

**HELP YOUR  
WYSIWYG HELP YOU:  
CREATING CUSTOM  
CKEDITOR WIDGETS**

## Who am I?

Les Lim, Developer

TEN7 Interactive

<http://ten7.com>

[les@ten7.com](mailto:les@ten7.com)

Twitter: @lesmana

August 9, 2014

Drupal Camp Twin Cities

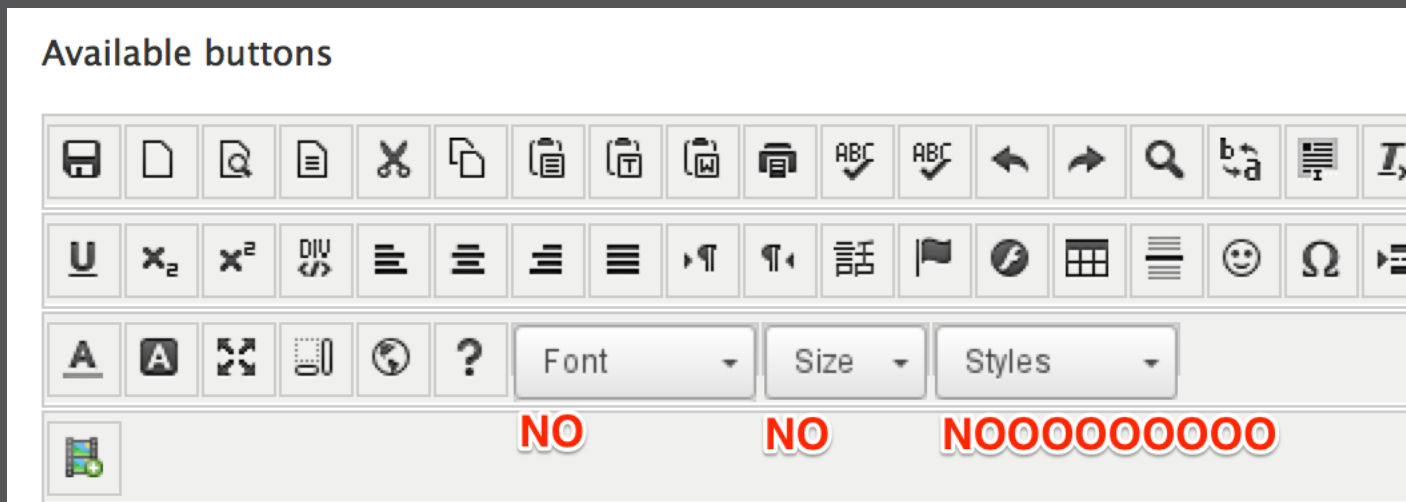
## Who are you?

To get the most out of this session, you should:

- Have some familiarity with JavaScript syntax

# Why are we here?

- Because your content creators demand more
- Because you don't want to give it to them



## What about Templates?











- Templates are pre-cooked HTML snippets that can be pasted into the editor
- After pasting, template elements behave just like normal HTML
- Targeting/selecting HTML elements for manipulation is difficult

# What about Templates?



# An all-too-real world example

Body

Normal ▾ **B** *I*           Source

**Here's a nice template.**

<b>Cost-effective</b>	<b>Consultative</b>
<p>You do not have to sacrifice service or support levels in order to meet budget constraints.</p>	<p>You have access to experts who understand storage hardware, capacity planning and environment planning.</p>

**I'll just add a paragraph in here...**







Insert paragraph here

IBM Storage System Product Support Listing

body div div p span Words: 220

# An all-too-real world example

Body

Normal **B** *I*      ABC  Source

**Cost-effective**

You do not have to sacrifice service or support levels in order to meet budget constraints.  
You d...wha... ah, @!\$.|

**Consultative**

You have access to experts who understand storage hardware, capacity planning and environment planning.

**GAAAAAAH.**

body div p Words: 224



## What's different with Widgets?

- Your widget template is treated as a single, unified element
- You can control which parts of your template are open for editing and manipulation, and lock down the parts that are not
- You can trust your widget to be structurally predictable

## What you'll need

- \*Either\* ckeditor.module OR wysiwyg.module
  - <https://www.drupal.org/project/ckeditor>
  - <https://www.drupal.org/project/wysiwyg>
- A build of the CKEditor library that includes the Widgets plugin
  - <http://ckeditor.com/builder>
- A custom module to put our custom code into

# Let's get started!

Let's make a simple “callout” widget for putting a call-out box in a node. The markup for a call-out box should look like this:

```
<div class="callout">Callout box contents here.</div>
```

We'll add styles for .callout in our theme CSS.

