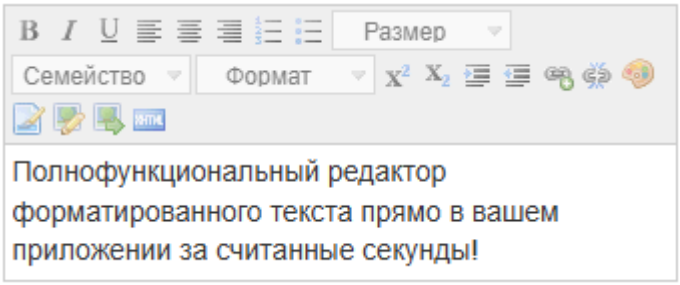
# NicEdit

NicEdit — это облегченный, кроссплатформенный редактор встроенного контента, позволяющий легко редактировать содержимое веб-сайта в режиме реального времени прямо в браузере.




NicEdit Javascript интегрируется в любой сайт за считанные секунды, позволяя сделать любой элемент/div редактируемым или преобразовать стандартные текстовые поля в редактор форматированного текста.

## Документация

This is the offical manual for the NicEdit WYSIWYG Editor. NicEdit is a Lightweight, Cross Platform, Inline Content Editor to allow easy editing of web site content on the fly in the browser. To download it go to http://nicedit.com/



### About NicEdit

NicEdit — это лёгкий, кроссплатформенный редактор встроенного контента, позволяющий легко редактировать содержимое веб-сайта на лету прямо в браузере. Этот небольшой и простой в использовании JavaScript-редактор интегрируется в любой сайт за считанные секунды, делая любой элемент/div редактируемым или преобразуя стандартные текстовые поля в форматированный текст.

NicEdit был разработан Брайаном Кирхоффом и может свободно использоваться в любых целях под лицензией MIT.

Функции:

- Небольшой размер файла (<35 КБ в сумме), <10 КБ в сжатом виде!

- Для работы требуется всего 2 файла (js + иконки).
- Гибкая конфигурация: замена текстовых полей или блоков div.
- При желании несколько редакторов могут использовать один элемент управления.
- Сохранение контента с помощью AJAX или HTTP POST
- IE 5.5+ / FF 2+ / Opera 9+ / Safari 3+

# NicEdit Reference

## Javascript API

### Class nicEditors

The nicEditors class provides convience methods to add NicEditor to the page or get a reference to a editor by ID

| | |
|---|---|
| `nicEditors.allTextAreas` | Converts all the textareas on the page info NicEditor instances. Returns a reference to the array of all NicEdit instances on the page. |
| `nicEditors.findEditor` | Finds a specific editor by ID if created using allTextAreas explitly For example to get the nicedit instance for `<textarea id=«myArea2»></textarea>` use: `nicEditors.findEditor('myArea2');` The method returns a nicedit instance that you call the functions below on. **Warning:** In order to use this and other API methods, the editor must be finshed loading. To see an example of how you can use addEvent to have your code run after editor load see Editor Events |
| `nicEditors.editors[]` | The raw array of all the NicEdit instances on the page |

### Class nicInstance

The nicEditor instance class creates the editable area for a single element and provides a number of useful methods to developers. You can get a reference to a specific nicInstance object using `nicEditors.findEditor`

| | |
|---|---|
| `[nicInstance].getContent()` | Returns the current HTML of the nicInstance For example: `nicEditors.findEditor('myArea2').getContent();` returns the HTML in the content editor that replaced the element on the page with ID 'myArea2'. |
| `[nicInstance].setContent(HTML)` | Set the current HTML in the editor instance For example: `nicEditors.findEditor('myArea2').setContent('<strong>Some HTML</strong>` here'); |
| `[nicInstance].saveContent()` | Only for nicInstances that are replacing a `<textarea>` this method syncs the content of the editor with the textarea value. This is done automatically if the form with the orginal `<textarea>` is submitted. However, you may want to explitly do the syncing yourself. |

