

nicEdit.js

<https://cdnjs.cloudflare.com/ajax/libs/NicEdit/0.93/nicEdit.js>

nicEdit.js

```
/* NicEdit - Micro Inline WYSIWYG
 * Copyright 2007-2008 Brian Kirchoff
 *
 * NicEdit is distributed under the terms of the MIT license
 * For more information visit http://nicedit.com/
 * Do not remove this copyright message
 */
var bkExtend = function() {
    var args = arguments;
    if (args.length == 1) args = [this, args[0]];
    for (var prop in args[1]) args[0][prop] = args[1][prop];
    return args[0];
};

function bkClass() {}
bkClass.prototype.construct = function() {};
bkClass.extend = function(def) {
    var classDef = function() {
        if (arguments[0] !== bkClass) {
            return this.construct.apply(this, arguments);
        }
    };
    var proto = new this(bkClass);
    bkExtend(proto, def);
    classDef.prototype = proto;
    classDef.extend = this.extend;
    return classDef;
};

var bkElement = bkClass.extend({
    construct: function(elm, d) {
        if (typeof(elm) == "string") {
            elm = (d || document).createElement(elm);
        }
        elm = $BK(elm);
        return elm;
    },

    appendTo: function(elm) {
        elm.appendChild(this);
        return this;
    },
});
```

```
appendBefore: function(elm) {
    elm.parentNode.insertBefore(this, elm);
    return this;
},

addEvent: function(type, fn) {
    bkLib.addEvent(this, type, fn);
    return this;
},

setContent: function(c) {
    this.innerHTML = c;
    return this;
},

pos: function() {
    var curleft = curtop = 0;
    var o = obj = this;
    if (obj.offsetParent) {
        do {
            curleft += obj.offsetLeft;
            curtop += obj.offsetTop;
        } while (obj = obj.offsetParent);
    }
    var b = (!window.opera) ? parseInt(this.getStyle('border-width') ||
this.style.border, 10) || 0 : 0;
    return [curleft + b, curtop + b + this.offsetHeight];
},

noSelect: function() {
    bkLib.noSelect(this);
    return this;
},

parentTag: function(t) {
    var elm = this;
    do {
        if (elm && elm.nodeName && elm.nodeName.toUpperCase() == t) {
            return elm;
        }
        elm = elm.parentNode;
    } while (elm);
    return false;
},

hasClass: function(cls) {
    return this.className.match(new RegExp('(\\s|^)nicEdit-' + cls +
' (\\s|$)'));
},
```

```
addClass: function(cls) {
    if (!this.hasClass(cls)) {
        this.className += " nicEdit-" + cls;
    }
    return this;
},

removeClass: function(cls) {
    if (this.hasClass(cls)) {
        this.className = this.className.replace(new
RegExp('(\\s|^)nicEdit-' + cls + '(\\s|$)'), ' ');
    }
    return this;
},

setStyle: function(st) {
    var elmStyle = this.style;
    for (var itm in st) {
        switch (itm) {
            case 'float':
                elmStyle['cssFloat'] = elmStyle['styleFloat'] = st[itm];
                break;
            case 'opacity':
                elmStyle.opacity = st[itm];
                elmStyle.filter = "alpha(opacity=" + Math.round(st[itm] * 100)
+ ")";
                break;
            case 'className':
                this.className = st[itm];
                break;
            default:
                //if(document.compatMode || itm != "cursor") { // Nasty
Workaround for IE 5.5
                elmStyle[itm] = st[itm];
                //}
        }
    }
    return this;
},

getStyle: function(cssRule, d) {
    var doc = (!d) ? document.defaultView : d;
    if (this.nodeType == 1) {
        if (doc && doc.getComputedStyle) {
            return doc.getComputedStyle(this,
null).getPropertyValue(cssRule);
        } else {
            return this.currentStyle[bkLib.camelize(cssRule)];
        }
    }
},
```

```
remove: function() {
    this.parentNode.removeChild(this);
    return this;
},

setAttributes: function(at) {
    for (var itm in at) {
        this[itm] = at[itm];
    }
    return this;
}
});

var bkLib = {
    isMSIE: (navigator.appVersion.indexOf("MSIE") != -1),

    addEvent: function(obj, type, fn) {
        if (obj.addEventListener) {
            obj.addEventListener(type, fn, false);
        } else {
            obj.attachEvent("on" + type, fn);
        }
    },

    toArray: function(iterable) {
        var length = iterable.length,
            results = new Array(length);
        while (length--) {
            results[length] = iterable[length];
        }
        return results;
    },

    noSelect: function(element) {
        if (element.setAttribute && element.nodeName.toLowerCase() !=
'input' && element.nodeName.toLowerCase() != 'textarea') {
            element.setAttribute('unselectable', 'on');
