

# Introduction to React, Components, and JSX

Skills Bootcamp in Front-End Web Development

Lesson 13.1





**WELCOME**

# Learning Objectives

---

By the end of this lesson, you will:



Begin to feel comfortable building static UIs with JSX.



Gain an initial understanding of the component-based paradigm in ReactJS.



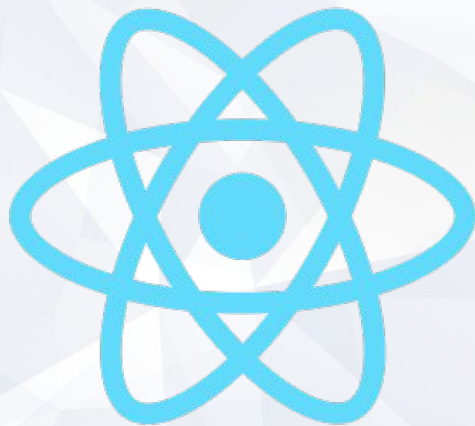
Dissect and build a few simple examples using ReactJS.



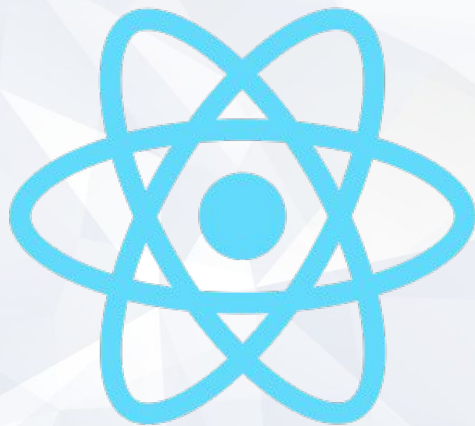
Today's class is meant to be a  
**gentle** introduction to React.