Proin viverra dui nec nisl pretium interdum. Nunc odio ipsum; sagittis et enim sed, pellentesque vestibulum nulla. Mauris pretium porta erat, nec vestibulum massa vehicula ut? Nullam quis ligula scelerisque nisi mattis rhoncus. Integer nec mi at justo hendrerit ullamcorper a at diam. Donec condimentum consequat venenatis. Proin ut purus ac sem consequat elementum id sed risus. Aenean in auctor nisl, id tincidunt augue. Nulla at iaculis nunc? Nulla faucibus eleifend quam, at luctus leo dapibus vel.

Donec sem neque, lobortis at ligula at, pharetra sagittis tellus. Morbi vehicula ligula ut dolor lobortis; non aliquet urna vestibulum. Ut congue lectus eu dignissim faucibus. Cras auctor; odio at volutpat fringilla, dolor massa consequat

turpis, a tincidunt lectus nulla quis eros. Maecenas adipiscing dolor sit amet accumsan mollis. Nullam tempor consequat scelerisque. Aliquam volutpat justo nec sem varius rhoncus. Nam blandit elementum mauris id bibendum. Vivamus tincidunt viverra placerat. Nulla vel suscipit dui. Morbi sed tellus a risus semper ultrices ut quis est. Aenean sed cursus ligula. Maecenas lobortis est ac metus vehicula, sollicitudin adipiscing lectus auctor. Aenean cras amet.

*Sidebar text here.*

# tc2014\_widgets module

We'll start by creating our example module

- sites/all/modules/custom/tc2014\_widgets
  - **tc2014\_widgets.info**
  - **tc2014\_widgets.module**
  - **plugins/**
  - **plugins/callout**
  - **plugins/callout/plugin.js**
  - **plugins/callout/icon-callout.png**

## **tc2014\_widgets.info**

```
name = TC2014 CKEditor Widgets  
description = Example CKEditor Widget  
implementations.  
core = 7.x
```

## tc2014\_widgets.module

Implement HOOK\_ckeditor\_plugin().

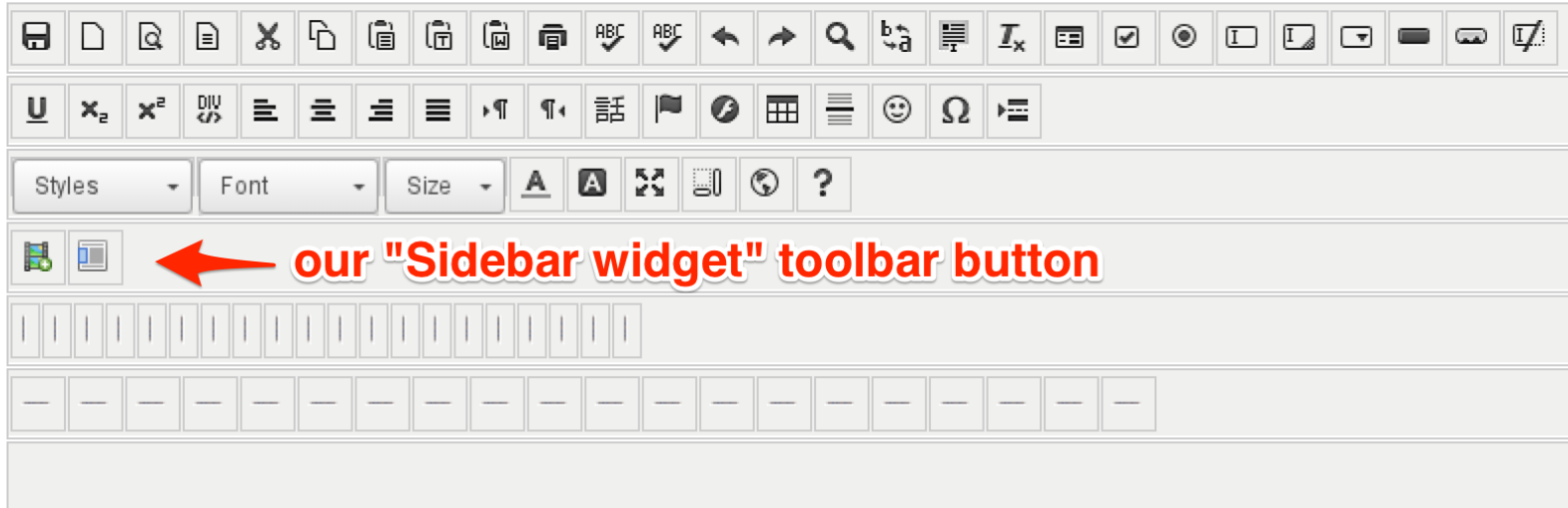
- Returns an array of configurations, keyed by plugin machine name
- Defines the path to the plugin's directory
- Describes the toolbar buttons for the Drupal configuration page for CKEditor.