### Class nicEditor

The nicEditor class is a container for a number of nicEditor instances and 1 nicPanel. Because of this

you can use nicEditor in 2 ways:

1. Create a single control panel with setPanel() then use addInstance multiple times to create as many editable areas on the page you need that are controlled from a single control panel.
2. Use the panelInstance() to create a combined content area/panel editor. If you use this option and want multiple editors you should create multiple nicEditor instances.

| | |
|---|---|
| `var myNicEditor = new nicEditor()` | Creates a new nicedit object. A single instance of nicEditor contains:- 1 or more editor instances (nicInstance/nicIFrameInstance) - 1 nicPanel |
| `myNicEditor.addInstance('someElementsID')` | Adds a nicedit instance to allow inline editing of the given elements ID |
| `myNicEditor.removeInstance('someElementID')` | Removes the nicedit instance with the given ID, returning it to the orginal element with the content within the element. |
| `myNicEditor.setPanel('someElementID')` | Append an editor control panel at the end of the given ID. Normally the element is blank and with a fixed CSS width you would like the panel set. |
| `myNicEditor.panelInstance('someElementID')` | Creates an inline content editor with attached control panel on top of the element. |
| `myNicEditor.instanceById('someElementID')` | Find a nicInstance object by ID that was added to this nicEditor |
| `myNicEditor.floatingPanel()` | Activate a floating panel. First add editor instances to this editor with addInstance() then call floatingPanel(). When a instance is selected, a floating panel will appear above/below it. * Requires nicFloating plugin |

## Configuration Options

NicEdit обладает широкими возможностями настройки: параметры конфигурации передаются при создании объекта NicEditor. Передавайте конфигурацию при вызове: new `nicEditor({CONFIG HERE})`

Добавьте `.panelInstance('ID TO TEXTAREA HERE')`, чтобы добавить редактор в текстовое поле.

Вы можете указать несколько параметров конфигурации, они представлены в формате JSON: `{fullPanel : true, iconsPath : 'nicEditorIcons.gif', maxHeight : 200}` Обратите внимание, что некоторые параметры конфигурации, такие как список кнопок, не будут работать, если вы не загрузите версию NicEdit и не используете её на своём веб-сервере.

**Доступные параметры конфигурации:**

| | |
|---|---|
| fullPanel | Если установлено значение true, будет создан экземпляр редактора со всеми доступными кнопками. Если значение `false` (по умолчанию), для управления отображением кнопок будет использоваться buttonList. |
| buttonList | Например, массив кнопок в редакторе.<br>['жирный','курсив','подчеркнутый','слева','по центру'] |
| iconsPath | Строковый путь к файлу значков, например, `images/nicEditorIcons.gif`. |
| maxHeight | Высота в пикселях, за пределы которой редактор не должен автоматически расширяться. Область редактирования может быть больше указанного значения, но в этом случае будут использоваться полосы прокрутки. Для редактора с фиксированной высотой (не динамически увеличивающегося) установите значение maxHeight и style=«height: X» равным одному и тому же значению в пикселях. |
| externalCSS | Относительный путь к внешней таблице стилей, применяемой к экземплярам iframe nicEditor (этот режим работы используется только в FF2). Пример: `{externalCSS : 'mysite.css'}` |
| `uploadURI` | Абсолютный или относительный URL-адрес скрипта-приемника nicUpload. Позволяет загружать изображения из nicedit на ваш собственный сервер. *Требуется nicUpload Пример:`{uploadURI: 'http://yourdomain.com/nicUpload.php '}`. По умолчанию ( http://files.nicedit.com/ ) используется сервер Nicedit, а файлы загружаются на Imageshack. PHP-версию скрипта для загрузки файлов можно найти по адресу: http://nicedit.com/svn/nicedit/trunk/nicUpload/php/nicUpload.php |

**List of Buttons for buttonList option**

- 'bold'
- 'italic'
- 'underline'
- 'left'
- 'center'
- 'right'
- 'justify'
- 'ol'
- 'ul'
- 'subscript'
- 'superscript'
- 'strikethrough'
- 'removeformat'
- 'indent'
- 'outdent'
- 'hr'
- 'image'
- 'upload' * requires nicUpload
- 'forecolor'
- 'bgcolor'
- 'link' * requires nicLink
- 'unlink' * requires nicLink
- 'fontSize' * requires nicSelect
- 'fontFamily' * requires nicSelect
- 'fontFormat' * requires nicSelect

- 'xhtml' * required nicCode

# Plugin Reference

## Saving via AJAX

NicEdit поддерживает наличие кнопки сохранения на панели инструментов, которую пользователи могут нажать для сохранения контента.

