JS - Object Manipulation

1. Basic Object Operations

In JavaScript, objects are collections of key-value pairs. Below are some basic operations to create and manipulate objects:

```
// Creating an object
let person = {
    name: "John",
    age: 30,
    job: "Developer"
};

// Accessing object properties
let name = person.name; // John
let age = person["age"]; // 30

// Adding new properties
person.country = "USA"; // Adding 'country' property

// Modifying existing properties
person.age = 31; // Changing 'age' to 31

// Deleting properties
delete person.job; // Deleting 'job' property
```

2. Common Object Methods

JavaScript provides several built-in methods for manipulating objects. Below are some commonly used ones:

2.1. `Object.keys()`

The `Object.keys()` method returns an array of the object's own enumerable property names (keys).

```
let keys = Object.keys(person);
// ['name', 'age', 'country']
```

2.2. `Object.values()`

The `Object.values()` method returns an array of the object's own enumerable property values.

```
let values = Object.values(person);
// ['John', 31, 'USA']
```

2.3. `Object.entries()`

The `Object.entries()` method returns an array of key-value pairs (as arrays) from the object.

```
let entries = Object.entries(person);
// [['name', 'John'], ['age', 31], ['country', 'USA']]
```

2.4. `Object.assign()`

The `Object.assign()` method copies all enumerable properties from one or more source objects to a target object. It performs a shallow copy.

```
let newPerson = Object.assign({}, person);
// Creates a new object with the same properties as 'person'
```

2.5. `Object.freeze()`

The `Object.freeze()` method freezes an object, preventing new properties from being added and existing properties from being modified or deleted.

```
Object.freeze(person);
person.age = 35; // This will not change the 'age' property.
```

2.6. `Object.seal()`

The `Object.seal()` method seals an object, preventing the addition of new properties but allows the modification of existing properties.

```
Object.seal(person);
person.city = "New York"; // This will fail, as 'city' cannot be added.
person.age = 32; // This will succeed, 'age' can still be modified.
```

2.7. `Object.hasOwnProperty()`

The `Object.hasOwnProperty()` method checks if the object has a specific property as its own (not inherited).

```
let hasAge = person.hasOwnProperty("age"); // true
let hasJob = person.hasOwnProperty("job"); // false
```

2.8. `Object.getOwnPropertyDescriptor()`

The `Object.getOwnPropertyDescriptor()` method returns a descriptor of an object's property, including details such as whether it's writable, enumerable, and configurable.

```
let descriptor = Object.getOwnPropertyDescriptor(person, "age");
// {value: 31, writable: true, enumerable: true, configurable: true}
```

3. Summary of Object Methods

Method	Example
Object.keys()	Object.keys(person)
Object.values()	Object.values(person)
Object.entries()	Object.entries(person)
Object.assign()	Object.assign({}, person)
Object.freeze()	Object.freeze(person)
Object.seal()	Object.seal(person)
Object.hasOwnProperty()	person.hasOwnProperty("age")
Object.getOwnPropertyDescriptor()	Object.getOwnPropertyDescriptor(person, "age")