Objects

What is Object?

In JavaScript, an object is a collection of key-value pairs, where each key is a string (or symbol) and each value can be of any data type, including other objects. Objects are one of the fundamental data types in JavaScript and are used extensively for structuring and organizing data. Object is the most important concept in javascript.

Example:

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

To create an object using Object Literal notation in JavaScript, you define the object properties and values within curly braces {}. Here's an example:

```
const person = {
  name: "John",
  age: 30,
  city: "New York"
};
```

In the above code, the person object is created using Object Literal notation. It has three properties: name, age, and city. Each property is defined with a key-value pair, where the key is a string (e.g., "name") and the value can be of any data type (e.g., "John", 30, "New York").

Insert in Object:

To insert or add properties to an existing object in JavaScript, you can use dot notation (object.property) or bracket notation (object["property"]) to assign a value to a new or existing property. Here's an example:

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

person.city = "New York"; // Using dot notation
person["occupation"] = "Engineer"; // Using bracket
notation
```

In the above code, two properties (city and occupation) are added to the person object using dot notation and bracket notation, respectively. After inserting, the object will have the following properties:

```
console.log(person);
/*
Output:
{
  name: "John",
  age: 30,
  city: "New York",
  occupation: "Engineer"
}
*/
```

Read in object

To read or access the values of properties in a JavaScript object, you can use dot notation (object.property) or bracket notation (object["property"]). Here's an example:

```
const person = {
  name: "John",
  age: 30,
  city: "New York"
};

console.log(person.name); // Output: John
console.log(person.age); // Output: 30
console.log(person["city"]); // Output: New York
```

In the above code, we have an object person with three properties: name, age, and city. To read or access these properties, we use dot notation or bracket notation followed by the property name.

- person.name retrieves the value of the name property, which is "John".
- person.age retrieves the value of the age property, which is 30.
- person["city"] retrieves the value of the city property, which is "New York".

Update in object

To update the value of a property in a JavaScript object, you can use either dot notation (object.property) or bracket notation (object["property"]) to assign a new value to the property. Here's an example:

```
const person = {
  name: "John",
  age: 30,
  city: "New York"
};

person.name = "Jane"; // Using dot notation
person["age"] = 31; // Using bracket notation
```

In the above code, the name property is updated to "Jane" using dot notation, and the age property is updated to 31 using bracket notation. After updating, the object will look like this:

```
console.log(person);
/*
Output:
{
  name: "Jane",
  age: 31,
  city: "New York"
}
*/
```

Deletion in Object

To delete a property from a JavaScript object, you can use the delete operator. The delete operator removes a property from an object. Here's an example:

```
const person = {
  name: "John",
  age: 30,
  city: "New York"
};

delete person.age;
```

In the above code, the age property is deleted from the person object using the delete operator. After deletion, the object will look like this:

```
console.log(person);
/*
Output:
{
  name: "John",
  city: "New York"
}
*/
```

Creating a Simple Nested Object

To create a simple nested object in JavaScript, you can define an object that contains other objects as its properties. Here's an example of creating a simple nested object:

```
const person = {
  name: "John",
  age: 30,
  address: {
    street: "123 Main St",
    city: "New York"
  }
};
```

In the above code, the person object has three properties: name, age, and address. The address property itself is another object with its own properties, street and city.

You can access the properties of the nested object using dot notation or bracket notation:

```
console.log(person.name); // Output: John
console.log(person.address.street); // Output: 123
Main St
console.log(person["address"]["city"]); // Output: New
York
```

You can also modify the properties of the nested object by chaining the dot or bracket notation:

```
person.address.street = "456 Elm St";
person["address"]["city"] = "San Francisco";
```

After modifying the nested object, it will look like this:

```
console.log(person);
/*
Output:
{
  name: "John",
  age: 30,
  address: {
    street: "456 Elm St",
    city: "San Francisco"
  }
}
*/
```

Complex nested Object:

To create a complex nested object in JavaScript, you can define an object that contains multiple levels of nesting, with objects nested within objects. Here's an example of creating a complex nested object:

```
const company = {
  name: "ABC Corp",
  location: {
    city: "New York",
    country: "USA"
  },
  departments: {
    sales: {
      manager: "John",
      employees: ["Alice", "Bob", "Charlie"]
    },
    marketing: {
      manager: "Sarah",
      employees: ["David", "Emily"]
    }
  }
};
```

In the above code, the company object has several levels of nesting. It contains properties like name, location, and departments, and within location, there is another object with properties like city and country. Similarly, within departments, there are objects for different departments (sales and marketing), each with their own properties.

You can access the properties of the complex nested object using dot notation or bracket notation:

```
console.log(company.name); // Output: ABC Corp
console.log(company.location.city); // Output: New
York
console.log(company.departments.sales.manager); //
Output: John
console.log(company["departments"]["marketing"]
["employees"]); // Output: ["David", "Emily"]
```

You can also modify the properties of the complex nested object by chaining the dot or bracket notation:

```
company.name = "XYZ Corp";
company.location.country = "Canada";
company.departments.sales.manager = "Michael";
company["departments"]["marketing"]
["employees"].push("Frank");
```

After modifying the complex nested object, it will look like this:

```
console.log(company);
/*
Output:
{
  name: "XYZ Corp",
  location: {
    city: "New York",
    country: "Canada"
  },
  departments: {
    sales: {
      manager: "Michael",
      employees: ["Alice", "Bob", "Charlie"]
    },
    marketing: {
      manager: "Sarah",
      employees: ["David", "Emily", "Frank"]
    }
  }
}
*/
```