Lesson:

Object and Manipulating Object

PW SKILLS

Topics

1. Define Object in JavaScript
2. Creating an object
3. Manipulating values in an object
4. Interview Point

Define Object in JavaScript

In JavaScript, objects can be seen as a collection of properties. It is a collection of key-value pairs, where each key is a string or symbol that uniquely identifies a value. It is a fundamental data type and plays a crucial role in the language as it allows you to present complex data structures and manipulate them.

Example of object -

```javascript
const user = {
      name: "Subham",
      lastName: "Sahu",
      city: "Bangalore"
}
```

Creating an object

There are majorly 3 ways to create an object in javascript:

1. By object literal
2. By creating an instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

By object literal

```javascript
// Syntax
let object = {name1: value1, name2:value2..... nameN: valueN}
```

As you can see, named and value are separated by a colon, this is how we create an object with the object literal method.

Let's see this simple example -

```javascript
let emp = {
  id: 101,
  name: "Alex",
  salary: 10000,
};
console.log(emp.id + " " + emp.name + " " + emp.salary);
// output - 101 Alex 10000
```

Here, emp is an object and it constitutes of name-value pairs of Id:101, name: "Alex" and salary: 10000

2. By creating an instance of an Object directly

```javascript
// Syntax --
var objectname = new Object()
```

Here, a new keyword is used to create an object.
Let's see example -

```javascript
const emp = new Object();
emp.id = 101;
emp.name = "Alex";
emp.salary = 10000;

console.log(emp.id + " " + emp.name + " " + emp.salary);
// output - 101 Alex 10000
```

Here we used the new keyword to create the instance of the object emp. The individual values are assigned with the help of "." operator with the object emp.

Creating an instance of an Object directly is not recommended due to the following reasons -

- Lack of encapsulation - Creating an instance directly bypasses any initialization or setup logic defined in constructors or factory functions, leading to potential inconsistencies and difficulties in maintaining the object's internal state.
- Limited Flexibility - Direct instantiation doesn't allow for passing arguments or setting default values during object creation, limiting the object's configurability and adaptability.
- Inconsistent object creation - By not following the established pattern of using constructors or factory functions, code readability and predictability may suffer, making it harder for others to understand and maintain the code.

3. By using an object constructor

Here, we create a function with arguments.
Each argument value can be assigned to the current object by using **this** keyword, which refers to the current object.

An example of creating an object by object constructor-

```javascript
function Emp(id, name, salary) {
  this.id = id;
  (this.name = name), (this.salary = salary);
}

const emp = new Emp(101, "Alex", 10000);
console.log(emp.id + " " + emp.name + " " + emp.salary);
// Output - 101 Alex 10000
```

Here, this keyword is used to assign values to the arguments id, name and salary

Manipulating values in an object

In data manipulation, we will cover the following -

1. Accessing data
2. Adding data
3. Changing data
4. Deleting data

1. Accessing data

In JavaScript objects, we can either use dot notation or square bracket notation to access object properties or alter values.
Since it is easier to read and use, dot notation is more is commonly used.

Format for dot notation: objectName.propertyName.
Format for the square bracket notation: objectName['propertyName'].

Use square bracket notation without quotes if you wish to use a variable as a property name.
Examples

```javascript
const emp = {
  id: 101,
  name: "Alex",
  salary: "Doe",
};
console.log(emp.id);
console.log(emp["name"]);
// output -
// 101

// Alex
```

2. Adding data

It is simple to add additional key-value pairs to an existing object. Dot notation or square bracket notation can be used to accomplish that.

Let us see the example below to understand it better:

```javascript
// adding
let emp = {
  id: 101,
  name: "Alex",
  salary: 10000,
};

emp.items = 10;
console.log(emp);

//Adding one more attribute to the emp object

emp["type"] = "intern";
console.log(emp);
/*************** output /*************** */
{ id: 101, name: 'Alex', salary: 10000, items: 10 }
{ id: 101, name: 'Alex', salary: 10000, items: 10, type:
'intern' }
```

3. Changing data

Sometimes all an object needs is a simple value change. We accomplish this using dot notation or square bracket notation, just as we do when adding new data.

Example -

```javascript
//update --
let emp = {
  id: 101,
  name: "Alex",
  salary: 10000,};
//Using dot notation
emp.id = 102; // changing id for an emp
console.log(emp);

//Using brackets notation
emp["name"] = "Sam"; // Changing name for an emp
console.log(emp);

/*********** update object ************/
{ id: 102, name: 'Alex', salary: 10000 }
{ id: 102, name: 'Sam', salary: 10000 }
```

4. Deleting data

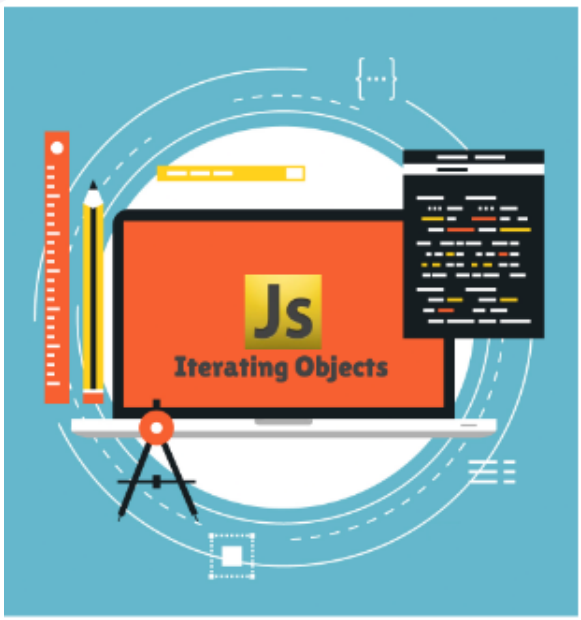
Data in an object can only be deleted with one method. It is done using the keyword delete. Data is not lost if we use undefined or null; the key is kept but the value is altered.

