Introduction to Object Oriented JavaScript

Object Oriented JavaScript

Lesson Objectives

- Object-Oriented Terminology
- Types of Objects
- Creating New Types of Objects (Reference Types)
- Accessing Object Values / Getter and Setter methods

Object-Oriented Terminology

Object-oriented programming (OOP) is a programming paradigm that uses abstraction to create models based on the real world.

OOP uses several techniques from previously established paradigms, including modularity, polymorphism, and encapsulation.

OOP promotes greater flexibility and maintainability in programming, and is widely popular in large-scale software engineering.

As per ECMA the object in JavaScript is define as –

Unordered collection of properties each of which contains a primitive value, object, or function.

ECMAScript has no formal classes.

ECMA-262 describes object definitions as the way for an object.

Even though classes don’t actually exist in JavaScript, we will refer to object definitions as classes , as functionally both are same.

Object-Oriented Terminology-Types

In ECMAScript, all objects are not created equal.

Three specific types of objects can be used and/or created in JavaScript.

- Host Object
- Native Objects
- Built-in Object

Host Object:

Host Objects are objects that are supplied to JavaScript by the browser environment.

All BOM and DOM objects are considered to be host objects

Examples of these are window, document, forms, etc

Native Object

JavaScript has a number of built-in objects that extend the flexibility of the language. These objects are Date, Math, String, Array, and Object.

Object-Oriented Terminology

Build-in Objects:

Developer does not require to explicitly instantiate a built-in object,it is already instantiated.

- Only two built-in objects are defined by ECMA

Global and

Math

Both are native objects because by definition, every built-in object is a native object

Object-Oriented Terminology-Creating New Objects

Inline object:

An object can be created with brackets {…} with an optional list of properties. A property is a “key: value” pair, where key is a string (and value can be anything.

```
var employee={};  //inline Empty Object
console.log(employee);
employee.empId=1001; //key value pair
console.log(employee.empId);
var user = new Object(); // "object constructor" syntax
```

Usually, the brackets {...} are used. That declaration is called an object literal.

Object-Oriented Terminology-Creating New Objects

Existence check: A notable objects feature is that it's possible to access any property. There will be no error if the property doesn't exist! Accessing a non-existing property just returns undefined.

```
var emp={};
alert(emp.getProperty==undefined);
```

There also exists a special operator "in" to check for the existence of a property.

```
var empOne={eid:1001,ename:'ABCD'};
alert("eid" in empOne);//true
alert("edep" in empOne);//false
```

"for…in" loop:

Syntax

```
for(key in object) {
// executes the body for each key among object properties
}
```

Object-Oriented Terminology-Creating New Objects

Example

```
for(key in empTwo){
      console.log(key);    //key
      console.log(empTwo[key]); //value
}
```

Order :“ordered in a special fashion”: integer properties are sorted

```
var productCode={"121":"Mobile","11":"Rice","9":"Pencil","23":"Shirt"}
for(code in productCode){
      console.log(code);    //key
      console.log(productCode[code]); //value
}
```

Output is: in order 9 Pencil 11 Rice 23 shirt 121 Mobile

Object-Oriented Terminology-Creating New Objects

We can immediately put some properties into {...} as "key: value" pairs:

A property has a key (also known as "name" or "identifier") before the colon ":" and a value to the right of it.

It stores values by key, with that we can assign or delete it using "dot notation" or "Square Brackets" (associative arrays).

using dot notation

```
var myEmployee={
        empId:1001,
        empName:"Rahul",
        empDep:"Java",
        isAdmin:false,
        empSalary:4543.88,
        address:{
                street:"MG Road",
                city:"pune",
                pincode:410017
        }
};

console.log(myEmployee.empId);//print 1001
console.log(myEmployee.address);//all address will
print
delete myEmployee.empId //delete empId
```

using [] –square Brackets

```
var myEmployee={
        empId:1001,
        empName:"Rahul",
        empDep:"Java",
        isAdmin:false,
        empSalary:4543.88,
        address:{
        street:"MG Road",
        city:"pune",
        pincode:410017
        }
};

console.log(myEmployee["empId"]);//print 1001
console.log(myEmployee["address"]["city"]);//print
pune
```