        }
        for (var i = 0; i < element.childNodes.length; i++) {
            bkLib.noSelect(element.childNodes[i]);
        }
    },

    camelize: function(s) {
        return s.replace(/\\-(.)/g, function(m, l) {
            return l.toUpperCase();
        });
    },

    inArray: function(arr, item) {
        return (bkLib.search(arr, item) !== null);
    }
};
```

```
    },
    search: function(arr, itm) {
        for (var i = 0; i < arr.length; i++) {
            if (arr[i] == itm)
                return i;
        }
        return null;
    },
    cancelEvent: function(e) {
        e = e || window.event;
        if (e.preventDefault && e.stopPropagation) {
            e.preventDefault();
            e.stopPropagation();
        }
        return false;
    },
    domLoad: [],
    domLoaded: function() {
        if (arguments.callee.done) return;
        arguments.callee.done = true;
        for (i = 0; i < bkLib.domLoad.length; i++) bkLib.domLoad[i]();
    },
    onDomLoaded: function(fireThis) {
        this.domLoad.push(fireThis);
        if (document.addEventListener) {
            document.addEventListener("DOMContentLoaded", bkLib.domLoaded,
null);
        } else if (bkLib.isMSIE) {
            document.write("<style>.nicEdit-main p { margin: 0;
}</style><scr" + "ipt id=__ie_onload defer " + ((location.protocol ==
"https:") ? "src='javascript:void(0)'" : "src=//0") + "><\scr" +
"ipt>");
            $BK("__ie_onload").onreadystatechange = function() {
                if (this.readyState == "complete") {
                    bkLib.domLoaded();
                }
            };
        }
        window.onload = bkLib.domLoaded;
    }
};

function $BK(elm) {
    if (typeof(elm) == "string") {
        elm = document.getElementById(elm);
    }
    return (elm && !elm.appendTo) ? bkExtend(elm, bkElement.prototype) :
elm;
}

var bkEvent = {
```

```
addEvent: function(evType, evFunc) {
    if (evFunc) {
        this.eventList = this.eventList || {};
        this.eventList[evType] = this.eventList[evType] || [];
        this.eventList[evType].push(evFunc);
    }
    return this;
},
fireEvent: function() {
    var args = bkLib.toArray(arguments),
        evType = args.shift();
    if (this.eventList && this.eventList[evType]) {
        for (var i = 0; i < this.eventList[evType].length; i++) {
            this.eventList[evType][i].apply(this, args);
        }
    }
}
};

function __(s) {
    return s;
}

Function.prototype.closure = function() {
    var __method = this,
        args = bkLib.toArray(arguments),
        obj = args.shift();
    return function() {
        if (typeof(bkLib) != 'undefined') {
            return __method.apply(obj,
args.concat(bkLib.toArray(arguments)));
        }
    };
};

Function.prototype.closureListener = function() {
    var __method = this,
        args = bkLib.toArray(arguments),
        object = args.shift();
    return function(e) {
        var target;
        e = e || window.event;
        if (e.target) {
            target = e.target;
        } else {
            target = e.srcElement;
        }
        return __method.apply(object, [e, target].concat(args));
    };
};
```

```
/* START CONFIG */

var nicEditorConfig = bkClass.extend({
  buttons: {
    'bold': {
      name: __('Click to Bold'),
      command: 'Bold',
      tags: ['B', 'STRONG'],
      css: {
        'font-weight': 'bold'
      },
      key: 'b'
    },
    'italic': {
      name: __('Click to Italic'),
      command: 'Italic',
      tags: ['EM', 'I'],
      css: {
        'font-style': 'italic'
      },
      key: 'i'
    },
    'underline': {
      name: __('Click to Underline'),
      command: 'Underline',
      tags: ['U'],
      css: {
        'text-decoration': 'underline'
      },
      key: 'u'
    },
    'left': {
      name: __('Left Align'),
      command: 'justifyleft',
      noActive: true
    },
    'center': {
      name: __('Center Align'),
      command: 'justifycenter',
      noActive: true
    },
    'right': {
      name: __('Right Align'),
      command: 'justifyright',
      noActive: true
    },
    'justify': {
      name: __('Justify Align'),
      command: 'justifyfull',
```

```
    noActive: true
  },
  'ol': {
    name: __('Insert Ordered List'),
    command: 'insertorderedlist',
    tags: ['OL']
  },
  'ul': {
    name: __('Insert Unordered List'),
    command: 'insertunorderedlist',
    tags: ['UL']
  },
  'subscript': {
    name: __('Click to Subscript'),
    command: 'subscript',
    tags: ['SUB']
  },
  'superscript': {
    name: __('Click to Superscript'),