Example -

```javascript
let emp = {
    id: 101,
    name: "Alex",
    salary: 10000  }
  emp.name = null;
  console.log(emp);
  delete emp.name;
  console.log(emp);
/********* output ********/
{ id: 101, name: null, salary: 10000 }

{ id: 101, salary: 10000 }
```

Iterating Objects in Javascript



In JavaScript, just like an array, an object can also be iterated. However, there are several approaches to loop over the properties of an object.
Which includes - for...in, Object.keys(), Object.entries(), Object.getOwnPropertyNames().

Note - Object.keys(), Object.entries(), Object.getOwnPropertyNames() will study in the object methods module.

for...in

In JavaScript, for...of is a looping statement that executes a loop that operates on a sequence of values sourced from an iterable object.

Example

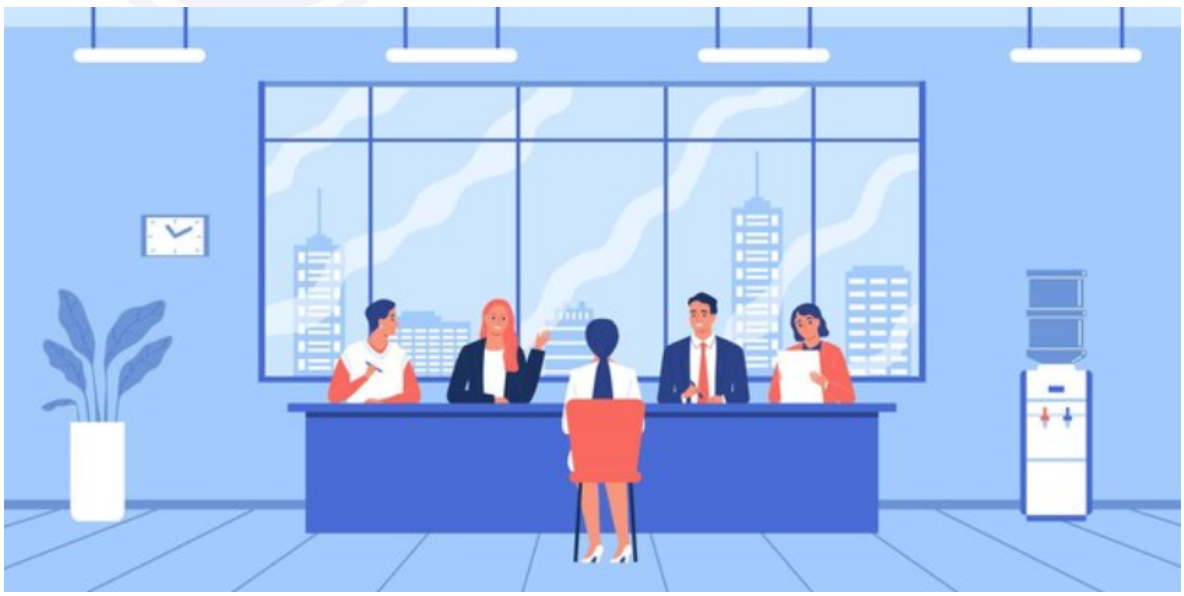
```javascript
const user = {
  name: "Suham",
  id: "1111",
  depart: "Wed dev",
};

for (const key in user) {
  console.log(user[key]);
}

// -- output --
// Suham
// 1111
// Wed dev

for (const key in user) {
  console.log(`${key} : ${user[key]}`);
}
// -- output --
// name : Suham
// id : 1111
// depart : Wed dev
```

Interview Point



Q. what is an object in javascript? Create a student object with name age roll number and then manipulate it (add new data to the object, delete data from the object, and update existing data from the object)

Ans. An object in JavaScript is a datatype that allows us to store and structure data using key-value pairs. Keys are strings or Symbols, and values can be of any data type, including other objects. Objects are used to represent entities and organize related information.

```
const student = {
  name: "Nasikh",
  age: 17,
  rollNumber: 43
};

//adding new data
student.status = 'active'

console.log(student)

//removing a data
delete student.age

console.log(student)

//updating an existing data
student.name = "Prabir"

console.log(student)
```

Output:

```
{ name: 'Nasikh', age: 17, rollNumber: 43, status: 'active' }
{ name: 'Nasikh', rollNumber: 43, status: 'active' }
{ name: 'Prabir', rollNumber: 43, status: 'active' }
```

Q. A student object is given below, I want you to print the keys of that object in the console.

```
const student = {
  name: "Nasikh",
  age: 17,
  branch: "CS",
  id: "18736"
};
```

Ans

JavaScript

```
for (const key in user) {
  console.log(key);
}
```