React - Introduction

1. React is a JavaScript library for building user interfaces.
2. React is used to build single page applications.
3. React allows us to create reusable UI components.

## Create React App

In order to learn and test React, you should set up a React Environment on your computer.

If you have NPM and Node.js installed, you can create a React application by first installing the create-react-app.

Install create-react-app by running this command in your terminal:



## Run the React Application

If you followed the two commands above, you are ready to run your first *real* React application!

Run this command to move to the myfirstreact directory:

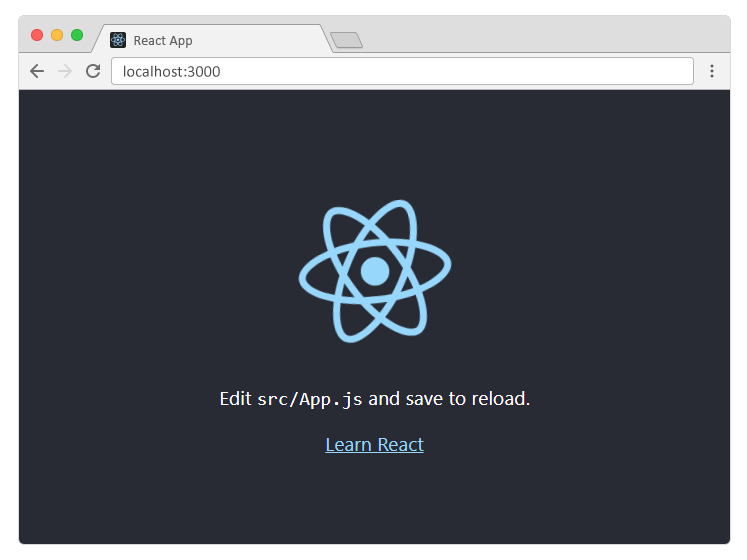


Run this command to execute the React application myfirstreact:



A new browser window will pop up with your newly created React App! If not, open your browser and type localhost:3000 in the address bar.

The result:

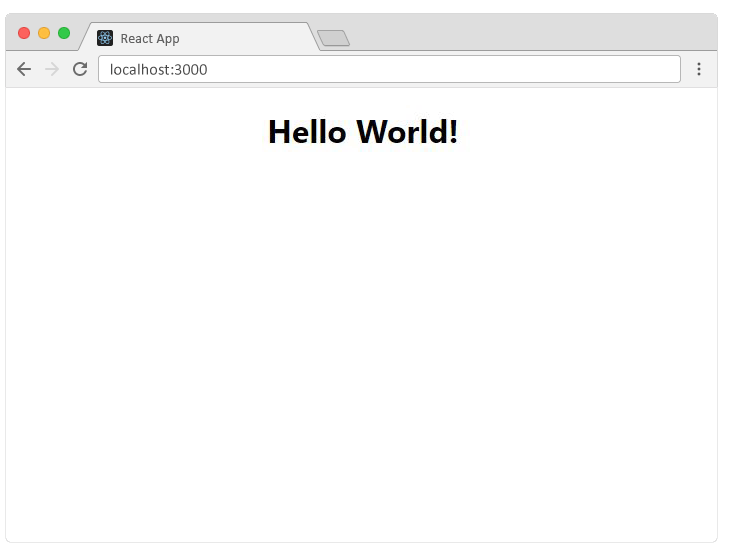


## Modify the React Application

So far so good, but how do I change the content?

Look in the myfirstreact directory, and you will find a src folder. Inside the src folder there is a file called App.js.

Now, the result is as follows:



# React ES6

## What is ES6?

ES6 stands for ECMAScript 6.

ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015.

## Why Should I Learn ES6?

React uses ES6, and you should be familiar with some of the new features like:

* Classes
* Arrow Functions
* Variables (let, const, var)

## Classes

ES6 introduced classes.

A class is a type of function, but instead of using the keyword function to initiate it, we use the keyword class, and the properties are assigned inside a constructor() method.

### Method in Classes

You can add your own methods in a class:

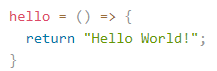


I have a ford will be the output of this code

## Arrow Functions

Arrow functions were introduced in ES6.

