REACT.JS

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

To install globally - 

Create React App

To learn and test React, you should set up a React Environment on your computer.

The create-react-app tool is an officially supported way to create React applications. Node.js is required to use create-react-app.

Open your terminal in the directory you would like to create your application.

Run this command to create a React application named my-react-app:

 - this will set up everything you need to run a React application.

**Note: If you've previously installed create-react-app globally, it is recommended that you uninstall the package to ensure npx always uses the latest version of create-react-app. To uninstall, run this command: npm uninstall -g create-react-app.**

Run this command to move to the my-react-app directory: **cd my-react-app**

Run this command to execute the React application my-react-app: **npm start**

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.

React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook. React is a tool for building UI components.

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM. React finds out what changes have been made, and changes only what needs to be changed.

React.JS was first used in 2011 for Facebook's Newsfeed feature. Facebook Software Engineer, Jordan Walke, created it.

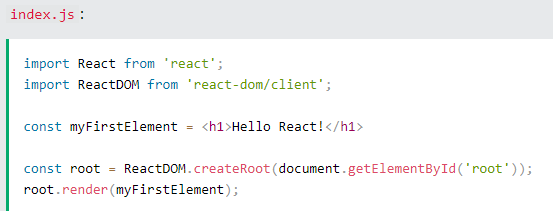
In order to use React in production, you need npm and Node.js installed.

Start by including three scripts, the first two let us write React code in our JavaScripts, and the third, Babel, allows us to write JSX syntax and ES6 in older browsers.



Look in the my-react-app directory, and you will find a src folder. Inside the src folder there is a file called App.js, open it and try changing the HTML content and save the file.

Typical react index.js –



ES6 stands for ECMAScript 6. ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript. 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.

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text, application

Description automatically generated

The constructor function is called automatically when the object is initialized. You can also create a method inside class ( eg. Present)

To create a class inheritance, use the extends keyword.A class created with a class inheritance inherits all the methods from another class.

Example – (<https://www.w3schools.com/react/tryit.asp?filename=tryreact_es6_class_inherit>)

The super() method refers to the parent class. By calling the super() method in the constructor method, we call the parent's constructor method and get access to the parent's properties and methods.

Arrow functions –

A picture containing graphical user interface

Description automatically generated Graphical user interface, website

Description automatically generated with medium confidence

If you have parameters, you pass them inside the parentheses

If you use var outside of a function, it belongs to the global scope. If you use var inside of a function, it belongs to that function. If you use var inside of a block, i.e. a for loop, the variable is still available outside of that block. const is a variable that once it has been created, its value can never change.

Array Methods

There are many JavaScript array methods.One of the most useful in React is the .map() array method.

The .map() method allows you to run a function on each item in the array, returning a new array as the result. In React, map() can be used to generate lists.

A picture containing text

Description automatically generated

Destructuring

We may have an array or object that we are working with, but we only need some of the items contained in these.

Destructuring makes it easy to extract only what is needed.

Graphical user interface, text

Description automatically generatedA picture containing chart

Description automatically generated

If we only want the car and suv we can simply leave out the truck but keep the comma: A picture containing text

Description automatically generatedAnother example is -

Graphical user interface, text, application

Description automatically generated

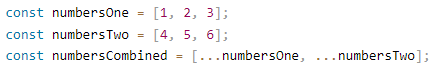
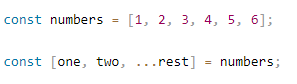
Similarly destructuring objects –

A picture containing text

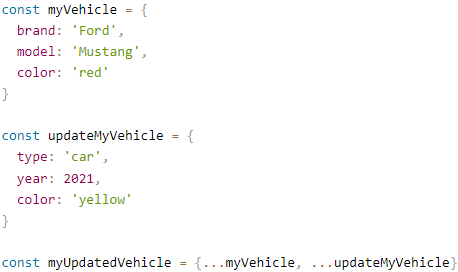
Description automatically generatedA picture containing text

Description automatically generated

The JavaScript spread operator (...) allows us to quickly copy all or part of an existing array or object into another array or object.

