JavaScript Date
JavaScript Boolean
JavaScript Math
JavaScript ReqExp Objects

JavaScript Date

1. Create a Date Object
2. Set Dates
3. Compare Two Dates
4. Examples
5. The Date object is used to work with dates and times.
6. Date object properties and methods

1. Create a Date Object

The Date object is used to work with dates and times.
Date objects are created with the Date() constructor.

There are four ways of instantiating a date:

a) new Date() // current date and time
b) new Date(milliseconds) //milliseconds since 1970/01/01
c) new Date(dateString)
d) new Date(year, month, day, hours, minutes, seconds, milliseconds)

All dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC) with a day containing 86,400,000 milliseconds.

Some examples of instantiating a date:

```
today = new Date()
d1 = new Date("October 13, 1975 11:13:00")
d2 = new Date(79,5,24)
d3 = new Date(79,5,24,11,33,0)
```

2. Set Dates

We can easily manipulate the date by using the methods available for the Date object.

In the following example, we set a Date object to a specific date **(14th January 2010)**:

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
```

And in the following example, we set a Date object to be five days into the future:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+5);
```

3. Compare Two Dates

The Date object is also used to compare two dates.
The following example compares today's date with the **14th January 2010:**

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
var today = new Date();
if (myDate>today)
{

alert("Today is before 14th January 2010");
}
else
{
alert("Today is after 14th January 2010");
}
```

4. Examples

The following example demonstrates how to use the **Date() method** to get today's date.

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
document.write(d);
</script>
</body>
</html>
```

The following example demonstrates how to use **getTime()** to calculate the **milliseconds since 1970.**

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
document.write(d.getTime() + " milliseconds since 1970/01/01");
</script>
</body>
</html>
```

The following example shows how to use **setFullYear()** to set a specific date.

```
<html>
<body>
<script type="text/javascript">
var d = new Date();
d.setFullYear(1992,11,3);
document.write(d);
```

```
</script>
</body>
</html>
```

The following example demonstrates how to use **toUTCString()** to convert today's date (according to UTC) to a string.

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
document.write("Original form: ");
document.write(d + "<br />");
document.write("To string (universal time): ");
document.write(d.toUTCString());
</script>
</body>
</html>
```

The **getDay() method** returns the day of the week as a number, with Sunday = 0. The following example demonstrates how to use getDay() and an array to display the day of the week as a text string rather than a number.

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
var weekday=new Array(7);
weekday[0]="Sunday";
weekday[1]="Monday";
weekday[2]="Tuesday";
weekday[3]="Wednesday";
weekday[4]="Thursday";
weekday[5]="Friday";
weekday[6]="Saturday";
document.write("Today is " + weekday[d.getDay()]);
</script>
</body>
</html>
```

The following example demonstrates how to display a clock on your Web page.

```
<html>
<head>
<script type="text/javascript">
function startTime()
```

```
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',500);
}
function checkTime(i)
{
if (i<10)
{
i="0" + i;
}
return i;
}
</script>
</head>
<body onload="startTime();">
<div id="txt"></div>
</body>
</html>
```

6. Date object properties and methods

Date Object Properties

Property	Description
constructor	Returns the function that created the Date object's prototype
prototype	Allows you to add properties and methods to an object

Date Object Methods

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year (four digits)
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)

getTime()	Returns the number of milliseconds since midnight Jan 1, 1970
getTimezoneOffset()	Returns the time difference between UTC time and local time, in minutes
getUTCDate()	Returns the day of the month, according to universal time (from 1-31)
getUTCDay()	Returns the day of the week, according to universal time (from 0-6)
getUTCFullYear()	Returns the year, according to universal time (four digits)
getUTCHours()	Returns the hour, according to universal time (from 0-23)
getUTCMilliseconds()	Returns the milliseconds, according to universal time (from 0-999)
getUTCMinutes()	Returns the minutes, according to universal time (from 0-59)
getUTCMonth()	Returns the month, according to universal time (from 0-11)
getUTCSeconds()	Returns the seconds, according to universal time (from 0-59)
getYear()	Deprecated. Use the getFullYear() method instead
parse()	Parses a date string and returns the number of milliseconds since midnight of January 1, 1970
setDate()	Sets the day of the month of a date object
setFullYear()	Sets the year (four digits) of a date object
setHours()	Sets the hour of a date object
setMilliseconds()	Sets the milliseconds of a date object
setMinutes()	Set the minutes of a date object
setMonth()	Sets the month of a date object
setSeconds()	Sets the seconds of a date object
setTime()	Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
setUTCDate()	Sets the day of the month of a date object, according to universal time
setUTCFullYear()	Sets the year of a date object, according to universal time (four digits)
setUTCHours()	Sets the hour of a date object, according to universal time
setUTCMilliseconds()	Sets the milliseconds of a date object, according to universal time
setUTCMinutes()	Set the minutes of a date object, according to universal time
setUTCMonth()	Sets the month of a date object, according to universal time
setUTCSeconds()	Set the seconds of a date object, according to universal time
setYear()	Deprecated. Use the setFullYear() method instead
toDateString()	Converts the date portion of a Date object into a readable string
toGMTString()	Deprecated. Use the toUTCString() method instead
toISOString()	Returns the date as a string, using the ISO standard
toJSON()	Returns the date as a string, formatted as a JSON date
toLocaleDateString()	Returns the date portion of a Date object as a string, using locale conventions
toLocaleTimeString()	Returns the time portion of a Date object as a string, using locale conventions
toLocaleString()	Converts a Date object to a string, using locale conventions
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string

toUTCString()	Converts a Date object to a string, according to universal time
UTC()	Returns the number of milliseconds in a date string since midnight of January 1, 1970, according to universal time
valueOf()	Returns the primitive value of a Date object

JavaScript Boolean

1. Create a Boolean Object
2. Examples
3. Boolean object properties and methods

The Boolean object is used to convert a non-Boolean value to a Boolean value (either true or false).

1. Create a Boolean Object

The Boolean object represents two values: true or false.
The following code creates a Boolean object called myBoolean:

