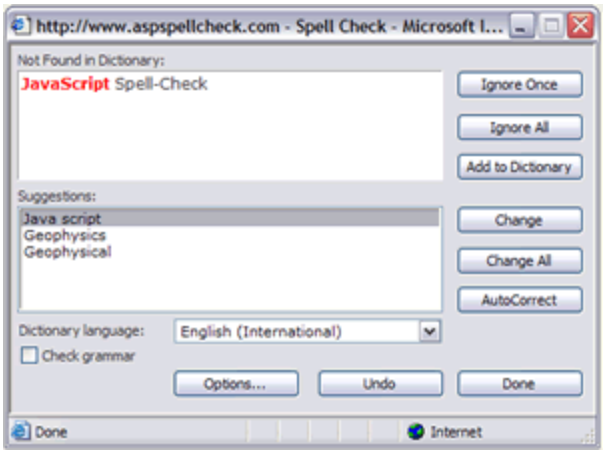
Introduction

Javascript Spell Check is a complete spell checking solution for JavaScript and Ajax. It has 3 main features:

Use 1: The Spell Check Window



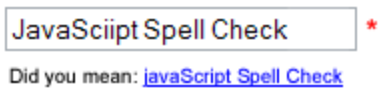
The JavaScript API allows you toautomatically add Microsoft Word style spell-checking to any HTML form field, string, HTML Element or iFrame.

This feature can be called using only a few lines of JavaScript.

The spell checker dialog supports custom user dictionaries , grammar checking , and international support for over 10 languages ; making this possibly the richest online spellchecking component available.

Find out more...

Use 2: AJAX Spell Check



Ajax (Asynchronous JavaScript And XML), is a Web development technique for creating highly interactive web applications using JavaScript.

The Ajax spell-checker allows you to spell check individual words or entire forms at any time. It even provides ranked spelling suggestions.

Find out more...

Use 3: Spell Check Function

Our new JavaScript spell-checking function allows you to spellcheck words and retrieve suggestions directly in JavaScript. This is an inline function that can be integrated in to your javascript function or application.

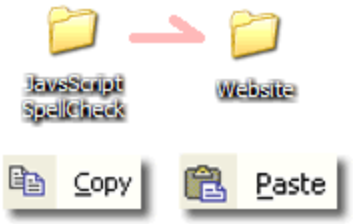
JavaScriptSpellCheck.spellCheck('Any String')

This function returns *true* if spelling is correct. If the input is spelled incorrectly, an ordered *array of spelling suggestions* is returned.

Find out more...

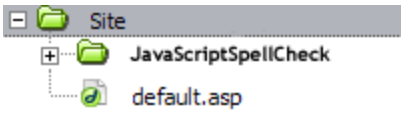
Installing JavaScript SpellCheck

JavaScript SpellCheck installation is simple. There are no installers to run. This makes it very easy to use, even on most virtual hosting packages.



Step 1: Installing Javascript SpellCheck

Inside the ZIP archive you downloaded there is a folder called ***JavaScriptSpellCheck***.
Copy this directory to the root folder of your website (normally where the home page is located).

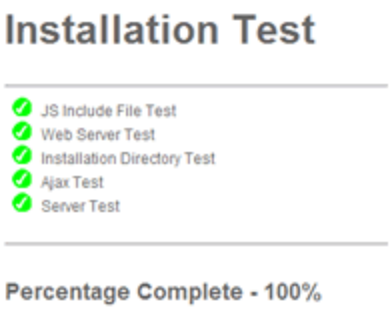


Step 2: Uploading Javascript Spell Check.

JavaScript Spell Check will only work when uploaded to Microsoft Web hosting with ASP Enabled. This covers most modern Windows Web Servers.

If you do not work with ASP on your own computer, don't worry - you can continue to work as normal. The spell-checker will be disabled until the website is uploaded to your live server.

Step 3: Test Your Installation



You can test and troubleshoot you installation by running the files **Sample.htm** and **TestInstall.htm** in your **JavaScriptSpellCheck**

directory on your web server .

Learn More...

Trouble Shooting your Installation

You can test troubleshoot your JavaScript SpellCheck installation using tools provided. The following scripts are located in the ***JavaScriptSpellCheck*** directory you installed .

Step 1. Test the Software in Action

Run the web page **Sample.htm** in your web browser.
You should access this page the same way a user would - via a web server.
The address bar should say *http://* or *https://* and not *file://* at the start.

You can test and play with JavaScript SpellCheck's basic features here.

Step 2. If you have problems, trouble-shoot your installation.

Run the web page **TestInstall.htm** in your web browser. It will verify the status of your installation, and prescribe fixes if any problems arise.

