Lesson:

Values and Datatypes

PW SKILLS

Topics Covered

1. Introduction to Data Types.
2. Datatypes in Javascript.

Introduction to Data Types.

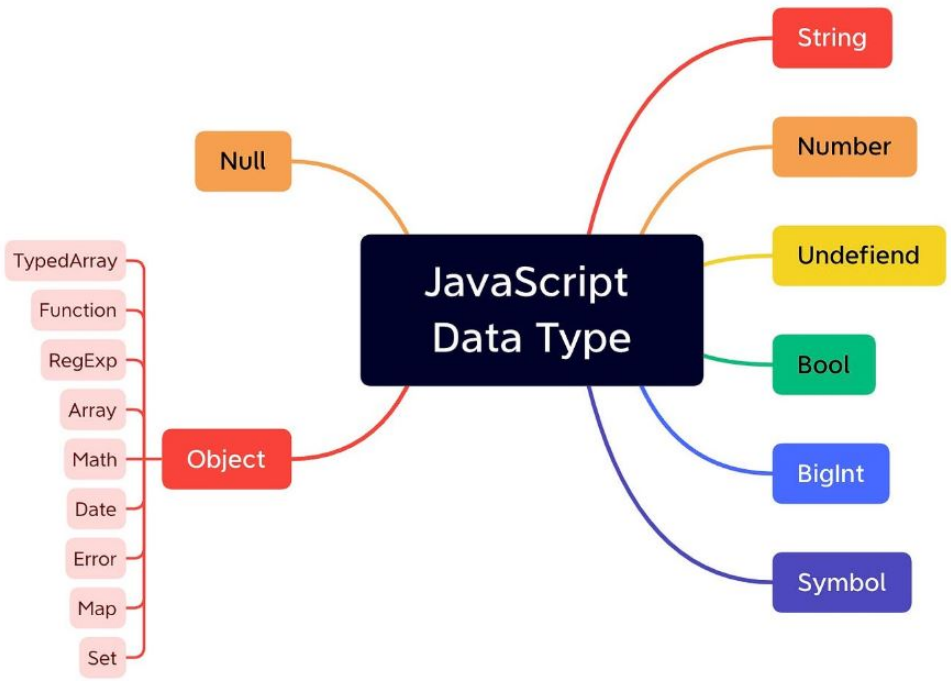
Data types are used to define the way the data is stored in memory. Storing data is an essential part of programming as it enables the manipulation, processing, and sharing of information within a program.

The data type is a classification of data according to the type of value that we want to operate on.

JavaScript is a dynamically typed language, which means the data type is identified during execution. The programmer need not explicitly declare the data type in code.

Following are the data types of JavaScript:

1. String - Primitive
2. Number - Primitive
3. Bigint - Primitive
4. Boolean - Primitive
5. Undefined - Primitive
6. Null - Primitive
7. Symbol - Primitive
8. Object - **Non Primitive**
9. Array - Non Primitive



Datatypes in Javascript

1. String: Strings are a data type used for representing text. A string is a sequence of characters, enclosed in single or double quotes.

```
"I am learning JavaScript"
‘I am happy to learn from PW Skills’
```

2. Number: Numbers are a data type used for representing numeric values. Numbers can be an integer, whole numbers, or decimal values [floating point values].

```
1
2
90
102.5
```

Other possible number values are **infinity** and **NaN.**

```
NaN
Infinity
```

Infinity is a special value that is greater than any number.

NaN stands for "Not a Number" and is a special value that represents the result of an undefined or unrepresentable mathematical operation.

3. Bigint: In JavaScript, there is a maximum safe value, which is approximately 2**^53 - 1**. Similarly, there is also a minimum safe value, which is approximate **-(2^53 - 1)**.

This means that integers less than min safe value or greater than max safe value, may lose precision when represented as a JavaScript number. So, for such numbers, we use **BigInt** data type.

The BigInt data type number can also be treated as a regular number by adding **n** to it at the end.

```
902345874n
```

4. Boolean: Boolean is a logical type that is either true or false.

```
true
false
```

Booleans are often used to represent the outcome of a logical comparison or the result of a logical operation. We will look into boolean operators in further lectures.

5. Undefined: Undefined is a special value that indicates that a variable or property has been declared but has not been assigned a value.

```
undefined
```

We will look into variables in the next lecture.

6. Null: Null means nothing or empty value. It is often used to indicate that a variable or property has no value.

```
null
```

We will look into variables in the next lecture.

7. Object: In javascript, numbers, strings, booleans, undefined, null are called as primitive data types.

Objects are non-primitive data because they are mutable (can be changed).

Object is a collection of properties, where each property has a key and a value, where

- **keys could be strings or symbols**
- **values could be of any datatype.**

Objects are usually created by curly brackets { } called object literal syntax.

```
{
  name: "Mithun",
  company: "PW Skills"
}
```

We will be looking at objects in depth in further lectures.

8. Symbol: A Symbol is a data type that can be used as an object property key. Symbols as object keys, avoids potential conflicts with keys that other code may add to the object, and every Symbol() call is guaranteed to return a unique Symbol.

```
Symbol("name")
```

9. Array - Non Primitive: Array is a data type containing data in sequential order. An array can contain multiple values with multiple data types also.
Example:-

```
[1, 2, 3, 4, 5]
[“prabir”, 23, “developer”]
```

Interview Point.

1. **Can you explain the difference between primitive and non-primitive data types in JavaScript?**

Ans: The key difference between primitive and non-primitive types lies in how they are stored and accessed:

- Primitive types: Stored directly in the variable's location in memory. When you assign a primitive value to another variable, a copy of the value is made.
- Non-primitive types: Stored by reference. When you assign a non-primitive value to another variable, both variables refer to the same object in memory. Changes made through one variable affect the other.

2. **What are template literals, and how do they differ from regular strings in JavaScript?**

Ans: Template literals, introduced in ECMAScript 6 (ES6), provide a more flexible and readable way to work with strings in JavaScript compared to regular strings.

- Primitive types: Stored directly in the variable's location in memory. When you assign a primitive value to another variable, a copy of the value is made.
- Non-primitive types: Stored by reference. When you assign a non-primitive value to another variable, both variables refer to the same object in memory. Changes made through one variable affect the other.

String Interpolation: Template literals allow you to embed expressions inside the string using ${} syntax, making it easier to concatenate variables and expressions.

```
let name = "John";
let greeting = `Hello, ${name}!`;
// Result: Hello, John!
```

Multiline Strings: Template literals support multiline strings without the need for explicit line breaks or concatenation.

```
let multiline = `
This is a
multiline
string.
`;
```

Expression Evaluation: Expressions inside ${} are evaluated, allowing you to include any valid JavaScript expression within the string.

```
let x = 5;
let y = 10;
let result = `The sum of ${x} and ${y} is ${x + y}.`;
// Result: The sum of 5 and 10 is 15.
```

3. Difference between null and undefined?

Ans: In JavaScript, null and undefined are both special values used to indicate the absence of meaningful values, but they are employed in different contexts.

When a variable is declared but not assigned any value, it is automatically assigned the value undefined. This typically occurs when a variable is created but hasn't been given a specific value yet or when a function does not explicitly return a value.

On the other hand, null is a value that must be explicitly assigned. It is often used by developers to signify the intentional absence of an object value, providing a way to express that a variable should have no meaningful content.

While undefined is a primitive data type, null is considered an object. When comparing them, in loose equality (==), null and undefined are equal to each other and to no other value, but in strict equality (===), they are not equal to each other or any other value.