Для использования этого режима вам потребуется базовое знание JavaScript, и вы должны выполнить следующие шаги:

- Сначала убедитесь, что вы настроили загрузку с помощью плагина nicSave.
- Используйте параметр onSave при объявлении экземпляра nicEditor, например:

script.js

```
bkLib.onDomLoaded ( function ( ) {
  new nicEditor ( { fullPanel: true , onSave: function ( content, id, instance ) {
    alert ( 'Кнопка сохранения нажата для элемента ' + id + ' = ' + content ) ;
  } } ) . panelInstance ( 'myArea2' ) ;
} ) ;
```

- Поскольку мы установили параметр onSave для нашей функции обратного вызова и добавили кнопку «Сохранить», значок сохранения появится на панели инструментов.
- При нажатии кнопки будет запущена функция обратного вызова onSave с тремя параметрами.
  - Первый параметр — это содержимое экземпляра nicEditor.
  - Второй параметр — это идентификатор элемента, преобразованного в nicEditor.
  - Третий параметр — это ссылка на nicInstance (подробнее об этом см. в документации по Javascript API).

## XHTML Compliant Output

=== В чем проблема? NicEdit генерирует некорректный, не соответствующий стандартам код, который выглядит по-разному в каждом браузере и приводит к сбоям в работе редактора, если редактирование выполняется в браузере, отличном от того, который использовался для создания контента.

### Почему это проблема?

Применение стилей и очистка HTML-кода — самая сложная задача при создании WYSIWYG-редактора, и я так и не смог её решить с помощью nicEdit. Для полностью совместимой с

XHTML системы необходимы два компонента:

Во-первых, нам нужно реализовать все кнопки nicEdit таким образом, чтобы они перестали использовать поведение `execCommand()` браузера и применяли свои стили непосредственно к выделению/диапазонам. Это решит проблемы, подобные тем, с которыми вы сталкивались при применении команд к различным способам применения одного и того же стиля (например, <b>, `<strong>`, `<span style=«font-weight: bold»>`), и позволит редактору обрабатывать все три стиля в своем содержимом, а не использовать стандартное поведение браузера, которое не отличается гибкостью. Во-вторых, nicEdit будет выполнять очистку HTML-кода во время доступа (подобно тому, как это пытается делать nicXHTML, но пока с ошибками), чтобы обрабатывать другой контент, который не применяется через элементы управления редактора (например, при вставке контента).

### Что доступно сейчас?

На данный момент плагин `nicXHTML` находится в экспериментальной стадии и не решает эту проблему полностью. Чтобы попробовать его в действии, скачайте плагин с nicXHTML, а затем добавьте `{xhtml : true}` в параметры. Известно множество проблем с его реализацией.

Ещё один вариант, который я рекомендовал и который может подойти некоторым, — это использование стратегий очистки на стороне сервера, таких как функции `tidy` в PHP, для очистки HTML-кода, создаваемого `nicEdit`. Однако это решение не будет доступно всем, усложняет процесс и не является идеальным.

### Когда я могу ожидать исправления этой проблемы?

Предстоит много работы, сделать это правильно немного сложнее, чем просто взять другую реализацию, например tinyMCE, и интегрировать её в nicEdit. NicEdit — это побочный проект, у меня есть основная работа, поэтому у меня нет нескольких свободных недель, которые потребовались бы для этого. Но я буду продолжать работать, когда смогу, и буду рад любым исправлениям для nicXHTML, а также любым новым плагинам, которые работают лучше (я включу любые сторонние плагины в загрузчик).