If you still have problems, you can contact us at support@javascriptspellcheck.com

Installing Dictionaries for JavaScript SpellCheck

You can install additional language dictionaries to JavaScript SpellCheck such as:

- English (Australia)
- English (Canada)
- English (International)
- English (UK)
- English (USA)
- French
- German
- Italian
- Dutch
- Portuguese
- Spanish
- Swedish
- Danish

You can download them from www.javascriptspellcheck.com/downloads.asp

To install these dictionaries simply place them in the "***JavaScriptSpellCheck/Dictionaries***" directory within your site. To use them, read up on the languages property.

You can add a custom dictionary of words specific to your website too.

The Custom Dictionary

The Custom dictionary allows you to add a list of custom words to your spell checker's vocabulary.
This is useful for adding words specific to a business, application or website.

Modifying the Custom Dictionary

The custom dictionary file is "**custom.txt"** within the ***JavaScriptSpellCheck/Dictionaries*** directory of your website.

To add words to your custom dictionary, simply place a list of up to 100,000 words, each on separate lines into this file.

For significantly improved performance, please sort these words in alphabetical (ANSI code) order. If you do not have a tool to do this, then EditPlus is an excellent choice. For performance issues, it is best to remove any spaces from the file.

Example contents of custom.txt:

```
Aaron
Jacob
James
Julie
Steve
Cybercon
JavaScript
```

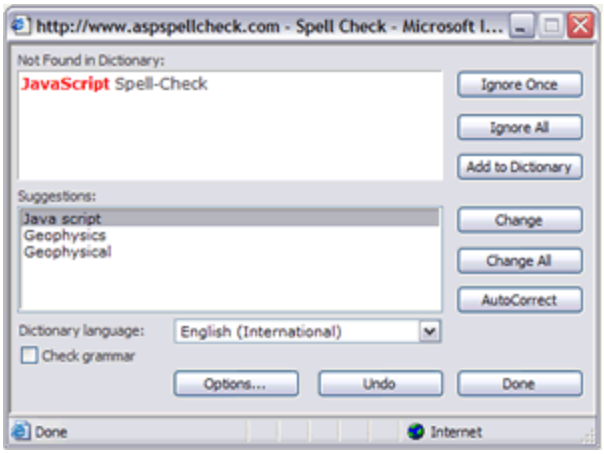
Common Typographic Errors

The Spell Checker has an awareness of common typos. This is located in a file called "**CommonTypos.txt"** within the International directory of your installation. This file can be amended by adding / removing entries (per language) in the format:

typo--->suggestion

The entries must be put in in exact alphabetical ASCII character code order.

Use 1: The Spell Check Window



The spell check window allows us to spellcheck any form field, string or HTML element in JavaScript.

It looks and works much like the Spell-Checker feature in MS Word.

The dialog is feature rich - including grammar checking, personal dictionaries , international support and much more.

What's more you can do this is a few quick steps!

Step 1. Install javascript Spell Check

This only takes a few minutes, Find out how .

Step 2. Include the JavaScript SpellCheck file in your web page

This code should normally included in the head section of your web page.

```
<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language= "javascript1.1"></script>
```

Step 3.Call the spell check window

```
<script>
var oSpell = new JavaScriptSpellCheck();
oSpell.spellCheckWindow('textarea1')
</script>
```

To spellcheck a Form Field or HTML Element:

```
oSpell.spellCheckWindow('textarea1')
```

or

```
oSpell.spellCheckWindow('document.getElementById("textarea1")')
```

To spellcheck Multiple Elements:

```
oSpell.spellCheckWindow('textarea1','textarea2')
```

To spellcheck a variable:

```
var strMyString='Any string or Variable'
oSpell.spellCheckWindow(strMyString)
```

To spellcheck an Iframe

```
oSpell.spellCheckWindow('iframe1.document.body.innerHTML')  // Where iframe1 is the id fo the iFrame
```

(Note that the above notation has been developed to avoid cross platform bugs. Read More..)

Advanced Settings

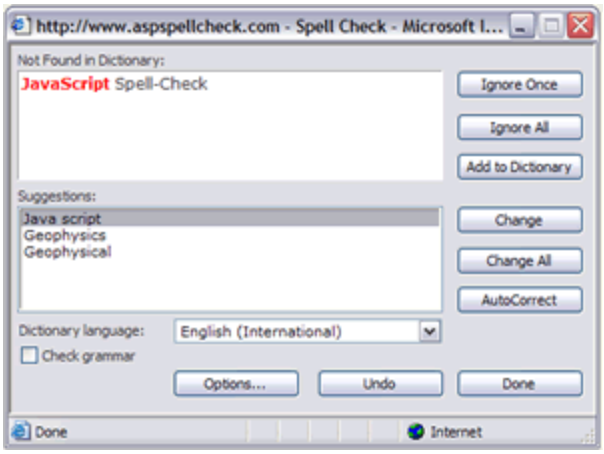
If you need to set up the spellchecker such as language settings - please refer to the Advanced Reference; .

