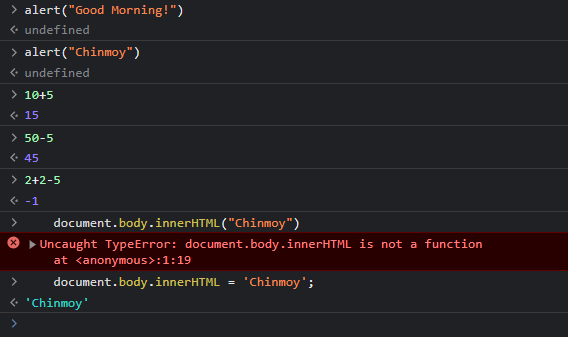
Exercise: 1

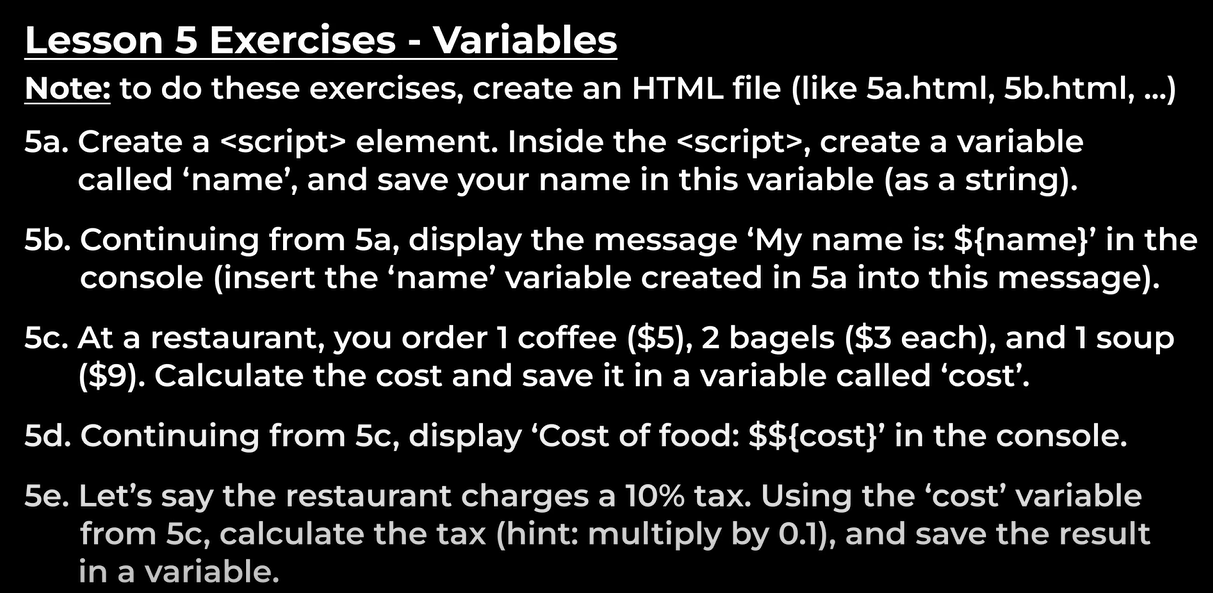


**Math.ceil()** rounds a number UP to the nearest integer.

The **Math.floor()** method rounds a number DOWN to the nearest integer.

