# getRange()

# ***S’applique à :***apps for SharePoint | Excel Services | SharePoint Server 2013

**Dans cet article**    
[Exemple](https://msdn.microsoft.com/fr-fr/library/office/jj730478.aspx#sectionSection2)

Obtient le [Ewa.Range Object](https://msdn.microsoft.com/fr-fr/library/office/ee589061.aspx) associé à un événement donné **rangeChange** .

VB

var value = Ewa.RangeChangeEventArgs.getRange()

## Valeur de retour

Type : **[Ewa.Range]**

## Remarques

La méthode **Ewa.RangeChangeEventArgs.getRange** renvoie la plage qui est associée à l'événement de changement de plage.

## Exemple

L'exemple de code suivant s'abonne à un gestionnaire d'événements pour l'événement **dataEntered** . Lorsque le Gestionnaire d'événements spécifié pour l'événement **dataEntered** est appelé, elle est appelée avec un seul argument de type [Ewa.RangeChangeEventArgs](https://msdn.microsoft.com/fr-fr/library/office/ee588967.aspx). L'objet **Ewa.RangeChangeEventArgs** contient des informations sur l'objet [Ewa.Range](https://msdn.microsoft.com/fr-fr/library/office/ee589061.aspx) associé à l'événement **dataEntered** .

Ensuite, l'exemple de code montre comment utiliser l'objet **Ewa.RangeChangeEventArgs** pour obtenir des informations relatives à la plage spécifiée. L'exemple de code suppose que vous travaillez avec un Excel Web Access composant WebPart sur SharePoint Server 2013.

JavaScript

<script type="text/javascript">

var ewa = null;

// Run the Excel load handler on page load

if (window.attachEvent) {

window.attachEvent("onload", loadEwaOnPageLoad);

} else {

window.addEventListener("DOMContentLoaded", loadEwaOnPageLoad, false);

}

// Event handler for page load event

function loadEwaOnPageLoad() {

if (typeof (Ewa) != "undefined") {

// Retrieve workbook from SharePoint location when EWA is ready

Ewa.EwaControl.add\_applicationReady(ewaApplicationReady);

}

else {

alert("Error - the EWA is not loaded.");

}

// ...

}

function ewaApplicationReady() {

// Get a reference to the EWA web part that represents the Ewa web part

// in SharePoint:

ewa = Ewa.EwaControl.getInstances().getItem(0);

var sheets = ewa.getActiveWorkbook().getSheets();

var sheet;

// Add the event handler to every sheet in the workbook

for (i = 0; i < sheets.getCount(); i++) {

sheet = sheets.getItem(i);

sheet.add\_dataEntered(sheetDataEnteredHandler);

}

}

function sheetDataEnteredHandler(rangeChangeArgs) {

var sheetName = rangeChangeArgs.getRange().getSheet().getName();

var col = rangeChangeArgs.getRange().getColumn();

var row = rangeChangeArgs.getRange().getRow();

// Get the formatted value

var value = rangeChangeArgs.getFormattedValues();

alert("The range was located at row " + (row + 1) + " and column " + (col + 1) + " with value " + "\"" + value + "\"" + ".");

}

</script>

The Selection.getRangeAt**()** method returns a range object representing one of the ranges currently selected.

**Syntax**

*range* = *sel*.getRangeAt(*index*)

Parameters

***range***

The [**Range**](https://developer.mozilla.org/en-US/docs/Web/API/Range) object that will be returned.

***index***

The zero-based index of the range to return. A negative number or a number greater than or equal to [Selection.rangeCount](https://developer.mozilla.org/en-US/docs/Web/API/Selection/rangeCount) will result in an error.

**Examples**

var ranges = [];

sel = window.getSelection();

for(var i = 0; i < sel.rangeCount; i++) {

ranges[i] = sel.getRangeAt(i);

}

/\* Each item in the ranges array is now

\* a range object representing one of the

\* ranges in the current selection \*/

**The Selection.rangeCount** read-only property returns the number of ranges in the selection.

Before the user has clicked a freshly loaded page, the **rangeCount** is 0. After the user clicks on the page, **rangeCount** is 1, even if no selection is visible.

