Javascript objects

Agenda

- objects

- In JavaScript, objects are king. If you understand objects, you understand JavaScript.

- In JavaScript, almost "everything" is an object.

- Booleans can be objects (if defined with the new keyword) var val = new Boolean(value);
- Numbers can be objects (if defined with the new keyword)
- Strings can be objects (if defined with the new keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects
- All JavaScript values, except primitives, are objects.

JavaScript Objects

- A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.
- JavaScript is an object-based language. Everything is an object in JavaScript.
- JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

- There are 3 ways to create objects.

By object literal

By creating instance of Object directly (using new keyword)

By using an object constructor (using new keyword)

1) JavaScript Object by object literal

- The syntax of creating object using object literal is given below:

object={property1:value1,property2:value2.....propertyN:valueN}
As you can see, property and value is separated by : (colon).

Let’s see the simple example of creating object in JavaScript.

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

2) By creating instance of Object

- The syntax of creating object directly is given below:

var objectname=new Object();

Here, new keyword is used to create object.

- Let’s see the example of creating object directly.

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

3) By using an Object constructor

- Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.
- The **this keyword** refers to the current object.

The example of creating object by object constructor is given below.

Constructor functions are one way of creating objects in JavaScript. Using a constructor for an object creates a ‘blueprint’. This allows for many instances of the same object to be created using a keyword.

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

Defining method in JavaScript Object

- We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
this.changeSalary=changeSalary;
function changeSalary(otherSalary){
this.salary=otherSalary;
}
}
e=new emp(103,"Sonoo Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
e.changeSalary(45000);
document.write("<br>"+e.id+" "+e.name+" "+e.salary);
</script>
```

Empty object

- An empty object (“empty cabinet”) can be created using one of two syntaxes:
- let user = new Object(); // "object constructor" syntax
- let user = {}; // "object literal" syntax

Properties

- We can put some properties into {...} as “key: value” pairs:

```
let user = {     // an object
  name: "John",  // by key "name" store value "John"
  age: 30        // by key "age" store value 30
};
```

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

In the user object, there are two properties:

The first property has the name "name" and the value "John".

The second one has the name "age" and the value 30.

- Property values are accessible using the dot notation:
- // get property values of the object:

```
alert( user.name ); // John
alert( user.age ); // 30
```

Adding a property

```
user.isAdmin = true;
```

To remove a property, we can use delete operator:

```
delete user.age;
```

- We can also use multiword property names, but then they must be quoted:
- let user = { name: "John", age: 30,

"likes birds": true // multiword property name must be quoted };

Access it by using array notation

user["likes birds"])

Checking if a property exists

- The in operator returns true if the propertyName exists in the objectName.
- The following example creates an employee object and uses the in operator to check if the ssn and employeeId properties exist in the object:

```
let employee = {
   firstName: 'Peter',
   lastName: 'Doe',
   employeeId: 1
};


console.log('ssn' in employee);
console.log('employeeId' in employee);
```

Output:

false

true

Iterating over properties of an object using for...in loop

- To iterate over all properties of an object without knowing property names, you use the for...in loop:

```
for(let key in object) {
   // ...
}
```

For example, the following statement creates a website object and iterates over its properties using the for...in loop:

```
let website = {
   title: 'JavaScript Tutorial',
   url: 'https://www.javascripttutorial.net',
   tags: ['es6', 'javascript', 'node.js', 'reactjs', 'react native']
};

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

Output:

```
'title'
'url'
'tags'
```

Nested Object example

```
var person = {
    "name":"Ram",
    "age":27,
    "vehicles": {
      "car":"limousine",
      "bike":"ktm-duke",
      "plane":"lufthansa"
    }
  }
```

Array of objects

```
let cars = [
    {
      "color": "purple",
      "type": "minivan",
      "registration": new Date('2017-01-03'),
      "capacity": 7
    },
    {
      "color": "red",
      "type": "station wagon",
      "registration": new Date('2018-03-03'),
      "capacity": 5
    }]
```

In build object :Math

- The JavaScript Math object allows you to perform mathematical tasks on numbers.
- Math.E // returns Euler's number
- Math.PI // returns PI
- Math.SQRT2 // returns the square root of 2
- Math.SQRT1_2 // returns the square root of 1/2
- Math.LN2 // returns the natural logarithm of 2
- Math.LN10 // returns the natural logarithm of 10
- Math.LOG2E // returns base 2 logarithm of E
- Math.LOG10E // returns base 10 logarithm of E

Math method

- Math.round(x) Returns x rounded to its nearest integer
- Math.ceil(x) Returns x rounded up to its nearest integer
- Math.floor(x) Returns x rounded down to its nearest integer
- Math.trunc(x) Returns the integer part of x (new in ES6)
- Math.pow(x, y) returns the value of x to the power of y
- Math.sqrt(x) returns the square root of x

Math.random()

- Math.random() returns a random number between 0 (inclusive), and 1 (exclusive):
- // Returns a random integer from 0 to 10:
- Math.floor(Math.random() * 11);
- // Returns a random integer from 0 to 99:
- Math.floor(Math.random() * 100);
- // Returns a random integer from 1 to 10:
- Math.floor(Math.random() * 10) + 1;

Date Objects

- By default, JavaScript will use the browser's time zone and display a date as a full text string:
- Fri Jan 28 2022 00:32:26 GMT+0530 (India Standard Time)
- const d = new Date();
- Date objects are created with the new Date() constructor.

- There are 4 ways to create a new date object:

new Date()

new Date(year, month, day, hours, minutes, seconds, milliseconds)

new Date(milliseconds)

new Date(date string)

Date object method

- Method Description
- getFullYear() Get the year as a four digit number (yyyy)
- getMonth() Get the month as a number (0-11)
- getDate() Get the day as a number (1-31)
- getHours() Get the hour (0-23)
- getMinutes() Get the minute (0-59)
- getSeconds() Get the second (0-59)
- getMilliseconds() Get the millisecond (0-999)
- getTime() Get the time (milliseconds since January 1, 1970)
- getDay() Get the weekday as a number (0-6)
- Date.now() Get the time. ECMAScript 5.

Prototype in JavaScript

- JavaScript is a dynamic language. You can attach new properties to an object at any time as shown below.

Example: Attach property to object

```
function Student() {
    this.name = 'John';
    this.gender = 'Male';
}

var studObj1 = new Student();
studObj1.age = 15;
alert(studObj1.age); // 15
var studObj2 = new Student();
alert(studObj2.age); // undefined
```

- As you can see in the above example, age property is attached to studObj1 instance. However, studObj2 instance will not have age property because it is defined only on studObj1 instance.
- So what to do if we want to add new properties at later stage to a function which will be shared across all the instances?
- The answer is Prototype.
- The prototype is an object that is associated with every functions and objects by default in JavaScript, where function's prototype property is accessible and modifiable and object's prototype property (aka attribute) is not visible.

- Every function includes prototype object by default.