# tc2014\_widgets.module

```
<?php
/**
 * Implements hook_ckeditor_plugin().
 */
function tc2014_widgets_ckeditor_plugin() {
  return array(
    'callout' => array(
      'name' => 'Callout box widget',
      'desc' => t('CKEditor widget for a callout box.'),
      // The full path to the CKEditor plugins directory, with the trailing slash.
      // Directory must contain a "plugin.js" file.
      'path' => drupal_get_path('module', 'tc2014_widgets') . '/plugins/callout/',
      // Register the toolbar icon to be used in the Drupal admin interface.
      'buttons' => array(
        'callout' => array(
          'label' => 'Callout box',
          'icon' => 'icon-callout.png',
        ),
      ),
    ),
  );
};
}
```

# Drupal CKEditor config page

## Available buttons



our "Sidebar widget" toolbar button

## Plugins

- ☐ Plugin to count symbols, symbols without blanks and words
- ☐ Plugin for inserting Drupal teaser and page breaks.
- ☐ Plugin for inserting Drupal embedded media
- ☐ CKEditor widget for creating a sidebar area.

Choose the plugins that you want to enable in CKEditor.

Plugin listed, disabled by default



## plugin.js

- Register our custom widget plugin
- When the plugin is initialized by CKEditor:
  - **Adds the toolbar button to the editor instance**
  - **Registers the widget definition when the plugin is loaded**

## plugin.js

```
// Registering our custom CKEditor plugin

CKEDITOR.plugins.add('callout', {
  requires: 'widget',
  init: function(editor) {
    /* Register the toolbar button. */
    /* Register the widget definition. */
  }
});
```

## plugin.js

```
// Add our button to the toolbar instance

CKEDITOR.plugins.add('callout', {
    requires: 'widget',
    init: function(editor) {
        /* Register the toolbar button. */
        editor.ui.addButton( 'callout', {
            label : 'Insert Callout box',
            icon : this.path + 'icon-callout.png',
            command : 'callout'
        });
    }
});
```

# CKEditor instance

it Basic page Sidebar!

VIEW

EDIT

DEVE

Title \*

Sidebar!

Body



Normal

**B**

*I*

~~S~~



Source

**...but it doesn't do anything yet.**

Maecenas aliquet mi a egestas pretium. Phasellus adipiscing ipsum sit amet quam laoreet; eget consectetur justo fringilla. In in diam ligula. Pellentesque sodales mollis interdum. Vivamus et urna tellus. In mattis non est a rhoncus. Vivamus lacinia quam eget mollis posuere. In ac tortor non enim venenatis vulputate. Duis rutrum ultrices elit vitae dignissim. Morbi sollicitudin, nulla tincidunt sodales laoreet, tellus ante fermentum libero; non sodales erat augue vel neque. Nulla eros tortor, laoreet ut hendrerit eget, convallis a est. Nulla ullamcorper lectus in rutrum tincidunt. |

## plugin.js

```
// Register our custom widget inside the plugin.  
  
CKEDITOR.plugins.add('callout', {  
  requires: 'widget',  
  init: function(editor) {  
    /* Register the toolbar button. */  
    /* Register the widget definition. */  
    editor.widgets.add('callout', {  
      template: ???  
      allowedContent: ???  
      editables: ???  
      upcast: ???  
    });  
  }  
});
```

## plugin.js (widget definition)

A widget definition should probably have at least these four properties:

- **template**  
a string of default markup to be inserted in the editor when the toolbar button is clicked
- **allowedContent**  
describes the valid HTML elements used or allowed in this widget, so that CKEditor's filter doesn't strip them
- **editables**  
an array of elements in the widget, defined by selector, that should be allowed to be edited
- **upcast**  
logic for determining if an element should be treated as this type of widget

