

# Introduction to JavaScript Training

**Built-In JavaScript Objects**

## Lesson 1, Activity 2: String

In JavaScript, there are two types of string data types: primitive strings and *String* objects. String objects have many methods for manipulating and parsing strings of text. Because these methods are available to primitive strings as well, in practice, there is no need to differentiate between the two types of strings. If you're feeling geeky, you can take a look at [BuiltInObjects/Demos/stringsAndStrings.html](#), which compares primitive type strings to String objects.

Some common string properties and methods are shown below. In all the examples, the variable `webucator` is the string "Webucator".

### Common String Properties

Property	Description
<code>length</code>	Read-only value containing the number of characters in the string.

### Common String Methods

Method	Description
<code>charAt(position)</code>	Returns the character at the specified position. Note that it is zero-based, so the first letter in the string is at position 0.
<code>charCodeAt(position)</code>	Returns the Unicode character code of the character at the specified position.
<code>fromCharCode(characterCodes)</code>	Returns the text representation of the specified comma-delimited character codes. Used with <code>String</code> rather than a specific String object.
<code>indexOf(substring, startPosition)</code>	Searches from <code>startPosition</code> for <code>substring</code> . Returns the position at which the <code>substring</code> is found. If <code>substring</code> is not found, returns -1.
<code>lastIndexOf(substring, endPosition)</code>	Searches from the end of the string for <code>substring</code> until <code>endPosition</code> is reached. Returns the position at which the <code>substring</code> is found. If <code>substring</code> is not found, returns -1.

`substring(startPosition,endPosition)`

Returns the substring beginning at `startPosition` and ending with the character before `endPosition`. `endPosition` is optional. If it is excluded, the substring continues to the end of the string.

`substr(startPosition,length)`

Returns the substring of `length` characters beginning at `startPosition`. `length` is optional. If it is excluded, the substring continues to the end of the string.

`slice(startPosition,endPosition)`

Same as `substring(startPosition, endPosition)`.

`slice(startPosition,positionFromEnd)`

`positionFromEnd` is a negative integer. Returns the substring beginning at `startPosition` and ending `positionFromEnd` characters from the end of the string.

`split(delimiter)`

Returns an array by splitting a string on the specified delimiter.

`toLowerCase()`

Returns the string in all lowercase letters.

`toUpperCase()`

Returns the string in all uppercase letters.

You can see these examples in a browser by opening [BuiltInObjects/Demos/StringPropertiesAndMethods.html](#). In the next activity, we'll show you how the `String.split()` method works.

Lesson 1, Activity 4: **Math**

The `Math` object is a built-in static object. The `Math` object's properties and methods can be accessed directly (e.g. `Math.PI`) and are used for performing complex math operations. Here are two commonly used properties of the `Math` object.

**Common Math Properties**

Property	Description
<code>Math.PI</code>	Pi (Π)
<code>Math.SQRT2</code>	Square root of 2.

Below are some of the common `Math` methods that you can use to perform mathematical operations:

**Common Math Methods**

Method	Description
<code>Math.abs (number)</code>	Absolute value of <code>number</code> .
<code>Math.ceil (number)</code>	<code>number</code> rounded up.
<code>Math.floor (number)</code>	<code>number</code> rounded down.
<code>Math.max (numbers)</code>	Highest Number in <code>numbers</code> .
<code>Math.min (numbers)</code>	Lowest Number in <code>numbers</code> .
<code>Math.pow (number, power)</code>	<code>number</code> to the power of <code>power</code> .
<code>Math.round (number)</code>	Rounded <code>number</code> .
<code>Math.random ()</code>	Random number between 0 and 1.

You can see these examples in a browser by opening [BuiltInObjects/Demos/MathPropertiesAndMethods.html](#). In the next activity, we'll use the `Math.round()` and `Math.random()` methods in a simple example.

#### **Method for Generating Random Integers**

You can easily generate random numbers in JavaScript using the `Math.random ()` method.

```
var low = 1;  
var high = 10;  
var rndDec = Math.random();  
  
var rndInt = Math.floor(rndDec * (high - low + 1) + low);
```

## Lesson 1, Activity 7: Date

The `Date` object has methods for manipulating dates and times. JavaScript stores dates as the number of milliseconds since January 1, 1970. The sample below shows the different methods of creating date objects, all of which involve passing arguments to the `Date()` constructor.

