JavaScript

part3

contents

- Arrays
- Objects
- Creating objects
- This keyword
- Constructor functions
- Date object

Arrays

we need an *ordered collection*, where we have a 1st, a 2nd, a 3rd element and so on. For example, we need that to store a list of something: users, goods, HTML elements etc.

JavaScript array is an object that represents a collection of similar type of elements.

```
let fruits = ["Apple", "Orange", "Plum"];

 alert( fruits[0] );
 alert( fruits[1] );
 alert( fruits[2] );
```

There are two syntaxes for creating an empty array:

```
1.let arr = new Array();
2.let arr = [];
```

```
let fruits = ["Apple", "Orange", "Plum"];
```

```
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";
```

JavaScript Array Methods

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.

One of the ways to cycle array items is the `for` loop over indexes:

```
let arr = ["Apple", "Orange", "Pear"];

for (let i = 0; i < arr.length; i++)
{
 alert( arr[i] );
 }
```

it is also possible to use `for..in`:

```
let arr = ["Apple", "Orange", "Pear"];
 for (let key in arr)
{
 alert( arr[key] );
}
```

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.

Creating Objects in JavaScript

There are 3 ways to create objects.

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

JavaScript Object by object literal

```
object={property1:value1,property2:value2.....propertyN:valueN}
```

Example:

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

By creating instance of Object

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

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

By using an Object constructor

The **this keyword** refers to the current object.

```
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);
```

We can add, remove and read files from it any time.
Property values are accessible using the dot notation:

The value can be of any type. Let's add a boolean one:

```
let user = {
name: "John",
age: 30
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
};
```

For instance, let's output all properties of `user`:

```
let user = {
 name: "John",
age: 30,
isAdmin: true
};
 for (let key in user)
 {
alert( key );
alert( user[key] );
}
```

Defining Methods for an Object

```
<html>

   <head>
   <title>User-defined objects</title>
      <script type = "text/javascript">

         function addPrice(amount) {
            this.price = amount;
         }
           function book(title, author) {
            this.title = title;
            this.author  = author;
            this.addPrice = addPrice; }
      </script>

   </head>
   <body>
```

```
   <script type = "text/javascript">
         var myBook = new book("Perl", "Mohtashim");
     myBook.addPrice(100);

         document.write("Book title is : " + myBook.title + "<br>");
         document.write("Book author is : " + myBook.author + "<br>");
         document.write("Book price is : " + myBook.price + "<br>");
      </script>

   </body>
</html>
```

“this” in methods

It’s common that an object method needs to access the information stored in the object to do its job.

For instance, the code inside `user.sayHi()` may need the name of the `user`.

To access the object, a method can use the `this` keyword.

The value of **this** is the object “before dot”, the one used to call the method.

```
let user = { name: "John", age: 30,
sayHi() {alert(this.name); } };
user.sayHi(); // John
```

JavaScript Constructor Method

A JavaScript constructor method is a special type of method which is used to initialize and create an object. It is called when memory is allocated for an object.

Constructor functions technically are regular functions. There are two conventions though:

1.They are named with capital letter first.

2.They should be executed only with "**new**" operator.

```
function User(name) {
 this.name = name;
 this.isAdmin = false;
 }

 let user = new User("Jack");
alert(user.name); // Jack
alert(user.isAdmin); // false
```

When a function is executed with new, it does the following steps:

1.A new empty object is created and assigned to this.

2.The function body executes. Usually it modifies this, adds new properties to it.

3.The value of this is returned.

Return from constructors

Usually, constructors do not have a return statement.

But if there is a return statement, then the rule is simple:

- If return is called with an object, then the object is returned instead of this.
- If return is called with a primitive, it's ignored.

In other words, return with an object returns that object, in all other cases this is returned.

For instance, here return overrides this by returning an object:

```
function BigUser() {
this.name = "John";
return { name: "Godzilla" }; // <-- returns this object
}
alert( new BigUser().name );
```

Or

```
function SmallUser() {
this.name = "John";
return; // <-- returns this
}
alert( new SmallUser().name );
```

The constructor function may have parameters that define how to construct the object, and what to put in it.

Of course, we can add to this not only properties, but methods as well.

For instance, new User(name) below creates an object with the given name and the method sayHi:

```
function User(name) {
 this.name = name;
this.sayHi = function() {
 alert( "My name is: " + this.name );
};
 }
let john = new User("John");
john.sayHi(); // My name is: John
/* john = { name: "John",
sayHi: function() { ... } } */
```

JavaScript Date Object

The **JavaScript date** object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

Constructor

You can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

JavaScript Date Methods

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.

```
let now = new Date();
alert( now ); // shows current date/time
```

```
var date=new Date();
var day=date.getDate();
var month=date.getMonth()+1;
var year=date.getFullYear();
document.write("<br>Date is: "+day+"/"+month+"/"+year);
```

```
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
```