Data-types in JavaScript:

In JavaScript, a primitive data type is data that is not an object and has no methods or properties. All types except Object define immutable values. These values are primitive values.

- string
- number
- bigint
- boolean
- undefined
- symbol
- null

All primitives are immutable; that is, they cannot be altered. The variable assigned to a primitive value, may be reassigned to a new value, but the existing value can never be changed in the ways that objects, arrays, and functions can be altered.

Primitives have no methods but behave as if they do. When properties are accessed or methods are called on primitives, JS auto-boxes the values into a wrapper object and access the property on that object instead. For examples:

"Ajay".includes("a") implicitly creates a String wrapper object and calls String.prototype.includes method on that object.

* Null and Undefined primitive types have "object" and "undefined" as their return type values, and do not have Object Wrapper.
*

- String:
A string is a sequence of characters used to represent text. In JS, a String is one of the primitive DT or value and the String object is a wrapper around a string primitive.

- number:
In JS, number is a ~~no~~ numeric data type in the double-precision 64-bit floating point format IEEE 754.

- bigint:
In JS, it is a numeric primitive that can represent integers with arbitrary magnitude.

Number.MAX_SAFE_INTEGER = 9007199254740991

Above that value, bigint is used, and is created using wrapper object, BigInt.

- boolean:
It represents a logical entity with two values: true or false.

- null:
It is inhabited by exactly one value: null.

- undefined:
It is inhabited by exactly one value: undefined

Conceptually, undefined indicates the absense of a value, while null indicates the absence of an object (typeof null === "object"). JS usually defaults to undefined when something is devoid of a value:
- A return statement with no value (return;) implicitly returns undefined.
- accessing a non-existent object property (obj.hablaExist) returns undefined.
- a variable declaration without initialisation implicitly initialise the variable to undefined.
- many methods like Array.prototype.find() return undefined when no element is found.

* NaN ("Not a Number") is a special kind of number value that's typically encountered when the result of an arithmetic operation cannot be expressed as number. It is the only value in JS. that is not equal to itself.

null is a keyword, whereas undefined is a normal identifier that happens to be a global property.

- Symbol :

Symbols are new primitive type. They are completely unique identifiers. They are created using factory function Symbol():

```
const symbol = Symbol('description');
```

A new and unique symbol is created. So, every symbol has its own identity:

```
Symbol() === Symbol()  // false
```

Use Cases :- Symbols are keys of non-public properties
Whenever there are inheritance heirarchies in JS, two kinds of properties are there:

1. public properties are seen by clients of the code.
2. private properties are used internally within the pieces that make up the inheritance hierarchy (eg: classes, objects).

For usability's sake, public properties usually have string keys. But for private properties with string keys, accidental name clashes can become a problem. Therefore, symbols are good choice.
Note: symbols only protects you from name clashes, not from unauthorized access.

Usages:

1. To allow us to define constants with semantic names and unique values.
2. When we want to prevent collisions with keys in objects.
3. When we do not want our object properties to be enumerable.
4. To define how an object can be iterated.