Array and String

Data types inJS

What types of values can be assigned to variables?

MDN- primitive

Primitives (value types)

string

number

boolean

undefined

null

bigInt

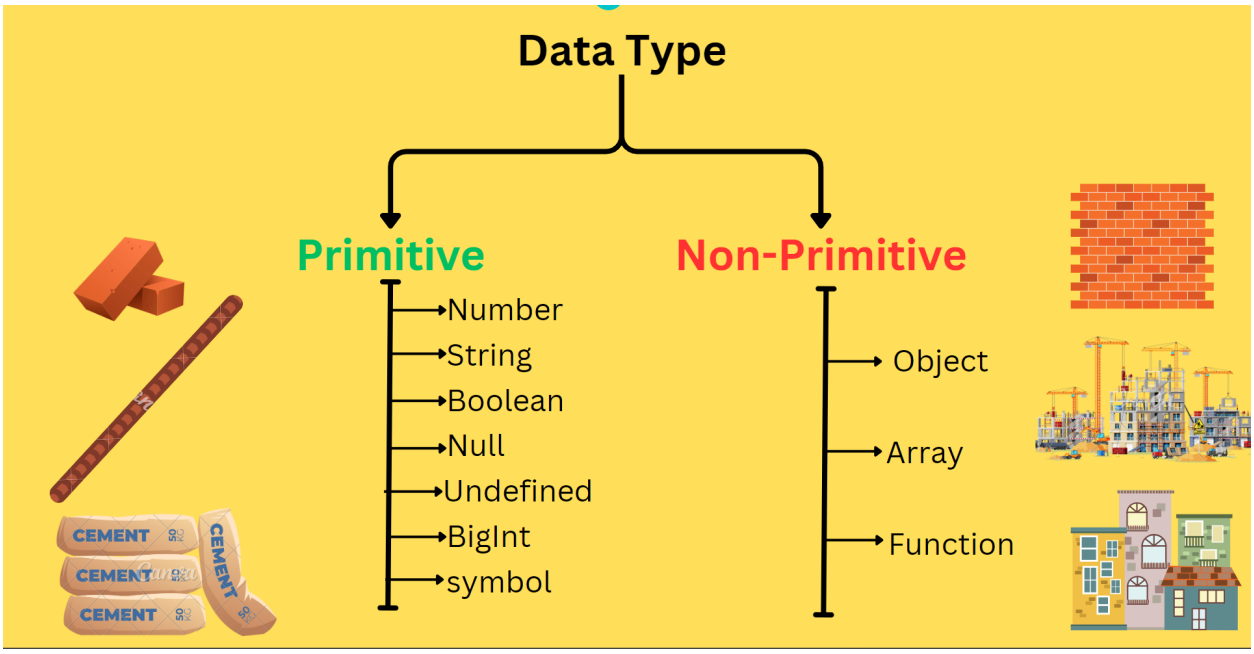
symbol

Non Primitives (reference types)

object

array

function



▼ Strings

The first type of data is a String. This is used to store a sequence of characters
used to represent text.

Example:

```
var name = "Masai School";  //using double quotes
var subject = 'Coding';     // using single quotes
var val =`Hi`;              //  using template literal
```

Any data within
" " (Double quotes) or ' ' (Single quotes) or ` ` (Template literal) is a String in JavaScript.

▼ Numbers

The second type of data we want to know is a Number, which is used to store any kind
of numbers. We have already seen this type of data in the variables example. Numbers
can store both Whole Numbers/Integers and Decimals.

Example:

```
var num = 100
var dec = 100.001
```

▼ Booleans

This data type has only two values true and false.

Example:

```
var x = true
var y = false
```

Checking the type of data:

Let's say you have some data but you don't know what type it is. You can use the
typeof() inbuilt code to find the type of the data.