Example

In this example the spell checking window is opened when a button is pressed.

```
<textarea id="textarea1" name="textarea1">Sentance to Spell Check</textarea>
<input type="button" value= "SpellCheck" onClick="spellCheck()">

<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language= "javascript1.1"></script>

<script>

function spellCheck(){
var oSpell = new JavaScriptSpellCheck();

oSpell.spellCheckWindow('textarea1')
}

</script>
```

Callback after Spell Checking



After you have spell Checking using the SpellCheck window, you may want to perform an operation in JavaScript such as:

- Submitting a form
- Alerting the user with a message
- Any other script you may wish to call

This is very easy; simply set the JavaScript SpellCheck **callBack** function.

JavaScriptSpellCheck.callBack=function

Example 1: Alerting the user

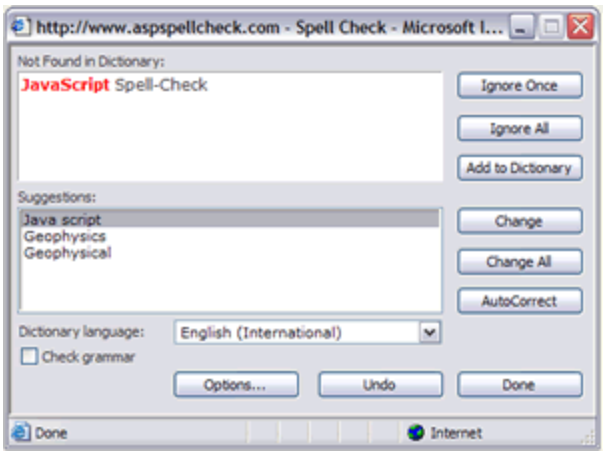
```
<textarea id="textarea1" name="textarea1">Sentance to Spell Check</textarea>
<input type="button" value="Submit" onClick="spellCheck()">

<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language="javascript1.1"></script>

<script>
function spellCheck()
{
    var oSpell = new JavaScriptSpellCheck();
   oSpell.callBack=myCallBack
   oSpell.spellCheckWindow('textarea1')
}
function myCallBack()
{
   alert('Spell Check is Complete')
}
</script>
```

Example 2: Submitting a Form

```
<form id="form1" name="form1">
<textarea id="textarea1" name="textarea1">Sentance to Spell Check</textarea>
<input type="button" value="Submit" onClick="spellCheck()">
</form>
```

```
<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language="javascript1.1"></script>

<script>
function spellCheck()
{
   var oSpell = new JavaScriptSpellCheck();
   oSpell.hideSummary=true;
   oSpell.callBack=function()
   {
      alert('Submitting Form')
      document.form1.submit()
   }
   oSpell.spellCheckWindow('textarea1')
}
</script>
```

Testing The Water



The **spellCheckWindowTest** function allows you to determine if a spellchecker window is needed **before** you start spell checking.

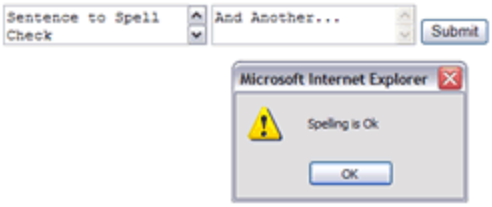
The function will tell you if there are any spelling mistakes in your fields, so you don't spellcheck unnecessarily. It returns true or false, and its inputs are conveniently the same as for the **spellCheckWindow** function.

spellCheckWindowTest ('field1','field2") returns *true* or *false*.

Example

```
<textarea id="textarea1" name="textarea1">Sentence to Spell Check</textarea>
<input type="button" value="Submit" onClick="spellCheck()">

<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language="javascript1.1">
</script>

<script>
function spellCheck()
{
   var oSpell= new JavaScriptSpellCheck();
   if (oSpell.spellCheckWindowTest('textarea1'))
   {
      alert('All Spelling is OK')
   }
   else
   {
      oSpell.spellCheckWindow('textarea1')
   }

}
</script>
```

Advanced Example

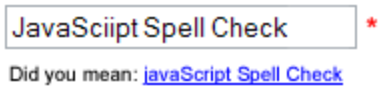


In this example we use spellCheckWindowTest and callBack

```
<textarea id="textarea1" name= "textarea1">Sentance to Spell Check</textarea>
<textarea id="textarea2" name="textarea2">Sentance to Spell Check</textarea>
<input type="button" value="Submit" onClick="spellCheck()">

<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language="javascript1.1"></script>
<script>
function spellCheck()
{
   var oSpell = new JavaScriptSpellCheck();
   oSpell.callBack=function()
   {
      alert ('Spellcheck is Complete')
   }
   if ( ! oSpell.spellCheckWindowTest('textarea1','textarea2') ){
   oSpell.spellCheckWindow('textarea1','textarea2')}else{
   alert('Spelling is Fine' )
}
</script>
```

Use 2: Ajax Spell Check

Ajax (Asynchronous JavaScript And XML), is a Web development technique for creating highly interactive web applications using JavaScript.



