Object:-

`Object()` :- Creates a new `Object` object. It is a wrapper for the given value.

Static Methods:-

`Object.assign()`
Copies the values of all enumerable own properties from one or more source objects to a target object.

`Object.create()`
Creates a new object with the specified prototype object and properties.

`Object.defineProperty()`
Adds the named property described by a given descriptor to an object.

`Object.defineProperties()`
Adds the named properties described by the given descriptors to an object.

`Object.entries()`
Returns an array containing all of the `[key, value]` pairs of a given object's **own** enumerable string properties.

`Object.freeze()`
Freezes an object. Other code cannot delete or change its properties.

`Object.fromEntries()`
Returns a new object from an iterable of `[key, value]` pairs. (This is the reverse of `Object.entries`).

`Object.getOwnPropertyDescriptor()`

Returns a property descriptor for a named property on an object.

`Object.getOwnPropertyDescriptors()`
Returns an object containing all own property descriptors for an object.

`Object.getOwnPropertyNames()`
Returns an array containing the names of all of the given object's **own** enumerable and non-enumerable properties.

`Object.getOwnPropertySymbols()`
Returns an array of all symbol properties found directly upon a given object.

`Object.getPrototypeOf()`
Returns the prototype (internal `[[Prototype]]` property) of the specified object.

`Object.is()`
Compare if two values are the same value. Equates all `NaN` values (which differs from both Abstract Equality Comparison and Strict Equality Comparison).

`Object.isExtensible()`
Determines if extending of an object is allowed.

`Object.isFrozen()`
Determines if an object was frozen.

`Object.isSealed()`
Determines if an object is sealed.

`Object.keys()`

Returns an array containing the names of all of the given object's **own** enumerable string properties.

`Object.preventExtensions()`
Prevents any extensions of an object.

`Object.seal()`
Prevents other code from deleting properties of an object.

`Object.setPrototypeOf()`
Sets the object's prototype (its internal `[[Prototype]]` property).

`Object.values()`
Returns an array containing the values that correspond to all of a given object's **own** enumerable string properties.

Instance methods:-

`Object.prototype.__defineGetter__()`

Associates a function with a property that, when accessed, executes that function and returns its return value.

`Object.prototype.__defineSetter__()`

Associates a function with a property that, when set, executes that function which modifies the property.

`Object.prototype.__lookupGetter__()`

Returns the function associated with the specified property by the `__defineGetter__()` **method.**

`Object.prototype.__lookupSetter__()`

Returns the function associated with the specified property by the

`__defineSetter__()` **method.**

`Object.prototype.hasOwnProperty()`

Returns a boolean indicating whether an object contains the specified property as a direct property of that object and not inherited through the prototype chain.

`Object.prototype.isPrototypeOf()`

Returns a boolean indicating whether the object this method is called upon is in the prototype chain of the specified object.

`Object.prototype.propertyIsEnumerable()`

Returns a boolean indicating if the internal ECMAScript [[Enumerable]] attribute is set.

`Object.prototype.toLocaleString()`

Calls `toString()`**.**

`Object.prototype.toString()`

Returns a string representation of the object.

`Object.prototype.valueOf()`

Returns the primitive value of the specified object.

Object Properties:-

`objectName.propertyName :-` `we can define different properties as shown.`