Example:



## Variables

Before ES6 there were only one way of defining your variables: with the var keyword. If you did not define them, they would be assigned to the global object. Unless you were in strict mode, then you would get an error if your variables were undefined.

Now, with ES6, there are three ways of defining your variables: var, let, and const.

# React Render HTML

1. React's goal is in many ways to render HTML in a web page.
2. React renders HTML to the web page by using a function called ReactDOM.render().

## The Render Function

The ReactDOM.render() function takes two arguments, HTML code and an HTML element.

The purpose of the function is to display the specified HTML code inside the specified HTML element.

## The Root Node

The root node is the HTML element where you want to display the result.

It is like a *container* for content managed by React.

It does NOT have to be a <div> element and it does NOT have to have the id='root'

# React JSX

## What is JSX?

JSX stands for JavaScript XML.

JSX allows us to write HTML in React.

JSX makes it easier to write and add HTML in React.

## Coding JSX

JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.

JSX converts HTML tags into react elements.

## Expressions in JSX

With JSX you can write expressions inside curly braces { }.

The expression can be a React variable, or property, or any other valid JavaScript expression. JSX will execute the expression and return the result:



# React Components

Components are like functions that return HTML elements.

## React Components

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and returns HTML via a render function.

Components come in two types, Class components and Function components, in this tutorial we will concentrate on Class components.

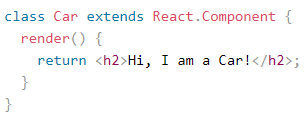
## Create a Class Component

When creating a React component, the component's name must start with an upper case letter.

The component has to include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a render() method, this method returns HTML.

Example,

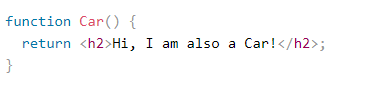


## Create a Function Component

Here is the same example as above, but created using a Function component instead.

A Function component also returns HTML, and behaves pretty much the same way as a Class component

Example,



## Component Constructor

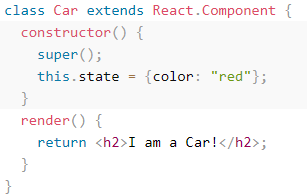
If there is a constructor() function in your component, this function will be called when the component gets initiated.

The constructor function is where you initiate the component's properties.

In React, component properties should be kept in an object called state.

You will learn more about state later in this tutorial.

The constructor function is also where you honor the inheritance of the parent component by including the super() statement, which executes the parent component's constructor function, and your component has access to all the functions of the parent component (React.Component).



## Components in Components

We can refer to components inside other components:

Example,



# React Props

1. Props are arguments passed into React components.
2. Props are passed to components via HTML attributes.
3. React Props are like function arguments in JavaScript *and* attributes in HTML.
4. To send props into a component, use the same syntax as HTML attributes

## Pass Data

Props are also how you pass data from one component to another, as parameters.

Example,



## Props in the Constructor

If your component has a constructor function, the props should always be passed to the constructor and also to the React.Component via the super() method.

Example,



# React State

1. React components has a built-in state object.
2. The state object is where you store property values that belongs to the component.
3. When the state object changes, the component re-renders.

## Creating the state Object

The state object is initialized in the constructor

Example,



## Using the state Object

Refer to the state object anywhere in the component by using the this.state.*propertyname* syntax.

Example,



## Changing the state Object

To change a value in the state object, use the this.setState() method.

When a value in the state object changes, the component will re-render, meaning that the output will change according to the new value(s).

# React Lifecycle

## Lifecycle of Components

Each component in React has a lifecycle which you can monitor and manipulate during its three main phases.

The three phases are: Mounting, Updating, and Unmounting.

## Mounting

Mounting means putting elements into the DOM.

React has four built-in methods that gets called, in this order, when mounting a component:

1. constructor()
2. getDerivedStateFromProps()
3. render()
4. componentDidMount()

The render() method is required and will always be called, the others are optional and will be called if you define them.

### constructor

The constructor() method is called before anything else, when the component is initiated, and it is the natural place to set up the initial state and other initial values.

The constructor() method is called with the props, as arguments, and you should always start by calling the super(props) before anything else, this will initiate the parent's constructor method and allows the component to inherit methods from its parent (React.Component).

Example,



### getDerivedStateFromProps

The getDerivedStateFromProps() method is called right before rendering the element(s) in the DOM.