 OR - 

Similarly combining two objects -

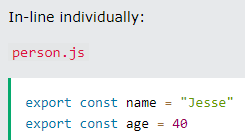


Modules

JavaScript modules allow you to break up your code into separate files. This makes it easier to maintain the code-base. ES Modules rely on the import and export statements.

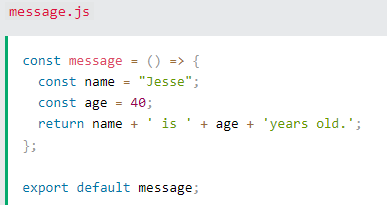
You can export a function or variable from any file. Let us create a file named person.js, and fill it with the things we want to export. There are two types of exports: Named and Default.

Named Exports



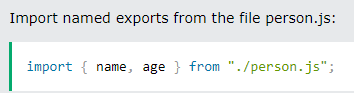
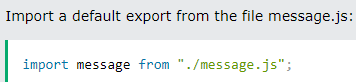
Default Exports

Let us create another file, named message.js, and use it for demonstrating default export. You can only have one default export in a file.



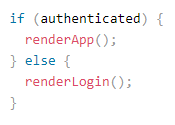
Import

You can import modules into a file in two ways, based on if they are named exports or default exports.

 OR 

Ternary Operator-

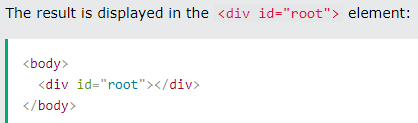
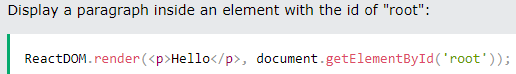




React renders HTML to the web page by using a function called ReactDOM.render().The ReactDOM.render() function takes two arguments, HTML code and an HTML element.

The purpose of the function is to display the specified HTML code inside the specified HTML element. But render where? You'll notice a single <div> in the body of this file. This is where our React application will be rendered.

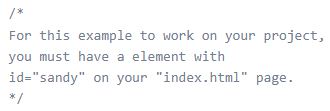
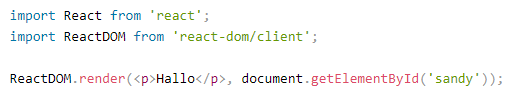




Another example of rendering - <https://www.w3schools.com/react/showreact.asp?filename=demo2_react_render2>

The Root Node

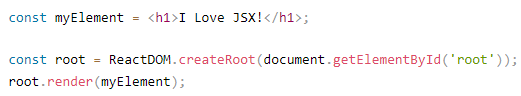
The root node is the HTML element where you want to display the result. It is like a container for content managed by React.It does NOT have to be a <div> element and it does NOT have to have the id='root':



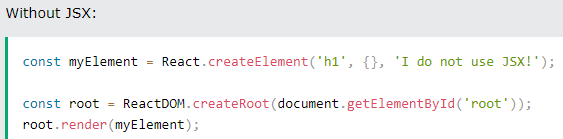
React JSX

JSX stands for JavaScript XML. JSX allows us to write HTML in React. JSX makes it easier to write and add HTML in React. JSX converts HTML tags into react elements.

Here are two examples. The first uses JSX and the second does not: JSX: As you can see in the first example, JSX allows us to write HTML directly within the JavaScript code.



Below one does not use’



Expressions in JSX

Using this Create a list with three list items or write two paragraphs, you must put them inside a parent element, like a div element.

JSX will throw an error if the HTML is not correct, or if the HTML misses a parent element.

NOTE –

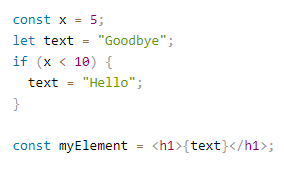
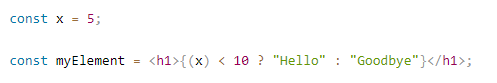
The class attribute is a much used attribute in HTML, but since JSX is rendered as JavaScript, and the class keyword is a reserved word in JavaScript, you are not allowed to use it in JSX.

Use attribute className instead.

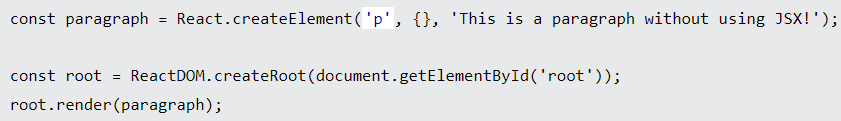
JSX solved this by using className instead. When JSX is rendered, it translates className attributes into class attributes. 