Ajax Spell Check allows you to easily perform spell checking operations in JavaScript in the background as a web page runs.

With JavaScript SpellCheck all the work is done for you, you can use AJAX for spell checking using just a few lines of code.

Step 1. Install javascript Spell Check

This only takes a few minutes, Find out how.

Step 2. Include the JavaScript SpellCheck file in your web page

This code is normally included in head section of your web page.

```
<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language= "javascript1.1"></script>
```

Step 3. Set up an Ajax callBack function

```
<script>
var oSpell = new JavaScriptSpellCheck();
oSpell.ajaxCallBack=function(result)
{
   alert (result)
}
</script>
```

Step 4. Call the Ajax spellchecker

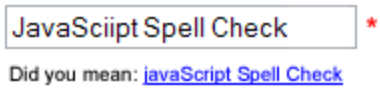
```
<script>
function spellCheck()
{
   oSpell.ajaxSpellCheck('test')
}
spellCheck()
</script>
```

Putting It All Together

```
<script src="/JavascriptSpellCheck/Include.js" type= "text/javascript" language= "javascript1.1"></script>

<input type="text" value= "Type Here" id="input1" name="input1" >
<input type="button" value= "SpellCheck" onClick="spellCheck()">

<script>
var oSpell = new JavaScriptSpellCheck();
oSpell.ajaxCallBack=function(result)
{
   if (result==true)
   {
      alert('spelling is OK')
   }
   else
   {
      var strAlert="Spelling Suggestion Array:\n"
      for ( i=0; i<result.length;i++)
      {
         strAlert+=(i+1)+") "+result[i]+"\n"
      }
      alert (strAlert)
   }
}
function spellCheck()
{
   oSpell.ajaxSpellCheck(document.getElementById('input1').value)
}
</script>
```

If you need to set up the spellchecker such as language settings - please refer to the Advanced Reference .

Testing for Ajax compatibility



Ajax is an emerging technology - and its not compatible with all web browsers.

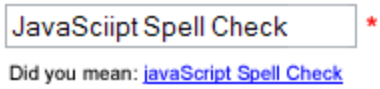
To avoid embarrassing mistakes - JavaScript SpellCheck can tell you weather the user's web browser is Ajax compatible.

ajaxEnabled returns *true* or *false*

Example

```
<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language= "javascript1.1"></script>
<script>
var oSpell = new JavaScriptSpellCheck();
if (oSpell.ajaxEnabled)
{
   alert('Ajax is Available')
}
else
{
   alert('Ajax is not Available')
}
</script>
```

Example




```
<html>
<head>
<title>JavaScript SpellCheck Ajax Example</title>
<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language="javascript1.1"></script>
</head>
<body>
<input name="input1" id="input1" value="Type Here..." onChange="ajaxTestSpelling()"/>
<span id="star1" style="color:#FF0000; display:none">*</span>
<div id="message1" style="font-size:small"></div>
<script type="text/javascript" language="javascript1.1">
var oSpell = new JavaScriptSpellCheck();
var objStar=document.getElementById("star1")
var objMessage=document.getElementById("message1")
oSpell .ajaxCallBack=function(result)
{
    if (result==true)
    {
        objStar.style.display='none'
        objMessage.innerHTML=''
    }
    else
    {
        objStar.style.display='inline'
        objMessage.innerHTML="Did You Mean : <a
href='javascript:updateInput1(\""+result[0]+"\")'>"+result[0]+"</a>"
    }
}
function updateInput1(strValue)
{
    document.getElementById("input1").value=strValue
    objStar.style.display='none'
    objMessage.innerHTML=''
}
function ajaxTestSpelling()
{
    var strInput=document.getElementById("input1").value
    oSpell.ajaxSpellCheck(strInput)
}
</script>
</body>
</html>
```

Use 3: The SpellCheck Function

The **spellCheck** Function allows you to perform spellchecking in your own Javascript functions and applications.

Its easy to use.

```
oSpell.spellCheck('ooooops')    // returns false
```

The result is ***true*** if the input is spelled correctly. If the input is not spelled correctly, the result is an ***Array*** of ranked spelling suggestions

Step 1. Insall javascript Spell Check

This only takes a few minutes, Find out how.

Step 2. Include the JavaScript SpellCheck file in your web page.

This code is normally included in head section of your webpage.

```
<script src="/JavascriptSpellCheck/Include.js" type="text/javascript" language= "javascript1.1"></script>
```

Step 3. Check spelling

```
<script>
var oSpell = new JavaScriptSpellCheck();
var result= oSpell.spellCheck('my input')
alert(result);
</script>
```

If you need to set up the spellchecker such as language settings - please refer to the Advanced Reference .