A user can normally only select one **range** at a time, so the **rangeCount** will usually be 1. Scripting can be used to make the selection contain more than 1 range.

Gecko browsers allow multiple selections across table cells.

**Syntax**

***value* = *sel*.rangeCount**

**Example**

The following example will show the **RangeCount** every second. Select text in the browser to see it change.

**JavaScript Content**

window.setInterval(function () {

console.log(window.getSelection().rangeCount);

}, 1000);

**ContentEditable**

**W3C Editor's Draft**

**This version:**

<http://w3c.github.io/editing/contentEditable.html>

**Latest published version:**

<https://www.w3.org/TR/ContentEditable/>

**Latest editor's draft:**

<http://w3c.github.io/editing/contentEditable.html>

***contentEditable=false element***

Any element that has the contentEditable attribute set to `false`.

4. contenteditable

The contenteditable attribute is an enumerated attribute whose keywords are the empty string (""), "events", "caret", "typing", "plaintext-only", "true", and "false". There is one additional state, the inherit state, which is the missing value default (and the invalid value default).

The empty string and the "true" keyword map to the true state. The other keywords map to their respective states.

The false state indicates that the element is not editable. The inherit state indicates that the element has the state of its parent.

The contentEditable IDL attribute, on getting, must return the string "true" if the content attribute is set to the true state, "plaintext-only" if the content attribute is set to the plaintext-only state, "typing" if the content attribute is set to the typing state, "caret" if the content attribute is set to the caret state, "events" if the content attribute is set to the events state, "false" if the content attribute is set to the false state, "plaintext-only" if the content attribute is set to the plaintext-only state, and inherit otherwise.

On setting, if the new value is an ASCII case-insensitive match for the string "inherit" then the content attribute must be removed, if the new value is an ASCII case-insensitive match for a string matching the name of a state, then the content attribute must be set to the name of that state, and otherwise the attribute setter must throw a SyntaxError exception.

The isContentEditable IDL attribute, on getting, must return true if the element is either an editing host or editable, and false otherwise.

5. Meaning of states

The states "events", "caret", "typing", "plaintext-only" and "true" are hierarchically ordered, so that the state "caret" also includes the features of the "events" state, the "typing" state includes the features of the "caret" state, the "plaintext-only" state includes the features of the "typing" state, and the "true" state includes all the features of the "plaintext-only" state.

The "events" state means that beforeinput events are triggered when the user asks for an editing operation. The "caret" state adds default browser controlled movement of the caret. The "typing" state adds handling of text input through IME and keyboard, and deletion within an IME composition. The "plaintext-only" state adds handling of text deletion within a text node. The "true" state adds handling of deletion deletion of non-textual content and editing commands through the execCommand command.

The states "events", "caret", "typing" and "plaintext-only" are defined in this document.

The state "true" is currently not well-defined and its usage is discouraged. An initial attempt has been made to specify the behavior of the "true" state in the [contentEditable=True spec](https://w3c.github.io/editing/contentEditableTrue.html).

6. contentEditable states

6.1 contentEditable=events state

In a focused [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is in the "events" state, a caret *MUST* be drawn if the selection is collapsed, and it *MUST* be possible to place the caret in all of the [Legal Caret Positions](https://w3c.github.io/editing/contentEditable.html#dfn-legal-caret-positions) programmatically.

All user editing intentions initiated while an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is in the events state is focused, *MUST* trigger a `beforeinput` event.

6.2 contentEditable=caret state

A focused [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is in the "caret" state *MUST* behave like an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) in the events state. Additionally, the default action of the `beforeSelectionChange` event in such an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) must be to move the caret in the indicated direction, if movement in that direction seems possible.