Conditions - if statements in JSX

React supports if statements, but not inside JSX. To be able to use conditional statements in JSX, you should put the if statements outside of the JSX, or you could use a ternary expression instead.

Outside if else etc OR Use ternary operator like thisI

If we want to render html without using JSX then we use React.createElement() like this –



React Components

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.

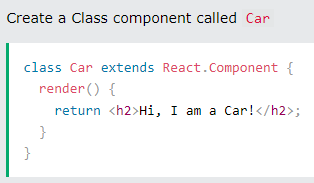
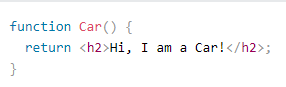
Components come in two types, Class components and Function components.

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

Class Component

A class component must 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.

 Function compo - 

Function Component - A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand.

Now your React application has a component called Car, which returns an <h2> element.

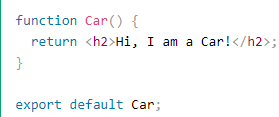
To use this component in your application, use similar syntax as normal HTML: <Car />

Components in Files

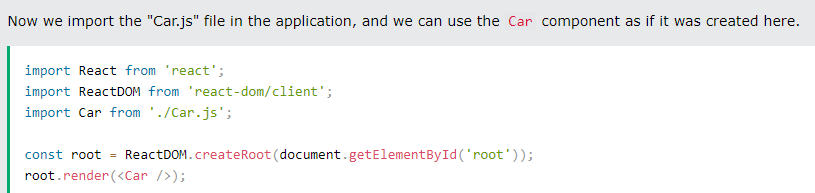
React is all about re-using code, and it is recommended to split your components into separate files.

To do that, create a new file with a .js file extension and put the code inside it.

This is the new file, we named it "Car.js"-



To be able to use the Car component, you have to import the file in your application.



React Class Components

Skipping this for now - <https://www.w3schools.com/react/react_class.asp>  
==================================================================================

Props

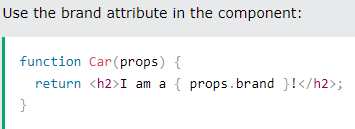
Components can be passed as props, which stands for properties. Props are arguments passed into React components. Props are passed to components via HTML attributes.

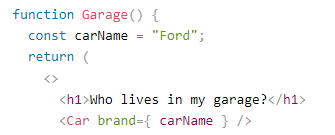
Components in Components - We can refer to components inside other components , just like function calling another function inside.

Eg - <https://www.w3schools.com/react/showreact.asp?filename=demo2_react_component_many>

React Props are like function arguments in JavaScript and attributes in HTML.

Eg -  The component receives the argument as a props object.

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

 like this - 

If you have a variable to send, and not a string as in the example above, you just put the variable name inside curly brackets as seen above. Const carName = “Ford” and brand={carName}

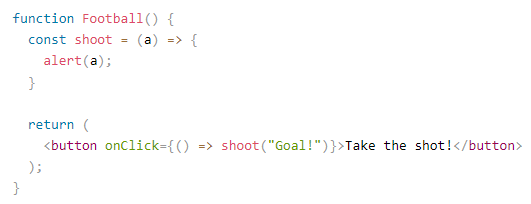
Similarly if it is an object-  using

Just like HTML DOM events, React can perform actions based on user events.’

NOTE - 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()".

**To pass an argument to an event handler, use an arrow function.**



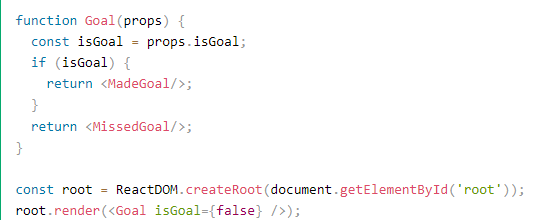
React Conditional Rendering

In React, you can conditionally render components. if Statement. We can use the if JavaScript operator to decide which component to render. If true then this or that