# What Is React?

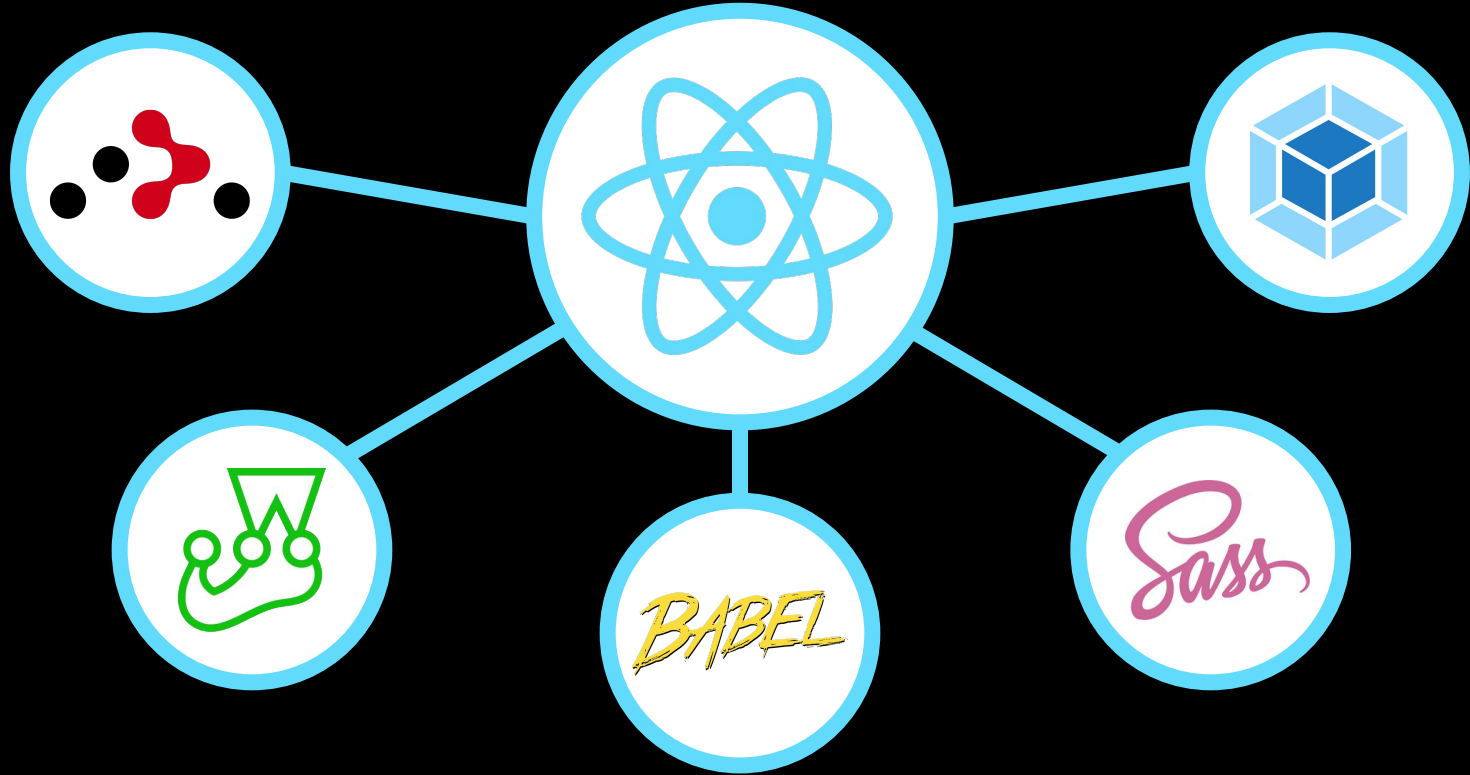


**React is one of the most powerful, in-demand front-end JavaScript libraries available today.**



**React helps you create complex and responsive single-page applications.**

Just about every popular UI library that came  
after React borrows from it.



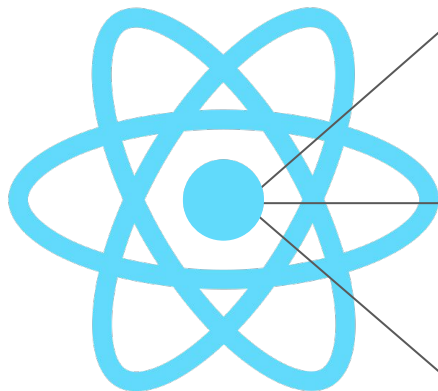


Libraries are like fads that come and go, but the ideas introduced by React appear to be here to stay.



# React Is...

---



An open-source JavaScript library for developing UIs developed by **facebook**.

---



Developed to build large apps with rapidly changing data.

---