NOTE

Notice the planned [beforeSelectionChange](https://github.com/w3c/selection-api/issues/56) event.

6.3 contentEditable=typing state

A focused [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is in the "typing" state *MUST* behave like an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) in the caret state, and additionally, it *MUST* handle text insertion by keyboard at the position of the caret if the caret is placed within a text node or it is possible to place a text node at the place of the caret. It must by default also handle composition by IME, both insertion as well as deletion of text input.

6.4 contentEditable=plaintext-only state

A focused [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is in the "plaintext-only" state *MUST* behave like an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) in the typing state, and additionally, it *MUST* handle text deletion.

7. Cutting and pasting

Within an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is in the "events", "caret" or "typing" state, cutting and pasting is disabled. Within an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) in the "plaintext-only" state, pasting only causes plaintext pasting by default. Independently of the state of the [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host), the events `paste` and `cut` should be triggered whenever the user expresses the intention to paste or cut.

8. Caret drawing and movement

8.1 Caret drawing

A caret is drawn in any [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is focused, that does not hold any other selections. Under such conditions, the caret represents a collapsed selection.

8.2 Caret positions

It *MUST* be possible to put the caret in any of the [Legal Caret Positions](https://w3c.github.io/editing/contentEditable.html#dfn-legal-caret-positions) programatically and for the caret to be visible in these in any [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is in the "events", "caret" or "typing" state.

**8.2.1 Initial caret placement**

Carets are initially placed at the first possible position within the [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host).

**8.2.2 Caret movement**

The specific location that the caret is moved to by default in an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) in the "caret" and "typing" state, is out of scope for this specification, but later versions of this specification or specifications covering other contentEditable specifications may further specify the movement patterns of the caret.

8.3 Replacing text/content

When the user indicates the wish to replace part of the contents of a [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host), by means of a browser-builtin spell checker or alike, a `beforeinput` input with `inputType` set to `replaceText` or `replaceContent` is triggered. No part of the DOM is being changed by default in an editing host that is in the "events", "caret" or "typing" state.

8.4 Removal of content

Content is not removed automatically through user input in an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is in the "events" or "caret" state. Instead, Del/Backspace key presses, etc. trigger user `beforeinput` events with `inputType` set to `deleteContentForward` or `deleteContentBackward`. The same is true for any [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) that is in the "typing" state, unless it is currently in the composition mode, in which case it will remove characters from the DOM if they are part of the composition.

In the "plaintext-only" state, an [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host) does handle text deletion within a single text node.

8.5 Advanced grammar checking

An implementation CAN provide spell check, grammar check and other advanced functionality that are not defined through a specification for any [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host). These features *MUST* be disabled by default.

8.6 Context menu

If the implementation provides a `context menu`, this `context menu` *SHOULD* contain items for editing operations such as `paste`, `cut`, `copy`, `delete` and CAN contain items for spellchecking for any [editing host](https://w3c.github.io/editing/contentEditable.html#dfn-editing-host). Triggering any of `context menu` items *MUST NOT* by default cause any change to the DOM, `paste`, `cut` and `beforeinput` events *SHOULD* be triggered. Menu items in a `context menu` that do not cause DOM changes by default *SHOULD* function as in any other editing context.

A.1 Normative references

**[css-cascade-3]**

[*CSS Cascading and Inheritance Level 3*](https://www.w3.org/TR/css-cascade-3/). Elika Etemad; Tab Atkins Jr.. W3C. 19 May 2016. W3C Candidate Recommendation. URL: <https://www.w3.org/TR/css-cascade-3/>

# WYSIWYG et WYSIWYM :

Découvrez les différences, avantages et inconvénients des éditeurs de textes avec options de formatage, les WYSIWYG, et leurs cousins semblables dans la forme mais pas dans le fond: les WYSIWYM

# Les WYSIWYG

# "Un ***WYSIWYG*** (prononcé [wɪziwig] ou [wiziwig]) est une interface utilisateur qui permet de composer visuellement le résultat voulu, typiquement pour un logiciel de mise en page, un traitement de texte ou d’image. C'est une interface « intuitive » : l’utilisateur voit directement à l’écran à quoi ressemblera le résultat final."