```
var myBoolean=new Boolean();
```

All the following lines of code create Boolean objects with an initial value of false:

```
var myBoolean=new Boolean();
var myBoolean=new Boolean(0);
var myBoolean=new Boolean(null);
var myBoolean=new Boolean("");
var myBoolean=new Boolean(false);
var myBoolean=new Boolean(NaN);
```

If the Boolean object has no initial value or if it is 0, -0, null, “”, false, undefined, or NaN, the object is set to false. Otherwise, it is true (even with the string “false”)! And all the following lines of code create Boolean objects with an initial value of true:

```
var myBoolean=new Boolean(true);
var myBoolean=new Boolean("true");
var myBoolean=new Boolean("false");
var myBoolean=new Boolean("Richard");
```

2. Examples:

The following example demonstrates how to check whether a Boolean object is true or false.

```
<html>
<body>
<script type=”text/javascript”>
var b1=new Boolean(0);
var b2=new Boolean(1);
var b3=new Boolean(“”);
var b4=new Boolean(null);
var b5=new Boolean(NaN);
var b6=new Boolean(“false”);
document.write(“0 is boolean “+ b1 +”<br />”);
document.write(“1 is boolean “+ b2 +”<br />”);
document.write(“An empty string is boolean “+ b3 + “<br
/>”);
document.write(“null is boolean “+ b4+ “<br />”);
document.write(“NaN is boolean “+ b5 +”<br />”);
document.write(“The string ‘false’ is boolean “+ b6 +”<br
/>”);
```

```
</script>
</body>
</html>
```

3. Boolean object properties and methods

Boolean Object Properties

Property	Description
constructor	Returns the function that created the Boolean object's prototype
prototype	Allows you to add properties and methods to a Boolean object

Boolean Object Methods

Method	Description
toString()	Converts a Boolean value to a string, and returns the result
valueOf()	Returns the primitive value of a Boolean object

JavaScript Math

1. Math Object
2. Mathematical Constants
3. Mathematical Methods
4. Examples
5. Math object properties and methods

1. Math Object

The Math object allows you to perform mathematical tasks.
The Math object includes several mathematical constants and methods.
The syntax for using properties/methods of Math is as follows:

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

2. Mathematical Constants

JavaScript provides eight mathematical constants that can be accessed from the Math object.
These are E, PI, square root of 2, square root of 1/2, natural log of 2, natural log of 10, base-2 log of E, and base-10 log of E.

You may reference these constants from your JavaScript like this:

1. Math.E
2. Math.PI
3. Math.SQRT2
4. Math.SQRT1_2
5. Math.LN2
6. Math.LN10
7. Math.LOG2E
8. Math.LOG10E

3. Mathematical Methods

In addition to the **mathematical constants** that can be accessed from the Math object, several methods also are available.

The following example uses the round() method of the Math object to round a number to the nearest integer:

```
document.write(Math.round(4.7));
```

The following example uses the random() method of the Math object to return a random number between 0 and 1:

```
document.write(Math.random());
```

The following example uses the floor() and random() methods of the Math object to return a random number between 0 and 10:

```
document.write(Math.floor(Math.random()*11));
```

4. Examples

The following example demonstrates how to use round().

