Object

An object is a collection of properties, and a property is an association between a name (or *key*) and a value. A property's value can be a function, in which case the property is known as a method.

Syantax:

{

Property : value,

Property : value,

……

}

Here, “property” is a variable and “ value” is data associated with that variable.

This “value” can be a primitive or non-primitive. That means the “value” can be a number,string,Boolean,array,object or function.

The function of an object is called a method.

Accessing properties of an Object

This can be done in two ways.

- ➔ **Using dot operator**
  - Syntax:
    - Object-name . property
      It returns value of that property.
- ➔ **Square bracket notation**
  - Syntax
    - Object-name [‘property’]
      This cab be useful when property is a multi word name.

Creating an object

Different ways are there to create objects in JavaScript.

1.Using Object initializer (Object literal)

An object initializer is a comma-delimited list of zero or more pairs of property names and associated values of an object, enclosed in curly braces ({}).

Ex:

```
let object = {
  foo: 'bar',
  age: 42,
  skills:[“c”,”c++”,”JavaScript”],
  baz: {myProp: 12}
}
```

From ES6 onwards, methods can also be represented as like below

```
let o = {

  property(parameters) {},
}
```

Here, the “function” keyword is no longer necessary.

2.Using constructor function

you can create an object with these two steps:

1. Define the object type by writing a constructor function. There is a strong convention, with good reason, to use a capital initial letter.
2. Create an instance of the object with new.

Ex: Let us define "Car" type.

```
function Car(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
}
```

Create object of Car type

```
var mycar = new Car('Suzuki', 'Swift', 2000);
```

3.Using Object.create method

Objects can also be created using the Object.create() method. This method can be very useful, because it allows you to choose the prototype object for the object you want to create, without having to define a constructor function.

```
const person = {
  isHuman: false,
  printIntroduction: function() {
    console.log(`My name is ${this.name}. Am I human? ${this.isHuman}`);
  }
};

const me = Object.create(person);

me.name = 'Matthew'; // "name" is a property set on "me", but not on "person"
me.isHuman = true; // inherited properties can be overwritten


me.printIntroduction();

// expected output: "My name is Matthew. Am I human? true"
```

Add or delete property of an object

To add a new property

Syntax:

Object-name . new-property = value

To delete an existing property

Syntax:

delete object-name.property;

Important static methods of Object

Object.assign()-

It copies all properties from one or more *source objects* to a *target object*. It returns the target object.

Syntax:

Object.assign(target,sources)

The target object — what to apply the sources' properties to, which is returned after it is modified.

The source object(s) — objects containing the properties you want to apply.

Finally, it returns the target object

Note: Properties in the target object are overwritten by properties in the sources if they have the same key. Later sources' properties overwrite earlier ones.

This method can be used

To clone an object

To merge objects

Object.create()-

Create method is used to create object instance with an already declared object properties and it's prototype and assign it to a newly created prototype object and return's empty object

Note: Difference between create() and assign() methods is, assign method assigns sources to target and returns target object where as create() creates a new empty object with an already declared object properties and its prototype.

Object.entries()

Returns an array containing all of the [key,value] pairs.

Object.freeze()

It freezes an object. Other code cannot delete or modify its properties

Object.keys()

Returns an array of properties

Object.values()

Returns an array of values

Iterating an object

To walk over all keys of an object, there exists a special form of the loop: `for..in`.

Syntax:

```
for( let key of object)
    {
    }
```