Javascript, first introduced by Netscape, changed the static fate of the HTML web pages. Before Javascript came into existence, HTML pages were just used to render static content. Inserting Javascript to a web page, can give it a significant degree of interactivity and functionality. It lets you control the behavior of the web browser and how the elements of the web page are displayed. Most web browsers have a built in javascript interpreter. When a browser downloads an html file that contains javascript, the javascript interpreter reacts to any script.   
  
  
Javascript is an object oriented programming language. It supports the concept of objects in the form of attributes. If an object attribute consists of function, then it is a method of that object, or if an object attribute consists of values, then it is a property of that object. For example,

var status=document.readyState;

readyState is a property of the document object which can contain values such as "unintialized","loading","interactive","complete" whereas,

document.write("Hello World");

write() is a method of the document object that writes the content "Hello World" on the web page. There are several Javascript built-in objects such as,

* Number
* String
* RegExp
* Array
* Math
* Date
* Boolean

Each of the above objects hold several built-in functions to perform object related functionality. Apart from these methods, Javascript provides few predefined functions which do not stick to a particular object type but are global.These global built-in functions are explained below with examples.

isNaN()

isNaN() method determines whether value of a variable is a legal number or not.

document.write(isNan(0));

document.write(isNan("Javascript"));

document.write(isNan(-2.45));

document.write(isNan("77"));

document.write(isNan("2012/4/8"));

Results:

**false**  
**true**  
**false**  
**false**  
**true**

Note: There is a property called NaN of the Number Object which can be used to assign a variable with 'not a number' value.

var year= Number.NaN;

document.write(year);

Result:

**NaN**

isFinite()

As the name indicates, this function is used to find whether a number is a finite legal number.

document.write(isFinite("5678"));

document.write(isFinite("ABCD"));

document.write(isFinite("123\_456"));

Result:

**true**

**false**

**false**

eval()

eval() is used to execute Javascript source code. It evaluates or executes the argument passed to it and generates output.

eval("var number=2;number=number+2;document.write(number)");

Result:

**4**

Note: Use of eval method is not advised for the following reasons,  
\* security risk  
\* debugging may be difficult  
\* may result at resource hog since each invocation of eval() creates a new instance of Javascript interpreter.

Number()

Number() method takes an object as an argument and converts it to the corresponding number value. If the object passed cannot be converted to a number, that is if the object is not in a format to be represented as a number, then it returns NaN(not a number).

var obj1=new String("123");

var obj2=new Boolean("false");

var obj3=new Boolean("true");

var obj4=new Date();

var obj5=new String("9191 9999");

document.write(Number(obj1));

document.write(Number(obj2));

document.write(Number(obj3));

document.write(Number(obj4));

document.write(Number(obj5));

Result:

**123**

**0**

**1**

**1342720050291**

**NaN**

Note: For date object it returns the number of milliseconds since January 1,1970 UTC.

String()

String() function converts the object argument passed to it to a string value.

var obj1=new Boolean(0);

var obj2=new Boolean(1);

var obj3=new Date();

document.write(String(obj1));

document.write(String(obj2));

document.write(String(obj3));

Result:

**false**

**true**

**Thu Jul 19 2012 23:28:08 GMT+0530 (India Standard Time)\**

parseInt()

parseInt() function takes string as a parameter and converts it to integer.

document.write(parseInt("50"));

document.write(parseInt("77 days"));

document.write(parseInt("this is 7"));

Result:

**50**

**77**

**NaN**

An optional radix parameter can also be used to specify the number system to be used to parse the string argument. For example,

document.write(parseInt("10",16));

Result:  
16

16 is the radix value passed, which means the string should be converted from hexadecimal to decimal value. If the radix is 8, then the string should be converted from octal to decimal value. The radix can be any number ranging from 2 to 36 that represents a numeral system.