## plugin.js (widget definition)

```
editor.widgets.add('callout', {  
  template: '<div class="callout">Callout  
box contents here.</div>',  
  allowedContent: ???  
  editables: ???  
  upcast: ???  
});
```

# CKEditor Advanced Content Filter

- Strips malicious or disallowed HTML elements and/or attributes
- Is turned on by default in CKEditor
- Strips “class” attributes by default
- “allowedContent” property can be used to define filter exceptions for a widget

More detail about creating “Allowed Content” rules:

[http://docs.ckeditor.com/#!/guide/dev\\_allowed\\_content\\_rules](http://docs.ckeditor.com/#!/guide/dev_allowed_content_rules)



## plugin.js (widget definition)











```
editor.widgets.add('callout', {  
  template: '<div class="callout">Callout  
box contents here.</div>',  
  allowedContent: 'div(!callout)',  
  editables: ???  
  upcast: ???  
});
```

# CKEditor instance

*Edit Basic page Sidebar!* **VIEW** **EDIT** **DEVEL**

**Title \***

**Body**

Format **B** **I** **S**           Source

idcus nunc: nulla idcus elementum quam, at idcus idcus idcus vel.

Donec sem neque, lobortis at ligula at, pharetra sagittis tellus. Morbi vehicula ligula ut dolor lobortis; non aliquet urna vestibulum. Ut congue lectus eu dignissim faucibus. Cras auctor; odio at volutpat fringilla, dolor massa consequat turpis, a tincidunt lectus nulla quis eros. Maecenas adipiscing dolor sit amet accumsan mollis. Nullam tempor consequat scelerisque. Aliquam volutpat justo nec sem varius rhoncus. Nam blandit elementum mauris id bibendum. Vivamus tincidunt viverra placerat. Nulla vel suscipit dui. Morbi sed tellus a risus semper ultrices ut quis est. Aenean sed cursus ligula. Maecenas

body div

[Switch to plain text editor](#)

**Hey, it worked!**

*Sidebar contents here.*

**...but I can't click inside it.**











## plugin.js (widget definition)

```
editor.widgets.add('callout', {  
  template: '<div class="callout">Callout box  
contents here.</div>',  
  allowedContent: 'div(!callout)',  
  editables: {  
    contents: {  
      selector: '.callout'  
    }  
  },  
  upcast: ???  
});
```

# CKEditor instance

Body

Normal (...

**B** *I* ~~S~~           Source

Donec sem neque, lobortis at ligula at, pharetra sagittis tellus. Morbi vehicula ligula ut dolor lobortis; non aliquet urna vestibulum. Ut congue lectus eu dignissim faucibus. Cras auctor; odio at volutpat fringilla, dolor massa consequat turpis, a tincidunt lectus nulla quis eros. Maecenas adipiscing dolor sit amet accumsan mollis. Nullam tempor consequat scelerisque. Aliquam volutpat justo nec sem varius rhoncus. Nam blandit elementum mauris id bibendum. Vivamus tincidunt viverra placerat. Nulla vel suscipit dui. Morbi sed tellus a risus semper ultrices ut quis est. Aenean sed cursus ligula. Maecenas lobortis est ac metus vehicula.

**I edited it!**

*"Janeway is the best captain," said the internet troll.*

body div div

[Switch to plain text editor](#)

Text format Full HTML [More information about text formats](#) ?

- Web page addresses and e-mail addresses turn into links automatically.

# Upcasting

**Upcasting** refers to the treatment of some simple object as a more complex type of object.

In CKEditor, we use the **upcast** property of the widget to define when a particular HTML element should be treated as an instance of this widget.

## plugin.js (widget definition)

```
editor.widgets.add('callout', {
  template: '<div class="callout">Callout box contents here.</div>',
  allowedContent: 'div(!callout)',
  editables: {
    contents: {
      selector: '.callout'
    }
  },
  upcast: function(element) {
    if (element.name == 'div' && element.hasClass('callout')) {
      return true;
    }
    else {
      return false;
    }
  }
});
```

# plugin.js (complete)

```
CKEDITOR.plugins.add('callout', {
    requires: 'widget',
    init: function(editor) {
        // Register the toolbar buttons for the CKEditor editor instance.
        editor.ui.addButton( 'callout', {
            label : 'Insert Callout box',
            icon : this.path + 'icon-callout.png',
            command : 'callout'
        });
        // Register the widget.
        editor.widgets.add('callout', {
            template: '<div class="callout">Callout box contents here.</div>',
            allowedContent: 'div(!callout)',
            editables: {
                contents: {
                    selector: '.callout'
                }
            },
            upcast: function(element) {
                if (element.name == 'div' && element.hasClass('callout')) {
                    return true;
                }
                else {
                    return false;
                }
            }
        });
    }
});
```

## Accordion widget

Now let's add a widget to generate markup for an expandable accordion.