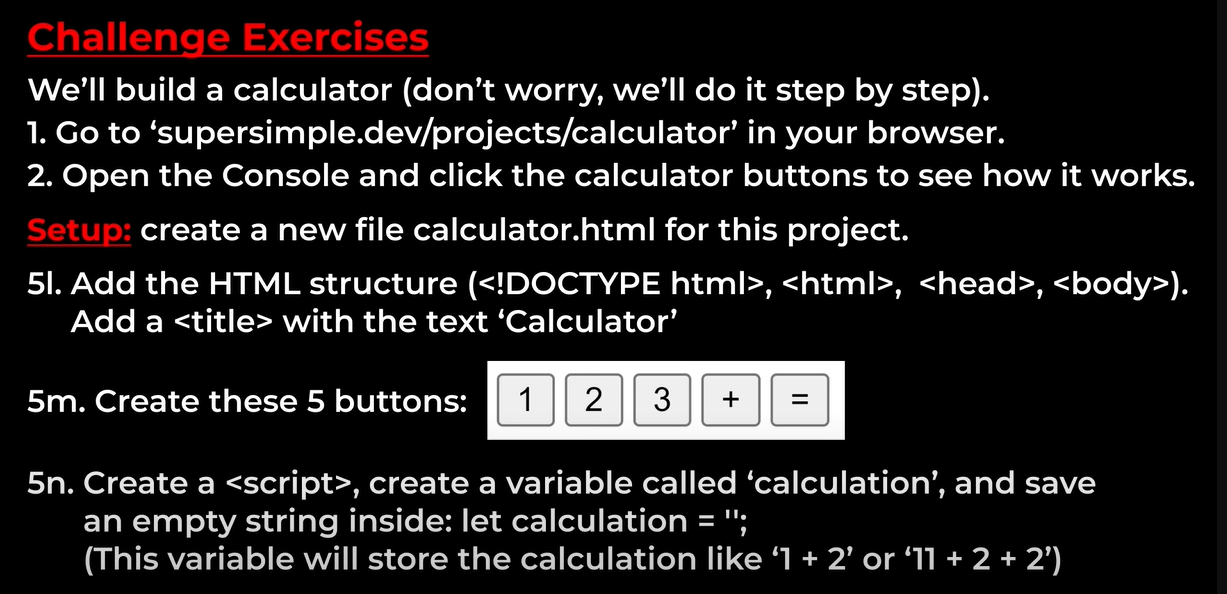
Concatination 🡪 Combining Strings Together.

Exercise 5:



A black and white text on a black background

Description automatically generated



Template Literals: `My name is ${name}`

Template literals (Template strings)

Template literals are literals delimited with backtick (`) characters, allowing for multi-line strings, string interpolation with embedded expressions, and special constructs called tagged templates.

Template literals are sometimes informally called template strings, because they are used most commonly for string interpolation (to create strings by doing substitution of placeholders). However, a tagged template literal may not result in a string; it can be used with a custom tag function to perform whatever operations you want on the different parts of the template literal.

A screenshot of a computer program

Description automatically generated

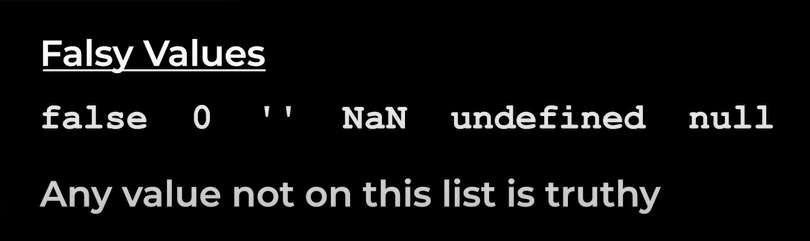
Result 🡪 A screenshot of a calculator

Description automatically generated

A screenshot of a calculator

Description automatically generated

Truthy value and Falsy Value



A black background with white text

Description automatically generated

As the most popular frontend library, everyone wants to learn React. ReactJS is fundamentally JavaScript. But that doesn't mean you must learn everything in JavaScript to move to ReactJS. Understanding the fundamental JavaScript concepts will help you grasp React concepts more easily, and ultimately it'll accelerate your ability to work on projects.

Let's outline essential concepts you should know about JavaScript before moving to ReactJS.

1. Arrow Functions

Arrow functions are widely used in React. As of version 16.8, React moved from class-based components to functional components. Arrow functions allow you to create functions with a shorter syntax.

Let's illustrate that in the following examples:

Regular function

function greeting() {

return 'hello'

}

console.log(greeting()) //hello

Arrow Function

let greeting = () => 'hello'

console.log(greeting()) //hello

The two functions above have the same output, although the syntax is different. The arrow function looks shorter and cleaner than the regular function. Usually, the React components have the following structure:

import React from 'react'

const Book = () => {

return (

<div>Book</div>

)

}

export default Book

Arrow functions do not have names. If you wish to name it, assign it to a variable. The difference between a regular and an arrow function is more than just the syntax. Learn more about arrow functions in Mozilla developers' documentation.

2. Destructuring

Destructuring is used to obtain data from complex data structures. In JavaScript, arrays and objects can store many values. You can manipulate the values and use them in different parts of the application.