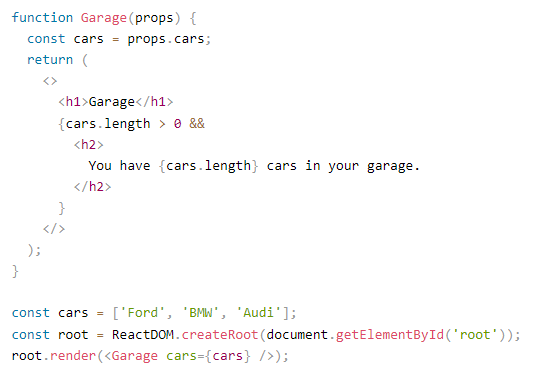
We'll use these two components:

function MissedGoal() { return <h1>MISSED!</h1>;}

function MadeGoal() { return <h1>Goal!</h1>;}

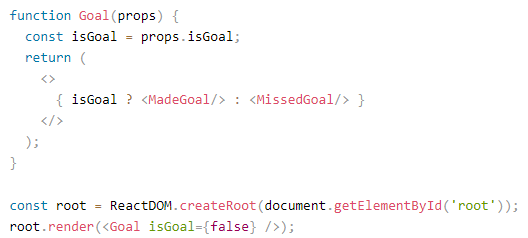
If true then output is – GOAL!

Another way to conditionally render a React component is by using the && operator.



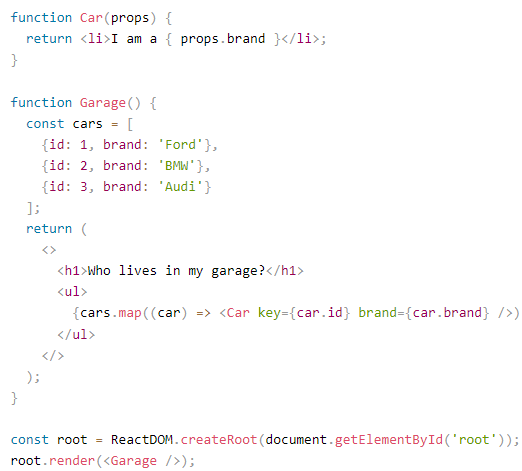
If cars.length > 0 is equates to true, the expression after && will render. If car array is emptied like cars =[]; then only h1 will render - ---------------- // It’s a stupid way of doing avoid I say

Another way to conditionally render elements is by using a ternary operator. ------best way 



React Lists

In React, you will render lists with some type of loop. The JavaScript map() array method is generally the preferred method.

1. B)

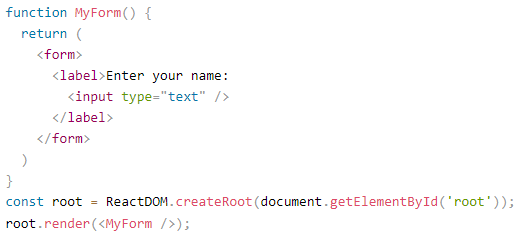
When you run a) this code in your create-react-app, it will work but you will receive a warning that there is no "key" provided for the list items. Keys allow React to keep track of elements. This way, if an item is updated or removed, only that item will be re-rendered instead of the entire list.

Keys need to be unique to each sibling. But they can be duplicated globally. Generally, the key should be a unique ID assigned to each item. As a last resort, you can use the array index as a key.

In a) its without keys in B) its with keys .

React Forms

You add a form with React like any other element:



this will work as normal, the form will submit and the page will refresh. But this is generally not what we want to happen in React. We want to prevent this default behavior and let React control the form.

In HTML, form data is usually handled by the DOM. In React, form data is usually handled by the components. When the data is handled by the components, all the data is stored in the component state. You can control changes by adding event handlers in the onChange attribute.

We can use the useState Hook to keep track of each inputs value and provide a "single source of truth" for the entire application.

Also You can control the submit action by adding an event handler in the onSubmit attribute for the <form>



You can control the values of more than one input field by adding a name attribute to each element.