#### Пример использования функции ''tidy'' в PHP для очистки кода:

script.js

```
tidy_repair_string(
         $_POST['body'],
            array(
                'show-body-only' => true,
                'doctype' => '-//W3C//DTD XHTML 1.0 Transitional//EN',
                'output-xhtml' => true
                  )
```

```
);
```

# Customization

## Developing with NicEdit

Разработка дополнительных плагинов и внесение других изменений в исходный код NicEdit не представляет сложности.

#### Ознакомьтесь с кодом.

Сначала получите копию исходного кода, клонировав репозиторий NicEdit из SVN.

svn co http://svn.nicedit.com/trunk src

#### Используйте Кодекс разработки

Для использования NicEdit с набором отдельных JavaScript-файлов для разработки вам потребуется включить их в раздел <head> создаваемой вами тестовой страницы. Вот пример для начала работы.

index.html

```
<html>
<head>

<script src="src/nicCore/bkLib.js" type="text/javascript"></script>
<script src="src/nicCore/nicConfig.js" type="text/javascript"></script>
<script src="src/nicCore/nicCore.js" type="text/javascript"></script>
<script src="src/nicCore/nicInstance.js"
type="text/javascript"></script>
<script src="src/nicCore/nicIFrameInstance.js"
type="text/javascript"></script>
<script src="src/nicCore/nicPanel.js" type="text/javascript"></script>
<script src="src/nicCore/nicButton.js" type="text/javascript"></script>
<script src="src/nicCore/nicPlugin.js" type="text/javascript"></script>
<script src="src/nicPane/nicPane.js" type="text/javascript"></script>
<script src="src/nicSelect/nicSelect.js"
type="text/javascript"></script>
<script src="src/nicButtonTips/nicButtonTips.js"
type="text/javascript"></script>
<script src="src/nicAdvancedButton/nicAdvancedButton.js"
type="text/javascript"></script>
```

```
<script src="src/nicLink/nicLink.js" type="text/javascript"></script>
<script src="src/nicImage/nicImage.js" type="text/javascript"></script>
<script src="src/nicCode/nicCode.js" type="text/javascript"></script>
<script src="src/nicColors/nicColors.js"
type="text/javascript"></script>

<script src="src/nicXHTML/nicXHTML.js" type="text/javascript"></script>
<script src="src/nicBBCode/nicBBCode.js"
type="text/javascript"></script>


</head>
<body>

<textarea style="width: 400px; height: 150px;" id="myArea2">
This is some TEST CONTENT
</textarea>

<script>
bkLib.onDomLoaded(function(){
  var myInstance = new nicEditor().panelInstance('myArea2');
});
</script>

</body>
</html>
```

## Creating a Plugin

Добавить плагин/кнопки в NiceDit несложно. Сначала убедитесь, что вы настроили свою среду разработки, а затем выполните следующие шаги:

1. Создайте новую папку для вашего плагина в каталоге src/, в этом примере я использовал nicExample.
2. Создайте файл с тем же именем в папке src/nicExample/nicExample.js
3. Вы можете использовать приведенный ниже код в качестве отправной точки для вашего плагина.

script.js

```
/**
* nicExample
* @description: An example button plugin for nicEdit
* @requires: nicCore, nicPane, nicAdvancedButton
* @author: Brian Kirchoff
* @version: 0.9.0
```

```
*/

/* START CONFIG */
var nicExampleOptions = {
    buttons : {
        'example' : {name : __('Some alt text for the button'), type :
'nicEditorExampleButton'}
    }/* NICEDIT_REMOVE_START */,iconFiles : {'example' :
'src/nicExample/icons/save.gif'}/* NICEDIT_REMOVE_END */
};
/* END CONFIG */

var nicEditorExampleButton = nicEditorButton.extend({
  mouseClick : function() {
    alert('The example save button icon has been clicked!');
  }
});

nicEditors.registerPlugin(nicPlugin,nicExampleOptions);
Car Shipping
```

- Скопируйте и вставьте код в свой JS-файл.
- Создайте папку icons в папке вашего плагина (например, src/nicExample/icons) и поместите туда свои иконки (в данном случае ).