Example:

```
var name = "Masai School"
console.log(typeof(name)) //output => string
```

▼ undefined

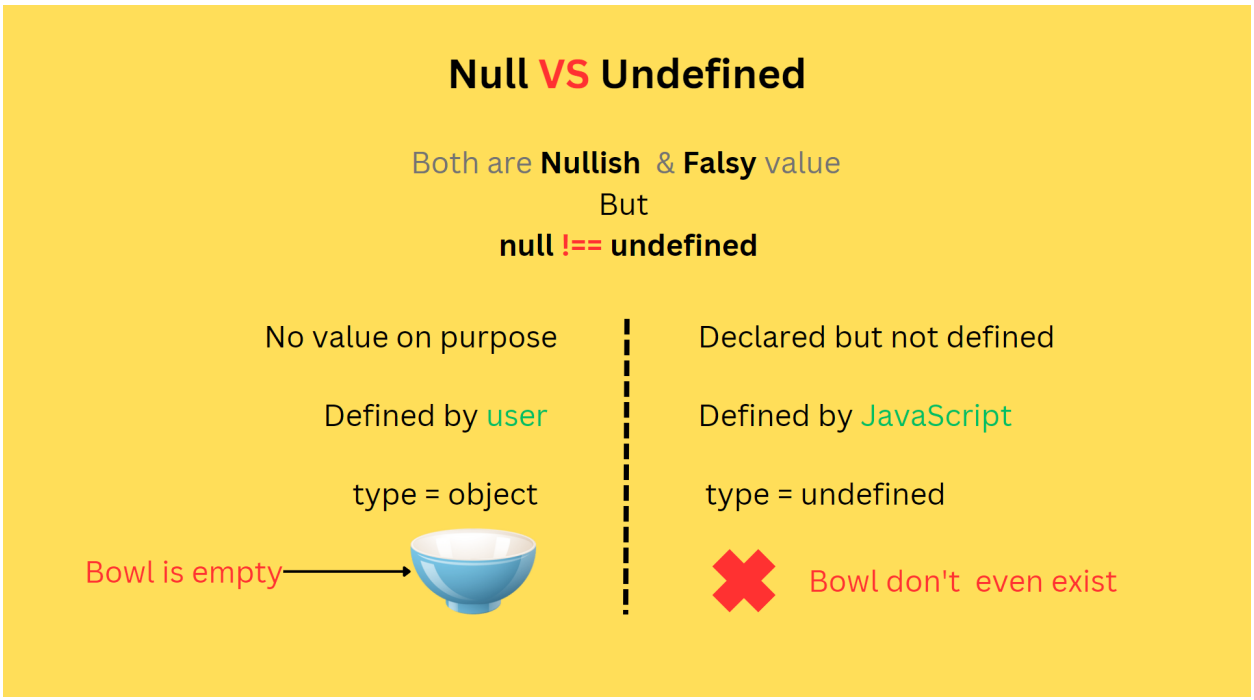
undefined is js is a falsey value. A variable that has been declared but has not been assigned a value is said to have an `"undefined"` value. It essentially means the variable exists but lacks a meaningful value.

```
var x; // Declares the variable x but leaves it undefined
console.log(x); // This will output "undefined"
```

▼ null

Null represents the intentional absence of any object value. It is often used to explicitly indicate that a variable should have no value or that an object property is empty.

```
var y = null; // Assigns the value null to the variable y
console.log(y); // This will output "null"
```



Array

Introduction

How do we store information?

We use variables to store data, such as the name of a student. However, when dealing with multiple values, using individual variables for each is not efficient.

For example:

```
var name1 = "Prateek";
var name2 = "Nrupul";
```

```
// ...and so on
```

To address this, we use **arrays**.

Arrays Basics

What is an Array?

An array is a contiguous chunk of memory that can store multiple values. It allows us to store and manage collections of data in a single variable.

Declaration of an array

```
var arr = [];
```

Storing Names in an Array

```
var namesArray = ["Prateek", "Nrupul", "Yogesh", "Aman", "Albert"];
```

Each value in the array is stored at an index, starting from 0.

