STRINGS

String In JavaScript:

A string is a piece of text.

It can be a word, a sentence, a single letter, or even empty—anything made of characters.

In JavaScript, we write a string by putting text inside quotes:

```
"School4U"
'JavaScript'
"123"
""
```

You can use double quotes (" "), single quotes (' '), or backticks ( ).

```
let greeting = "Hello, world!";
let name = 'Ali';
let empty = "";
let sentence = `My name is ${name}`;
```

Placeholders

Template Literals

A template literal is a special way to write strings in JavaScript using backticks (`).

String literals let you insert variables or expressions directly inside the string, which is called string interpolation.

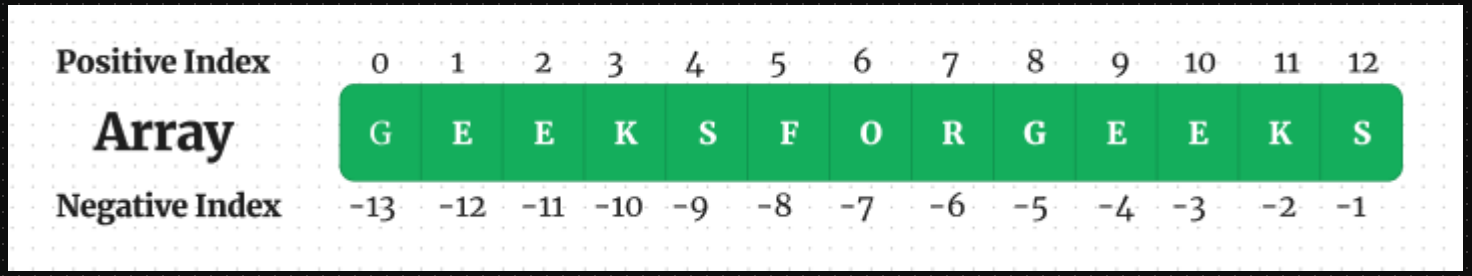
Create string using string constructor:

```
let str = new String("Manas Kumar");
```

Note:

- 🔒 Strings are immutable in JavaScript.

String has index (position)



Escape Sequences

An escape sequence is a special character combo that lets you do things in a string that aren't normally allowed—like adding quotes, new lines, or tabs. (e.g. \n, \t, \\, \", \', etc)

Escape sequences look like 2 characters in code,but they count as 1 character in string length.

String Properties & Methods in JavaScript

Properties:

- str.length – tells how many characters are in a string. (length of string)

Methods:

- str.toUpperCase() – convert each letter into uppercase
- str.toLowerCase() – convert each letter into lowercase
- str.trim() – remove whitespaces
- str.concat(str2) – joins str2 with str1
- str.includes() – checks if string contains given piece of string
- str.indexOf() – gives the position (index) of the first match
- str.charAt() – gives the character at given index
- str.replace(old, new) – replaces first matched part of the string with something else
- str.replaceAll(old, new) – replaces all matched parts of the string with something else
- str.slice(start, end) – cuts out a piece of the string
- str.split() – the .split() method is used to break a string into parts and turn it into an array.

1. Create a program to take full name from user and generate a username start with @, followed by their full name and ends with underscore followed by the length of full name.

2. Take a string and a character from the user and:
    a) count how many times that character appears in the string.
    b) Case-Insensitive Version
    c) Find All Occurrence Positions

3. Count the words present in a given string.

ARRAY

Array In JavaScript:

An array is a list that can store multiple values in one place.
Think of it like a row of boxes, where each box can hold one item (like a number, word, etc.).
You can use it to group similar things together.

Create array using array Literal method (recommended):

```
let colors = ["red", "blue", "green"];

let marks = [23, 94, 55, 26, 84, 89];

let personDetails = ["Muskan", 18, "Bhagalpur"];
```

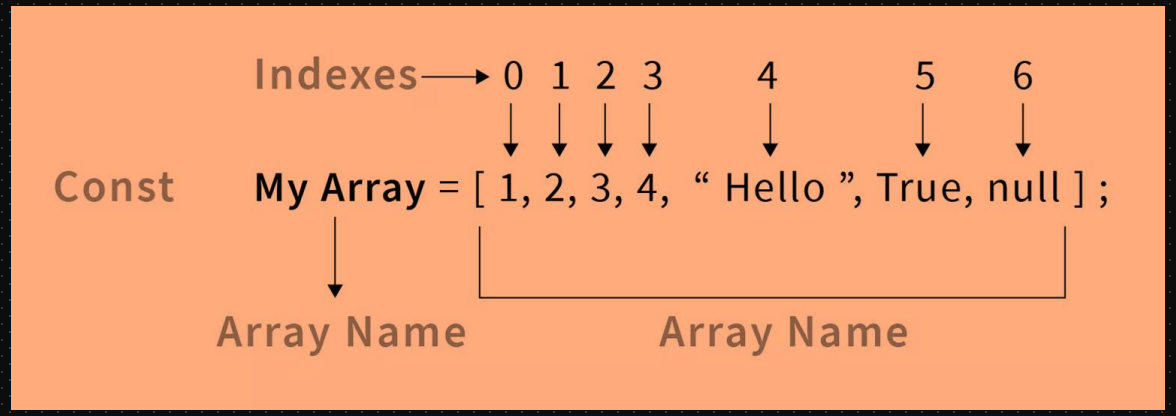
Create array using array constructor:

```
let arr = new Array("spiderman", "ironman", "thor");
```

Note:

- ❑ Array constructor is mostly used when we want to create an empty array with the given length

Indexing in Array



Arrays are mutable

change

Original

Note:

- typeof array is not "Array", it's an "Object".
- Array is a special type of object
- Arrays are mutable.