**Чтобы объяснить различные части примера:**

```
/**
* nicExample
* @description: Пример плагина кнопок для nicEdit
* @requires: nicCore, nicPane, nicAdvancedButton
* @автор: Брайан Кирхофф
* @version: 0.9.0
*/
```

Это специальный блок комментариев, используемый скриптами сжатия nicEdit, и он должен присутствовать в каждом плагине, который необходимо включить в загрузчик.

Большинство полей понятны. Параметр @requires должен указывать, какие плагины являются зависимостями для вашего плагина (используется конфигуратором загрузки).

```
/* НАЧАЛО КОНФИГУРАЦИИ */
var nicExampleOptions = {
  кнопки: {
        'example' : {name : __('Некоторый альтернативный текст для кнопки'), type
: 'nicEditorExampleButton'}
    }/* NICEDIT_REMOVE_START */,iconFiles : {'example' :
'src/nicExample/icons/save.gif'}/* NICEDIT_REMOVE_END */
};
```

```
/* КОНЕЦ КОНФИГУРАЦИИ */
```

В разделе config вы настраиваете конфигурацию кнопок для вашего плагина. Параметр 'type' в объекте buttons указывает конкретный класс, который будет использоваться (по умолчанию это будет nicButton), куда вы поместите код ваших кнопок.

```
/* NICEDIT_REMOVE_START */
```

Блоки удаляются из выходных данных скриптами сборки.

В поле iconFiles следует указывать относительное местоположение отдельных значков кнопок в формате GIF, находящихся в разработке. В данном случае значение ключа 'example' и элемент iconFiles с указанным путем должны совпадать.

```
/* НАЧАЛО КОНФИГУРАЦИИ */
```

Используется только для предотвращения сжатия пробелов в разделах конфигурации скриптом сборки.

script.js

```
var nicEditorExampleButton = nicEditorButton.extend({
  mouseClick : function() {
    alert('Пример значка кнопки сохранения был нажат!');
  }
});
```

Это класс, который определяет вашу кнопку; в данном случае мы переопределяем метод `mouseClick()` для обработки нажатия кнопки.

Существует также класс `nicEditorAdvancedButton`, который могут использовать кнопки плагинов, которым требуется, чтобы функциональность панели имела форму (см. src/nicLink/nicLink.js) для примера того, как это можно сделать.
`nicEditors.registerPlugin(nicPlugin,nicExampleOptions);`
Это зарегистрирует ваш плагин, чтобы все новые созданные экземпляры могли его использовать (убедитесь, что вы используете `{fullPanel : true}` или задаете buttonList с идентификатором кнопки, в данном случае 'example'), и вы сможете увидеть свою новую кнопку на панели инструментов nicEdit.

**Вскоре:**

- Как использовать скрипты сборки для упаковки вашего плагина (а не просто запускать его как отдельный JS-файл) после того, как он заработает (пока что напишите мне на почту или опубликуйте свой плагин на форуме, если он окажется полезным, я даже включу его в официальный загрузчик Nicedit).

## Integration with NicEdit events

Ядро nicEdit отправляет несколько внутренних событий, которые можно использовать в плагинах и в вашем собственном JavaScript при расширенной интеграции редактора в ваши веб-сайты.

| | |
|---|---|
| blur | Отправляется, когда экземпляр редактора теряет фокус. |
| focus | Отправлять, когда редактор получает фокус (то есть, когда кто-то щелкает внутри него). |
| key | Когда пользователь нажимает сочетание клавиш (например, Ctrl+B). |
| add | Событие срабатывает при добавлении нового экземпляра. |
| panel | Событие срабатывает при инициализации панели инструментов для новых экземпляров (это предпочтительное событие, если вы хотите добавить функцию при создании экземпляров NicEdit). |