Advanced Settings

This section gives a full technical documentation for JavaScript SpellCheck

Basics

To use the *JavascriptSpellCheck* class first include the ***Include.js*** file in your ***JavascriptSpellCheck*** Directory.

```
<script src="/JavascriptSpellCheck/Include.js" type="text/javascript"
language="javascript1.1"></script>
```

Then create a new instance of the JavascriptSpellCheck object.

<script>
var oSpell = new JavaScriptSpellCheck();
...

Properties

- setupPath
- languages
- windowLanguage
- hideSummary
- externalCSS
- caseSensitive
- checkGramar
- ignoreAllCaps
- ignoreWebAddresses
- ignoreNumbers
- newSentanceOnEachNewLine
- useServerSession
- ajaxEnabled (read only)

Methods (functions)

- spellCheckWindow
- spellCheckWindowTest
- spellCheck
- ajaxSpellCheck

Event Handlers (CallBack Functions)

- callBack
- ajaxCallBack

Example


```
<script src="/JavascriptSpellCheck/Include.js" type= "text/javascript" language= "javascript1.1">
</script>
<input type="text" value= "Type Here" id="input1" name="input1" >
<input type="button" value= "SpellCheck" onClick= "spellCheck()">
<script>
function spellCheck()
{
   var oSpell = new JavaScriptSpellCheck();
   strInput=document.getElementById("input1").value
   var result= oSpell.spellCheck(strInput);
   if (result==true)
   {
      alert('Spelling is OK')
   }
   else
   {
      var strAlert="Spelling Suggestion Array:\n"
      for ( i=0; i<result.length;i++)
      {
         strAlert+=(i+1)+") "+result[i]+"\n"
      }
      alert (strAlert)
   }
}
</script>
```

setupPath

This property is used to set the full path to the JavaScriptSpellCheck directory within your website. It should be used as an *absolute* path from the root of your website.

The default value of JavaScriptSpellCheck.setupPath is ***/JavaScriptSpellCheck/***

Example

```
oSpell.setupPath="/JavaScriptSpellCheck/"
```

languages

The ***languages*** property is used to decide which dictionary language(s) the spell-check engine will use.

The value will be the same as the name of any dictionary you have installed in the Dictionaries directory within JavaScriptSpellCheck (do not use the ".dic" at the end though).

- To choose a specific language, choose a dictionary name. **E.g. "English (USA)" etc.**
- To choose multiple dictionaries - use a comma separated list of dictionary names. **E.g. "English (Canada),Francais"**

Note: These dictionaries must, of course, be installed to be used.

You may use up to 10 dictionaries simultaneously!

The default value of languages is "*English (International)*"

Example

```
oSpell.languages="English (International)"
```

windowLanguage

This property is used to decide the *User Interface* language of the spell-checking window .

The possible values of windowLanguage are:

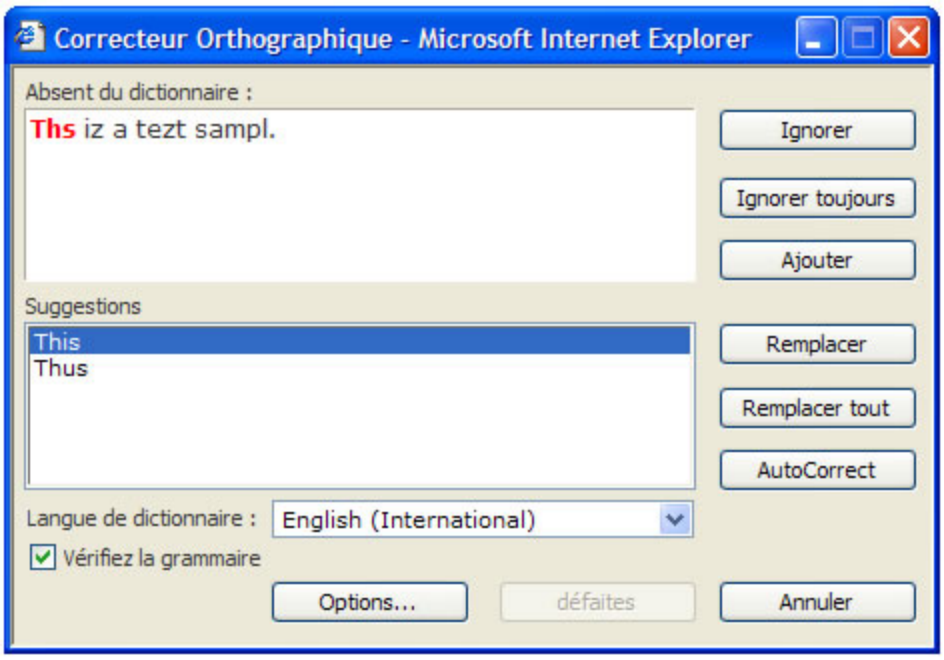
- "EN" - For English
- "ES" - For Spanish
- "DE" - For German
- "NL" - For Dutch
- "FR" - For French
- "IT" - For Italian
- "PT" - For Portuguese
- "NO" - For Norwegian
- "SV" - For Swedish
- "DK" - For Danish

The default value is "EN" for English.