To get these values, you have to destructure the variable. Depending on the data structure you are dealing with, you can use the dot (.) notation or the bracket notation. For example:

const student = {

'name': 'Mary',

'address': 'South Park, Bethlehem',

'age': 15

}

Destructuring:

console.log(student.name) // output Mary

In the above example, the dot notation accesses the value of the key 'name'. In ReactJS, you will use the destructuring concept to obtain and share values in your application. Destructuring helps to avoid repetition and makes your code more readable.

3. Array Methods

You'll encounter arrays several times while working on React projects. An array is a collection of data. You store data in arrays, so you can reuse them when needed.

Array methods are primarily used to manipulate, retrieve, and display data. Some commonly used array methods are map(), filter(), and reduce(). You must be familiar with the array methods to understand when to apply each.

For example, the map() method iterates over all the items in an array. It acts on each element of the array to create a new array.

const numbers = [9, 16, 25, 36];

const squaredArr = numbers.map(Math.sqrt) // 3,4,5,6

You will use array methods a lot in ReactJS. You will use them to convert arrays to strings, join, add items, and remove items from other arrays.

4. Short Conditionals

Conditionals are statements that JavaScript uses to make decisions in code. Short conditionals include && (and), II (or), and the Ternary Operator. These are shorter expressions of conditions and the if/else statements.

The following example shows how to use a ternary operator.

Code with if/else statement:

function openingTime(day) {

if (day == SUNDAY) {

return 12;

}

else {

return 9;

}

}

Code with Ternary Operator:

function openingTime(day) {

return day == SUNDAY ? 12 : 9;

}

Learn about different types of conditionals with a special focus on short conditionals. These are widely used in React.

5. Template Literals

Template Literals use back-ticks (``) to define a string. Template literals allow you to manipulate string data making them more dynamic. Tagged template literals allow you to perform operations within a string. These are shorter expressions of conditions and the if/else statements.

For example:

let firstName = "Jane";

let lastName = "Doe";

let text = `Welcome ${firstName}, ${lastName}!`; // Welcome Jane Doe!

6. Spread Operators

The Spread operator (...) copies the values of an object or array into another. Its syntax consists of three dots followed by the name of the variable. For example (...name). It merges the properties of two arrays or objects.

The following example shows how to use the spread operator to copy the values of one variable to another.

const names = ['Mary', 'Jane'];

const groupMembers = ['Fred', ...names, 'Angela']; // ["Fred", "Mary", "Jane", "Angela"]

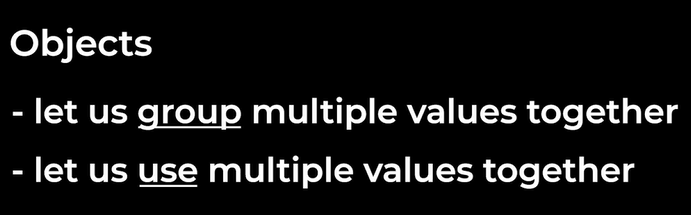
You can use the spread operator to do a number of operations. These include copying the contents of an array, inserting an array into another, accessing nested arrays, and passing arrays as arguments. You may use the spread operator in ReactJS to handle the state changes in components.

Why Learn ReactJS?

ReactJS is popular for good reason. It has a short learning curve, reliable, and renders quickly to the DOM. It supports standalone components and has great debugging tools.

ReactJS incorporates new JavaScript concepts from ECMAScript 6 (ES6). Learning fundamental concepts in JavaScript will make it easier to develop projects in ReactJS.

To top it all, ReactJS has a great community that is constantly releasing new updates. If you want to learn a JavaScript library, ReactJS would be a great pick. The Next.js framework complements the limitations of ReactJS. A combination of the two makes ReactJS a powerful front-end library.