Component-based: UI elements are broken into self-contained components.



# What Problem Does React Solve?

# What Problem Does React Solve?

---

- DOM operations are quite expensive in terms of performance, so React creates a **virtual DOM (VDOM)**.
- The VDOM is a representation of the page structure in memory. It tracks what needs to be updated and only updates those specific things.
- React is not opinionated like many other frameworks. It gives the developer the freedom to use Javascript the way they want to use it.



**Can You Give Me an Example?**

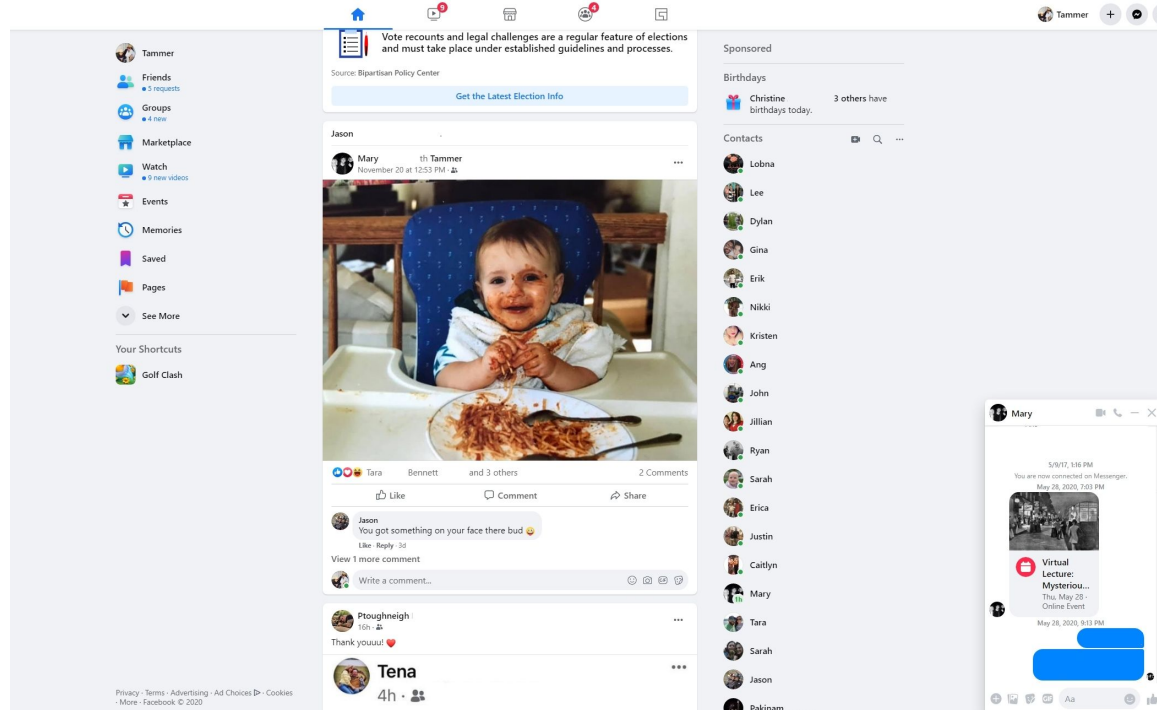
# Can You Give Me an Example?