Accessing Array Elements

```
console.log(namesArray[0]); // Output: Prateek
console.log(namesArray[1]); // Output: Nrupul
// ...and so on
```

Problem Solving with Arrays

Array of Vegetables

```
var vegetables = ["Tomato", "Beans", "Onion"];
console.log(vegetables[0]); // Output: Tomato
console.log(vegetables[1]); // Output: Beans
console.log(vegetables[2]); // Output: Onion
```

Finding Array Length

Use the `length` property to determine how many elements are present in an array.

Find the length of the vegetables array.

```
var vegetables = ["Tomato", "Beans", "Onion"];
console.log(vegetables.length); // Output: 3
```

Print the price of the last product.

```
var prices = [45, 71, 29];
console.log(prices[prices.length - 1]); // Output: 29
```

Modifying Arrays

Adding Elements with `push()`

```
var movies = [];
```

```
movies.push("Bahuballi");
movies.push("Avengers");
movies.push("Spider Man");
```

Problem Solving with `push()`

Push 4 superheroes into the array

```
var superheroes = [];
superheroes.push("batman");
superheroes.push("superman");
superheroes.push("ironman");
console.log(superheroes);
```

Updating Array Elements

Updating the first index value in an array:

```
superheroes[0] = "Thor";
```

Looping Through Arrays

Print all elements of the array using a loop.

```
var movies = ["batman", "superman", "ironman"];
for (var i = 0; i < movies.length; i++) {
  console.log(movies[i]);
```

```
}
```

Print movies with actors

```
var movies = ["bahuballi", "Spider Man", "Iron Man", "Super M
an"];
var actors = ["Prabhas", "Tom Holland", "Robert Downey", "Hen
ry Cavill"];
for (var i = 0; i < movies.length; i++) {
  console.log(movies[i], " - ", actors[i]);
}
```

Removing Elements

Using `pop()` to Remove Elements

Pop the last 2 elements from an array

```
var movies = ["batman", "superman", "ironman"];
movies.pop();
movies.pop();
console.log(movies);
```

Delete last 3 numbers from an array

```
var numbers = [2, 3, 4, 5, 6, 7];
numbers.pop();
numbers.pop();
```

```
numbers.pop();
console.log(numbers);
```

or

```
var numbers = [2, 3, 4, 5, 6, 7];
for (var i = 1; i <= 3; i++) {
  numbers.pop();
}
console.log(numbers);
```

Reversing an Array

To reverse the order of elements in an array, you can create a new array and iterate through the original array in reverse order.

```
var fruits = ["apple", "banana", "orange", "grape"];
var reversedFruits = [];

for (var i = fruits.length - 1; i >= 0; i--) {
  reversedFruits.push(fruits[i]);
}

console.log(reversedFruits); // Output: ["grape", "orange",
"banana", "apple"]
```

Counting Even and Odd Elements

You can iterate through an array and count the number of even and odd elements without using inbuilt functions.

```
var numbers = [2, 7, 6, 9, 14, 5];
var evenCount = 0;
var oddCount = 0;

for (var i = 0; i < numbers.length; i++) {
  if (numbers[i] % 2 === 0) {
    evenCount++;
  } else {
    oddCount++;
  }
}

console.log("Even Count:", evenCount); // Output: 3
console.log("Odd Count:", oddCount);   // Output: 3
```

Finding Maximum and Minimum Values

To find the maximum and minimum values in an array without using inbuilt functions:

```
var scores = [87, 92, 78, 94, 89];
var maxScore = scores[0];
var minScore = scores[0];

for (var i = 1; i < scores.length; i++) {
  if (scores[i] > maxScore) {
    maxScore = scores[i];
  }

  if (scores[i] < minScore) {
    minScore = scores[i];
  }
```

```
}

console.log("Maximum Score:", maxScore); // Output: 94
console.log("Minimum Score:", minScore); // Output: 78
```