If the radix parameter is not specified, then Javascript

* Assumes radix to be 16(hexadecimal), if the string starts with "0x"
* Assumes radix to be 8(octal), if the string starts with 0
* Assumes radix to be 10(decimal), if the string starts with any other number.

parseFloat()

parseFloat() function takes a string as parameter and  parses it to a floating point number.

document.write(parseFloat("10.33"));

document.write(parseFloat("15 66 75"));

document.write(parseFloat("this is 77"));

document.write(pareFloat("    77    "));

Result:

**10.33**

**15**

**NaN**

**77**

Note: This function allows leading and trailing spaces. If the first character in the string is not a number, then it returns NaN. If the string has more than one set of number separated by delimiters such as spaces, semicolons,commas then it returns only the first set of number before the first delimiter.

escape()

escape() function encodes the string passed to it so that it can be used across any network, say for example in query strings.

document.write(escape("testing escape function!!"));

**Result:**

**testing%20escape%20function%21%21**

escape() function leaves digits,latin letters and the characters + - \* / . \_ @ unchanged and replaces all other characters with ASCII code of the original character preceded by % symbol. Use unescape() method to decode the encoded strings.

Hope this blog helped you in identifying some useful javascript built-in functions. There are many other useful javascript functions for each built-in javascript object type. In case you want to learn about each of those functions, they are clearly explained [here](http://www.tutorialspoint.com/javascript/javascript_builtin_functions.htm).  
  
  
***Hope this post helped you in understanding some of the useful javascript built-in functions. Please leave your comments and queries about this post in the comment sections in order for me to improve my writing skills and to showcase more useful posts.*** 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **The Message Box** | |  |      |  | | --- | | **A Simple Message Box** | |  |   To help with script programming, the JavaScript language provides many functions already built-in the browser. These functions are readily available and you can use them as you see fit.  A message box is a rectangular window used to display a message to the user. To display a message box, use the **alert()** function. This function is used to display a message box that is equipped with a button with a caption as OK. As long as the message box displays, the user must close it in order to return to the browser, meaning the browser is not accessible while the message box is available.  The syntax of the **alert()** function is:  alert(*WhatToDisplay*);  The argument of this function can be a number. For example, to display the number 12 when the user clicks a button, you can use code as follows:   |  | | --- | | <Script Language="JavaScript">  function displayNumber()  {  alert(12);  }  </Script>  <input type="button" value="Show Me" onClick="displayNumber()"> | | http://www.functionx.com/javascript/images/preview7a.gif |   The above program could also have been written as:  <input type="button" value="Show Me" onClick="alert(12)">  The item to display on the message box can also be a word, a sentence, or a paragraph. In this case, provide the argument between double-quotes. Here is an example:   |  | | --- | | <Script Language="JavaScript">  function displayReminder()  {  alert("Please make sure you submit your time sheet");  }  </Script>  <input type="button" value="Reminder" onClick="displayReminder()"> | | http://www.functionx.com/javascript/images/preview7b.gif |   If you want to display the message on various lines, end each line with the new line constant represented as "\n". Here is an example:  <Script Language="JavaScript">  function displayReminder()  {  alert("Please make sure you submit your time sheet\nto Human Resource today");  }  </Script>  <input type="button" value="Reminder" onClick="displayReminder()">  The item to display can also be an expression or a combination of items.   |  | | --- | | **The Message Box With Options** | |  |  |  |  |  | | --- | --- | --- | | A message box created with the **alert()** function is used only to display a message to the user and it is equipped with the OK button. In some cases you may want to ask a specific question to the user and expect the user to make a decision. To do this, the JavaScript language allows you to create a message equipped with two buttons, namely OK and Cancel.  To display a message box with more than one button, use the **confirm()** function. Its syntax is:  confirm(QuestionToAsk);  In the strict sense, the **confirm()** uses the same argument terminology as the **alert()** function. This means that you can use it to display a number, a word, a sentence, or a paragraph. To make it effective, the **confirm()** function is better used to ask/display a question to the user. The user makes a decision by clicking OK or Cancel. If the user clicks OK, the function returns **true**. If the user clicks Cancel, the function returns **false**.  Here is an example:   |  | | --- | | <Script Language="JavaScript">  function applicationReminder()  {  confirm("The application you filled out is not complete.\nDo you still wanto to quit?");  }  </Script>  <input type="button" value="Reminder" onClick="applicationReminder()"> | | http://www.functionx.com/javascript/images/preview8.gif | | |  |