Note that changing the Window Language *does not* automatically select the Dictionary Language used for spell-checking.

Example

oSpell.windowLanguage= **"FR"**



hideSummary

This property is used to disable the spellchecker window's summary screen. Possible values are *true* and *false* .

The default value of *hideSummary* is false.

Example

```
oSpell.hideSummary = false
```

externalCSS

This property is to attach an optional CSS stylesheet to the spell checking window .
If this property is not set, the default CSS will be resemble Windows® XP.

There is a sample stylesheet included in the Assets folder within ***JavascriptSpellCheck*** . To use this effectively, it is advisable to study the output HTML from the window.

Example

```
oSpell.externalCSS = "assets/sample.css"
```

caseSensitive

This Boolean property is used to decide if the spell-checker will act in a case sensitive manner. The default value is defined *true*.

Also see the checkGrammar property. If *checkGrammar* is set to true then the spellchecker can pickup irregularities in sentence casing even if caseSensitive is set to false.

Example

oSpell. caseSensitive= false

checkGrammar

This Boolean property is used to decide if the spell-check engine will find basic grammar mistakes such as repeated words, poor sentence casing etc.

The default value is *true.*

Example

If you want your spellchecker to ignore grammatical mistakes :

```
oSpell.checkGrammar = false
```

ignoreAllCaps

This Boolean property is used to decide if the spell-check engine will ignore words which are in ALL CAPITALS, such as acronyms.

The default value is *true.*

Example

```
oSpell.ignoreAllCaps = true
```

ignoreWebAddresses

This Boolean property is used to decide if the spell check engine will ignore words which resemble:

- web addresses (such as http://www.JavaScriptSpellCheck.com or "Ebay.com ")
- email addresses (such as " info@javascriptspellcheck.com ").

The default value is *true.*

Example

```
oSpell.ignoreWebAddresses = true
```

ignoreNumbers

This Boolean property is used to decide if the spell-check engine will ignore words containing numbers such as "$100mill" or "8Ball".

The default value is *true.*

Example

```
oSpell. ignoreNumbers = true
```

newSentenceOnEachNewLine

This Boolean property is used to decide if the spell-check engine will expect a new sentence after each carriage return (new line). This is useful to ensure proper grammar as per a word-processor, but may not be appropriate for more 'casual' web forms.

The default value is *f*alse.

Example

```
oSpell.newSentenceOnEachNewLine = true
```

useServerSession

This Boolean property is used to decide if the spell-check engine will use the IIS Session object to accelerate the dialog for each user.

Using the session will reduce the bandwidth used by the spellcheck window, but will use more server memory.

The default value is *true.*

Example

If you don't want your spellchecker to use the session:

```
oSpell.useServerSession = false
```

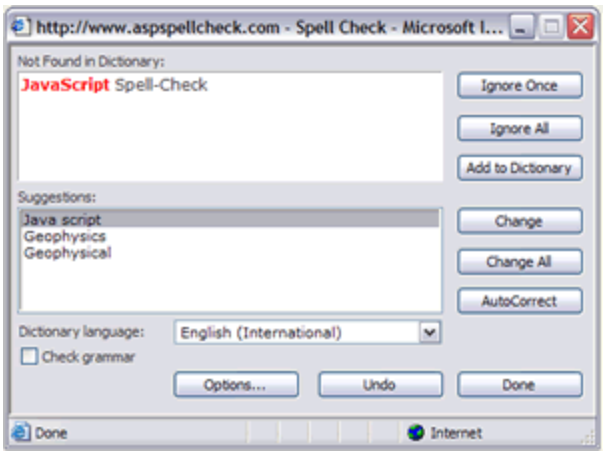
ajaxEnabled (read only)

The ajaxEnabled property allows you to determine if the client browser can use Ajax . Ajax is needed for the spellCheck, ajaxSpellCheck and spellCheckWindowTest methods.

The results is Boolean - either *true* or *false* .

Read More...

spellCheckWindow(strfield1 [,strfield2...])



The spellCheckWindow function opens a spellchecker dialog similar to that of MS Word.

Any number of fields, HTML elements or iFrames can be spell checked at one time. The strings or properties to be spell checked are added as a comma separated list of object paths in string format

Examples

```
oSpell.spellCheckWindow('textarea1')
oSpell.spellCheckWindow('document.getElementById("textarea1")')
oSpell.spellCheckWindow(''textarea1'',''textarea2'',"textfield1")
oSpell.spellCheckWindow('strMyString')
oSpell.spellCheckWindow(any_property_or_variable)
oSpell.spellCheckWindow('iframe1.document.body.innerHTML')
```

Read More...

