JS OBJECTS

Objects

- JavaScript objects are containers for **named values** called properties or methods.
- **Object definition**: You define (and create) a JavaScript object with an object literal

  var person = {firstName:“f.name", lastName:“l.name", age:50, eyeColor:“e_color"};
- The **name:values** pairs in JavaScript objects are called **properties.**

Accessing Object Properties

- You can access object properties in two ways:
  - *objectName.propertyName*
  - *objectName["propertyName"]*

Eg:

```
<p id="demo"></p>
<script>
// Create an object:
var person = {
firstName: “ABC",
 lastName : “DEF",
id    :  5566
};
// Display some data from the object:
document.getElementById("demo").innerHTML =
person.firstName + " " + person["lastName"];
</script>
```

Object Methods

- Objects can also have **methods**.
- Methods are **actions** that can be performed on objects.
- Accessing Object Methods
  - *objectName.methodName()*
  - *Eg:* name = person.fullName();

Cont..

- When a JavaScript variable is declared with the keyword "new", the variable is created as an object:
- ```
  var x = new String();        // Declares x as a String object
  var y = new Number();        // Declares y as a Number object
  var z = new Boolean();       // Declares z as a Boolean object
  ```
  ```

String

- A JavaScript string is zero or more characters written inside quotes.
  - var x = “ABC DEF";
  - var x = ’ABC DEF’;

- strings can also be defined as objects with the keyword new:
- Eg:
  - var x = “ABC";
  - var y = new String(“ABC");

JavaScript String Methods

length	The length property returns the length of a string
indexOf()	returns the index of (the position of) the first occurrence of a specified text in a string
lastIndexOf()	returns the index of the last occurrence of a specified text in a string
search()	searches a string for a specified value and returns the position of the match
slice()	extracts a part of a string and returns the extracted part in a new string.
substring()	similar to slice(). The difference is that substring() cannot accept negative indexes.
substr()	is similar to slice(). The difference is that the second parameter specifies the length of the extracted part.
replace()	method replaces a specified value with another value in a string
toUpperCase()	string is converted to upper case
toLowerCase()	string is converted to lower case
concat()	joins two or more strings
trim()	method removes whitespace from both sides of a string
charAt()	returns the character at a specified index (position) in a string
charCodeAt()	returns the unicode of the character at a specified index in a string
split()	string can be converted to an array

string can be converted to an array

```
<p id="demo"></p>
        <script>
        function myFunction() {
         var str = "a,b,c,d,e,f";
         var arr = str.split(",");
        document.getElementById("demo").innerHTML = arr[5];
        }
        </script>
```

JavaScript Numbers

- ```
  var x = "10";
  var y = "20";
  var z = x + y;          // z will be 1020 (a string)
  ```
- ```
  var x = 10;
  var y = "20";
  var z = x + y;          // z will be 1020 (a string)
  ```
- ```
  var x = "10";
  var y = 20;
  var z = x + y;          // z will be 1020 (a string)
  ```
  ```

JavaScript Number Methods

toString()	returns a number as a string
toExponential()	returns a string, with a number rounded and written using exponential notation
toFixed()	returns a string, with the number written with a specified number of decimals
toPrecision()	returns a string, with a number written with a specified length valueOf() returns a number as a number
Number()	Returns a number, converted from its argument.
parseFloat()	Parses its argument and returns a floating point number
parseInt()	Parses its argument and returns an integer

Arrays

- JavaScript arrays are used to store multiple values in a single variable.
- Creating an Array
  - var *array_name* = [*item1*, *item2*, ...];
  - var cars = [“Benz", "Volvo", "BMW"];
  - var cars = new Array(“Benz",, "Volvo", "BMW");

Looping Array Elements

```
<p id="demo"></p>
<script>
var fruits, text, fLen, i;
fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "<ul>";
    for (i = 0; i < fLen; i++)
    {
      text += "<li>" + fruits[i] + "</li>";
    }
text += "</ul>";
document.getElementById("demo").innerHTML = text;
</script>
```

Cont..

```
<p id="demo"></p>
    <script>
        var fruits, text;
        fruits = ["Banana", "Orange", "Apple", "Mango"];
        text = "<ul>";
                fruits.forEach(myFunction);
        text += "</ul>";
        document.getElementById("demo").innerHTML = text;
            function myFunction(value) {
              text += "<li>" + value + "</li>";
            }
    </script>
```

Adding Array Elements

```
<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

    <script>
    var fruits = ["Banana", "Orange", "Apple", "Mango"];
    document.getElementById("demo").innerHTML = fruits;
        function myFunction() {
          fruits.push("Lemon");
          document.getElementById("demo").innerHTML = fruits;
        }
    </script>
```

Cont..

```
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
        <script>
        var fruits = ["Banana", "Orange", "Apple", "Mango"];
        document.getElementById("demo").innerHTML = fruits;
              function myFunction() {
                fruits[fruits.length] = "Lemon";
                document.getElementById("demo").innerHTML = fruits;
              }
        </script>
```

Associative Arrays

- Many programming languages support arrays with named indexes.
- Arrays with named indexes are called associative arrays (or hashes).
- JavaScript does **not** support arrays with named indexes.
- In JavaScript, **arrays** always use **numbered indexes**.

Eg:

```
<p id="demo"></p>
        <script>
        var person = [];
        person[0] = "John";
        person[1] = "Doe";
        person[2] = 46;
        document.getElementById("demo").innerHTML =
        person[0] + " " + person.length;
        </script>
```

Output: John 3

Cont..

```
<script>
var person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
document.getElementById("demo").innerHTML =
person[0] + " " + person.length;
</script>
```

Output: undefined 0

Methods

toString()	converts an array to a string of (comma separated) array values.
join()	also joins all array elements into a string. It behaves just like toString(), but in addition you can specify the separator
pop()	removes the last element from an array
push()	adds a new element to an array (at the end)
shift()	removes the first array element and "shifts" all other elements to a lower index
unshift()	adds a new element to an array (at the beginning), and "unshifts" older elements
length	property provides an easy way to append a new element to an array
splice()	add/remove items to an array by specifying the position
concat()	creates a new array by merging (concatenating) existing arrays
slice()	method slices out a piece of an array into a new array

JavaScript Date Objects

- 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*)
- Get Date Method: Used for getting information from a date object
- Set Date Method: Set date values (years, months, days, hours, minutes, seconds, milliseconds) for a Date Object

JavaScript Get Date Methods

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.

JavaScript Set Date Methods

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)