**DIFF B/W LOCAL STORAGE AND SESSION STORAGE**

**LocalStorage**

* **Scope**: LocalStorage is scoped to the origin (domain, protocol, and port). Data stored in LocalStorage is accessible to all pages with the same origin.
* **Persistence**: Data stored in LocalStorage has no expiration time. It remains available even after the browser is closed and reopened.
* **Storage Limit**: Typically, LocalStorage can store around 5-10 MB of data per origin, though this limit can vary between browsers.
* **Use Case**: Suitable for data that needs to persist across sessions, such as user preferences, application settings, or cached content.

**SessionStorage**

* **Scope**: SessionStorage is scoped to the browser tab. Data stored in SessionStorage is accessible only within the same tab (or window). Different tabs or windows will have different SessionStorage.
* **Persistence**: Data stored in SessionStorage is only available for the duration of the page session. It is cleared when the tab or window is closed.
* **Storage Limit**: Typically, SessionStorage can store around 5-10 MB of data per origin, similar to LocalStorage.
* **Use Case**: Suitable for data that only needs to persist for a single page session, such as temporary form data, one-time calculations, or transient user interactions.  
    
  => javascript is giving instruction to our computer

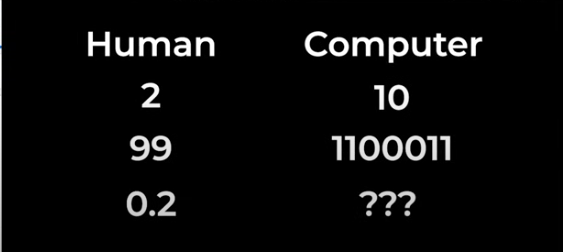
document.body.innerHTML="HI"  
=>after executing this code removed all website content and replaced to Hi;  
used to modify the webpage  
  