Before opening the spellchecker window, you can double check for spelling mistakes in all of the the fields at once using the spellCheckWindowTest method.

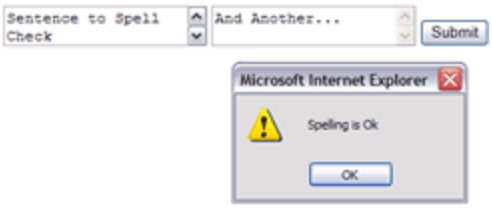
Note on Iframes

`oSpell.spellCheckWindow('iframe1.document.body.innerHTML')` // Where *iframe1* is the id of an iframe

This shorthand notation for accessing an iFrame (above) avoids some cross platform bugs. It replaces:

- `oSpell.spellCheckWindow('document.frames["iframe1"].document.body.innerHTML')` // in Internet Explorer
- `oSpell.spellCheckWindow('document.getElementById("iframe1").contentDocument.body.innerHTML')` // in W3C compliant browsers.

spellCheckWindowTest (strfield1 [,strfield2...])



The **spellCheckWindowTest** function allows you to determine if there are any spelling error in a group of fields.
The inputs of this function conveniently exacty match those of spellCheckWindow.

This is useful for form validation - where a spellchcker will only be opened if there are spelling errors in the form.

It returns *true* of *false* .

Example...

Note: the users web browser must have Ajax compatability for this function to work. Otherwise this function will return *null*.

spellCheck(strInput)

The spellCheck function allows you to check the spelling of any variable or property in JavaScript.

e.g. oSpell.spellCheck('Any string or variable you like')

The result is *true* if the input is spelled correctly. If the input is not spelled correctly, the result is an Array of ordered spelling suggestions.

Read More...

Note: the users web browser must have Ajax comparability for this function to work. Otherwise it will return *null*.

ajaxSpellCheck(strInput)

The ajaxSpellCheck function automates an Ajax interface with the spellchecker engine.

ajaxSpellCheck is useless until integrated with the ajaxCallBack event function.

Read More...

Note: the users web browser must have Ajax compatibility for this function to work. Otherwise it will return *null*.

Example

```
oSpell.ajaxCallBack=function(result){
 alert (result)
 }

oSpell.ajaxSpellCheck('Any string or variable you like')
```

callBack

After the SpellCheck Window is finished spell checking, a callBack event is triggered.

By setting the value of the callBack function, you can customize what happens at upon this even. The callBack function has no arguments.

Read More...

Example

```
function myCallBack() {
alert('SpellCheck is Complete"')
}

JSSpell.callBack=myCallBack
```

or

```
JSSpell.callBack= function () {
alert('You can develop your own custom actions')
}
```

ajaxCallBack

Ajax is an asynchronous technology. A request is sent for data, and then it is returned some time later. The code we use to handle this returned data is called a callBack function.

The arguments sent to this function are:

- *true* if spelling is ok
- *an array of ranked spelling suggestions* if the spelling is bad
- *null* in case of a server error or if Ajax is unavailable

Read More...

Example

```
oSpell.ajaxCallBack=function(result){
alert (result)
}

oSpell.ajaxSpellCheck('Any string or variable you like')
```

Changing Default Property Values

The default values of properties can be modified by most developers.

1. Back up the file /***JavaScriptSpellCheck/Inclue.js***
2. *Open this file*
3. *Look for and modify the following code:*

```
/* Default Values - May be amended */
   var setupPath = "/javascriptspellcheck/"
   var languages = "English (International)"
    var windowLanguage = "EN"
   var useServerSession = true
   var hideSummary = false
   var ignoreAllCaps = true
   var ignoreWebAddresses = true
   var ignoreNumbers = true
   var newSentanceOnEachNewLine = false
   var checkGramar = true
   var caseSensitive = true
   var externalCSS = ""
/* End of Default Values          */
```

If you are unsure of the meaning of any of these properties you can look them up here.

ASP.Net and JavaScript SpellCheck

JavaScript SpellCheck can easily be used as part of an ASP.Net Application.

Step 1. Install javascript Spell Check

This only takes a few minutes, Find out how .