Object-Oriented Terminology-Creating New Objects

Function Execution
By calling a function

```
function getEmployeeData(){
            console.log("Welcome to JavaScript OOPS")
}

getEmployeeData();
```

Referring to function--function Expression

```
var obj={};
var obj.callGet=function getEmployeeData(){
            console.log("Welcome to JavaScript OOPS")
};
 obj.callGet();
```

Anonymous function

```
var callGet=function{
            console.log("Welcome to JavaScript OOPS")
};

callGet();
```

Object-Oriented Terminology-Creating New Objects

Self invoking function

```
(function getEmployeeData(){
          console.log("Self invoking functions")
})();
```

Object-Oriented Terminology-Creating New Objects

Regular Function way in javascript

```
function createEmployee(empId,empName,empSalary,empDep){
      var emp={};
      emp.empId=empId;
      emp.empName=empName;
      emp.empSalary=empSalary;
      emp.empDep=empDep;
      return emp;
}

var empone=createEmployee(1001,'Rahul',2000.12,'JAVA');
console.log('Employee Id is '+empone.empId);
console.log('Employee Name is '+empone.empName);
console.log('Employee Salary is '+empone.empSalary);
console.log('Employee Department is '+empone.empDep);
```

Now going for constructor in a function

Object-Oriented Terminology-Creating New Objects

- A constructor is a function that instantiates a particular type of Object
- new Operator can be used for creating an object using Constructor (predefined/user defined).
- Object created using constructor will be reusable.
- When a function is called from the object, this becomes a reference to this object.

```
function createEmployee(empId,empName,empSalary,empDep){
        this.empId=empId;
        this.empName=empName;
        this.empSalary=empSalary;
        this.empDep=empDep;
}

var empone=new createEmployee(1001,'Rahul',2000.12,'JAVA');

console.log('Employee Id is '+empone.empId);
console.log('Employee Name is '+empone.empName);
console.log('Employee Salary is '+empone.empSalary);
console.log('Employee Department is '+empone.empDep);
```

Object-Oriented Terminology-Creating New Objects

with Function

```
function createEmployee(empId,empName,empSalary,empDep){
        this.empId=empId;
        this.empName=empName;
        this.empSalary=empSalary;
        this.empDep=empDep;
        this.totalSalary;
        this.getTakeHomeSalary=function(){
                this.totalSalary=this.empSalary-(this.empSalary*0.12);
                console.log("Employee Take Home Salary"+this.totalSalary)
        }
}
var empone=new createEmployee(1001,'Rahul',2000.12,'JAVA');
console.log('Employee Id is '+empone.empId);
console.log('Employee Name is '+empone.empName);
console.log('Employee Salary is '+empone.empSalary);
console.log('Employee Department is '+empone.empDep);
empone.getTakeHomeSalary();
```

Object-Oriented Terminology- Getter and Setter methods

getter & setter in Javascript

```
function createEmployee(empName){
      var empId;           //local
      this.empName=empName;
      this.getName=function(){
            alert("My Name is "+empName);
      }
      this.setEmpId=function(id){
      empId=id;
      }
      this.getEmpId=function(){
            return empId;
      }
}
var emp=new createEmployee("ABCD");
emp.setEmpId(1001);
console.log(emp.getEmpId());
emp.getName();
```

Object-Oriented Terminology- Other Way to create Object

The Object() constructor

We can use the Object() constructor to create a new object.

```
var person1 = new Object();
```

```
var person1 = new Object({
        name: 'Abcd',
        age: 18,
        greeting: function() {
            alert('Hi! I\'m ' + this.name + '.');
        }
});
```

Object-Oriented Terminology- Other Way to create Object

Create()

JavaScript has a built-in method called create() that allows us to create object.

```
var person1 = new Object({
        name: 'Abcd',
        age: 18,
        greeting: function() {
            alert('Hi! I\'m ' + this.name + '.');
        }
});
var person2 = Object.create(person1);
person2.greeting();
```

Demo

Demo1
Demo2
Demo3
Demo4
Demo5
Demo6
Demo7

Lab

Lab 1

Summary

In this lesson we have learned about -

- Object-Oriented concept with JavaScript
- Types of Objects
- How to Create New Types of Objects
- How to Access Object Values
- How to create Getter and Setter methods