    command: 'superscript',
    tags: ['SUP']
  },
  'strikethrough': {
    name: __('Click to Strike Through'),
    command: 'strikeThrough',
    css: {
      'text-decoration': 'line-through'
    }
  },
  'removeformat': {
    name: __('Remove Formatting'),
    command: 'removeformat',
    noActive: true
  },
  'indent': {
    name: __('Indent Text'),
    command: 'indent',
    noActive: true
  },
  'outdent': {
    name: __('Remove Indent'),
    command: 'outdent',
    noActive: true
  },
  'hr': {
    name: __('Horizontal Rule'),
    command: 'insertHorizontalRule',
    noActive: true
  }
},
```

```
    iconsPath: '../nicEditorIcons.gif',
    buttonList: ['save', 'bold', 'italic', 'underline', 'left', 'center',
'right', 'justify', 'ol', 'ul', 'fontSize', 'fontFamily', 'fontFormat',
'indent', 'outdent', 'image', 'upload', 'link', 'unlink', 'forecolor',
'bgcolor'],
    iconList: {
        "xhtml": 1,
        "bgcolor": 2,
        "forecolor": 3,
        "bold": 4,
        "center": 5,
        "hr": 6,
        "indent": 7,
        "italic": 8,
        "justify": 9,
        "left": 10,
        "ol": 11,
        "outdent": 12,
        "removeformat": 13,
        "right": 14,
        "save": 25,
        "strikethrough": 16,
        "subscript": 17,
        "superscript": 18,
        "ul": 19,
        "underline": 20,
        "image": 21,
        "link": 22,
        "unlink": 23,
        "close": 24,
        "arrow": 26,
        "upload": 27
    },
    initWithLineBreak: true
});
/* END CONFIG */

var nicEditors = {
    nicPlugins: [],
    editors: [],

    registerPlugin: function(plugin, options) {
        this.nicPlugins.push({
            p: plugin,
            o: options
        });
    },

    allTextAreas: function(nicOptions) {
        var textareas = document.getElementsByTagName("textarea");
```

```
        for (var i = 0; i < textareas.length; i++) {
            nicEditors.editors.push(new
nicEditor(nicOptions).panelInstance(textareas[i]));
        }
        return nicEditors.editors;
    },

    findEditor: function(e) {
        var editors = nicEditors.editors;
        for (var i = 0; i < editors.length; i++) {
            if (editors[i].instanceById(e)) {
                return editors[i]; // r is an instance of nicEditorInstance
therefore it does not have removeInstance or removePanel methods
            }
        }
    }
};

var nicEditor = bkClass.extend({
    construct: function(o) {
        this.options = new nicEditorConfig();
        bkExtend(this.options, o);
        this.nicInstances = [];
        this.loadedPlugins = [];

        var plugins = nicEditors.nicPlugins;
        for (var i = 0; i < plugins.length; i++) {
            this.loadedPlugins.push(new plugins[i].p(this, plugins[i].o));
        }
        nicEditors.editors.push(this);
        bkLib.addEvent(document.body, 'mousedown',
this.selectCheck.closureListener(this));
    },

    panelInstance: function(e, o) {
        e = this.checkReplace($BK(e));
        var panelElm = new bkElement('DIV').setStyle({
            width: (parseInt(e.getStyle('width'), 10) || e.clientWidth) +
'px'
        }).appendBefore(e);
        this.setPanel(panelElm);
        return this.addInstance(e, o);
    },

    checkReplace: function(e) {
        var r = nicEditors.findEditor(e);
        if (r) {
            r.removeInstance(e);
            r.removePanel();
        }
    }
});
```

```
    }
    return e;
  },

  addInstance: function(e, o) {
    var newInstance;
    e = this.checkReplace($BK(e));
    if (e.contentEditable || !!window.opera) {
      newInstance = new nicEditorInstance(e, o, this);
    } else {
      newInstance = new nicEditorIFrameInstance(e, o, this);
    }
    this.nicInstances.push(newInstance);
    return this;
  },

  removeInstance: function(e) {
    e = $BK(e);
    var instances = this.nicInstances;
    for (var i = 0; i < instances.length; i++) {
      if (instances[i].e == e) {
        instances[i].remove();
        this.nicInstances.splice(i, 1);
      }
    }
  },

  removePanel: function(e) {
    if (this.nicPanel) {
      this.nicPanel.remove();
      this.nicPanel = null;
    }
  },

  instanceById: function(e) {
    e = $BK(e);
    var instances = this.nicInstances;
    for (var i = 0; i < instances.length; i++) {
      if (instances[i].e == e) {
        return instances[i];
      }
    }
  },

  setPanel: function(e) {
    this.nicPanel = new nicEditorPanel($BK(e), this.options, this);
    this.fireEvent('panel', this.nicPanel);
    return this;
  },

  nicCommand: function(cmd, args) {
```

```
        if (this.selectedInstance) {