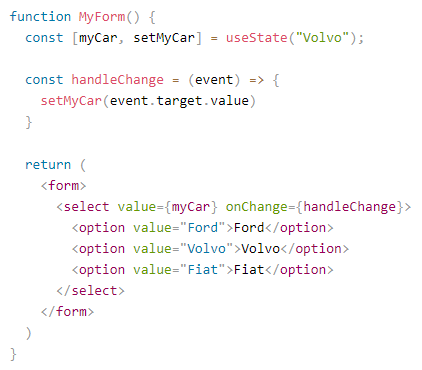
We will initialize our state with an empty object.To access the fields in the event handler use the event.target.name and event.target.value syntax. To update the state, use square brackets [bracket notation] around the property name. Like this –

<https://www.w3schools.com/react/showreact.asp?filename=demo2_react_forms_multiple>

Textarea

In React the value of a textarea is placed in a value attribute. We'll use the useState Hook to mange the value of the textarea: like this - <https://www.w3schools.com/react/showreact.asp?filename=demo2_react_forms_textarea>

Select - A drop down list, or a select box, in React is also a bit different from HTML. In React, the selected value is defined with a value attribute on the select tag. A simple select box, where the selected value "Volvo" is initialized in the constructor.

   
React Router

React Routing is a feature of the React JavaScript library that allows you to handle client-side routing in a Single Page Application (SPA).

In a traditional multi-page web application, each page has its own URL, and when a user clicks a link or enters a URL in the browser's address bar, the server sends a new HTML page to the client. In contrast, in an SPA, the server sends a single HTML page, and the client dynamically updates the content based on user interaction. This means that the URL in the address bar does not change, and the browser's back and forward buttons do not work as expected.

React Routing provides a solution to this problem by allowing you to define a set of routes that correspond to different components of your SPA. When the user clicks a link or enters a URL, React Routing matches the URL to a route and renders the corresponding component. This ensures that the URL in the address bar corresponds to the current view, and allows the browser's back and forward buttons to work as expected.

React Router is a popular library that provides a set of components and utilities for React Routing. It allows you to define routes using a declarative syntax, and provides hooks and APIs for accessing the current route and manipulating the browser history.

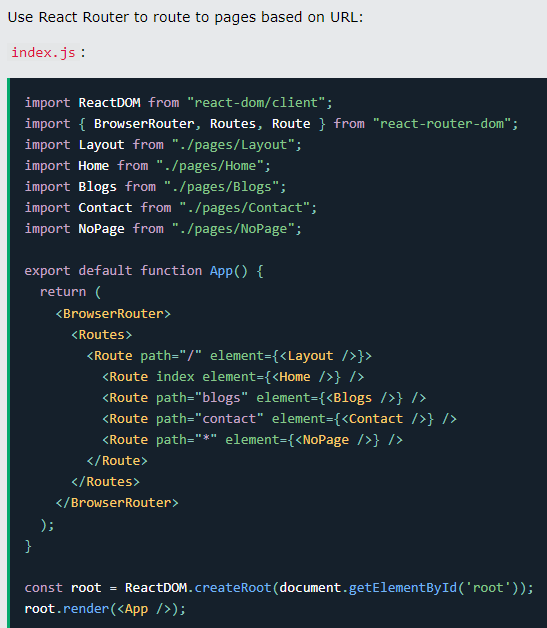
To add React Router in your application, run this in the terminal from the root directory of the application:  or 

To create an application with multiple page routes, let's first start with the file structure.

Within the src folder, we'll create a folder named pages with several files:

src\pages\: like - Layout.js, Home.js, Blogs.js, Contact.js, NoPage.js

Each file will contain a very basic React component.



We wrap our content first with <BrowserRouter>.

Then we define our <Routes>. An application can have multiple <Routes>. Our basic example only uses one.

<Route>s can be nested. The first <Route> has a path of / and renders the Layout component.

The nested <Route>s inherit and add to the parent route. So the blogs path is combined with the parent and becomes /blogs.

The Home component route does not have a path but has an index attribute. That specifies this route as the default route for the parent route, which is /.