Syntax  
the rules that we follow when using a programming Language  
  
  
NUMBERS AND MATHS  
  
order of operation  
  
10\*3  
10/2

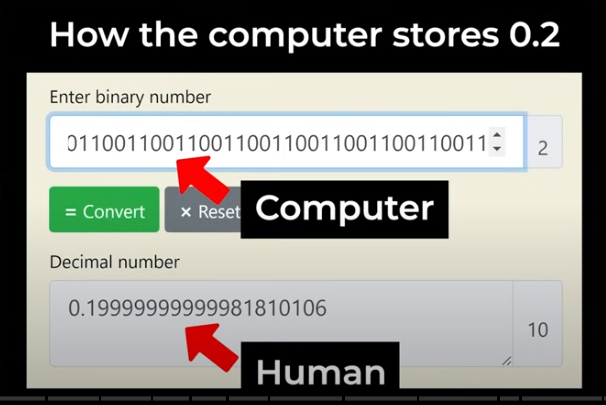
2+ 2

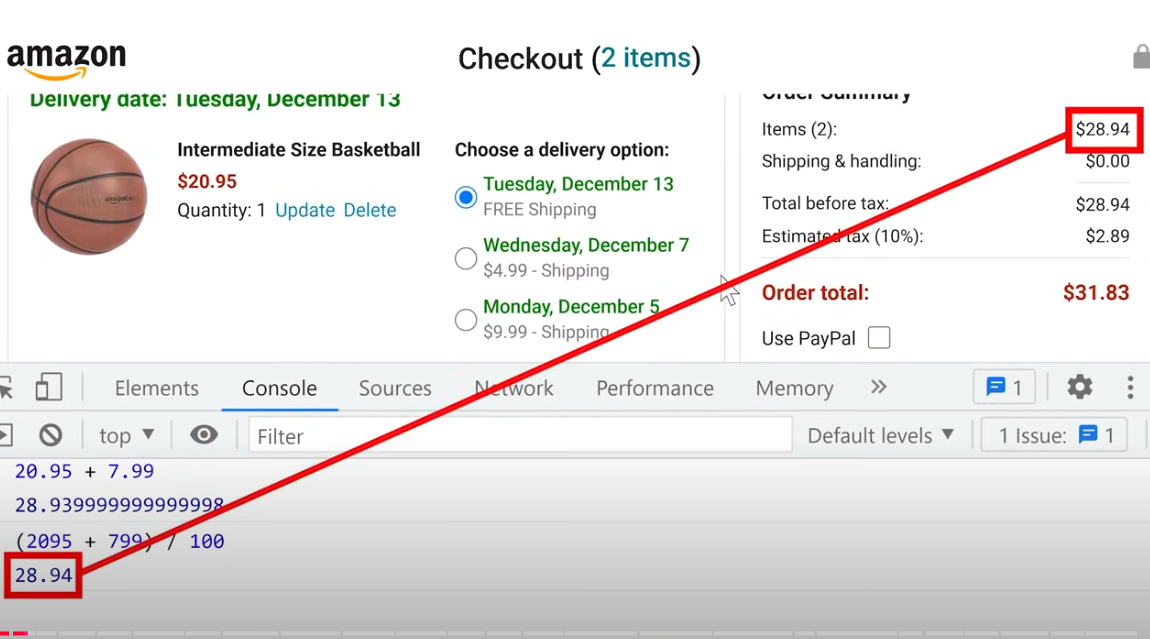
2 – 2

the characters in the middle called operators.

We can use (…) to control which part gets done first  
  
  
intigers , floats  
  
adding folating numbers its weird , you will not get accurate number  
  
the reason is because how computers stored numbers  
  
computers can store only 0 and 1 so we can see mismatch while calculating







Calculation with floats are some times inaccurate ,  
   
when working with money best practice is to do calculation in cents and convert back in to dollars  
  
How to round a number in javascript?  
  
Math.round(2.2);  
  
***TEXT /STRING***  
‘first string’  
+ =>con catination  
  
to find syntax ?  
  
typeof 2

typeof 'hello'  
  
  
<style>

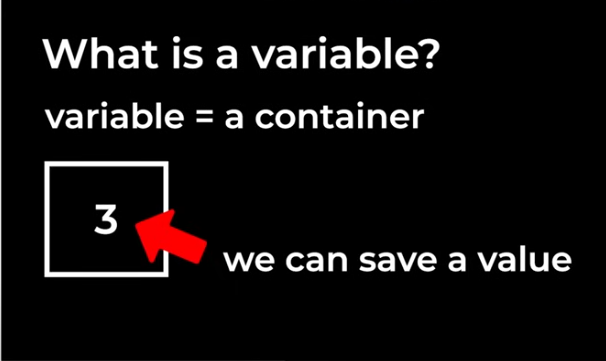
button{

background-color: red; -🡪 css property : value

color:yellow;

}

</style>



In JavaScript, a variable is used to store data that can be used later in the program. Variables can hold different types of data such as numbers, strings, objects, arrays, etc. Here's how to declare and use variables in JavaScript:

**Declaring Variables**

There are three ways to declare variables in JavaScript:

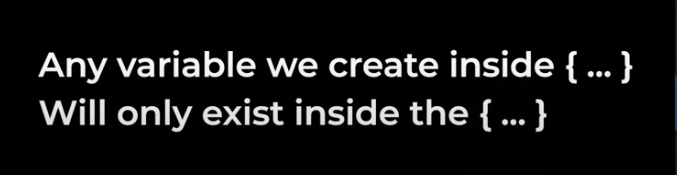
1. var: The oldest way, with function scope or globally scoped.
2. let: Introduced in ES6, has block scope.
3. const: Also introduced in ES6, used for values that won't be reassigned, and has block scope.

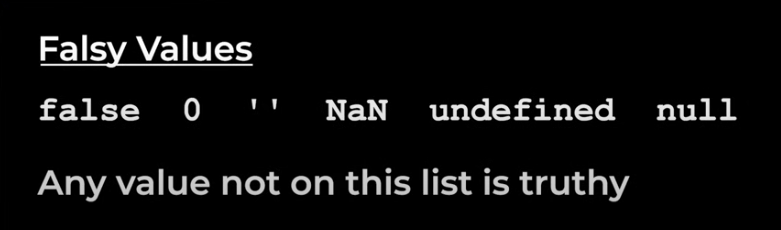
**Boolean**

In JavaScript, a boolean is a data type that can hold one of two values: true or false. Booleans are often used in conditional statements, such as if statements, to control the flow of a program based on conditions.

