JavaScript Objects Notes

Introduction to Objects

- **Objects** store key-value pairs: `const obj = { key: value }`.
- Objects provide methods like `Object.keys()`, `Object.values()`, and `Object.entries()` to retrieve keys, values, and key-value pairs.

Different Ways to Create Objects in JavaScript

1. **Object Literal (`{}`)** → Best for creating single objects.
2. **Using `new Object()`** → Creates an object using the built-in Object constructor.
3. **Using a Constructor Function** → Used to create multiple instances of an object.
4. **Using `Object.create()`** → Creates a new object with a specified prototype. Useful for inheritance.
5. **Using Class (ES6)** → Used to create multiple instances of an object.

Object Creation Examples

1. Object Literal (`{}`)

```
let person = { name: "Alice" };
```

2. Using `new Object()`

```
let person = new Object();
person.name = "Alice";
```

3. Using a Constructor Function

```javascript
function Person(name, age, city) {
    this.name = name;
    this.age = age;
    this.city = city;
}
let person1 = new Person("Alice", 25, "New York");
```

4. Using `Object.create()`

```javascript
let personPrototype = {
    greet: function () {
        console.log("Hello, my name is " + this.name);
    }
};
let person = Object.create(personPrototype);
person.name = "Alice";
person.greet(); // Hello, my name is Alice
```

5. Using ES6 Classes

```javascript
class Person {
    constructor(name, age, city) {
        this.name = name;
        this.age = age;
        this.city = city;
    }
    greet() {
        console.log("Hello, my name is " + this.name);
    }
}
let person1 = new Person("Alice", 25, "New York");
person1.greet(); // Hello, my name is Alice
```

Deleting Properties from an Object

```
let person = { name: "Alice", age: 15 };
delete person.name;
```

Printing Object Keys

```
let person = { name: "Alice", age: 25, city: "New York" };
console.log(Object.keys(person));
// Output: ["name", "age", "city"]
```

Using `Object.create()` for Inheritance

```
const functionsBundle = {
    addMoney: function () {
        this.accountBalance++;
    },
    fetchBalance: function () {
        console.log('The balance is ' + this.accountBalance);
    },
};
const account = Object.create(functionsBundle);
account.accountBalance = 100;
account.addMoney(); // Inherits addMoney from functionsBundle
account.fetchBalance();
```

Copying Objects

Using `Object.assign()`

```
let obj1 = { name: "Alice" };
let obj2 = { age: 25, city: "New York" };
let person = Object.assign({}, obj1, obj2);
console.log(person); // { name: "Alice", age: 25, city: "New York" }
```

What is `Object.freeze()` ?

- Prevents modifications to an object.
- No new properties can be added.
- Existing properties cannot be modified or deleted.

```
const person = { name: "Alice", age: 25 };
Object.freeze(person);
person.age = 30; // No effect
console.log(person); // { name: "Alice", age: 25 }
```

Checking if an Object is Frozen

```
console.log(Object.isFrozen(person)); // true
```

Limitations of `Object.freeze()`

- It only applies to the top-level properties.
- Nested objects can still be modified.

```
const user = {
    name: "Bob",
    details: { age: 30, city: "New York" }
};
Object.freeze(user);
user.details.age = 35; // Change happens!
console.log(user.details.age); // 35
```

Freezing Nested Objects

```
function deepFreeze(obj) {
    Object.keys(obj).forEach(key => {
        if (typeof obj[key] === "object" && obj[key] !== null) {
            deepFreeze(obj[key]);
        }
    });
    return Object.freeze(obj);
}
const deepUser = {
    name: "Charlie",
    details: { age: 40, city: "Los Angeles" }
};
deepFreeze(deepUser);
deepUser.details.age = 45; // No effect
console.log(deepUser.details.age); // 40

Another Alternative: Using JSON.stringify()
If the object contains only serializable data (i.e., no functions or circular reference
```

Summary

- Objects in JavaScript store key-value pairs and can be created using literals, constructors, `Object.create()` , and ES6 classes.
- Objects support deletion of properties, property enumeration, and inheritance.
- `Object.freeze()` is used to make objects immutable, but it does not freeze nested objects unless deep freezing is applied.