A screenshot of a computer program

Description automatically generated

Auto-Boxing: -

A black background with white text

Description automatically generated

A screen shot of a computer program

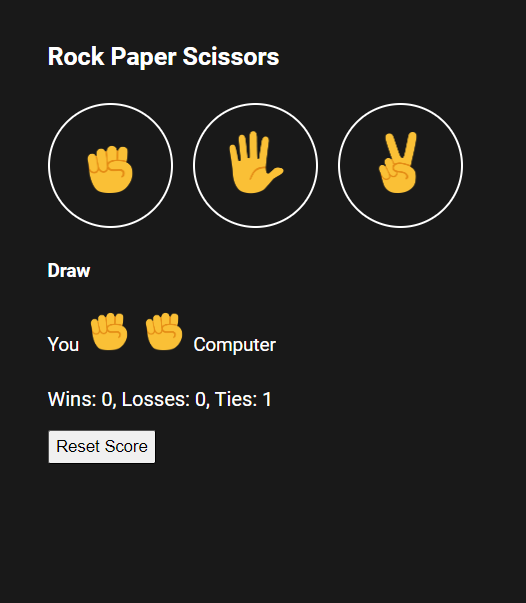
Description automatically generated



A screen shot of a computer program

Description automatically generated

Project Rock Paper Scissors:-

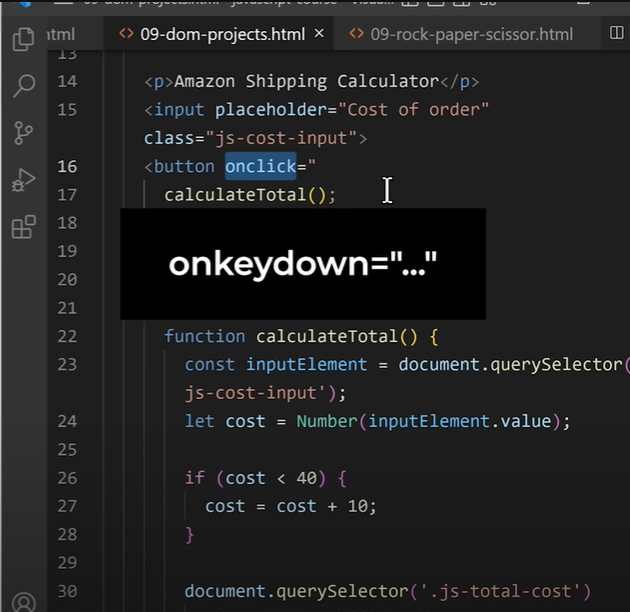


innerText and innerHTML

A screen shot of a computer program

Description automatically generated

Onkeydown



A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

A screen shot of a computer

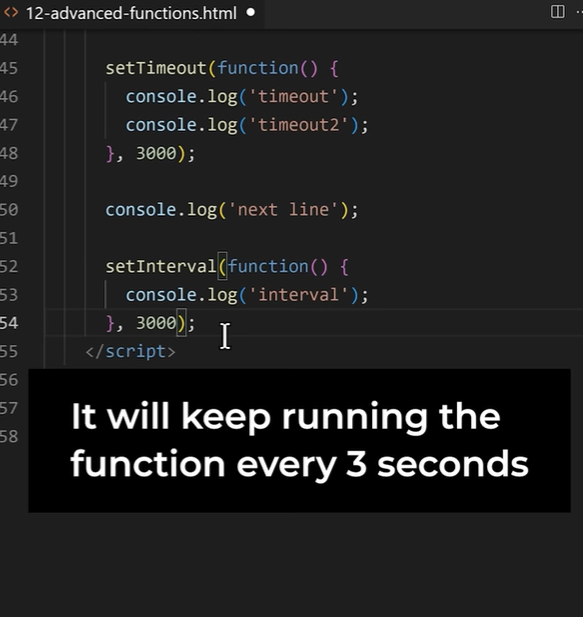
Description automatically generated

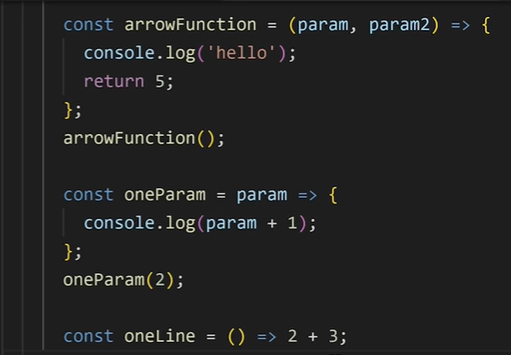
For - \* / string convert to number

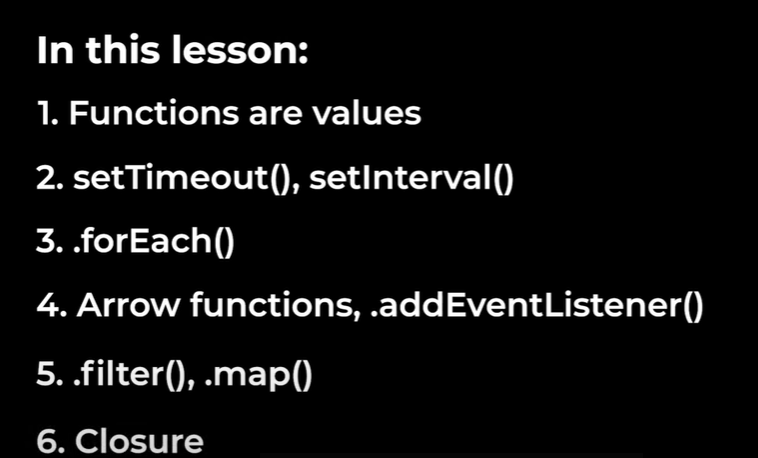
Document object -🡪 Webpage