**If statement**  
  
An if statement evaluates a condition, and if it's true, the code block inside the if statement is executed. If the condition is false, it can either skip the block or move to an else or else if block.  
  
Math.random (generate a number in between o to 1)  
  
 **AND (&&)**

 **OR (||)**

 **NOT (!)**



Object  
  
In JavaScript, an **object** is a collection of key-value pairs where each key is a property (or method) and the corresponding value can be any data type (number, string, array, another object, or a function). Objects are used to store and organize data in a structured way.

### Creating an Object

There are several ways to create an object in JavaScript:

#### 1. **Object Literal Syntax:**

The most common way to create an object is using the object literal syntax, which is simply a pair of curly braces {} containing key-value pairs.

javascript

Copy code

// Creating an object

let person = {

firstName: "John",

lastName: "Doe",

age: 30,

isMarried: true

};

console.log(person); // Outputs the entire object

### Converting an Object to a JSON String

javascript

Copy code

const person = { name: "John", age: 30, isMarried: false };

const jsonString = **JSON.stringify**(person);

console.log(jsonString);

### // Output: {"name":"John","age":30,"isMarried":false} Converting an Array to a JSON String

javascript

Copy code

const numbers = [1, 2, 3, 4];

const jsonString = **JSON.stringify**(numbers);

console.log(jsonString);

**// Output: [1,2,3,4]**

### Converting a JSON String into an Object

javascript

Copy code

const jsonString = '{"name": "John", "age": 30, "isMarried": false}';

const person = **JSON.parse**(jsonString);

console.log(person);

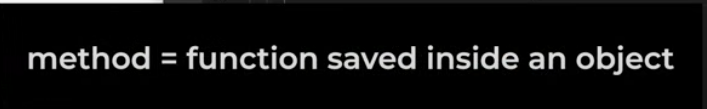
// Output: {name: "John", age: 30, isMarried: false}

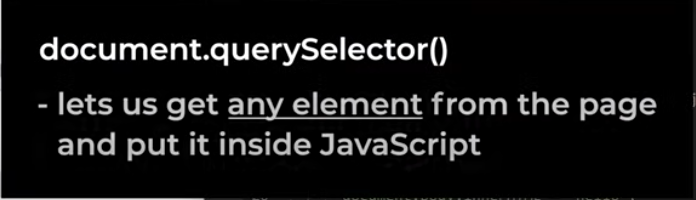
**console.log(person.name); // Output: John**The **DOM (Document Object Model)** is an essential concept in web development, representing the structure of an HTML or XML document in a way that makes it accessible and manipulable by JavaScript.

**Document is a object**

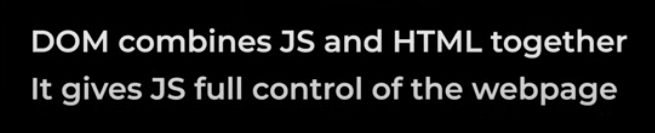
**document.body.innerHTML = "ssssss";**

**document.title = "new Title";**

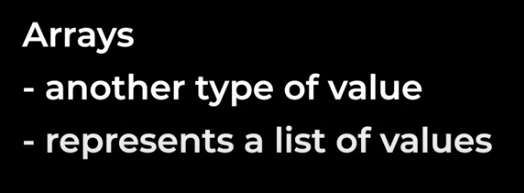




document.querySelector(‘button’) console.log(document.querySelector('button').innerHTML);



Array



JavaScript arrays are data structures used to store multiple values in a single variable. They are versatile and allow for a variety of operations. Here's a detailed overview:

### ****Creating Arrays****

1. **Using an Array Literal (Preferred)**

const fruits = ["apple", "banana", "cherry"];

**Using the Array Constructor**

const numbers = new Array(1, 2, 3, 4); // Creates [1, 2, 3, 4]

const emptyArray = new Array(5); // Creates an array with 5 empty slots

### ****Accessing Elements****

* Array elements are accessed using their **index**, which starts at 0.

