Primitive data types are the most basic types of data in JavaScript. These types are immutable, meaning their value cannot be changed once created. JavaScript has six primitive data types:

- **Number**
- **String**
- **Boolean**
- **Null**
- **Undefined**
- **Symbol**
- **bigInt**

String:

A string is a data type in JavaScript that represents text or a sequence of characters. Strings are used to store and manipulate textual data, such as words, sentences, or symbols. In JavaScript, strings are enclosed in either single quotes (') or double quotes ("), and they can contain letters, numbers, symbols, and whitespace characters.

Here are some key points about strings in JavaScript:

1. **Declaration**:
   - Strings are declared by enclosing text in quotes. Example:

```
// Declaration
var greeting = 'Hello, World!';
var message = "JavaScript is awesome!";
```

2. **Concatenation**:
   - Strings can be concatenated (joined together) using the + operator. Example:

```
// 2.  Concatenation
var firstName = 'Mahesh';
var lastName = 'H';
var fullName = firstName + ' ' + lastName; // fullName will be 'Mahesh H'
```

3. **Escape Characters**:
   - JavaScript allows the use of escape characters to include special characters in strings. Common escape characters include:
     - **\n**: Newline
     - **\t**: Tab
     - Tab (\t)
     - Vertical Tab (\v)
     - Carriage Return (\r)

4. **String Length**:
    - The length of a string, i.e., the number of characters it contains, can be determined using the **length** property. Example:

```
// String Length
var text = 'Hello';
var length = text.length; // length will be 5
```

5. **Accessing Characters**:
    - Individual characters in a string can be accessed using bracket notation (**[]**) with the character's index. Indexing starts at 0. Example:

```
// Accessing Characters
var word = 'JavaScript';
var firstCharacter = word[0]; // firstCharacter will be 'J'
```

6. **String Methods**:
    - JavaScript provides many built-in methods for working with strings, such as:
        - **toUpperCase()**, **toLowerCase()**: Convert string to uppercase or lowercase.
        - **split()**, **join()**: Split and join strings based on delimiters.

Number Data Type:

The number data type in JavaScript represents numeric values, including integers, floating-point numbers, and special numeric values like **Infinity**, **-Infinity**, and **NaN** (Not-a-Number). Here are the key points about the number data type in JavaScript:

1. **Integer and Floating-Point Numbers**:
    - Integers: Whole numbers without a decimal point, such as **-10**, **0**, **42**.
    - Floating-Point Numbers: Numbers with a decimal point or in exponential notation, such as **3.14**, **-0.5**, **2.5e3** (equivalent to **2500**).
    - JavaScript represents all numbers as floating-point numbers internally, including integers.
2. **Special Numeric Values**:
    - **Infinity**: Represents positive infinity, typically resulting from arithmetic operations like dividing a number by zero (**1 / 0**).
    - **-Infinity**: Represents negative infinity, often resulting from operations like dividing a negative number by zero (**-1 / 0**).
    - **NaN** (Not-a-Number): Represents an invalid or undefined numeric value, resulting from operations that cannot produce a meaningful result, such as dividing zero by zero (**0 / 0**) or attempting arithmetic operations on non-numeric values.

```
var a = 10; // integer
var b = 3.14; // floating-point
var sum = a + b; // sum will be 13.14
var product = a * b; // product will be 31.4

var c = 2.5e3
```

Type Coercion:

- JavaScript performs automatic type coercion during operations involving different data types, such as converting strings to numbers or vice versa.

```
var num = 10 + '5'; // num will be '105' (string concatenation)
```

Number Methods:

- JavaScript provides built-in methods for working with numbers, such as **parseInt()**, **parseFloat()**, **toFixed()**, **toPrecision()**, **isNaN()**, **isFinite()**, etc.

```
var num1 = parseInt('10'); // num1 will be 10
var num2 = parseFloat('3.14'); // num will be 3.14
var num3 = 3.14159;
var rounded = num3.toFixed(3); // rounded will be '3.14'
console.log(rounded)
```

Math Object:

