JavaScript Objects

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

All JavaScript values, except primitives, are objects.

JavaScript Primitives

A **primitive value** is a value that has no properties or methods.

A **primitive data type** is data that has a primitive value.

JavaScript defines 5 types of primitive data types:

- string
- number
- boolean
- null
- undefined

Primitive values are immutable (they are hardcoded and therefore cannot be changed).

if x = 3.14, you can change the value of x. But you cannot change the value of 3.14.

Value	Type	Comment
"Hello"	string	"Hello" is always "Hello"
3.14	number	3.14 is always 3.14
True	boolean	true is always true
False	boolean	false is always false
Null	null (object)	null is always null
Undefined	undefined	undefined is always undefined

Objects in JavaScript, just as in many other programming languages, can be compared to objects in real life. The concept of objects in JavaScript can be understood with real life, tangible objects.

In JavaScript, an object is a standalone entity, with properties and type. Compare it with a cup, for example. A cup is an object, with properties. A cup has a color, a design, weight, a material it is made of, etc. The same way, JavaScript objects can have properties, which define their characteristics.

Objects and Properties

A JavaScript object has properties associated with it. A property of an object can be explained as a variable that is attached to the object. Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects. The properties of an object define the characteristics of the object. You access the properties of an object with a simple dot-notation:

```
objectName.propertyName
```

Like all JavaScript variables, both the object name (which could be a normal variable) and property name are case sensitive. You can define a property by assigning it a value. For example, let's create an object named myCar and give it properties named make, model, and year as follows:

```
var myCar = new Object();

myCar.make = 'Ford';

myCar.model = 'Mustang';

myCar.year = 1969;
```

Objects are Variables

JavaScript variables can contain single values:

Example
<html>

<body>

<h2>JavaScript Variables</h2>

<script>

// Create and display a variable:

var person = "John Doe";

document.getElementById("demo").innerHTML = person;

</script>

</body>

</html>

Objects are variables too. But objects can contain many values.

The values are written as **name : value** pairs (name and value separated by a colon).

Example

```
<html>
<body>
<p>Creating a JavaScript Object.</p>
<p id="demo"></p>
<script>
var person = {
  firstName : "John",
  lastName  : "Doe",
  age     : 50,
  eyeColor  : "blue"
};
document.getElementById("demo").innerHTML = person.firstName + " " + person.lastName;
</script>
</body>
</html>
```

The above example could also be written using an **object initializer**, which is a comma-delimited list of zero or more pairs of property names and associated values of an object, enclosed in curly braces ({}):

```
var myCar = {
    make: 'Ford',
    model: 'Mustang',
    year: 1969
};
```