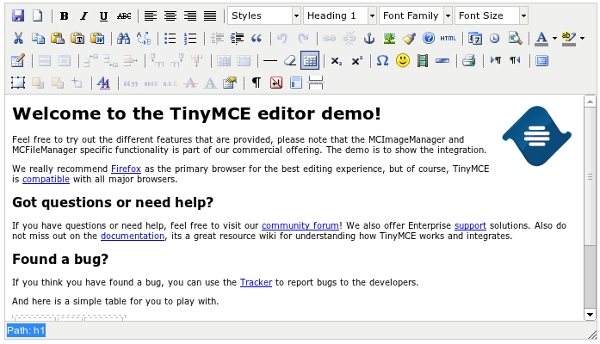
Le mot WISYWIG est l'acronyme de **Wh**at **Y**ou **S**ee **I**s **Wh**at **Y**ou **G**et. En gros, ce que vous voyez est ce que vous obtenez.

Ce terme regroupe tout les logiciels de traitement de texte comme par exemple [**libreoffice**](http://fr.libreoffice.org/), **microsoft office**, ... mais aussi les logiciels d'édition web dans le style de **Dreamweaver d'Adobe**, ou [**BlueGriffon**](http://bluegriffon.org/).

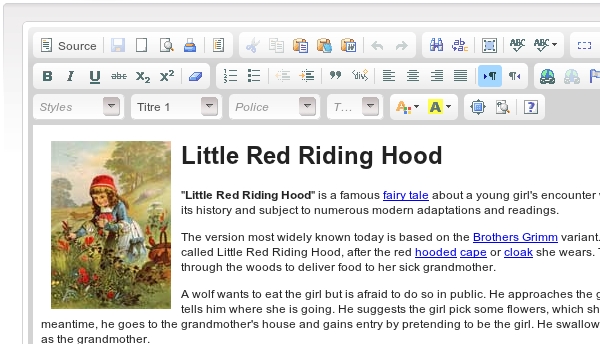
Dans le monde du webdesign, les **WYSIWYG** sont des éditeurs en javascript, qui s'installent sur un text area, et qui fournissent des options de formattage comme l'écriture en gras/italique et le changement de taille et de police directement visible dans le text area.

Quelques exemples:

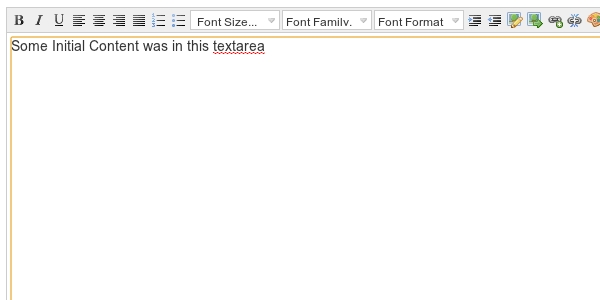
## ****TinyMCE****

[](http://www.tinymce.com/tryit/full.php)

## ****CKeditor****

**[](http://ckeditor.com/demo)**

## ****NiceEdit****

[](http://nicedit.com/demos.php?demo=1)

Les points important d'un éditeur **WYSIWYG** étant d'avoir une **possibilité pour éditer directement le HTML** (c'est parfois nécessaire), d'être en full javascript pour ne pas dépendre d'une configuration serveur particulière et de **générer le code le moins "crade" possible.**

Pourquoi un code le **"moins crade"** possible, et pas un code le **"plus propre"** possible? Car c'est impossible. Ces éditeurs génèrent tous un code avec du style dans les balises, des br de partout et beaucoup trop de balises inutiles (comme des <p><br/></p>, par exemple, pour **tinyMCE**, qui génère un des codes les moins encombrés.