---

- Facebook's UI is a great example of React in action.
- Each section of the page is a component that itself has tons of real-time updates happening every second.
- The component design pattern allows Facebook to add a search bar and messenger to nearly every page that the user visits.

# Facebook's UI Complexities

Facebook uses multiple components with interactive options, live-updating data, and tightly interacting elements. This poses a challenge to simple DOM.





# Why Separate UI Components?



# Why Separate UI Components?

---

- Logically decompose a UI into unique parts.
- Easily reuse these parts without rewriting code.
- Separate components are easier to test.
- Helps isolate bugs, saving time.



# What Are the Pros and Cons?

# React Pros and Cons

---

## Pros



- 1 Components are reusable.
- 2 UI updates in response to state change, reducing DOM manipulation code needed.
- 3 Can build applications on web, server, and native applications.
- 4 Easier to learn and more popular than other front-end JavaScript libraries and frameworks.

## Cons



- 1 React is a view library concerned with rendering user interfaces. You have to pull in other libraries to accomplish things like HTTP requests.
- 2 Can require more configuration than other libraries.

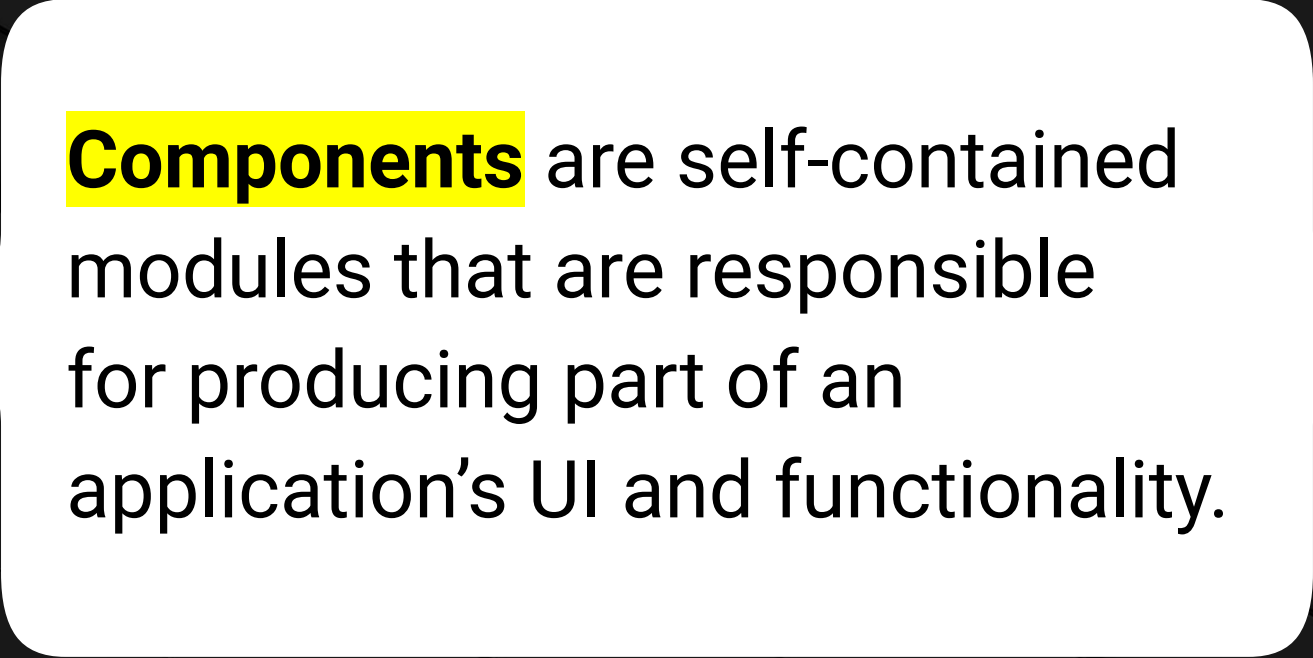


You may feel a little overwhelmed with the new unusual syntax.

Don't worry—at the end of the day, we're still just working with JavaScript .



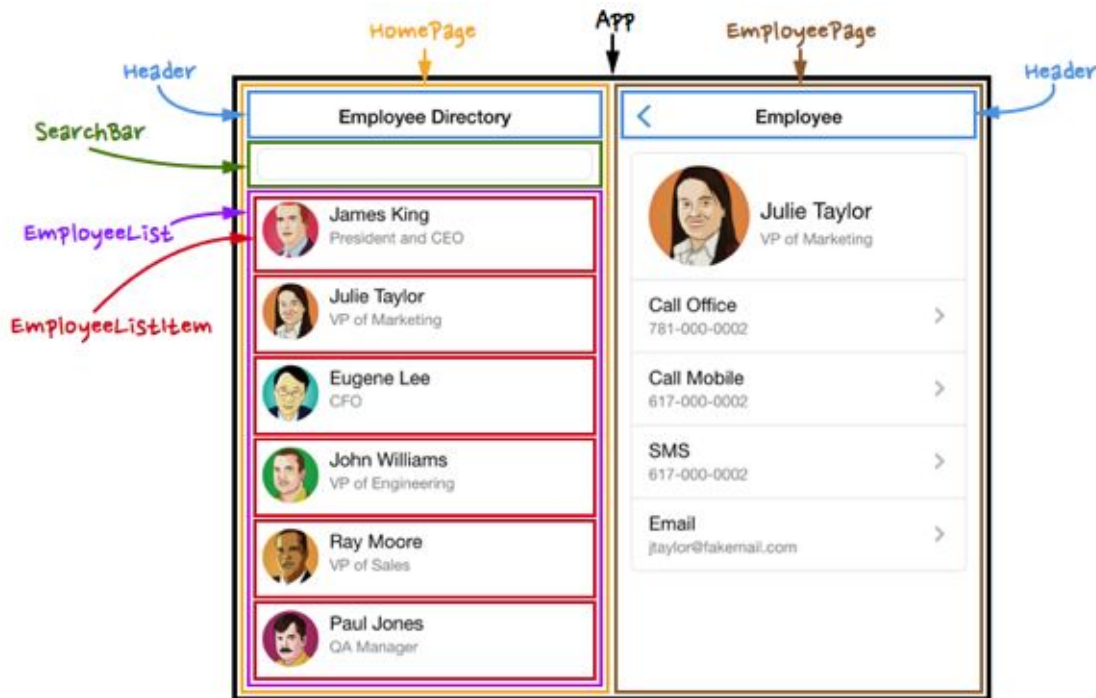
The most important concept to  
understand today is **components**.



**Components** are self-contained modules that are responsible for producing part of an application's UI and functionality.

# The Concept of Components

Using React, UI elements are broken down into reusable components. Each subcomponent behaves in a way that it is fully contained.



# The Power of Components

---

Why separate UI elements into components?



Logically decompose a UI into unique parts.



Easily reuse these parts without re-coding.



Easier to test.



Helps you find bugs and saves time.





# Instructor Demonstration

---

## Create React App

# Review: Create React App

---

To scaffold out a React app with Create React App, we run `npx create-react-app` followed by a name for our application.



**Welcome to React**

To get started, edit `src/App.js` and save to reload.



We're going to see this setup over and over again. There's no need to completely memorize every aspect of the Create React App boilerplate right now.

# Review: Create React App

---

The most important takeaways are:



We're going to be writing most of our code inside of the `src` folder.



The “entry” file to our React application will be the `index.js` file.