Arrays with Loop and Break

Code 12: Print the first 3 items in the array using a loop.

```
First Way
var movies = ["bahuballi", "SpiderMan", "IronMan", "SuperMan"];
for(var i = 1; i<=3; i++)
{
console.log(movies[i]);
}
```

```
Second Way [ Using Break ]
var movies = ["bahuballi", "SpiderMan", "IronMan", "SuperMan"];
for(var i = 0; i<movies.length; i++)
{
if(i==3)
{
break;
}
console.log(movies[i]);
}
```

Arrays with Loop and Continue

Code 12: Print all movies except the third movie.

```
var movies = ["bahuballi", "SpiderMan", "IronMan", "SuperMan"];
for(var i = 0; i<movies.length; i++)
{
if(i==2)
{
continue;
}
console.log(movies[i]);
}
```

Code 13: Print all movies except the third and fifth movies.

```
var movies = ["bahuballi", "SpiderMan", "IronMan", "SuperMan","I
for(var i = 0; i<movies.length; i++)
{
if(i==2 || i==4)
{
continue;
}
console.log(movies[i]);
}
```

Array methods

JavaScript

```
array.splice(0,1)
//1 - index of first element
//2 - no. of. elements to be removed

let fruits = ['apple', 'banana', 'orange', 'grape'];

let arr=fruits.splice(0,1)
console.log(arr,fruits)
//[ 'apple' ] [ 'banana', 'orange', 'grape' ]


let arr=fruits.splice(0)
console.log(arr,fruits)
//[ 'apple', 'banana', 'orange', 'grape' ] []
```

Removing elements from the first and adding them to the last

```
JavaScript

let fruits = ['apple', 'banana', 'orange', 'grape'];

let arr=fruits.splice(0,2)

console.log(arr,fruits)

//Creating new array
let newArray=fruits.concat(arr)
console.log(newArray)



JavaScript
```

```
let arr=fruits.splice(-1,1)
console.log(arr) //[ 'apple', 'banana', 'orange']
```

String

Introduction

What are Strings?

In JavaScript, a string is a sequence of characters. Strings are used to represent text and are enclosed in either single (`''`) or double (`""`) quotes.

```
var message = "Hello, World!";
var name = 'John Doe';
```

String Operations

Concatenation

Concatenation is the process of combining strings.

```
var firstName = "John";
var lastName = "Doe";
var fullName = firstName + " " + lastName;
```

```
console.log(fullName); // Output: John Doe
```

String Length

To find the length of a string, use the `length` property.

```
var greeting = "Hello, World!";
console.log(greeting.length); // Output: 13
```

Accessing Characters

You can access individual characters in a string using their index.

```
var word = "JavaScript";
console.log(word[0]); // Output: J
console.log(word[4]); // Output: S
```

Immutable Nature of Strings

Strings in JavaScript are immutable, meaning their values cannot be changed after they are created. Any operation that seems to modify a string actually creates a new string.

Example

```
var text = "Hello";
text[0] = "C"; // This will not modify the string.
console.log(text); // Output: Hello
```

Mutable Operations

String Concatenation

While strings are immutable, you can achieve a form of mutation through concatenation.

```
var message = "Hello";
message = message + ", World!";
console.log(message); // Output: Hello, World!
```

String Methods

JavaScript provides various string methods that create new strings based on the original string.

Example: Uppercase and Lowercase

```
var word = "JavaScript";
var upperCaseWord = word.toUpperCase();
var lowerCaseWord = word.toLowerCase();

console.log(upperCaseWord); // Output: JAVASCRIPT
console.log(lowerCaseWord); // Output: javascript
```

String Comparison

Equality Comparison

You can compare strings for equality using the `===` operator.

```
var str1 = "Hello";
var str2 = "Hello";
console.log(str1 === str2); // Output: true
```

Inequality Comparison

```
var str1 = "Hello";
var str2 = "World";
console.log(str1 !== str2); // Output: true
```

Reversing a String

Print the given string in reverse.

```
let reversed = "";
let input1 = "hello";
for (let i = input1.length - 1; i >= 0; i--) {
  reversed += input1[i];
}
console.log(reversed); // Expected Output: olleh

reversed = "";
let input2 = "JavaScript";
for (let i = input2.length - 1; i >= 0; i--) {
  reversed += input2[i];
}
console.log(reversed); // Expected Output: tpircSavaJ
```

