# Data flow

### The props data is data outside the component and cannot mutate.

That’s right! State data is a component’s internal data, which it can control and mutate. Props data is outside of the component and is immutable, meaning it cannot change.

### State data is data inside the component, and the component can control and mutate the data.

That’s right! State data is a component’s internal data, which it can control and mutate. Props data is outside of the component and is immutable, meaning it cannot change.

# Hooks in react

Hooks can be called only at the top level and only from React functions.

Hooks also come with a set of rules, that you need to follow while using them. This applies to all React hooks, including the **useState** hook that you just learned.

* You can only call hooks at the top level of your component or your own hooks.
* You cannot call hooks inside loops or conditions.
* You can only call hooks from React functions, and not regular JavaScript functions.

## useState hook

useState accepts an initial state and returns two values:

* The current state.
* A function that updates the state.

import {useState} from”react”;

let [“value”, function will performed on the value] = useState(“initial value”)

## useContext hook

Context provides a way to pass data through the component tree without having to pass props down manually at every level.

## useMemo hook

The React useMemo Hook returns a memoized value.

## useCallback hook

## 

## useRef hook

the Hook allows you to persist values between renders.

It can be used to store a mutable value that does not cause a re-render when updated.

It can be used to access a DOM element directly.