|  |
| --- |
| **Conversion Functions** |
|  |

|  |
| --- |
| **Integer Conversion** |
|  |

|  |
| --- |
| When the user types a value in an edit box or selects a value from a text-based control, the value is considered a simple group of words and the browser may not be able to identify or categorize what exists in a text-based control. The **parsetInt()** function is used to analyze what is in a text-based control. Its syntax is:  parseInt(*Value*)  When calling this function, you must provide the argument. The interpreter will examine it from left to right. If the value is a natural number, **parseInt()** returns that number. For example, if you call it as  parseInt("124")  the function will return 124. If the whole value is not a (natural) number, the interpreter will consider the numbers from left to right. If the first character of the argument is not a recognizable digit, the function would return a value known as **NaN** (not a number). If the first character from left is a digit, the interpreter will continue examining the other characters. When it finds a character that cannot be considered a digit, it would stop. It would then convert to number the digits found on the left side of the first non-digit character. For example, if you call it as  parseInt("216PLL")  the function would return 216. |

|  |
| --- |
| **Practical Learning: Converting a String to an Integer** |
|  |

|  |  |  |
| --- | --- | --- |
| 1. Start your text editor and type the following:  |  | | --- | | <html>  <head>  <title>CD Publisher</title>  </head>  <body>  <h1>CD Publisher</h1>  <form name="frmOrder">  <table border="0" width="344">  <tr>  <td width="231">Number of CDs ordered:</td>  <td width="93"><input type="text" name="txtQuantity" size="10" value="0"></td>  <td width="1"><input type="button" value="Calculate" name="btnCalculate"></td>  </tr>  <tr>  <td width="211" colspan="3">At $2.95 for each CD</td>  </tr>  <tr>  <td>Order Total:</td>  <td width="70"><input type="text" name="txtOrderTotal" size="10"></td>  <td width="70"><input type="reset" value="Reset" name="btnReset"></td>  </tr>  </table>  </form>  </body>  </html> |  1. Save the file as CDPublisher.htm and preview it in your browser 2. After previewing the page, return to your text editor 3. To use the parseInt() function, change the file as follows:  |  | | --- | | <html>  <head>  <SCRIPT language="JavaScript">  function calculateOrder()  {  var quantity, totalOrder;  quantity = parseInt(document.frmOrder.txtQuantity.value);  totalOrder = quantity \* 2.95;  document.frmOrder.txtOrderTotal.value = totalOrder;  }  </SCRIPT>  <title>CD Publisher</title>  </head>  <body>  <h1>CD Publisher</h1>  <form name="frmOrder">  <table border="0" width="344">  <tr>  <td width="231">Number of CDs ordered:</td>  <td width="93"><input type="text" name="txtQuantity" size="10" value="0"></td>  <td width="1"><input type="button" value="Calculate" name="btnCalculate" onClick="calculateOrder()"></td>  </tr>  <tr>  <td width="211" colspan="3">At $2.95 for each CD</td>  </tr>  <tr>  <td>Order Total:</td>  <td width="70"><input type="text" name="txtOrderTotal" size="10"></td>  <td width="70"><input type="reset" value="Reset" name="btnReset"></td>  </tr>  </table>  </form>  </body>  </html> |  1. Save the file. Return to the browser and refresh it 2. Enter a natural number in the top text box and click Calculate   http://www.functionx.com/javascript/images/operations2.gif 3. After preview |