We'll start by applying the steps we've already covered.



# Accordion widget

The HTML structure generated by our widget should look like this:

```
<div class="accordion">
  <div class="accordion__title">Accordion title</div>
  <div class="accordion__content">
    <p>Accordion contents</p>
  </div>
</div>
```

In a module or theme, we'll write JavaScript to implement jQuery Accordion on any DIV with the class "accordion".

```
$('#div.accordion').accordion({
  header: ".accordion__title",
  collapsible: true,
  active: false,
  animated: false
});
```

## tc2014\_widgets module

We'll add a plugin directory for “accordion”:

- sites/all/modules/custom/tc2014\_widgets
  - **tc2014\_widgets.info**
  - **tc2014\_widgets.module**
  - **plugins/**
  - **plugins/callout**
  - **plugins/callout/plugin.js**
  - **plugins/callout/icon-callout.png**
  - **plugins/accordion**
  - **plugins/accordion/plugin.js**
  - **plugins/accordion/icon-accordion.png**

# tc2014\_widgets.module

```
function tc2014_widgets_ckeditor_plugin() {
  return array(
    'callout' => array( ... ),
    'accordion' => array(
      'name' => 'accordion',
      'desc' => t('CKEditor widget for an accordion.'),
      'path' => drupal_get_path('module', 'tc2014_widgets') . '/
plugins/accordion/',
      'buttons' => array(
        'accordion' => array(
          'label' => 'Accordion',
          'icon' => 'icon-accordion.png',
        ),
      ),
    ),
  );
}
```

## accordion/plugin.js

```
CKEDITOR.plugins.add('accordion', {
    requires: 'widget',
    init: function(editor) {
        // Register the toolbar buttons for the CKEditor
        editor instance.
        editor.ui.addButton('accordion',
            {
                label : 'Insert Accordion',
                icon : this.path + 'icon-accordion.png',
                command : 'accordion'
            });
        // Register the widget.
        editor.widgets.add('accordion', { ... });
    }
});
```

## accordion/plugin.js (widget definition)

```
// Register the widget.  
editor.widgets.add('accordion', { ... });
```

Some differences from the “callout” widget:

- We have more element classes to describe in **allowedContent**.
- We require **two** editable areas to be defined in **editables**: the accordion title, and the accordion content.

## accordion/plugin.js (widget definition)

```
editor.widgets.add('accordion', {
  template:
    '<div class="accordion">' +
      '<div class="accordion__title">Accordion title</div>' +
      '<div class="accordion__content"><p>Accordion contents</p></div>' +
    '</div>',
  allowedContent: 'div(!accordion); div(!accordion__content); div(!accordion__title)',
  editables: {
    title: {
      selector: '.accordion__title',
    },
    content: {
      selector: '.accordion__content'
    }
  },
  upcast: function(element) {
    return element.name == 'div' && element.hasClass('accordion');
  }
});
```

## accordion/plugin.js (widget definition)

We should further limit allowed HTML tags in the **.accordion\_title** editable area.

```
editables: {  
  title: {  
    selector: '.accordion__title',  
    allowedContent: 'strong em'  
  },  
  content: {  
    selector: '.accordion__content'  
  }  
},
```

# accordion/plugin.js (widget definition)

```
editor.widgets.add('accordion', {
  template:
    '<div class="accordion">' +
      '<div class="accordion__title">Accordion title</div>' +
      '<div class="accordion__content"><p>Accordion contents</p></div>' +
    '</div>',
  allowedContent: 'div(!accordion); div(!accordion__content); div(!accordion__title)',
  editables: {
    title: {
      selector: '.accordion__title',
      allowedContent: 'strong em'
    },
    content: {
      selector: '.accordion__content'
    }
  },
  upcast: function(element) {
    return element.name == 'div' && element.hasClass('accordion');
  }
});
```



# Drupal node view

## Accordion!

[View](#)[Edit](#)[Devel](#)**It works!**[Home](#)

► Accordion title

Proin viverra dui nec nisl pretium interdum. Nunc odio ipsum; sagittis et enim sed, pellentesque vestibulum nulla. Mauris pretium porta erat, nec vestibulum massa vehicula ut? Nullam quis ligula scelerisque nisi mattis rhoncus. Integer nec mi at justo hendrerit ullamcorper a at diam. Donec condimentum consequat venenatis. Proin ut purus ac sem consequat elementum id sed risus. Aenean in auctor nisl, id tincidunt augue. Nulla at iaculis nunc? Nulla faucibus eleifend quam, at luctus leo dapibus vel.