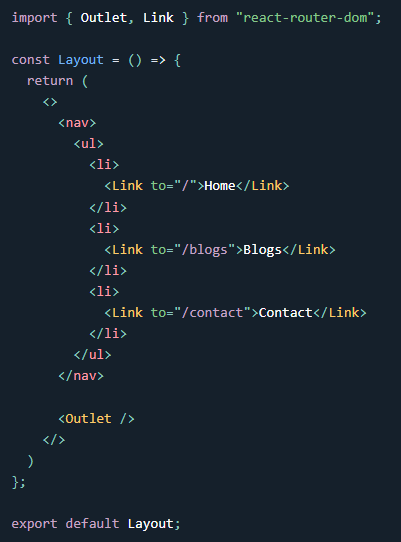
Setting the path to \* will act as a catch-all for any undefined URLs. This is great for a 404 error page.

Pages / Components

The Layout component has <Outlet> and <Link> elements. The <Outlet> renders the current route selected. <Link> is used to set the URL and keep track of browsing history.

Anytime we link to an internal path, we will use <Link> instead of <a href="">.

The "layout route" is a shared component that inserts common content on all pages, such as a navigation menu.

React Memo

Skipping this for now - <https://www.w3schools.com/react/react_memo.asp>

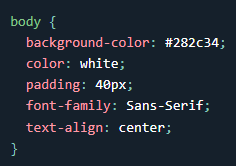
Styling React Using CSS- There are many ways to style React with CSS, this tutorial will take a closer look at three common ways:Inline styling, CSS stylesheets, CSS Modules

Inline Styling 

In JSX, JavaScript expressions are written inside curly braces, and since JavaScript objects also use curly braces, the styling in the example above is written inside two sets of curly braces {{}}.

NOTE - Since the inline CSS is written in a JavaScript object, properties with hyphen separators, like background-color, must be written with camel case syntax. 

You can also create an object with styling information, and refer to it in the style attribute.

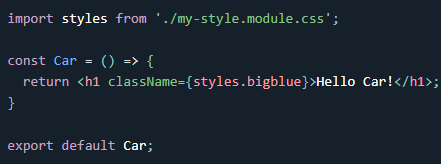
OR 

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. As shown above after OR –

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.

The CSS inside a module is available only for the component that imported it, and you do not have to worry about name conflicts. Create the CSS module with the .module.css extension, example: my-style.module.css.



Import it on application site – in this .bigblue{ css} will be written which will be applied on it upon import.

Styling React Using Sass

Will DO THIS LATER - <https://www.w3schools.com/react/react_sass_styling.asp>

React Hooks

Hooks were added to React in version 16.8.

Hooks allow function components to have access to state and other React features. Because of this, class components are generally no longer needed. Hooks allow us to "hook" into React features such as state and lifecycle methods.

Example - <https://www.w3schools.com/react/showreact.asp?filename=demo2_react_hooks>



There are 3 rules for hooks:

* Hooks can only be called inside React function components.
* Hooks can only be called at the top level of a component.
* Hooks cannot be conditional

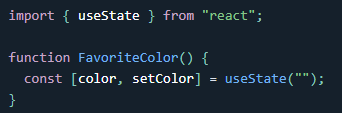
NOTE - If you have stateful logic that needs to be reused in several components, you can build your own custom Hooks.

useState Hook

The React useState Hook allows us to track state in a function component. To use the useState Hook, we first need to import it into our component. At the top of your component, import the useState Hook. 

Initialize useState - We initialize our state by calling useState in our function component.

useState accepts an initial state and returns two values:The current state & A function that updates the state.

 The first value, color, is our current state. The second value, setColor, is the function that is used to update our state.

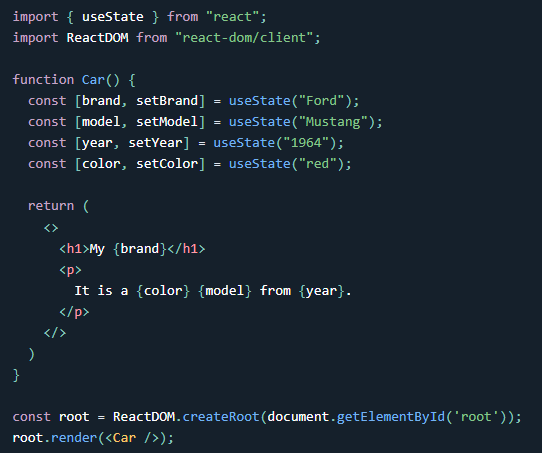
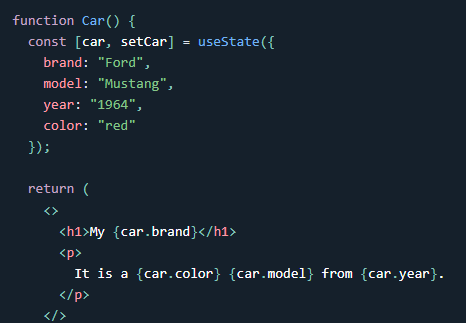
Lastly, we set the initial state to an empty string: useState("").