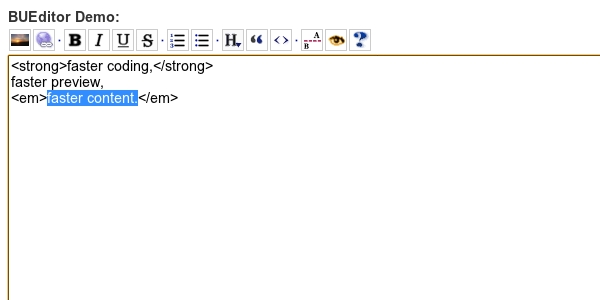
Il existe pourtant une alternative aux WYSIWIG: les WYSIWYM (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **M**ean), ce que vous voyez est ce que vous voulez dire.

# Les WYSIWYM

Les **WYSIWYM** ont la particularité de ne pas montrer directement ce que vous aurez comme rendu final (il existe quand même généralement une prévisualisation du résultat), mais un rendu schématique de ce qui a été demmandé. Cela permet un bien plus grand contrôle: rien ne se fait derrière notre dos.

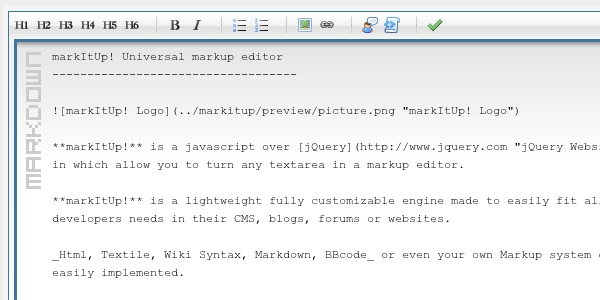
Un excelent exemple de **WISYWIM** est **BUEditor**: Un Textarea, et des boutons qui insèrent directement dans une textarea du code HTML.

## ****BUEditor****

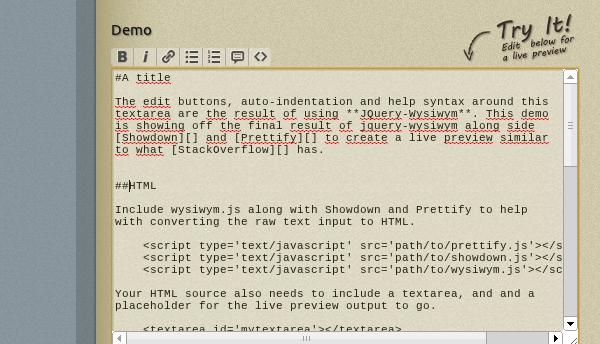
[](http://www.blogduwebdesign.com/javascript-ressources/wysiwyg-wysiwym-faisons-le-point/%EF%BB%BFhttp:/ufku.com/drupal/bueditor/demo)

L'inconvénient, ici, est que l'on jongle avec du HTML et que j'aime de moins en moins le HTML en temps que markup language: il est trop verbeux, trop fouilli. Heureusement, il existe des alternatives, comme le [**markdown**](http://fr.wikipedia.org/wiki/Markdown). Et re-heureusement, il existe des éditeurs qui vont avec.

## ****Markitup****

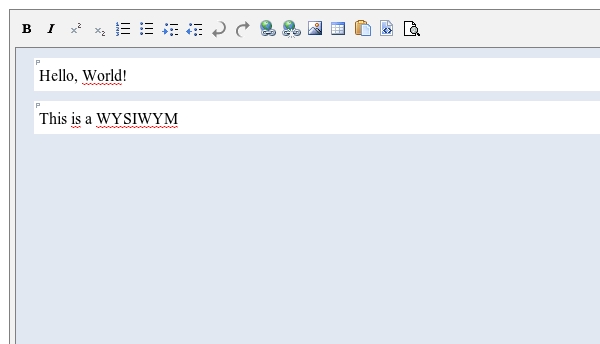
[](http://www.blogduwebdesign.com/javascript-ressources/wysiwyg-wysiwym-faisons-le-point/%EF%BB%BFhttp:/markitup.jaysalvat.com/)