Counting Vowels

Count the vowels in given string.

```
let count = 0;
let vowels = "aeiou";
let input1 = "programming";
for (let i = 0; i < input1.length; i++) {
  if (input1[i] === "a" || input1[i] === "e" || input1[i] ===
"i" || input1[i] === "o" || input1[i] === "u") {
    count++;
  }
}
console.log(count); // Expected Output: 2
```

Palindrome Check

Check palindrome in given string.

```
let input1 = "level";
let reversed1 = "";
for (let i = input1.length - 1; i >= 0; i--) {
  reversed1 += input1[i];
}
console.log(input1 === reversed1); // Expected Output: true

let input2 = "hello";
let reversed2 = "";
for (let i = input2.length - 1; i >= 0; i--) {
  reversed2 += input2[i];
}
console.log(input2 === reversed2); // Expected Output: false
```

String methods

`charAt(index)`: Returns the character at a specific index in the string.

```
let str = "Hello World!";
console.log(str.charAt(0)); // Output: "H"
```

`charCodeAt(index)`: Returns the Unicode character code at a specific index.

```
let str = "Hello World!";
console.log(str.charCodeAt(0)); // Output: 72 (ASCII code for "H
```

`toLowerCase()`: Converts the string to all lowercase letters.

```
let str = "HeLlO WoRlD!";
console.log(str.toLowerCase()); // Output: "hello world!"
```

`toUpperCase()`: Converts the string to all uppercase letters.

```
let str = "HeLlO WoRlD!";
console.log(str.toUpperCase()); // Output: "HELLO WORLD!"
```

`indexOf(substr, [fromIndex])`: Returns the first index at which a substring can be found.

```
let str = "Hello World, how are you?";
console.log(str.indexOf("World")); // Output: 7
```

`slice(start, end)`: Extracts a section of the string and returns a new string.

```
let str = "Hello World!";
console.log(str.slice(0, 5)); // Output: "Hello"
```

`substring(start, end)`: Similar to `slice`, but allows negative indices for end.

`split(separator)`

```
let str = "Hello,World,how,are,you?";
console.log(str.split(",")); // Output: ["Hello", "World", "how"
```

`concat`

```
let str1 = "Hello";
let str2 = " World!";
console.log(str1.concat(str2)); // Output: "Hello World!"
```

- `trim()`: Removes leading and trailing whitespace characters.

```
let str = "  Hello World!  ";
console.log(str.trim()); // Output: "Hello World!"
```

- `trimStart()`: Removes leading whitespace characters. (introduced in ES2019)

- `trimEnd()`: Removes trailing whitespace characters. (introduced in ES2019)

- `replace(searchValue, replaceValue)`: Replaces all occurrences of a substring with another substring.

```
let str = "Hi Hello Hi";
console.log(str.replace("Hi", "Hey")); // Output: "Hey Hello Hey
```

1. `includes(searchString, position)` - Determines whether the calling string contains the `searchString`.

```
let text = "Learning JavaScript is fun!";
console.log(text.includes("JavaScript")); // true
```

1. `lastIndexOf(searchValue, fromIndex)` - Returns the index of the last occurrence of the specified value. Searches backward, starting at `fromIndex`.

```
let text = "Hello, World!";
console.log(text.lastIndexOf("o")); // 8
```

2. `repeat(count)` - Returns a new string consisting of the elements of the object repeated the given times.

```
let text = "abc";
console.log(text.repeat(3); // "abcabcabc"
```

3. `padStart(targetLength, padString)` - Pads the current string from the start with another string until the resulting string reaches the given length.

```
let text = "5";
console.log(text.padStart(3, "0")); // "005"
```

`padEnd(targetLength, padString)` - Pads the current string from the end with another string until the resulting string reaches the given length.

```
let text = "5";
console.log(text.padEnd(3, "0")); // "500"
```