### Code Sample:

[BuiltInObjects/Demos/DateObject.html](#)

```

---- C O D E   O M I T T E D ----

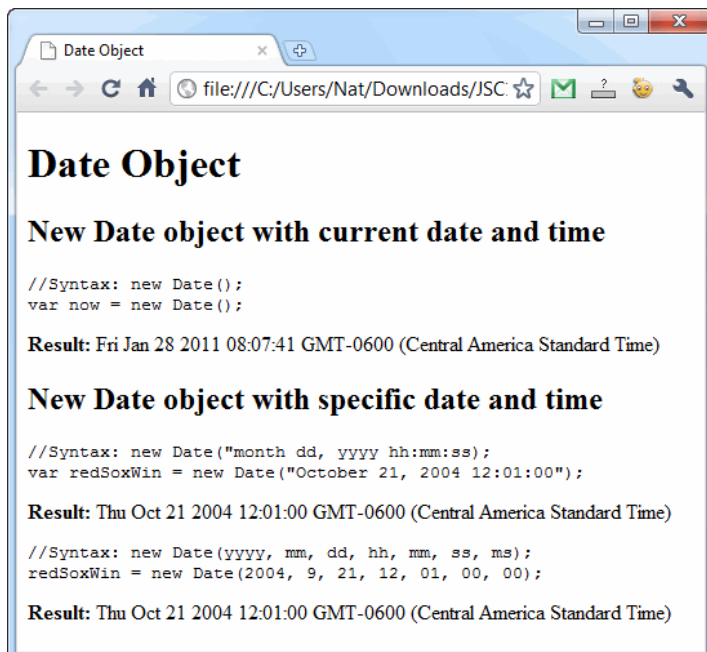
<body>
<h1>Date Object</h1>
<h2>New Date object with current date and time</h2>
<pre>
//Syntax: new Date();
var now = new Date();
</pre>
<strong>Result:</strong>
<script type="text/javascript">
  var now = new Date();
  document.write(now);
</script>

<h2>New Date object with specific date and time</h2>
<pre>
//Syntax: new Date("month dd, yyyy hh:mm:ss");
var redSoxWin = new Date("October 21, 2004 12:01:00");
</pre>
<strong>Result:</strong>
<script type="text/javascript">
  var redSoxWin = new Date("October 21, 2004 12:01:00");
  document.write(redSoxWin);
</script>

<pre>
//Syntax: new Date(yyyy, mm, dd, hh, mm, ss, ms);
redSoxWin = new Date(2004, 9, 21, 12, 01, 00, 00);
</pre>
<strong>Result:</strong>
<script type="text/javascript">
  redSoxWin = new Date(2004, 9, 21, 12, 01, 00, 00);
  document.write(redSoxWin);
</script>
</body>
</html>

```

This page is shown in a browser below.



A few things to note:

1. To create a `Date` object containing the current date and time, the `Date()` constructor takes no arguments.
2. When passing the date as a string to the `Date()` constructor, the time portion is optional. If it is not included, it defaults to 00:00:00. Also, other date formats are acceptable (e.g, "4/14/2011" and "14-04-2011").
3. When passing date parts to the `Date()` constructor, `dd`, `hh`, `mm`, `ss`, and `ms` are all optional. The default of each is 0.
4. Months are numbered from 0 (January) to 11 (December). In the example above, 9 represents October.

Some common date methods are shown below. In all the examples, the variable `rightNow` contains "Thu Apr 14 00:23:54:650 EDT 2011".

#### Common Date Methods

Method	Description
<code>getDate()</code>	Returns the day of the month (1-31).
<code>getDay()</code>	Returns the day of the week as a number (0-6, 0=Sunday, 6=Saturday).
<code>getMonth()</code>	Returns the month as a number (0-11, 0=January, 11=December).
<code>getFullYear()</code>	Returns the four-digit year.
<code>getHours()</code>	Returns the hour (0-23).
<code>getMinutes()</code>	Returns the minute (0-59).
<code>getSeconds()</code>	Returns the second (0-59).
<code>getMilliseconds()</code>	Returns the millisecond (0-999).

