What are Objects

Everything in JavaScript is either primitive datatype,
like **Strings**, **Numbers**, **Boolean** , **undefined**, **null** or an **Object**. **Arrays**, **Functions** are also **Objects**, but they are built-in objects of javascript.

JavaScript Objects just like any entity with properties and values. Values can be of any datatype. Like a car is an **object** which can have properties like *length*, *width*, *height*, *power*, *torque* etc and methods like *start()*, *reverse()*, *ignition()* etc.

JavaScript Objects are collection of *Multiple datatypes* inside a single variable, just like Arrays, but Arrays use index notation and **Objects** use key-value pair.

Built-in Objects in JavaScript

1. console
2. document
3. window
4. screen
5. navigator
6. history
7. location

How to declare Object in JavaScript

Objects in javascript can be declared using curly brackets {}. We can also use *new Object()* constructor function to declare Objects. Both {} and *new Objects()* works same. **new Object()** is constructor form, and curly brackets {} is literal form of **JS Objects**.

JavaScript Object Literal

```
var month={};            // blank Object
```

JavaScript Object Constructor

```
var month=new Object();        // blank Object
```

Check Object datatype using instanceof

Objects are Reference data types. typeof operator will return "object" for both Arrays and Objects. To check datatype of an object, use **instanceof(Object)** method. **instanceof()** method will check whether the passed value is an Object or not.

instanceOf() array, object and function is true. This means **instanceof** can check datatype of all reference datatypes (Arrays, Objects and Functions).

Instance Of Object

```
var x={};
typeof x;                 // return "object"
x instanceof(Object);     // return true
```

1. instanceof of Arrays, Objects and Functions is Object.
2. For Arrays, use Array.isArray().
3. For functions, use *typeof* operator.

Traversal values of Object

To check element inside Object, we can call object.property or object followed by property in brackets. To find first element, call array[0]. This index notation starts from 0 to array.length-1.

```
var user={
    "full-name":"avinash malhotra",
```

```
    profile:"UI Developer",
    location:"Noida, Delhi NCR",
    pincode:201301,
};

    user["full-name"];  // return "avinash malhotra"
    user.profile;         // return "UI Developer"
    user.location;        // return "Noida, Delhi NCR"
    user.pincode;         // return 201301

      or

    user["full-name"];     // return "avinash malhotra"
    user["profile"];  // return "UI Developer"
```

Check property in object

An **Object** can have both custom properties and built-in properties (*like toString, toLocaleString etc*). **in** operator is used to check whether given property or method (*function*) exists or not. **in** operator return value in boolean form, either true or false. See example

```
    var user={
            "full-name":"avinash malhotra",
            profile:"UI Developer",
            location:"Noida, Delhi NCR",
            pincode:201301,
            };

    "location" in user;   // returns true
    "toString" in user;   // returns true
    "age" in user;        // returns false
```

hasOwnProperty

The **hasOwnProperty** property check whether the property is his own property or global property of all JS Objects. It returns boolean value, true or false. *in* operator returns true for all properties, including global. See example

```
    var user={
            "full-name":"avinash malhotra",
            profile:"UI Developer",
            location:"Noida, Delhi NCR",
            pincode:201301,
            };

user.hasOwnProperty("pincode");   // returns true
user.hasOwnProperty("profile");   // returns true
user.hasOwnProperty("toString");   // returns false
```

Add and remove property in objects

An **JS Object** can **add and remove properties** in object. Even if **object** is already declared, still we can **add and properties** in object

Add Property in Object

```
    var user={
            "full-name":"avinash malhotra",
            profile:"UI Developer",
            location:"Noida, Delhi NCR",
            pincode:201301,
            };

    user.country="India";
```

Remove Property in Object

```
var user={
        "full-name":"avinash malhotra",
        profile:"UI Developer",
        location:"Noida, Delhi NCR",
        pincode:201301,
        };

delete user["full-name"];
```

Change Property value in Object

```
var user={
        "full-name":"avinash malhotra",
        profile:"UI Developer",
        location:"Noida, Delhi NCR",
        pincode:201301,
        };

user.pincode=110022;
```