Step 2. Develop Javascript

Develop JavaScript functions and HTML buttons for your custom spell checking activity

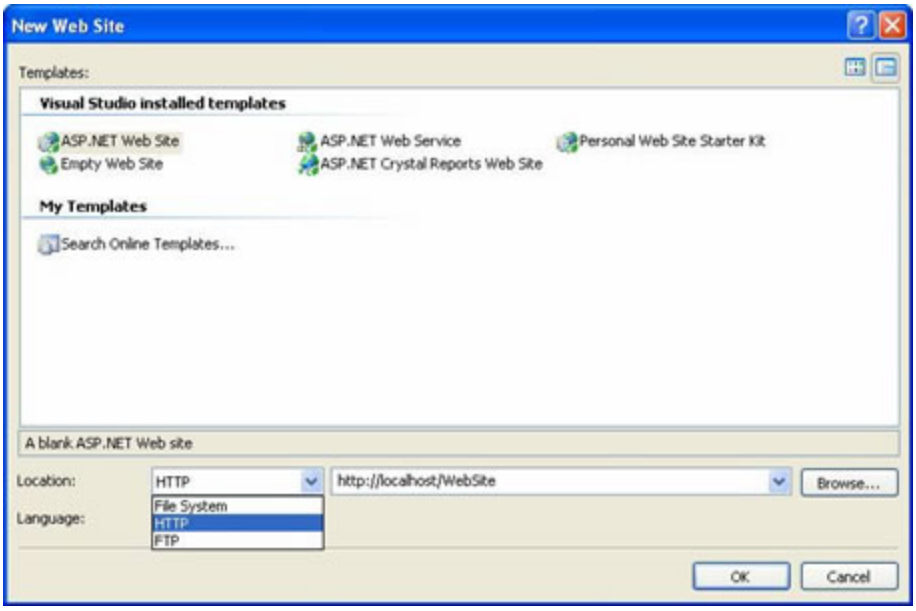
Step 3. Wrap the Javascript in an ASP.Net control

Create your own custom control ASP.Net containing the HTML and JavaScript your developed.

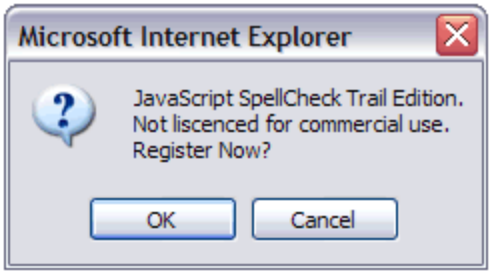
Visual Studio 2005 Compatability

Visual Studio 2005 uses its own web server (Cassini) and this doesn't allow asp pages to be rendered. It will appear that the spell checker doesn't work until uploaded to an independent web server.

To work around this, you must open your web project as an http: and place it inside of the IIS folder structure (**C:\inetpub\wwwroot**).



Free Trial & Registering



The free trial is not licensed for commercial or public use! A "Registration Reminder" may pop up from time to time. This is removed by purchasing and installing a full version of JavaScriptSpellCheck .

- To purchase a suitable license for JavaScriptSpellCheck please visit http://www.javascriptspellcheck.com/purchase.asp

- You will then receive a unique license key. Copy this into the file **/javascriptspellcheck/regkey.asp**

  E.G. <% LicenseKey = "MY-LICENSE-12345-KEY-0001" %>

License Agreement

JavaScriptSpellCheck.com hereby grants you a non-exclusive license to the JavaScriptSpellCheck Software Component (the Software). By downloading or using the Software, the Licensee agrees not to utilize the software in a manner which is disparaging to JavaScriptSpellCheck.com, and not to rent, lease or otherwise transfer rights to the Software.

The Licensee agrees that no attempt will be made by the Licensee or associated parties to translate, reverse engineer, modify, decompile, disassemble or distribute the Software.

License terms are offered on the following terms, as purchased. If no purchase or insufficient purchase has been made then the Free Trial License terms apply.

Free Trial License - Allows the use of the TRIAL VERSION of the software for private evaluation purposes only. The software should not be used for public or commercial use of any kind.

Single Website License - Allows the use of the software on any number of web / intranet pages belonging to a single domain.

Single Server License - This license allows the usage of the software on any number of web / intranet pages for any number of domains located on a single server.

Enterprise License - This license allows the usage of the software on any number of web / intranet pages for any number of domains located on any number of servers belonging to one company or organization.

UPGRADES
If a new release of the software is produced within 12 months from the date of purchase then you will be entitled to a free upgrade. This license does not grant you any right to any enhancement or update beyond the initial 12 month period, commencing from the date of purchase.

COPYRIGHT
Title, ownership rights, and intellectual property rights in and to the Software shall remain with JavaScriptSpellCheck.com. The Software is protected by the international copyright laws. Title, ownership rights, and intellectual property rights in and to the content accessed through the Software is the property of the applicable content owner and may be protected by applicable copyright or other law. This License gives you no rights to such content.

LIMITATION OF LIABILITY.
THIS SOFTWARE IS PROVIDED "AS IS," WITHOUT A WARRANTY OF ANY KIND. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. JAVASCRIPTSPELLCHECK.COM AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL JAVASCRIPTSPELLCHECK.COM OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF JAVASCRIPTSPELLCHECK.COM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

MISCELLANEOUS
This software is not designed or intended for use in on-line control of aircraft, air traffic, aircraft navigation or aircraft communications; or in the design, construction, operation or maintenance of any nuclear facility. Licensee represents and warrants that it will not use or redistribute the Software for such purposes.