**JavaScript**

1. **Primitive Data Types**

In JavaScript a Primitive is data that is not an Object and has no method or Properties. There are 7 primitive data types: string, number, Bigint, Boolean, undefined, symbol, and null.

All primitives are **immutable**, i.e., they cannot be altered. It is important not to confuse a primitive itself with a variable assigned a primitive value. The variable may be reassigned a new value, but the existing value cannot be changed in the ways that objects, arrays, and functions can be altered.

**String =>** In [JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript), a String is one of the [primitive values](https://developer.mozilla.org/en-US/docs/Glossary/Primitive) and the [String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String) object is a [wrapper](https://developer.mozilla.org/en-US/docs/Glossary/Wrapper) around a String primitive.

The **String** object is used to represent and manipulate a sequence of characters.

String literals (denoted by double or single quotes) and strings returned from String calls in a non-constructor context (that is, called without using the [new](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/new) keyword) are primitive strings. JavaScript automatically converts primitives to String objects, so that it's possible to use String object methods for primitive strings. In contexts where a method is to be invoked on a primitive string or a property lookup occurs, JavaScript will automatically wrap the string primitive and call the method or perform the property lookup.

let s\_prim = 'foo'

let s\_obj = new String(s\_prim)

console.log(typeof s\_prim) // Logs "string"

console.log(typeof s\_obj) // Logs "object"

**Number =>** In [JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript), **Number** is a numeric data type. It is capable of storing floating-point numbers between 2^-1074 and 2^1024, but can only safely store integers in the range -(2^53 − 1) to 2^53 − 1. Values outside of the range from [Number.MIN\_VALUE](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/MIN_VALUE) to [Number.MAX\_VALUE](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/MAX_VALUE) are automatically converted to either +Infinity or -Infinity, which behave similarly to mathematical infinity, but with some slight differences; see [Number.POSITIVE\_INFINITY](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/POSITIVE_INFINITY) for details.

Console.log(Number.MAX\_VALUE) // 1.7976931348623157e+308

Console.log(Number.MIN\_VALUE) // 5e-324

**BigInt** => The BigInt type is a numeric primitive in JavaScript that can represent integers with arbitrary precision. With BigInts, you can safely store and operate on large integers even beyond the safe integer limit for Numbers.

A BigInt is created by appending n to the end of an integer or by calling the constructor.

**Boolean** => In computer science, a **Boolean** is a logical data type that can have only the values true or false.

The value passed as the first parameter is converted to a Boolean value, if necessary. If the value is omitted or is 0, -0, [null](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/null), false, [NaN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/NaN), [undefined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined), or the empty string (""), the object has an initial value of false. All other values, including any object, an empty array ([]), or the string "false", create an object with an initial value of true.

**Undefined** => **undefined** is a [primitive](https://developer.mozilla.org/en-US/docs/Glossary/Primitive) value automatically assigned to [variables](https://developer.mozilla.org/en-US/docs/Glossary/Variable) that have just been declared, or to formal [arguments](https://developer.mozilla.org/en-US/docs/Glossary/Argument) for which there are no actual arguments.

**Symbol** => In [JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript), Symbol is a [primitive value](https://developer.mozilla.org/en-US/docs/Glossary/Primitive). A value having the data type **Symbol** can be referred to as a "Symbol value". In a JavaScript runtime environment, a symbol value is created by invoking the function [Symbol](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol), which dynamically produces an anonymous, unique value. A symbol may be used as an object property.

**Null** => In computer science, a **null** value represents a reference that points, generally intentionally, to a non-existent or invalid [object](https://developer.mozilla.org/en-US/docs/Glossary/Object) or address. The meaning of a null reference varies among language implementations.