# Getting Started with Javascript

## JavaScript data types and data structures

JavaScript is a loosely typed or a dynamic language. Variables in JavaScript are not directly associated with any particular value type, and any variable can be assigned (and re-assigned) values of all types.

let foo = 42; // foo is now a number

foo = 'bar'; // foo is now a string

foo = true; // foo is now a Boolean

### Data and Structure types

The latest ECMAScript standard defines nine types:

* Six **Data Types** that are [primitives](https://developer.mozilla.org/en-US/docs/Glossary/Primitive), checked by [typeof](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/typeof) operator:
  + [undefined](https://developer.mozilla.org/en-US/docs/Glossary/Undefined) : typeof instance === "undefined"
  + [Boolean](https://developer.mozilla.org/en-US/docs/Glossary/Boolean) : typeof instance === "boolean"
  + [Number](https://developer.mozilla.org/en-US/docs/Glossary/Number) : typeof instance === "number"
  + [String](https://developer.mozilla.org/en-US/docs/Glossary/String) : typeof instance === "string"
  + [BigInt](https://developer.mozilla.org/en-US/docs/Glossary/BigInt) : typeof instance === "bigint"
  + [Symbol](https://developer.mozilla.org/en-US/docs/Glossary/Symbol) : typeof instance === "symbol"
* [null](https://developer.mozilla.org/en-US/docs/Glossary/Null) : typeof instance === "object". Special [primitive](https://developer.mozilla.org/en-US/docs/Glossary/Primitive) type having additional usage for it's value: if object is not inherited null is shown at the end of [Prototype Chain](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Inheritance_and_the_prototype_chain);
* [Object](https://developer.mozilla.org/en-US/docs/Glossary/Object) : typeof instance === "object". Special non data but Structural type for any [constructed](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects#The_Constructor) instance instance also used as data structures: new [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object), new [Array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array), new [Map](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map), new [Set](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set), new [WeakMap](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/WeakMap), new [WeakSet](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/WeakSet), new [Date](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date) and almost everything made with [new keyword](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/new);
* [Function](https://developer.mozilla.org/en-US/docs/Glossary/Function) non data structure, though it also answers for typeof operator: typeof instance === "function". This answer is done as a special shorthand for Functions, though every Function constructor is derived from Object constructor.

### Primitive values

All types except objects define immutable values (that is, values which can't be changed). For example (and unlike in C), Strings are immutable. We refer to values of these types as "primitive values".

The number type has only one integer with two representations: 0 is represented as both -0 and +0. (0 is an alias for +0.)

+0===-0

> 42 / +0

Infinity

> 42 / -0

-Infinity

### String Type

JavaScript's [String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String) type is used to represent textual data. It is a set of "elements" of 16-bit unsigned integer values. Each element in the String occupies a position in the String. The first element is at index 0, the next at index 1, and so on. The length of a String is the number of elements in it.

Unlike some programming languages (such as C), JavaScript strings are immutable. This means that once a string is created, it is not possible to modify it.

However, it is still possible to create another string based on an operation on the original string. For example:

* A substring of the original by picking individual letters or using [String.substr()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/substr).
* A concatenation of two strings using the concatenation operator (+) or [String.concat()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/concat).

### Object type

In computer science, an object is a value in memory which is possibly referenced by an [identifier](https://developer.mozilla.org/en-US/docs/Glossary/Identifier).

Properties:

In JavaScript, objects can be seen as a collection of properties. With the object literal syntax, a limited set of properties are initialized; then properties can be added and removed. Property values can be values of any type, including other objects, which enables building complex data structures. Properties are identified using key values. A key value is either a String or a Symbol value.

Examples:->

### Data property:

Associates a key with a value, and has the following attributes:

| **Attributes of a data property** | | | |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Description** | **Default value** |
| [[Value]] | Any JavaScript type | The value retrieved by a get access of the property. | undefined |
| [[Writable]] | Boolean | If false, the property's [[Value]] cannot be changed. | false |
| [[Enumerable]] | Boolean | If true, the property will be enumerated in [for...in](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in) loops. See also [Enumerability and ownership of properties](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Enumerability_and_ownership_of_properties). | False |
| [[Configurable]] | Boolean | If false, the property cannot be deleted, cannot be changed to an  accessor property, and attributes other than [[Value]] and  [[Writable]] cannot be changed. | false |

#### Accessor property

Associates a key with one of two accessor functions (get and set) to retrieve or store a value, and has the following attributes:

| **Attributes of an accessor property** | | | |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Description** | **Default value** |
| [[Get]] | Function object or undefined | The function is called with an empty argument list and retrieves the property value whenever a get access to the value is performed. See also [get](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/get). | undefined |
| [[Set]] | Function object or undefined | The function is called with an argument that contains the assigned value and is  executed whenever a specified property is attempted to be changed. See also [set](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/set). | undefined |
| [[Enumerable]] | Boolean | If true, the property will be enumerated in [for...in](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in) loops. | false |
| [[Configurable]] | Boolean | If false, the property can't be deleted and can't be changed to a data property. | false |

Example->TODO

A JavaScript object is a mapping between keys and values. Keys are strings (or [Symbol](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol)s), and values can be anything. This makes objects a natural fit for [hashmaps](http://en.wikipedia.org/wiki/Hash_table).

Functions are regular objects with the additional capability of being callable.

### Symbol type

Symbols are new to JavaScript in ECMAScript 2015. A Symbol is a unique and immutable primitive value and may be used as the key of an Object property. In some programming languages, Symbols are called "atoms".

### Dates ;

When representing dates, the best choice is to use the built-in [Date utility](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date) in JavaScript.

Example

Indexed Collection : Array

[Arrays](https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Global_Objects/Array) are regular objects for which there is a particular relationship between integer-key-ed properties and the length property.

Examples ->

The **slice()** method returns a shallow copy of a portion of an array into a new array object selected from begin to end (end not included) where begin and end represent the index of items in that array. The original array will not be modified.

The **splice()** method changes the contents of an array by removing or replacing existing elements and/or adding new elements [in place](https://en.wikipedia.org/wiki/In-place_algorithm).

The **reduce()** method executes a **reducer** function (that you provide) on each element of the array, resulting in a single output value.

arr.reduce(callback( accumulator, currentValue[, index[, array]] )[, initialValue])

### JSON

JSON (**J**ava**S**cript **O**bject **N**otation) is a lightweight data-interchange format, derived from JavaScript, but used by many programming languages. JSON builds universal data structures.

Methods

JSON.parse(text[, reviver])

Parse the string text as JSON, optionally transform the produced value and its properties, and return the value. Any violations of the JSON syntax, including those pertaining to the differences between JavaScript and JSON, cause a SyntaxError to be thrown. The reviver option allows for interpreting what the replacer has used to stand in for other datatypes.

JSON.stringify(value[, replacer[, space]])

Return a JSON string corresponding to the specified value, optionally including only certain properties or replacing property values in a user-defined manner. By default, all instances of undefined are replaced with null, and other unsupported native data types are censored. The replacer option allows for specifying other behavior.