Donec sem neque, lobortis at ligula at, pharetra sagittis tellus. Morbi vehicula

*"Janeway is the best captain."*

# CKEditor instance

*Edit Basic page* Accordion!

VIEW

EDIT

DEVEL

Title \*

Accordion!

Body

Format

**B**

*I*

S



























Source

Accordion title

...sort of.

Accordion contents

why does it look awful?

Proin viverra dui nec nisl pretium interdum. Nunc odio ipsum; sagittis et enim sed, pellentesque vestibulum nulla. Mauris pretium porta erat, nec vestibulum massa vehicula ut? Nullam quis ligula scelerisque nisi mattis rhoncus. Integer nec mi at justo hendrerit ullamcorper a at diam. Donec condimentum consequat venenatis. Proin ut purus ac sem consequat

## CKEditor instance

- jQuery Accordion reformats the markup on the viewed page, but not in the CKEditor instance.
- We can add editor-only CSS to better represent the widget for editing.

# tc2014\_widgets module

We'll add a CSS file for the plugin:

- sites/all/modules/custom/tc2014\_widgets
  - **tc2014\_widgets.info**
  - **tc2014\_widgets.module**
  - **plugins/**
  - **plugins/callout**
  - **plugins/callout/plugin.js**
  - **plugins/callout/icon-callout.png**
  - **plugins/accordion**
  - **plugins/accordion/plugin.js**
  - **plugins/accordion/icon-accordion.png**
  - **plugins/accordion/editor-accordion.css**

## editor-accordion.css

```
.accordion {  
    position: relative;  
    margin: 10px;  
    background: #eee;  
    border: 1px dashed #ddd;  
}  
.accordion__title,  
.accordion__content {  
    margin: 5px;  
    box-shadow: 0 1px 1px #ddd inset;  
    border: 1px solid #cccccc;  
    background: #fff;  
}
```

## editor-accordion.css (cont.)

```
.accordion:before {  
    background: none repeat scroll 0 0 #141;  
    color: #fff;  
    content: ".accordion";  
    opacity: 0.65;  
    position: absolute;  
    right: 0;  
    top: 0;  
}
```

## Accordion widget

- Register the accordion-specific editor CSS file in the plugin's **init** method, not the widget definition.

## accordion/plugin.js

```
CKEDITOR.plugins.add('accordion', {
    requires: 'widget',
    init: function(editor) {
        // Register the toolbar buttons for the CKEditor editor instance.
        editor.ui.addButton('accordion',
            {
                label : 'Insert Accordion',
                icon : this.path + 'icon-accordion.png',
                command : 'accordion'
            });
        // Add our plugin-specific CSS to style the widget within CKEditor.
        editor.addContentsCss(this.path + 'editor-accordion.css');
        // Register the widget.
        ...
    }
});
```



# CKEditor instance

*Edit Basic page* Accordion!

VIEW

EDIT

DEVEL

Title \*

Accordion!

Body

Format

**B**

*I*

S



Source

Accordion title

.accordion

Accordion contents

Proin viverra dui nec nisl pretium interdum. Nunc odio ipsum; sagittis et enim sed, pellentesque vestibulum nulla. Mauris pretium porta erat, nec vestibulum massa vehicula ut? Nullam quis ligula scelerisque nisi mattis rhoncus. Integer nec mi at justo hendrerit ullamcorper a et diam. Donec condimentum consequat venenatis. Proin ut purus ac sem consequat

## Resources

CKEditor's widget example tutorial:

[http://docs.ckeditor.com/#!/guide/widget\\_sdk\\_tutorial\\_1](http://docs.ckeditor.com/#!/guide/widget_sdk_tutorial_1)

[http://docs.ckeditor.com/#!/guide/widget\\_sdk\\_tutorial\\_2](http://docs.ckeditor.com/#!/guide/widget_sdk_tutorial_2)

tc2014\_widgets example module:

[http://ten7.com/ckeditor\\_widgets](http://ten7.com/ckeditor_widgets)