Array Methods

Method	Changes the Original Array?	What It Returns	What It Does
push()	Yes	New length of the array	Add item at the end
pop()	Yes	The removed last item	Delete item from end
unshift()	Yes	New length of the array	Add item to start
shift()	Yes	The removed first item	Delete item from start
splice()	Yes	An array of removed items	Add remove or doing both at the same time
slice()	No	A new array (sliced portion)	Returns a piece of array
concat()	No	A new array (combined arrays)	Joins multiple arrays (we can also use spread operator instead)
join()	No	A string made by joining elements	join the elements of array on the basis of some string or character
toString()	No	A string of array elements	Converts array to string
includes()	No	true or false	Check whether given item is present in array or not
indexOf()	No	Index of the item or -1 if not found	returns the index of given item if it is present in array, otherwise returns -1
reverse()	Yes	The reversed array	Reverse the order of items
sort()	Yes	The sorted array	Sort the array
find()	No	The first matching item (or undefined)	Returns the first element in the array that satisfies a condition
flat()	No	A new array with nested arrays flattened	returns a new array with nested arrays flattened

Sort method

```
arr.sort(compareFunction)
```

It works on Tim sort under the hood. (combination of merge sort and insertion sort)

By default, sort() converts elements to strings and sorts them alphabetically (Unicode order) or Lexicographical order.

```
const nums = [10, 5, 20];
nums.sort();  // ❌ [10, 20, 5] — because it sorts as ["10", "20", "5"]
```

So, numbers don’t always sort correctly without a compare function.

Return Value of the function	Meaning	Effect on Order
🔻 **Negative** (< 0)	a should come **before** b	✅ Keep order as-is
🟰 **Zero** (=== 0)	a and b are **equal**	🤝 Keep their order (stable)
🔺 **Positive** (> 0)	a should come **after** b	🔁 Swap a and b

Array iteration methods or Functional array methods

Method	Returns	Modifies Original Array?
map()	New array	No
filter()	New array	No
reduce()	Single value	No
forEach()	undefined	No (unless you do it manually)

So if you just want to loop through an array and do something, forEach() is great. But if you want to create a new array or value, use map(), filter(), or reduce().

MDN Web Docs

- **JavaScript** array-copy operations **create** shallow copies. (All standard built-in copy operations with *any* JavaScript objects create shallow copies, rather than deep copies).

A **shallow copy** of an object is a copy whose properties share the same references (point to the same underlying values) as those of the source object from which the copy was made. As a result, when you change either the source or the copy, you may also cause the other object to change too. That behavior contrasts with the behavior of a deep copy, in which the source and copy are completely independent.

String v/s Array

Feature	String	Array
Purpose	Stores **text** (a sequence of characters)	Stores **multiple values** (any data type)
Storage	Text only (characters)	Can be numbers, strings, booleans, etc.
Example	"hello"	["h", "e", "l", "l", "o"]
Indexing	Yes (e.g. "hello"[0] is 'h')	Yes (e.g. ["red", "blue"][0] is "red")
Mutable?	**No** (characters can't be changed directly)	**Yes** (you can change items easily)
Length Property	Yes: "hello".length → 5	Yes: ["a", "b"].length → 2
Common Use	Names, messages, text	Lists of things (colors, numbers, etc.)
Datatype	String	Object
Iterable	**Yes**	**Yes**

1. For an array with marks of students find the average marks of the entire class.

2. Create an array with the given length(n) and fill with 0.

3. Create an array with the given length (n) and store natural numbers from 1 to n.

4. Consider an array of mcu heroes ([ironman, captain, black widow, wanda, hulk, black panther]).
   Now:

   a) Add spiderman at the end and thor at the start.

   b) Remove black widow and add hawkeye in its place.

   c) Check whether captain is present in the array.

5. How to check if given thing is array or not? How to convert other datatypes to array? What if we try to convert an object into an array?

6. We have three variables a, b, c, a contains any number, b contains any string, c contains any object, and d contains any array. Can we create an array from all these four variables? If yes, How?

7. Check whether given string is palindrome or not.

8. Capitalize the first letter of every word in a sentence.

OBJECT

Object In JavaScript:

An object is a collection of key-value pairs or stores related information as a set of key-value pairs. It's a way to group data and functions together.