- JavaScript's **Math** object provides additional mathematical functions and constants, such as **Math.sqrt()**, **Math.abs()**, **Math.max()**, **Math.min()**, etc.

```
var squareRoot = Math.sqrt(25); // squareRoot will be 5
var absoluteValue = Math.abs(-10); // absoluteValue will be 10
var largestNumber = Math.max(10, 20, 5, 30); // largestNumber will be 30
var smallestNumber = Math.min(10, 20, 5, 30); // smallestNumber will be 5
```

Boolean:

A Boolean is a data type in JavaScript that represents a logical value indicating either true or false. Booleans are commonly used for conditions and control flow in programming to make decisions based on whether a condition is true or false. Here are the key points about Booleans in JavaScript

```
// Using Boolean literals directly
var isSunny = true; // true represents a true Boolean value
var isRainy = false; // false represents a false Boolean value

// Using Boolean function to convert other values to Booleans
var num = 10;
var isNumTruthy = Boolean(num); // Convert num to a Boolean value (true for non-zero numbers)
var isZeroTruthy = Boolean(0); // Convert 0 to a Boolean value (false for zero)

// Displaying the results
console.log(isSunny); // Output: true
```

```
console.log(isRainy); // Output: false
console.log(isNumTruthy); // Output: true
console.log(isZeroTruthy); // Output: false
```

undefined:

- The **undefined** data type represents a variable that has been declared but not assigned a value or a property that does not exist in an object.

```
var x;
console.log(x); // Output: undefined
```

null:

- The **null** data type represents the intentional absence of any value or an empty value. It is often used to explicitly indicate that a variable or object property has no value

```
var car = null; // No value assigned to car
```

Interview Questions

What are the primitive data types in JavaScript?

- **Answer**: The primitive data types in JavaScript are Number, String, Boolean, Null, Undefined, Symbol, and BigInt.

What is the difference between null and undefined?

- **Answer**: null is an assignment value that represents the intentional absence of any object value. undefined means a variable has been declared but has not yet been assigned a value.

Describe how type coercion works with primitive data types in JavaScript.

- **Answer**: Type coercion in JavaScript is the automatic or implicit conversion of values from one data type to another. This can happen during operations involving different types, such as adding a number to a string (1 + "1" results in "11") .

How does JavaScript handle arithmetic operations involving NaN?

- **Answer**: NaN stands for "Not-a-Number" and is a result of invalid or undefined mathematical operations. Any arithmetic operation involving NaN will result in NaN. For example, 1 + NaN results in NaN, and NaN * 2 also results in NaN.

Explain the significance of the toString method for primitive data types.

- **Answer**: The toString method converts a primitive value to its string representation. This method is available for most primitive types like Number, Boolean, and Symbol. For example, (123).toString() returns "123", and true.toString() returns "true".

Why Use void 0?

- In JavaScript, the undefined value can technically be reassigned, although this is generally considered bad practice. Using void 0 ensures that you always get undefined, even if undefined has been reassigned.

What is the difference between the global undefined property and void 0?

- **Answer**: The global undefined property is the value undefined. The expression void 0 also evaluates to undefined, but it is used to ensure that the value undefined has not been overwritten. Using void 0 guarantees you get undefined even if someone has reassigned the undefined variable.

What is NaN and how can you check if a value is NaN?

- **Answer:** NaN stands for "Not-a-Number". It is a special value in JavaScript that represents a computational error resulting from an operation that doesn't produce a valid number. You can check if a value is NaN using Number.isNaN(value).

How can you convert a string to a number in JavaScript?

- **Answer:** You can use the Number() function, parseInt(), parseFloat(), or the unary + operator. For example, Number("42"), parseInt("42"), parseFloat("42.5"), and +"42" all convert the string "42" to the number 42.

Basic Task:

1. **String Task**: Create a variable **favoriteFood** and assign it a string value representing your favorite food. Then, log this string to the console.
2. **Number Task**: Declare a variable **myAge** and assign it your age as a number. Then, log this number to the console.
3. **Boolean Task**: Create a variable **isStudent** and assign it a boolean value that represents whether you are currently a student (true) or not (false). Log this boolean value to the console.
4. **Undefined Task**: Declare a variable **myPet** without assigning any value to it. Log this variable to the console to see the output as undefined.
5. **Null Task**: Create a variable **carBrand** and assign it a null value to represent that you don't currently own a car. Log this variable to the console.