**Javasript objects:** In JavaScript, objects are king. If you understand objects, you understand JavaScript.

In JavaScript, almost "everything" is an object.

* Booleans can be objects (if defined with the new keyword)
* Numbers can be objects (if defined with the new keyword)
* Strings can be objects (if defined with the new keyword)
* Dates are always objects
* Maths are always objects
* Regular expressions are always objects
* Arrays are always objects
* Functions are always objects
* Objects are always objects

**JavaScript Primitives:**

* A **primitive value** is a value that has no properties or methods.
* **3.14** is a primitive value
* A **primitive data type** is data that has a primitive value.
* JavaScript defines 7 types of primitive data types:
* string
* number
* boolean
* null
* undefined
* symbol
* bigint

**Creating a JavaScript Object**

With JavaScript, you can define and create your own objects.

There are different ways to create new objects:

* Create a single object, using an object literal.
* Create a single object, with the keyword new.
* Define an object constructor, and then create objects of the constructed type.
* Create an object using Object.create().

**Objects internal representation in Javascript:**

|  |  |
| --- | --- |
| By object literal | By creating instance of Object directly (using new keyword) |
| syntax :    Property and value is separated by colon(:). | syntax :    Here, **new keyword** is used to create object. |
| **Example 1:**  Var person= {fname:”xxx”, lname:”yyy”age:25} | **Example2 :**  Var emp=new Object();  Emp.id=101;  Emp.name=”xxx”;  Emp.salary=400000; |
| Accessing JavaScript Objects: objectName.propertyName; Accessing ‘fname’ from example 1 using dot operator,Person.fname; | Accessing JavaScript Objects: objectName[“propertyName”]  Accessing ‘name’ form example 2 using [],  **emp[“name”]** |

## JavaScript Objects are Mutable:

Objects are mutable: They are addressed by reference, not by value.

If person is an object, the following statement will not create a copy of person:

Example:

const x = person;  // Will not create a copy of person.

* The object x is **not a copy** of person. It **is** person. Both x and person are the same object.
* Any changes to x will also change person, because x and person are the same object