## ****jQuery-wysiwym****

[](http://pushingkarma.com/projects/jquery-wysiwym/)

Ces deux là sont personnellement mes deux chouchous. Leur seul inconvénient par rapport à un WYSIWYG est que les images ne sont pas affichées, mais seulement leurs liens. Pour palier à ce problème, il y a **wymeditor**, l'editeur **WYSIWYM** qui se la joue **WYSIWYG**. Si vous êtes fan des wysiwyg, prenez le temps de le tester, et de bien apprecier les différences. Il mérite vraiment d'être connu.

## ****Wymeditor****

[](http://www.wymeditor.org/)

# Conclusion

**Le WYSIWYG** est un outil qui simplifie grandement l'édition, c'est vrai, mais à quel prix? Un code indigeste, pas standard, et pas forcément excellent pour le référencement.

Les **WYSIWYM** résolvent bien ces problèmes et le fait de pouvoir utiliser plusieurs languages (BBCode pour la plupart des forums, Markdown pour le traitement de texte, latex pour les mathematiques, l'impression, ...) les rendent bien plus puissants. Il est peut-être temps de changer nos habitudes et de remplacer le bouton "**edit HTML** source" par **"preview".**

Vous en pensez quoi ?

### Articles similaires

[Opus Magnum, un premier pas dans le monde de la programmation ?](http://www.blogduwebdesign.com/javascript-ressources/Opus-Magnum-premier-pas-programmation/2662)[Découvrez Semantris, un jeu de mots issu de Google A.I. Experiments](http://www.blogduwebdesign.com/javascript-ressources/Semantris-jeu-de-mots-Google-Ai-Experiments/2658)[Synthétiser de la musique 8bit en HTML5 ?](http://www.blogduwebdesign.com/javascript-ressources/Synthetiser-de-la-musique-8bit-en-HTML5-/2624)[Synchronisez plusieurs objets Javascript avec Automerge](http://www.blogduwebdesign.com/javascript-ressources/automerge-synchronisez-plusieurs-objets-javascript/2620)[Transformez n'importe quel site en API grace à Apify](http://www.blogduwebdesign.com/javascript-ressources/Transformez-nimporte-quel-site-APi-Apify/2612)

## QU’EST-CE QU’UN ÉDITEUR DE SITES WEB ?

Avant de rentrer dans le vif du sujet, il est nécessaire d’expliquer ce qu’est un éditeur web.

**Un éditeur de sites web ou plus complètement un éditeur de sites WYSIWYG (What You See Is What You Get) est un logiciel en ligne qui vous permet de développer indépendamment des sites Web en ligne sans avoir de compétences en développement** PHP, HTML, JavaScript. Le principe de fonctionnement est réduit à un mouvement visuel des éléments du site comme le souhaite le webmaster avec un simple mouvement de souris.

Ces capacités sont obtenues avec le principe de Drag & Drop. Ces éditeurs sont conçus pour deux groupes de publics différents. La plupart des clients de ces éditeurs sont des webmasters débutants avec ou sans budget. Pour le deuxième groupe d’utilisateurs, il s’agit d’entreprises ou de particuliers qui commencent un business en ligne. Ces clients ne veulent pas investir tout de suite des sommes importantes dans le développement de sites Web pour leur business.**En règle générale, ces éditeurs ne sont pas totalement gratuits**.

Ces éditeurs en ligne ont commencé à naître au début des années 2000, puis ont explosé**à partir de 2005 avec l’apparition de Wix en 2006,** par exemple.

[www.tinymce.com](http://www.tinymce.com) (Conseil de Robin)

**Créer un Plugin (Jquery)**

**https://youtu.be/s18tL6dohxM**