Read State- We can now include our state anywhere in our component. ( const [color, setColor] = useState("red") )

Update State

To update our state, we use our state updater function. We should never directly update state. Ex: color = "red" is not allowed use a button instead. 

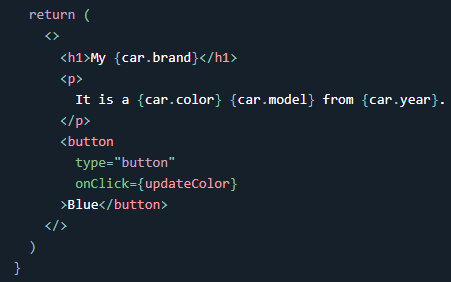
The useState Hook can be used to keep track of strings, numbers, booleans, arrays, objects, and any combination of these! We could create multiple state Hooks to track individual values.

1.  b) 

Or, we can just use one state and include an object instead! Like In b).

Updating Objects and Arrays in State –

What if we only want to update the color of our car? If we only called setCar({color: "blue"}), this would remove the brand, model, and year from our state. We can use the JavaScript spread operator to help us.

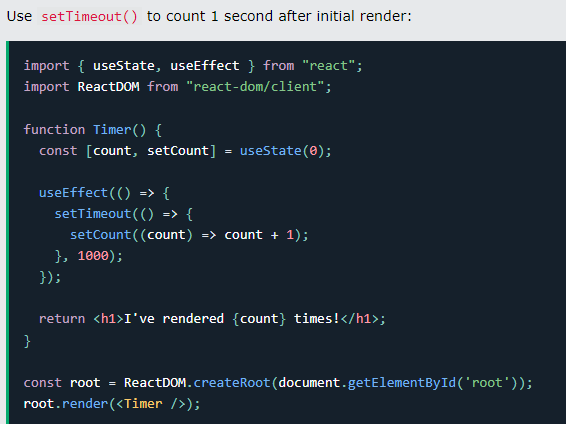
<https://www.youtube.com/watch?v=Bn62TH0ulps&ab_channel=CodeStepByStep>



The useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers.

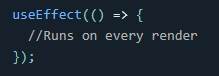
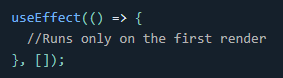
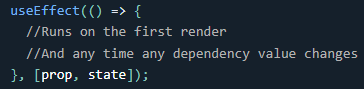
useEffect accepts two arguments. The second argument is optional.

useEffect(<function>, <dependency>)

 0,1,2…..n

But wait!! It keeps counting even though it should only count once!

useEffect runs on every render. That means that when the count changes, a render happens, which then triggers another effect. This is not what we want. There are several ways to control when side effects run. We should always include the second parameter which accepts an array. We can optionally pass dependencies to useEffect in this array. Like –

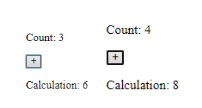
  

Here is an example of a useEffect Hook that is dependent on a variable. If the count variable updates, the effect will run again.

<https://www.youtube.com/watch?v=zJeyz5mIXGQ&ab_channel=CodeStepByStep>

UseEffect with props and with conditions on specific states –

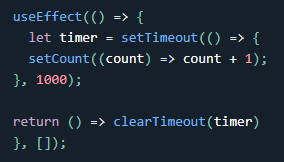
<https://www.youtube.com/watch?v=80u8D2-Ro3s&ab_channel=CodeStepByStep>

If there are multiple dependencies, they should be included in the useEffect dependency array.

Effect Cleanup

Some effects require cleanup to reduce memory leaks. Timeouts, subscriptions, event listeners, and other effects that are no longer needed should be disposed. We do this by including a return function at the end of the useEffect Hook.

