JavaScript Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

Data types that are known as primitive values in JavaScript are numbers, strings, Booleans, null, undefined. Objects such as functions and arrays are referred to as non-primitive values.The fundamental difference between primitives and non-primitives is that primitives are immutable and non-primitives are mutable.

1. Primitive data type
2. Non-primitive (reference) data type

Primitive data type

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents Boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

The undefined type

The `undefined` type is a primitive type that has one special value `undefined`. By default, when a variable is declared but not initialized, it is assigned the value of `undefined`.
Example:

```
let foo;

console.log(foo);       // undefined

console.log(typeof foo); // undefined
```

The number, string, Boolean type

The **string *data type*** is used to represent textual data (i.e. sequences of characters). Strings are created using single or double quotes surrounding one or more characters

The ***number* data type** is used to represent positive or negative numbers with or without decimal place, or numbers written using exponential notation e.g. 1.5e-4

The **Boolean data type** can hold only two values: true or false. It is typically used to store values like yes (true) or no (false), on (true) or off (false)

Example:

```
let foo = 120; // foo is a number
```

```
foo = false;   // foo is now a boolean
```

```
foo = "foo";   // foo is now a string
```

The null type

The null type is the second primitive data type that has only one value: null. Javascript defines that null is an empty object pointer.
example:

```
let obj = null;
console.log(typeof obj); // object
```

Non-Primitive data type

Non-primitive values are mutable data types. The value of an object can be changed after it gets created.

```
var arr = [ 'one', 'two', 'three', 'four', 'five' ];
arr[1] = 'TWO';
console.log(arr) // [ 'one', 'TWO', 'three', 'four', 'five' ];
```

Objects are not compared by value. This means that even if two objects have the same properties and values, they are not strictly equal. Same goes for arrays. Even if they have the same elements that are in the same order, they are not strictly equal.

```
var obj1 = { 'cat': 'playful' };
var obj2 = { 'cat': 'playful' };
obj1 === obj2;
// false var
 arr1 = [ 1, 2, 3, 4, 5 ];
var arr2 = [ 1, 2, 3, 4, 5 ];
arr1 === arr2;  // false
```

Non primitive values can also be referred to as reference types because they are being compared by reference instead of value. Two objects are only strictly equal if they refer to the same underlying object.

```
var obj3 = { 'car' : 'purple' }
var obj4 = obj3;
obj3 === obj4;
 // true
```

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
Function	Represent a block of statement