Window object 🡪 Browser









Advanced Function Exercises: -

A black and white screen with white text

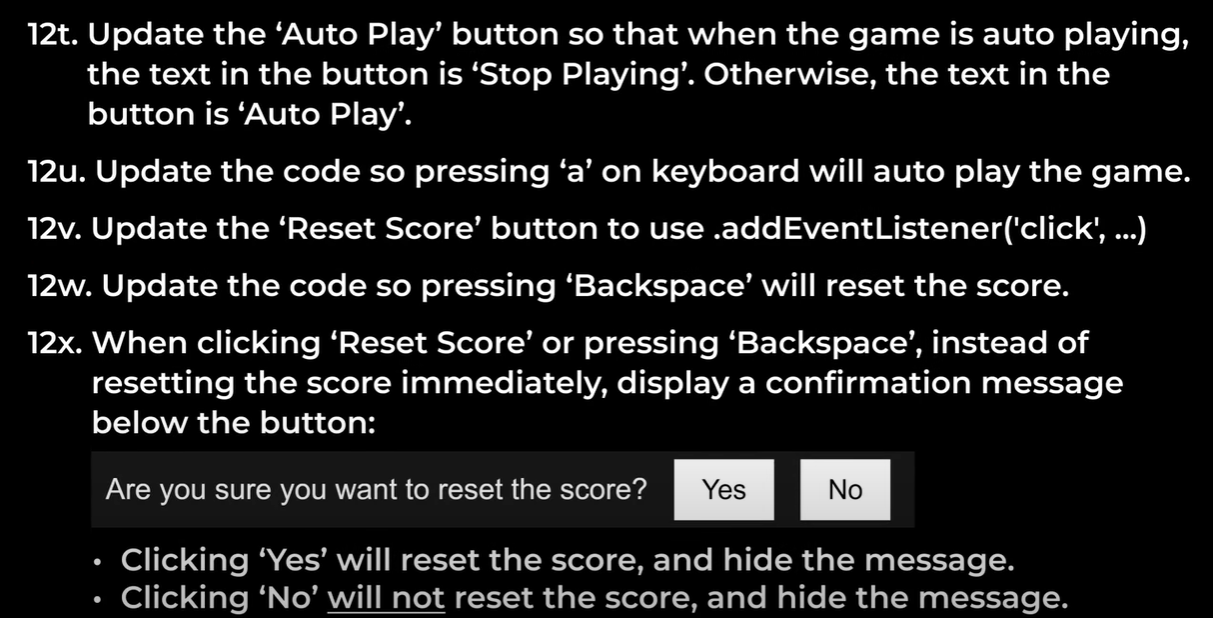
Description automatically generated

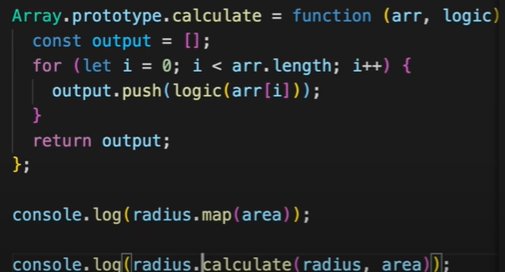
A black and white text on a black background

Description automatically generated

A screenshot of a black and white screen

Description automatically generated







Web API

Microtask Queue 🡪 Promises

Callback Queue -> Set Time Out Dom API

Event Loop

Call Stack

Starvation:-

Microtask Queue ---- many

Javascript Engine:- <https://www.youtube.com/watch?v=2WJL19wDH68>

Callback ->

Importance of call back

Issue with callback

* Callback Hell
* Inversion of Control

Some Topics:-

Event Delegation,

Bubbling and Capturing,

CORS – Cross Origin Resource Sharing

Throttling and Debouncing