 Not sure how imp it is

React uses a tool called "Babel" to compile JSX. Babel is a popular JavaScript compiler that can be configured to transform modern JavaScript syntax and features into a form that can be understood by older browsers. JSX is a syntax extension that allows developers to write HTML-like syntax within JavaScript code, and Babel is used to transform this syntax into regular JavaScript code that can be executed by the browser. In addition to compiling JSX, Babel can also be configured to perform other transformations such as code minification and optimizations.

useContext Hook

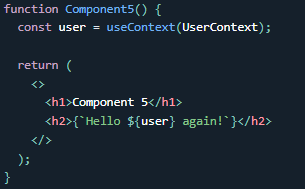
It can be used together with the useState Hook to share state between deeply nested components more easily than with useState alone.

State should be held by the highest parent component in the stack that requires access to the state. To do this without Context, we will need to pass the state as "props" through each nested component. This is called "prop drilling".

Even though components 2-4 did not need the state, they had to pass the state along so that it could reach component 5. The solution is to create context. To create context, you must import it - 

Next we'll use the Context Provider to wrap the tree of components that need the state Context. Wrap child components in the Context Provider and supply the state value. Now, all components in this tree will have access to the user Context. Like in A).

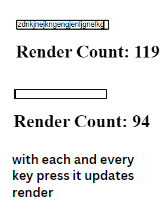
1.  B) 

And in order to use the Context in a child component, we need to access it using the useContext Hook. Like in B)

Check out full example - <https://www.w3schools.com/react/showreact.asp?filename=demo2_react_context2>



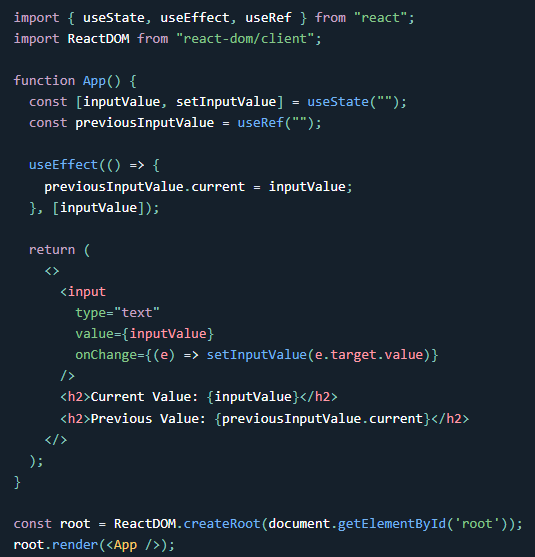
The useRef Hook allows you to persist values between renders. It can be used to store a mutable value that does not cause a re-render when updated. It can be used to access a DOM element directly. Use useRef to track application renders.

useRef() only returns one item. It returns an Object called current. When we initialize useRef we set the initial value: useRef(0). Its useful but will require little practise to be used smartly in applications.

It's like doing this: const count = {current: 0}. We can access the count by using count.current.

Tracking State Changes - The useRef Hook can also be used to keep track of previous state values. This is because we are able to persist useRef values between renders.



useRef is a hook provided by React that allows you to create a mutable object that persists across renders. In this example, previousInputValue is initialized using useRef to an empty string.

In the useEffect hook, we are setting the current property of previousInputValue to the current value of inputValue whenever inputValue changes. The useEffect hook only runs when the value of inputValue changes because of the dependency array passed as the second argument.

In the return statement, the current value of inputValue and the previous value of inputValue (which is stored in previousInputValue.current) are rendered to the screen.

The useRef hook is useful for storing mutable values that do not trigger a re-render when they are updated. In this example, previousInputValue.current is updated without triggering a re-render of the component. This is because updating the current property of a useRef object does not trigger a re-render, unlike updating the state using useState.

In the useEffect, we are updating the useRef current value each time the inputValue is updated by entering text into the input field.



React concept which is called context. Context allows us to pass data between any components in our application without using props and that is very useful if you have some general application-wide data like is the user logged in, so such a central vital question which affects a lot of parts in your application, which you simply don't want to pass around with props because that would be madness, you would have to pass props from everywhere to everywhere, it would be super annoying. So context is some behind the scenes data transfer mechanism you could say and it's built right into.