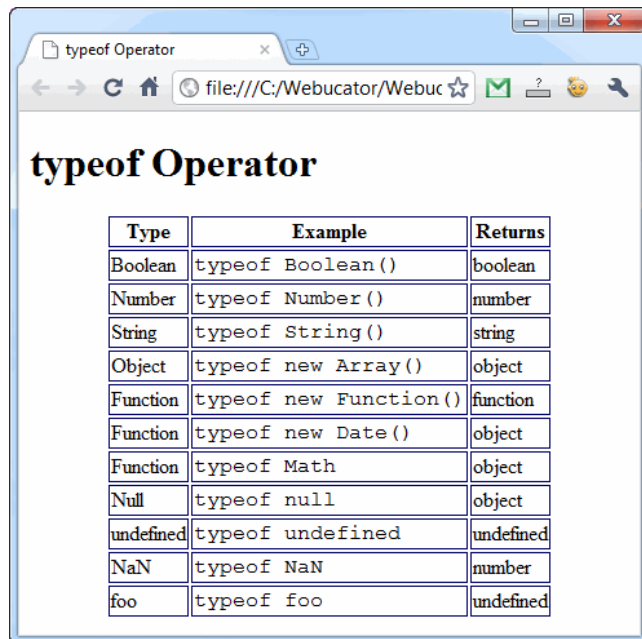


<code>getTime()</code>	Returns the number of milliseconds since midnight January 1, 1970.
<code>getTimezoneOffset()</code>	Returns the time difference in minutes between the user's computer and GMT.
<code>toLocaleString()</code>	Returns the Date object as a string.
<code>toGMTString()</code>	Returns the Date object as a string in GMT timezone.

You can see these examples in a browser by opening [BuiltInObjects/Demos/DateMethods.html](#).

## Lesson 1, Activity 9: `typeof` Operator

The `typeof` operator is used to find out the data type of a variable. As we learned earlier in the course, we can have variables that hold String values, Numeric values, and more. Using the `typeof` operators lets us identify the type of data the variable references. The screenshot below (of [BuiltInObjects/Demos/typeof.html](file:///C:/Webucator/Webucator/BuiltInObjects/Demos/typeof.html)) shows what the `typeof` operator returns for different data types



Type	Example	Returns
Boolean	<code>typeof Boolean()</code>	boolean
Number	<code>typeof Number()</code>	number
String	<code>typeof String()</code>	string
Object	<code>typeof new Array()</code>	object
Function	<code>typeof new Function()</code>	function
Function	<code>typeof new Date()</code>	object
Function	<code>typeof Math</code>	object
Null	<code>typeof null</code>	object
undefined	<code>typeof undefined</code>	undefined
NaN	<code>typeof NaN</code>	number
foo	<code>typeof foo</code>	undefined

## Lesson 1, Activity      **Helper Functions**

11:

Some languages have functions that return the month as a string. JavaScript doesn't have such a built-in function. The sample below shows a user-defined "helper" function that handles this and how the `getMonth()` method of a `Date` object can be used to get the month.

### Code Sample:

---

#### [BuiltInObjects/Demos/MonthAsString.html](#)

```

---- C O D E   O M I T T E D ----

<script type="text/javascript">
function monthAsString(num){
    var months = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"];
    return months[num-1];
}

function enterMonth(){
    var userMonth = prompt("What month were you born?", "");
    alert("You were born in " + monthAsString(userMonth) + ".");
}

function getCurrentMonth(){
    var today = new Date();
    alert(monthAsString(today.getMonth()+1));
}
</script>
---- C O D E   O M I T T E D ----

```

## Lesson 1, Activity 12: Returning the Day of the Week as a String

Duration: 15 to 25 minutes.

In this exercise, you will create a function that returns the day of the week as a string.

1. Open [BuiltInObjects/Exercises/DateUDFs.html](#) for editing.
2. Write a `dayAsString()` function that returns the day of the week as a string.
3. Write an `enterDay()` function that prompts the user for the day of the week and then alerts the string value of that day by calling the `dayAsString()` function.
4. Write a `getCurrentDay()` function that alerts today's actual day of the week according to the user's machine.
5. Add a "CHOOSE DAY" button that calls the `enterDay()` function.
6. Add a "GET CURRENT DAY" button that calls the `getCurrentDay()` function.
7. Test your solution in a browser.

### Solution:

#### [BuiltInObjects/Solutions/DateUDFs.html](#)

```

---- C O D E   O M I T T E D ----

<script type="text/javascript">
---- C O D E   O M I T T E D ----
function dayAsString(num){
    var days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];
    return days[num-1];
}

---- C O D E   O M I T T E D ----

function getCurrentMonth(){
    var today = new Date();
    alert(monthAsString(today.getMonth()+1));
}

function enterDay(){
    var userDay = prompt("What day of the week is it?", "");
    alert("Today is " + dayAsString(userDay) + ".");
}

function getCurrentDay(){
    var today = new Date();
    alert(dayAsString(today.getDay()+1));
}
</script>

---- C O D E   O M I T T E D ----

<input type="button" value="CHOOSE MONTH" onclick="enterMonth();">
<input type="button" value="GET CURRENT MONTH" onclick="getCurrentMonth();"><br>
<input type="button" value="CHOOSE DAY" onclick="enterDay();">
<input type="button" value="GET CURRENT DAY" onclick="getCurrentDay();">

---- C O D E   O M I T T E D ----

```