Example: Think of a real-life object “car”:

Create an object using literal syntax:

```
let car = {
    brand: "Toyota",
    color: "red",
    speed: 120,
    drive: function () {
        console.log("The car is driving");
    }
};
```

Properties (brand, color, speed)

Behaviors / Methods (drive, stop)

Create an object using the Object constructor:

```
let person = new Object();
person.name = "Alice";
person.age = 25;
```

Note:

- ❑ Objects are mutable.

Accessing Object Properties

You can access properties in two ways:

1. Dot Notation (Most Common)

```
console.log(person.name);
```

2. Bracket Notation (Useful with variables or special characters or strings with white spaces)

```
console.log(person["full name"]);
console.log(obj["*"]);
console.log(obj.variableName);
```

Updating or Adding New Properties

Update if already existing property and add if not exist.

```
person.city = "New York";
person["hobby"] = "Reading";
```

Deleting Properties

```
delete person.isStudent;
```

this Keyword in object

In case of object, this refers to the object itself.

```
let obj = {
    name: "Manas Kumar Lal",
    greet : function(){
        console.log(this.name)
    }
}
obj.greet() // Manas Kumar Lal
```

Object Methods

Method	Changes Original Object?	What It Returns	What It Does
Object.keys(obj)	❌ No	Array of keys	Returns an array of all **enumerable property names (keys)**
Object.values(obj)	❌ No	Array of values	Returns an array of all **enumerable property values**
Object.entries(obj)	❌ No	Array of [key, value]	Returns an array of **[key, value]** pairs
Object.assign(target, source)	✅ Yes (if target is modified)	Modified target object	Copies properties from source to target and returns the updated object
Object.freeze(obj)	✅ Yes (locks the object)	The frozen object	Prevents any changes (no adding, removing, or modifying properties)
Object.seal(obj)	✅ Yes (locks structure)	The sealed object	Prevents adding/removing properties, but allows modifying existing ones
ObjName.hasOwnProperty(key)	❌ No	Boolean (true / false)	Checks if the object has the specified property directly

What is singleton object in js?

A singleton is just a single, unique object created once in your code.

Creating an object using literal syntax is considered a singleton because it creates one specific object instance that is not used as a template to create others.

Feature	Singleton Object	Class-Based Object
Syntax	Object literal {}	class or function
Purpose	One specific object	Blueprint for many objects
Instances	Only one	Many objects can be created
Example Use Case	Settings, config, logger	Users, cars, products
Example Code	let config = {}	class User { ... }
Feature	Singleton Object	Class-Based Object

Destructuring

```
const person = {
    name: 'Alice',
    age: 25,
    city: 'New York'
};

const { name, age } = perso
n;
console.log(name); // "Alic
console.log(age);  // 25
```

```
const colors = ['red', 'green', 'blu
e'];
const [first, second, third] = colors;

console.log(first);  // "red"
console.log(second); // "green"
console.log(third);  // "blue"
```

Object v/s Array

Feature	Object	Array
Purpose	Store data as **key-value pairs**	Store **ordered list** of values
Syntax	{ key: value }	[value1, value2, value3]
Access by	Keys (names)	Index (number, starts from 0)
Example	{ name: 'Alice', age: 25 }	['Alice', 25]
Ordering	Not guaranteed	Preserves order
Best Use Case	Describing **properties of a thing**	Working with a **list of items**
Type Check	typeof obj === 'object' && !Array.isArray(obj)	Array.isArray(arr)
Iterable	❌ No	✅ Yes
Iteration	for...in, Object.entries()	for...of, forEach(), map()
Can be nested?	Yes (objects inside objects)	Yes (arrays or objects inside arrays)
Common methods	Object.keys(), Object.values()	push(), pop(), map(), filter()

1. Create a person object with properties: name, age, and city. Then
   a) Log each property as: value of “name” property is “manas” using loop.
   b) Add a new property called email to the person object.
   c) Delete “city” property from person object.

2. Create a function that takes an object with firstName, middleName, lastName properties and returns the full name.

3. Write a function that takes object and returns the number of properties in an object.

4. Write a function that returns an array of names of users who have the role “admin”.

```
const users = [
    { name: "Alice", role: "admin" },
    { name: "Bob", role: "user" },
    { name: "Charlie", role: "admin" }
];
```

5. Write a function searchProducts(products, keyword) that returns an array of products whose name includes the given keyword (case-insensitive).

```
const products = [
    { id: 1, name: "iPhone 14" },
    { id: 2, name: "Samsung Galaxy" },
    { id: 3, name: "Google Pixel" }
];
```

6. Write a function groupByPost(comments) that returns an object grouping comments by postId:

```
const comments = [
    { postId: 1, text: "Great post!" },
    { postId: 2, text: "Thanks!" },
    { postId: 1, text: "Very helpful" }
];
```

```
{
    1 : ["Great post!", "Very helpful"],
    2 : ["Thanks!"]
}
```

7. Write a function buildQuery(params) that returns

```
const params = { search: "laptop", page: 2, sort: "price" };

// Output -> "search=laptop&page=2&sort=price"
```