Objects

An object in JavaScript is a collection of key-value pairs where each key is a string (or a Symbol) and each value can be of any data type, including other objects, functions, arrays, and primitive data types like strings, numbers, and Booleans. Objects are created using curly braces {}.

Creating Objects:

1. **Literal notation:**

```
let person = { name: "John", age: 30 };
```

2. **Using the Object constructor:**

```
let person = new Object();
person.name = "John";
person.age = 30;
```

Accessing Object Properties:

You can access object properties using dot notation or square bracket notation:

```
console.log(person.name); // Dot notation
 console.log(person['age']); // Square bracket notation
```

Adding and Modifying Properties:

```
person.gender = "Male"; // Adding a new property
person.age = 31; // Modifying an existing property
```

Deleting Properties:

```
delete person.age;
```

Object Methods:

Methods are functions stored as object properties.

```
let person = {
name: "John",
greet: function() { console.log("Hello, my name is " + this.name); } };
person.greet(); // Output: Hello, my name is John
```

Object Iteration:

You can iterate over an object's properties using loops or methods like **Object.keys()**, **Object.values()**, or **Object.entries()**.

```
for (let key in person) { console.log(key + ": " + person[key]); } Object.keys(person).forEach(function(key) { console.log(key + ": " + person[key]); });
```

Object Methods in JS

1. **Object.keys()**: Returns an array of a given object's property names.

```
const obj = { a: 1, b: 2, c: 3 };
console.log(Object.keys(obj)); // Output: ["a", "b", "c"]
```

Object.values(): Returns an array of a given object's own enumerable property values.

```
const obj = { a: 1, b: 2, c: 3 };
console.log(Object.values(obj)); // Output: [1, 2, 3]
```

3. **Object.entries()**: Returns an array of a given object's own enumerable string-keyed property [key, value] pairs.

```
const obj = { a: 1, b: 2, c: 3 };
 console.log(Object.entries(obj)); // Output: [["a", 1], ["b", 2],
["c", 3]]
```

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

```
const target = { a: 1, b: 2 };
const source = { b: 3, c: 4 };
Object.assign(target, source);
console.log(target); // Output: { a: 1, b: 3, c: 4 }
```

5. **Object.create():** Creates a new object with the specified prototype object and properties.

```
const obj = Object.create({ foo: 1 });
console.log(obj.foo); // Output: 1
```

6. **Object.freeze():** Freezes an object, preventing new properties from being added to it, existing properties from being removed, and values from being changed.

```
const obj = { a: 1, b: 2 };
Object.freeze(obj);
obj.c = 3; // This will not add 'c' to the object
console.log(obj); // Output: { a: 1, b: 2 }
```

7. **Object.seal():** Seals an object, preventing new properties from being added to it and marking all existing properties as non-configurable.

```
const obj = { a: 1, b: 2 };
Object.seal(obj);

delete obj.a; // This will not delete 'a' from the object
console.log(obj); // Output: { a: 1, b: 2 }
```

8. **Object.hasOwnProperty():** Returns a boolean indicating whether the object has the specified property as its own property

```
console.log(obj.hasOwnProperty("name"));
```

How to Iterate objects

for...in Loop

```
var obj={
  name:"johnn",
  age:20,
  city:{
    name:"new york",
  }
}
for ( key in obj) {
  console.log( key + obj[key]);
}
```

for...of Loop with Object.keys()

```
var obj={
  name:"johnn",
  age:20,
  city:{
    name:"new york",
  }
}
for ( key of Object.keys(obj)) {
  console.log(`${key}: ${obj[key]}`);
}
```

for...of Loop with Object.values()

```
var obj={
  name:"johnn",
  age:20,
  city:{
    name:"new york",
  }
}
for ( value of Object.values(obj)) {
  console.log(value);
}
```

for...of Loop with Object.entries()

```

var obj={
  name:"johnn",
  age:20,
  city:{
    name:"new york",
  }
}
for ( [key, value] of Object.entries(obj)) {
  console.log(`${key}: ${value}`);
}
```