**console.log(fruits[0]); // "apple"**

**console.log(fruits[2]); // "cherry"**

**Modify elements by assigning values:**

fruits[1] = "blueberry"; // Updates the second element to "blueberry"

**Common Methods**

1. **Adding Elements**
   * push(): Adds to the end

javascript

Copy code

fruits.push("orange");

* + unshift(): Adds to the start

javascript

Copy code

fruits.unshift("grape");

1. **Removing Elements**
   * pop(): Removes the last element

javascript

Copy code

fruits.pop(); // Removes "orange"

* + shift(): Removes the first element

javascript

Copy code

fruits.shift(); // Removes "grape"

1. **Finding Elements**
   * indexOf(): Finds the index of a value

javascript

Copy code

console.log(fruits.indexOf("banana")); // 1

* + includes(): Checks if an element exists

javascript

Copy code

console.log(fruits.includes("apple")); // true

1. **Iterating Over Arrays**
   * Using for loop:

for (let i = 0; i < fruits.length; i++) {

console.log(fruits[i]);

}

* + Using for...of:

for (const fruit of fruits) {

console.log(fruit);

}

* + Using forEach():

fruits.forEach(fruit => console.log(fruit));

1. **Transforming Arrays**
   * map(): Creates a new array by applying a function to each element

const lengths = fruits.map(fruit => fruit.length);

* + filter(): Creates a new array with elements that pass a condition

const shortFruits = fruits.filter(fruit => fruit.length <= 5);

1. **Sorting and Reversing**
   * sort(): Sorts elements alphabetically or using a comparator function

fruits.sort();

* + reverse(): Reverses the array order

fruits.reverse();

1. **Combining and Splitting**
   * concat(): Combines two or more arrays

const moreFruits = ["kiwi", "pineapple"];

const allFruits = fruits.concat(moreFruits);

* + join(): Joins elements into a string

console.log(fruits.join(", "));

1. **Slicing and Splicing**
   * slice(): Extracts a portion of the array (non-destructive)

const citrus = fruits.slice(1, 3); // ["banana", "cherry"]

* + splice(): Adds/removes elements (destructive)fruits.splice(1, 1, "mango"); // Replaces "banana" with "mango"

**Special Features**

1. **Length Property**
   * Automatically adjusts when adding/removing elements.

console.log(fruits.length); // Current number of elements

1. **Sparse Arrays**
   * Arrays can have "holes" (undefined elements).

const sparseArray = [1, , 3];

console.log(sparseArray[1]); // undefined

1. **Multidimensional Arrays**
   * Arrays can contain other arrays.

const matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9],

];

console.log(matrix[1][2]); // 6

**Example: Putting It All Together**

javascript

Copy code

const numbers = [10, 20, 30, 40];

numbers.push(50); // Add 50 to the end

numbers.shift(); // Remove 10 from the start

const doubled = numbers.map(num => num \* 2); // [40, 60, 80, 100]

const filtered = doubled.filter(num => num > 50); // [60, 80, 100]

console.log(filtered); // [60, 80, 100]

**Advanced Functions**

A function is another type of value we can store it in variable  
  
const variable = function abc(){

console.log("hello");

}

console.log(variable); // output abc()

console.log(typeof variable); // output= Fuction  
variable(); // output = hello

in javascript we not need function name to access function , we can use variable too access function

**Anonymous Functions**A function without function name is called anonymous function  
  
const variable = function(){

console.log("hello");

}

console.log(variable); output variable()

console.log(typeof variable); // output= Fuction  
variable(); // output = hello  
  
**We can save the value in an object**

const object1 = {

id:2,

msg : function(){

console.log('greeting asdasd');

}

};

console.log(object1.msg()); // output greeting asdasd

**Callback Function**

**Callback Function:**

A callback function can be defined and passed as an argument to another function.

function greet(name, callback) {

    console.log(`Hello, ${name}`);

    callback();

}

function sayGoodbye() {

    console.log("Goodbye!");

}

// Passing `sayGoodbye` as a callback to `greet`

greet("Alice", sayGoodbye);

/\* Output:

Hello, Alice

Goodbye

Fuctions are like values , anything we can do values , we can do functions



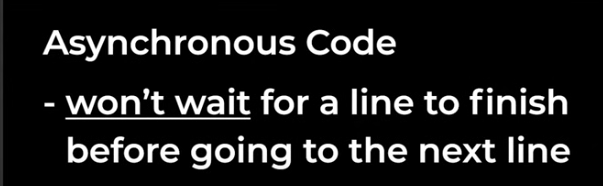
setTimeout(function(){

    console.log('hello');