This is the natural place to set the state object based on the initial props.

It takes state as an argument, and returns an object with changes to the state.

### componentDidMount

The componentDidMount() method is called after the component is rendered.

This is where you run statements that requires that the component is already placed in the DOM.

## Updating

The next phase in the lifecycle is when a component is *updated*.

A component is updated whenever there is a change in the component's state or props.

React has five built-in methods that gets called, in this order, when a component is updated:

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

The render() method is required and will always be called, the others are optional and will be called if you define them.

### getDerivedStateFromProps

Also at *updates* the getDerivedStateFromProps method is called. This is the first method that is called when a component gets updated.

This is still the natural place to set the state object based on the initial props.

The example below has a button that changes the favorite color to blue, but since the getDerivedStateFromProps() method is called, which updates the state with the color from the favcol attribute, the favorite color is still rendered as yellow:



### shouldComponentUpdate

In the shouldComponentUpdate() method you can return a Boolean value that specifies whether React should continue with the rendering or not.

The default value is true.

### getSnapshotBeforeUpdate

In the getSnapshotBeforeUpdate() method you have access to the props and state *before* the update, meaning that even after the update, you can check what the values were *before* the update.

If the getSnapshotBeforeUpdate() method is present, you should also include the componentDidUpdate() method, otherwise you will get an error.

## Unmounting

The next phase in the lifecycle is when a component is removed from the DOM, or *unmounting* as React likes to call it.

React has only one built-in method that gets called when a component is unmounted:

* componentWillUnmount()

### componentWillUnmount

The componentWillUnmount method is called when the component is about to be removed from the DOM.

# React Events

1. Just like HTML, React can perform actions based on user events.
2. React has the same events as HTML: click, change, mouseover etc.

## Adding Events

React events are written in camelCase syntax:

onClick instead of onclick.

React event handlers are written inside curly braces:

onClick={shoot} instead of onClick="shoot()"

Event Handlers

A good practice is to put the event handler as a method in the component class

Example,



## React Event Object

Event handlers have access to the React event that triggered the function.

In our example the event is the "click" event. Notice that once again the syntax is different when using arrow functions or not.

With the arrow function you have to send the event argument manually:

Example,



# React Forms

## Adding Forms in React

You add a form with React like any other element

Example,



## Handling Forms

1. Handling forms is about how you handle the data when it changes value or gets submitted.
2. In HTML, form data is usually handled by the DOM.
3. In React, form data is usually handled by the components.
4. When the data is handled by the components, all the data is stored in the component state.
5. You can control changes by adding event handlers in the onChange attribute

## Conditional Rendering

If you do not want to display the h1 element until the user has done any input, you can add an if statement.

Look at the example below and note the following:

1. We create an empty variable, in this example we call it header.
2. We add an if statement to insert content to the header variable if the user has done any input.
3. We insert the header variable in the output, using curly brackets.

## Textarea

The textarea element in React is slightly different from ordinary HTML.

In HTML the value of a textarea was the text between the start tag <textarea> and the end tag </textarea>, in React the value of a textarea is placed in a value attribute.

Example,



# Styling React Using CSS

There are many ways to style React with CSS, this tutorial will take a closer look at inline styling, and CSS stylesheet.

## Inline Styling

To style an element with the inline style attribute, the value must be a JavaScript object:

Example,



The output of the above code as shown below:



## CSS Stylesheet

You can write your CSS styling in a separate file, just save the file with the .css file extension, and import it in your application.

## CSS Modules

Another way of adding styles to your application is to use CSS Modules.

CSS Modules are convenient for components that are placed in separate files.

# React Sass

## What is Sass

Sass is a CSS pre-processor.

Sass files are executed on the server and send CSS to the browser.

## Can I use Sass?

If you use the create-react-app in your project, you can easily install and use Sass in your React projects.

Install Sass by running this command in your terminal:



-------reference credits: w3schools, tutorialspoint, [https://github.com/hegdenaveen1/Become-A-Full-Stack-Web-Developer#start-here](https://www.google.com/url?q=https://github.com/hegdenaveen1/Become-A-Full-Stack-Web-Developer%23start-here&sa=D&source=hangouts&ust=1599916002593000&usg=AFQjCNFqAEgym0xUC_MkOX8mYy5AtiOv1g) ------