```
<html>
<body>
<script type=”text/javascript”>
document.write(Math.round(0.60) + “<br />”);
document.write(Math.round(0.50) + “<br />”);
document.write(Math.round(0.49) + “<br />”);
document.write(Math.round(-4.40) + “<br />”);
document.write(Math.round(-4.60));
</script>
</body>
</html>
```

The following example demonstrates how to use random() to return a random number between 0 and 1

```
<html>
<body>
<script type=”text/javascript”>
//return a random number between 0 and 1
document.write(Math.random() + “<br />”);
//return a random integer between 0 and 10
document.write(Math.floor(Math.random()*11));
</script>
</body>
</html>
```

The following example demonstrates how to use max() to return the largest of the specified values.

```
<html>
<body>
<script type=”text/javascript”>
document.write(Math.max(5,10) + “<br />”);
document.write(Math.max(0,150,30,20,38) + “<br />”);
document.write(Math.max(-5,10) + “<br />”);
document.write(Math.max(-5,-10) + “<br />”);
document.write(Math.max(1.5,2.5));
</script>
</body>
</html>
```

The following example shows how to use min() to return the smallest of the specified values.

```
<html>
<body>
```

```
<script type="text/javascript">
document.write(Math.min(5,10) + "<br />");
document.write(Math.min(0,150,30,20,38) + "<br />");
document.write(Math.min(-5,10) + "<br />");
document.write(Math.min(-5,-10) + "<br />");
document.write(Math.min(1.5,2.5));
</script>
</body>
</html>
```

5. Math object properties and methods

Math Object Properties

Property	Description
E	Returns Euler's number (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14)
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of Ex
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer

sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

JavaScript ReqExp Objects

1. What Is RegExp?
2. RegExp Modifiers
3. test()
4. exec()
5. RegExp object properties and methods

RegExp is short for regular expression

1. What Is RegExp?

A regular expression is an object that describes a pattern of characters. When you search in a text, you can use a pattern to describe what you are searching for.

A simple pattern can be a single character. A more complicated pattern can consist of more characters and can be used for parsing, format checking, substitution, and more.

Regular expressions are used to perform powerful pattern-matching and search and-replace functions on text.

The syntax is as follows:
var txt=new RegExp(pattern,modifiers);
or more simply:
var txt=/pattern/modifiers;

The syntax follows a couple of general guidelines:

- The pattern specifies the pattern of an expression.
- The modifiers specify whether a search should be global, case-sensitive, and so on.

2. RegExp Modifiers
Modifiers are used to perform case-insensitive and global searches.
The i modifier is used to perform case-insensitive matching.
The g modifier is used to perform a global match (find all matches rather than stopping after the first match).

The following example demonstrates how to do a case-insensitive search for “google” in a string:

```
var str="Visit Google";
var patt1=/google/i;
var txt=/pattern/modifiers;

<html>
<body>
<script type="text/javascript">
var str = "Visit Google";
var patt1 = /google/i;
document.write(str.match(patt1));
</script>
</body>
</html>
```

The following example demonstrates how to do a global search for “is”:

```
var str="Is this all there is?";
var patt1=/is/g;

<html>
<body>
<script type="text/javascript">
```

```
var str="Is this all there is?";
var patt1=/is/g;
document.write(str.match(patt1));
</script>
</body>
</html>
```

The following example demonstrates how to do a global, case-insensitive search for “is”:

```
var str="Is this all there is?";
var patt1=/is/gi;
```

```
<html>
<body>
<script type="text/javascript">
var str="Is this all there is?";
var patt1=/is/gi;
document.write(str.match(patt1));
</script>
</body>
</html>
```

3. test()
The test() method searches a string for a specified value and returns true or false, depending on the result.

The following example searches a string for the character “e”:

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
```

Because there is an “e” in the string, the output of the preceding code is as follows:
true

```
<html>
<body>
<script type="text/javascript">
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
</script>
</body>
</html>
```

4. exec()
The exec() method searches a string for a specified value and returns the text of the found value. If no match is found, it returns null.

The following example searches a string for the character “e”:

```
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
```

Because there is an “e” in the string, the output of the preceding code is:
e

```
<html>
<body>
<script type="text/javascript">
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
</script>
</body>
</html>
```

5. RegExp object properties and methods

RegExp Object Properties

Property	Description
constructor	Returns the function that created the RegExp object's prototype
Global	Specifies if the "g" modifier is set
ignoreCase	Specifies if the "i" modifier is set
lastIndex	Specifies the index at which to start the next match
Multiline	Specifies if the "m" modifier is set
Source	Returns the text of the RegExp pattern

RegExp Object Methods

Method	Description
compile()	Deprecated in version 1.5. Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false
toString()	Returns the string value of the regular expression