},3000);

Syntax

setTimeout(function , milliseconds );



setTimeout(function(){

    console.log('hello');

},3000);

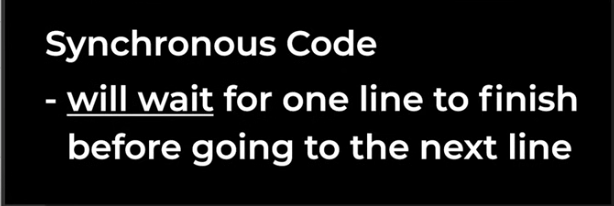
console.log('next line');

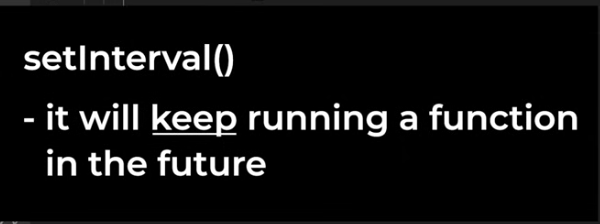
//output

next line

hello



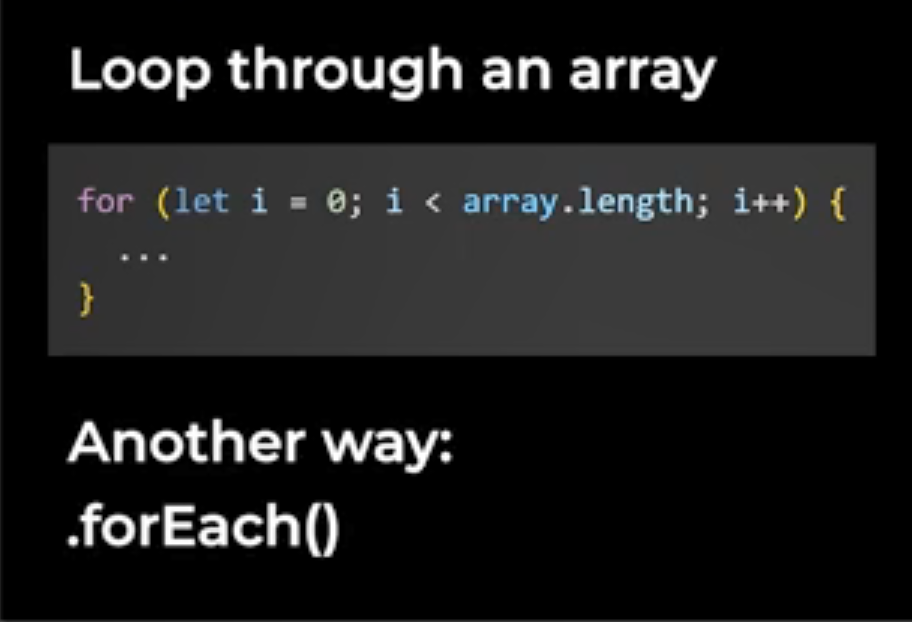




setInterval(function(){

    console.log('hello');

},3000);



['value1',

'value2',

'value3'].forEach(function(value , index){

    console.log(value);

    console.log(index);

    });

Instead of **continue** we can use **return** here , while using return its didn’t execute while condition is satisfied and remain will will execute

['value1',

'value2',

'value3'].forEach(function(value , index){

    if(value === 'value2'){

      return;

    }

    console.log(value);

    console.log(index);

    });

//output

value1

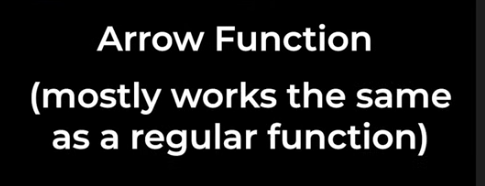
0

value3

2

**Syntax of arrow function**

(param1, param2, ..., paramN) => { /\* function body \*/ }



const abc = ()=>{

    console.log('arrow function created')

}

abc();

//if it is one param () is optional

const abc = param =>{

    console.log(param + 1)

