Javascript Object Methods

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

Javascript

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

2) Object.values(obj): Returns an array of the object's own enumerable property values.

Javascript

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

Javascript Object Methods

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

Javascript

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

4) Object.isSealed(obj): Returns true if the object is sealed, otherwise false.

Javascript

```
const obj = Object.seal({ a: 1 });
console.log(Object.isSealed(obj));
// Output: true
```

Javascript Object Methods

5) Object.assign(): Copies the values of all enumerable properties from one or more source objects to a target object. It returns the target object.

Javascript

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

6) Object.freeze(): Freezes an object, preventing new properties from being added or existing

Javascript

```
const obj = { name: 'Khabib' };
Object.freeze(obj);
obj.name = 'Bob'; // This won't change the value
console.log(obj.name); // Output: 'Khabib'
```

Javascript Object Methods

7) Object.seal(): Seals an object, preventing new properties from being added, but allowing existing properties to be modified.

Javascript

```
const obj = { name: 'Alice' };
Object.seal(obj);
obj.name = 'Bob'; // This will update the value
obj.age = 25; // This won't add a new property
console.log(obj); // Output: { name: 'Bob' }
```

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

Javascript

```
const person = {greet() {console.log('Hello!');}};
const student = Object.create(person);
student.greet();
// Output: 'Hello!'
```

Javascript Object Methods

9) Object.defineProperty(): Defines a new property directly on an object or modifies an existing property.

Javascript

```
const obj = {};
Object.defineProperty(obj, 'name', {
  value: 'Alice',
  writable: false });
console.log(obj.name); // 'Alice'
```

10) Object.defineProperties(): Defines multiple new properties or modifies existing properties on an object.

Javascript

```
const obj = {};
Object.defineProperties(obj, {
  name: { value: 'Cormier', writable: false },
  age: { value: 30, writable: true } });
console.log(obj.name); // 'Cormier'
```

Javascript Object Methods

11) Object.isExtensible(): Determines if an object is extensible (i.e., whether new properties can be added).

Javascript

```
const obj = {};
console.log(Object.isExtensible(obj)); // true
Object.preventExtensions(obj);
console.log(Object.isExtensible(obj)); // false
```

12) Object.isFrozen(): Determines if an object is frozen (i.e., not extensible and all properties are non-writable).

Javascript

```
const obj = Object.freeze({ name: 'Gregor' });
console.log(Object.isFrozen(obj));
// output: true
```

Javascript Object Methods

13) Object.hasOwn(): Returns true if the specified object has the specified property as its own property, even if the property's value is undefined.

Javascript

```
const obj = { name: 'Alice' };
console.log(Object.hasOwn(obj, 'name')); // true
console.log(Object.hasOwn(obj, 'age')); // false
```

14) Object.hasOwnProperty(): Determines if an object is frozen (i.e., not extensible and all properties are non-writable).

Javascript

```
const obj = { name: 'Alice' };
console.log(obj.hasOwnProperty('name')); // true
console.log(obj.hasOwnProperty('age'));  // false
```

Javascript Object Methods

15) Object.preventExtensions(): Prevents new properties from ever being added to an object.

Javascript

```
const obj = {};
Object.preventExtensions(obj);
obj.name = 'Khabib'; // Won't be added
console.log(obj); // {}
```

16) Object.setPrototypeOf(): Sets the prototype (the internal [[Prototype]] property) of a specified object.

Javascript

```
const proto = { greet() {console.log('Hello!');}};
const obj = {};
Object.setPrototypeOf(obj, proto);
obj.greet(); // 'Hello!'
```

Javascript Object Methods

17) Object.fromEntries(): Transforms a list of key-value pairs into an object.

Javascript

```
const entries = [['name', 'Rock'], ['age', 35]];
const obj = Object.fromEntries(entries);
console.log(obj); // { name: 'Rock', age: 35 }
```

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

Javascript

```
const obj = {};
const proto = Object.getPrototypeOf(obj);
console.log(proto === Object.prototype); // true
```

Javascript Object Methods

19) `Object.getOwnPropertySymbols()`: Returns an array of all symbol properties found on the object.

Javascript

```
const symbol = Symbol('id');
const obj = { [symbol]: 123 };
const symbols = Object.getOwnPropertySymbols(obj);
console.log(symbols); // [Symbol(id)]
console.log(obj[symbols[0]]); // 123
```

20) `Object.getOwnPropertyDescriptor()`: Returns a property descriptor for a specific property of a given object.

Javascript

```
const obj = { name: 'Alice', age: 26 };
const descriptor =
Object.getOwnPropertyDescriptor(obj, 'name');
console.log(descriptor);
// Output: { configurable: true, enumerable: true,
value: "Alice", writable: true }
```

Javascript Object Methods

21) Object.getOwnPropertyNames(): Returns an array of all properties found on the object (including non-enumerable properties).

Javascript

```
const obj = { name: 'Ferguson', age: 30 };
const propertyNames =
Object.getOwnPropertyNames(obj);
console.log(propertyNames); // ['name', 'age']
```

22) Object.is(): Compares if two values are the same.

Javascript

```
console.log(Object.is('foo', 'foo')); // true
console.log(Object.is({}, {})); // false
```

Javascript Object Methods

23) Object.getOwnPropertyDescriptors(): Returns all own property descriptors of an object.

Javascript

```
const obj = { name: 'Khabib', age: 28 };
const descriptors =
Object.getOwnPropertyDescriptors(obj);
console.log(descriptors);
// Output: {
            age: {
                  configurable: true,
                  enumerable: true,
                  value: 28,
                  writable: true },
           name: {
                  configurable: true,
                  enumerable: true,
                  value: "Khabib",
                  writable: true
                 }
           }
```