Для привязки функций к внутренним событиям NicEdit используйте метод `addEvent` объекта nicInstance.
`[some nicedit instance].addEvent([event name], [callback function]);`
События также часто используются для того, чтобы сообщить о завершении загрузки редактора.

script.js

```
bkLib.onDomLoaded(function(){
  var myEditor = new nicEditor({fullPanel : true
}).panelInstance('myArea2');
  myEditor.addEvent('add', function() {
    alert( myEditor.instanceById('myArea2').getContent() );
  });
});
```

Ещё один пример использования событий для определения момента размытия изображения в редакторе.

script.js

```
<script>
bkLib.onDomLoaded(function(){
  var myInstance = new nicEditor().panelInstance('myArea2');
  myInstance.addEvent('blur', function() {
    // Your code here that is called whenever the user blurs (stops
editing) the nicedit instance
  });
});
</script>
```

## How to translate nicEdit (with Spanish example)

- Скачать версию для разработчиков
- Откройте файл nicEdit.js
- Перейдите к этим нескольким строкам и отредактируйте текст справки, например, перевод на испанский будет выглядеть так:

**Примерно строка 252:**

script.js

```
var nicEditorConfig = bkClass.extend({
    buttons : {
        'bold' : {name : __('Negrita'), command : 'Bold', tags :
['B','STRONG'], css : {'font-weight' : 'bold'}, key : 'b'},
        'italic' : {name : __('Cursiva'), command : 'Italic', tags :
['EM','I'], css : {'font-style' : 'italic'}, key : 'i'},
        'underline' : {name : __('Subrayado'), command : 'Underline',
tags : ['U'], css : {'text-decoration' : 'underline'}, key : 'u'},
        'left' : {name : __('Alinear texto a la izquierda'), command :
'justifyleft', noActive : true},
        'center' : {name : __('Centrar texto'), command :
'justifycenter', noActive : true},
        'right' : {name : __('Alinear texto a la derecha'), command :
'justifyright', noActive : true},
        'justify' : {name : __('Justificar texto'), command :
'justifyfull', noActive : true},
        'ol' : {name : __('Insertar lista ordenada'), command :
'insertorderedlist', tags : ['OL']},
        'ul' :     {name : __('Insertar lista desordenada'), command :
'insertunorderedlist', tags : ['UL']},
        'subscript' : {name : __('SubIndice'), command : 'subscript',
tags : ['SUB']},
        'superscript' : {name : __('SuperIndice'), command :
'superscript', tags : ['SUP']},
        'strikethrough' : {name : __('Tachar'), command :
'strikeThrough', css : {'text-decoration' : 'line-through'}},
        'removeformat' : {name : __('Borrar formato'), command :
'removeformat', noActive : true},
        'indent' : {name : __('Identar'), command : 'indent', noActive
: true},
        'outdent' : {name : __('Eliminar identado'), command :
'outdent', noActive : true},
        'hr' : {name : __('Barra separadora'), command :
'insertHorizontalRule', noActive : true}
```

**Примерно строка 1046:**

script.js

```
var nicLinkOptions = {
    buttons : {
        'link' : {name : 'Enlazar', type : 'nicLinkButton', tags :
['A']},
        'unlink' : {name : 'Eliminar enlace',  command : 'unlink',
noActive : true}
    }
```

**Примерно строка 1120:**

script.js

```
var nicCodeOptions = {
    buttons : {
        'xhtml' : {name : 'Visualizar HTML', type : 'nicCodeButton'}
    }
};
```

**Примерно строка 1131:**

script.js

```
    addPane : function() {
        this.addForm({
            '' : {type : 'title', txt : 'Editar HTML'},
            'code' : {type : 'content', 'value' :
this.ne.selectedInstance.getContent(), style : {width: '340px', height
: '200px'}}
        });
    },
```

# Дополнения и Файлы

- Сжатый NicEdit (без пробелов и комментариев). Рекомендуется для развертывания NicEdit.
- Несжатая версия NicEdit для разработки. Используйте её только для отладки или разработки NicEdit.

- 
- 
- nicEdit-latest.js
- nicEdit.js
- 
-