}

abc(2);

output will be 3

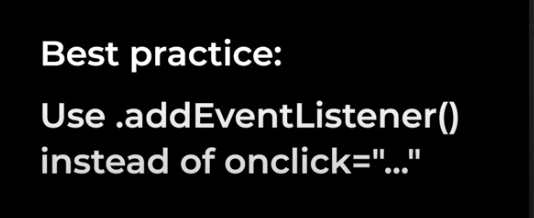
//if it is one line { } is optional , we can write it same line

const abc = param => console.log(param + 1)

abc(2);

output will be 3





addEventListener is a method provided by the DOM to attach event handlers to elements. It allows you to:

* Add multiple event listeners to the same element.
* Use event capturing or bubbling.
* //SYNTAX
* element.addEventListener(event, listener, options);

 **element**: The DOM element to attach the listener to.

 **event**: A string representing the event type (e.g., 'click', 'mouseover').

 **listener**: The function to execute when the event occurs.

 **options** *(optional)*:

* An object specifying event options or a boolean to indicate the capture phase.

<button class="js-button" onclick="">click</button>

<script>

    const val = document.querySelector('.js-button');

    val.addEventListener('click' , () =>{

            console.log('val');

    });

    //we can add multiple function same onclick

    val.addEventListener('click' , () =>{

            console.log('val 2');

    });

    //stop eventListener make function as variable

    const fun = () =>{

    console.log('val');

    }

    val.removeEventListener('click' , fun );

### ****Common Event Types****

Here’s a quick overview of some frequently used events:

#### 1. **Mouse Events**

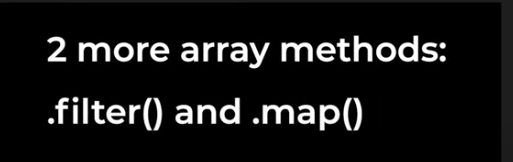
* click: Fired when the user clicks an element.
* dblclick: Fired when the user double-clicks.
* mouseover: Fired when the mouse pointer enters an element.
* mouseout: Fired when the mouse pointer leaves an element.

#### 2. **Keyboard Events**

* keydown: Fired when a key is pressed.
* keyup: Fired when a key is released.
* keypress (deprecated): Use keydown or keyup.

#### 3. **Form Events**

* submit: Fired when a form is submitted.
* change: Fired when an input value changes.
* input: Fired when the user types in an input field.



***First array method is .forEach***

**.filter()**

The .filter() method in JavaScript is used to create a **new array** containing elements from the original array that pass a specified condition. It doesn’t modify the original array but instead returns a new array with the filtered elements.

const array1 = [1, -20 ,30].filter((value,index)=>{

    if(value > 0){

        return true

    }else{

        return false;

    }

});

console.log(array1);

also we can write like below

const array1 = [1, -20 ,30].filter((value,index)=>{

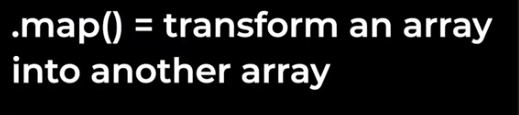
    return value > 0;

});

console.log(array1);

**.map ()**





const Maparray = [1,2,3,8].map((value , index )=>{

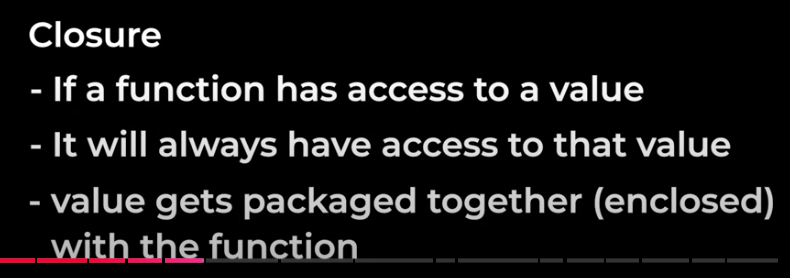
    return value + 10 ;

});

console.log(Maparray);

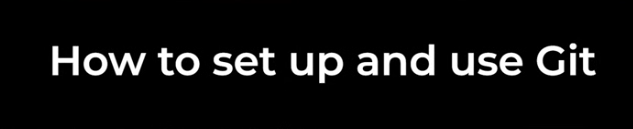
//output

Array(4) [ 11, 12, 13, 18 ]





Using GIT we can easily undo changes in our code.



Install GIT