We start our React app in development mode with the command `npm start`. This means that our app will live update as we change it, which is why we're running our app on a server.

# Questions?





## Pair Programming Activity:

---

# Hello World Example

In this activity, you will dissect a simpler example and answer some questions.

Suggested Time:

15 Minutes



Time's Up! Let's Review.



JSX can **either** represent primitive HTML tags or React components.



# Review: Hello World Example

---

The most important takeaways are:



`const` works like `var`, but it's meant for values that aren't going to be reassigned, otherwise `let` is preferred.



`ES2015 modules` are part of a new module system introduced with ES6. So far we've been working with CommonJS modules (`module.exports` and `require syntax`)—which for our purposes today will work similarly.



As we'll see a bit later, ES2015 modules allow for finer-tuned control over what is exported and imported from a module. For now, just compare the new syntax to what they're used to.

# Questions?





## Activity: HelloDiv

In this activity, you will write a React component that displays your name and some information about themselves.

Suggested Time:

10 Minutes



Time's Up! Let's Review.

# Review: HelloDiv

---

The most important takeaways are:



`HelloDiv` is exported and rendered inside of `App`.



`App` is exported and then rendered inside of `index.js` as the first argument to the `ReactDOM.render` method.



The second argument to the `ReactDOM.render` method is the real DOM element that our React application should be rendered inside of.

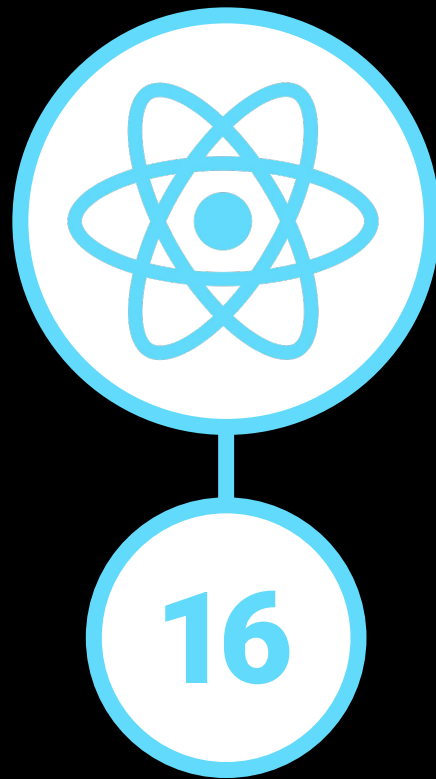


`HelloDiv`, like most components we'll write, is a JavaScript function; it returns some JSX.



**Normally**, we aren't able to render multiple JSX elements from a component without an enclosing parent tag.

**Note:** With the release of React 16, we do have another option for returning multiple JSX elements from a component—**we can return an array.**



# Questions?





  
Break



# Instructor Demonstration

---

## HelloBootstrap

# HelloBootstrap

---

So far, we've only just begun to work with React, but we've still managed to learn a few key things:

01

In React, we structure our code into components.

02

A component is a JavaScript function that describes some part of our application's UI.

03

Inside of our components, we describe our application's UI using JSX: a markup syntax that resembles HTML.



**We're now going to go over how we can  
add Bootstrap to a React project.**



## Activity: HelloBootstrap

In this activity, you will be given slightly less starter code and tasked with creating a React application that renders Bootstrap components to the page.

Suggested Time:

10 Minutes



Time's Up! Let's Review.

# Review: HelloBootstrap

---

The most important takeaways are:



If we want to render multiple JSX elements, they should be contained within a single parent element, such as a `div`.



Void elements, such as `input` tags, are represented by JSX tags with a self-closing forward slash (i.e., `<input />`).



We need to import the `react` library anywhere that we are utilizing JSX.



We use `className` instead of `class` because `class` is a reserved word in JavaScript.

# Questions?







# Instructor Demonstration

---

## JSX Variables



## Activity: JSX Variables

In this activity, you will render JavaScript expressions inside of JSX curly braces.

Suggested Time:

10 Minutes



Time's Up! Let's Review.

# Questions?





# Instructor Demonstration

---

## CSS Demo



## Activity: CSS Props

In this activity, you will change the appearance of an application's components using inline styles.

---

Suggested Time:

10 Minutes



Time's Up! Let's Review.

# Questions?





*The  
End*