Get All properties of Object

To **get all properties of object**, **for in loop** is required. **for in loop** can access all properties inside object and array.

```
full-name - avinash malhotra
profile - UI Developer
location - Noida, Delhi NCR
pincode - 201301
```

```
    var user={
            "full-name":"avinash malhotra",
            profile:"UI Developer",
            location:"Noida, Delhi NCR",
            pincode:201301,
            };
    for( var i in user){
    console.log(i + " - " + user[i]);
    }
```

Object Methods

Objects can have both properties and methods. For **object methods**, we have to assign function to object property and use return value as output.

this keyword in value of object will refer to current object, i.e car.

102.85714285714286 bhp/ton

129.14285714285714 Nm/ton

```
var car={
  name:"swift",
  power:90,
  torque:113,
  weight:875,

  powerToWeight:function(){return this.power/this.weight * 1000},

  torqueToWeight:function(){return this.torque/this.weight * 1000}
};

  console.log(car.powerToWeight()+ " bhp/ton");
```

```
console.log(car.torqueToWeight() + " Nm/ton");
```

In the object example, we had a car name *swift* with 90bhp power and 113Nm Torque. The weight of car is 875kg.

To calculate powerToWeight ratio, we have to divide car power by car weight and then multiply by 1000 and for and torqueToWeight ratio, we have to divide car torque by car weight and then multiply by 1000.

Object.Keys

Object.keys() method return an array of given objects keys.

['name', 'power', 'torque', 'weight']

```
let car={
        name:"swift",
        power:90,
        torque:113,
        weight:875
    };

let keys=Object.keys(car);
```

Object.values

Object.values() method return an array of given objects values.

['swift', 90, 113, 875]

```
let car={
        name:"swift",
```

```
        power:90,
        torque:113,
        weight:875
    };

let keys=Object.values(car);
```

Convert Object to Array using Object.entries()

Objects,entries() methods convert an **JavaScript Object to Array** with each key-value pair in nested array. This method is also useful to **count number of key-values in Object.**

[["name","swift"], ["power",90], ["torque",113], ["weight",875]]

```
let car={
  name:"swift",
  power:90,
  torque:113,
  weight:875,
};

console.log(Object.entries(car));
```

Check length of Object

In JavaScript *length* is the property of Arrays, Strings and Functions. To **check length of Object**, use Object.entries(*obj*).length. Object.entries(*obj*) will convert to Array and then we can check length property of array.

4

```
let car={
  name:"swift",
```

```
  power:90,
  torque:113,
  weight:875,
};

console.log(Object.entries(car).length);
```

Iterate over Object.entries

We can also use **for-of loop** to iterate over array build using **Object.entries()**. This is similar to using for-in loop for Object, but for-in will iterate over both own and prototype properties.

name swift

power 90

torque 113

weight 875

```
let car={
  name:"swift",
  power:90,
  torque:113,
  weight:875,
};

let obj=Object.entries(car);

for(let [i,j] of obj){
  console.log(i,j);
}
```

Array like Objects

In JavaScript, there are both Array and Arrays-like Object. Here is a list of some **array-like objects** in javascript. Arrays-like Objects behave like array and both length property and indexing, but the datatype is not array.

Array like Objects example

1. document.images
2. document.getElementsByTagName
3. document.getElementsByClassName
4. document.getElementsByName
5. document.querySelectorAll
6. strings

Custom array-like Object

```
let obj={ 0: 'a', 1:'b', 2:'c'};
```

datatype of array-like objects

```
let obj=document.querySelectorAll('p');
console.log(typeof obj);                                    // object
console.log(Array.isArray(obj));                             // false
```

Arrays-like Objects to Array

To convert **Array like Objects to Arrays** , use **Array.from()** method in ES6.

```
let obj=document.querySelectorAll('p');
let arr=Array.from(obj);
```

```
console.log(Array.isArray(obj));                    // false
console.log(Array.isArray(arr));                     // true
```

String to Array

['a', 'b', 'c', 'd']

```
let str="abcd";
let arr=Array.from(str);

console.log(arr);
```