            this.selectedInstance.nicCommand(cmd, args);
        }
    },

    getIcon: function(iconName, options) {
        var icon = this.options.iconList[iconName];
        var file = (options.iconFiles) ? options.iconFiles[iconName] : '';
        return {
            backgroundImage: "url('" + ((icon) ? this.options.iconsPath :
file) + "')",
            backgroundPosition: ((icon) ? ((icon - 1) * -18) : 0) + 'px 0px'
        };
    },

    selectCheck: function(e, t) {
        var found = false;
        do {
            if (t.className && t.className.indexOf('nicEdit') != -1) {
                return false;
            }
        } while (t = t.parentNode);
        this.fireEvent('blur', this.selectedInstance, t);
        this.lastSelectedInstance = this.selectedInstance;
        this.selectedInstance = null;
        return false;
    }

});
nicEditor = nicEditor.extend(bkEvent);

var nicEditorInstance = bkClass.extend({
    isSelected: false,

    construct: function(e, options, nicEditor) {
        this.ne = nicEditor;
        this.elm = this.e = e;
        this.options = options || {};

        newX = parseInt(e.getStyle('width'), 10) || e.clientWidth;
        newY = parseInt(e.getStyle('height'), 10) || e.clientHeight;
        this.initialHeight = newY - 8;

        var isTextarea = (e.nodeName.toLowerCase() == "textarea");
        if (isTextarea || this.options.hasPanel) {
            var ie7s = (bkLib.isMSIE && !(typeof
document.body.style.maxHeight != "undefined") && document.compatMode ==
"CSS1Compat"));
            var s = {
```

```
        width: newX + 'px',
        border: '1px solid #ccc',
        borderTop: 0,
        overflowY: 'auto',
        overflowX: 'hidden'
    });
    s[(ie7s) ? 'height' : 'maxHeight'] = (this.ne.options.maxHeight)
    ? this.ne.options.maxHeight + 'px' : null;
    this.editorContain = new
    bkElement('DIV').setStyle(s).appendBefore(e);
    var editorElm = new bkElement('DIV').setStyle({
        width: (newX - 8) + 'px',
        margin: '4px',
        minHeight: newY + 'px'
    }).addClass('main').appendTo(this.editorContain);

    e.setStyle({
        display: 'none'
    });

    editorElm.innerHTML = e.innerHTML;
    if (isTextarea) {
        this.copyElm = e;
        var f = e.parentTag('FORM');
        if (f) {
            bkLib.addEvent(f, 'submit', this.saveContent.closure(this));
        }
    }
    editorElm.setStyle((ie7s) ? {
        height: newY + 'px'
    } : {
        overflow: 'hidden'
    });
    this.elm = editorElm;
}
this.ne.addEvent('blur', this.blur.closure(this));

this.init();
this.blur();
},

init: function() {
    this.elm.setAttribute('contentEditable', 'true');
    if (this.getContent() === "" && this.options.initWithLineBreak) {
        this.setContent('<br />');
    }
    this.instanceDoc = document.defaultView;
    this.elm.addEvent('mousedown',
    this.selected.closureListener(this)).addEvent('keypress',
    this.keyDown.closureListener(this)).addEvent('focus',
    this.selected.closure(this)).addEvent('blur',
```

```
this.blur.closure(this)).addEvent('keyup',
this.selected.closure(this));
    this.ne.fireEvent('add', this);
},

remove: function() {
    this.saveContent();
    if (this.copyElm || this.options.hasPanel) {
        this.editorContain.remove();
        this.e.setStyle({
            'display': 'block'
        });
        this.ne.removePanel();
    }
    this.disable();
    this.ne.fireEvent('remove', this);
},

disable: function() {
    this.elm.setAttribute('contentEditable', 'false');
},

getSel: function() {
    return (window.getSelection() ? window.getSelection() :
document.selection);
},

getRng: function() {
    var s = this.getSel();
    if (!s) {
        return null;
    }
    return (s.rangeCount > 0) ? s.getRangeAt(0) :
        s.createRange && s.createRange() || document.createRange();
},

selRng: function(rng, s) {
    if (window.getSelection) {
        s.removeAllRanges();
        s.addRange(rng);
    } else {
        rng.select();
    }
},

selElm: function() {
    var r = this.getRng();
    if (r.startContainer) {
        var contain = r.startContainer;
        if (r.cloneContents().childNodes.length == 1) {
```

```
        for (var i = 0; i < contain.childNodes.length; i++) {
            var rng = contain.childNodes[i].ownerDocument.createRange();
            rng.selectNode(contain.childNodes[i]);
            if (r.compareBoundaryPoints(Range.START_TO_START, rng) != 1
&&
                r.compareBoundaryPoints(Range.END_TO_END, rng) != -1) {
                return $BK(contain.childNodes[i]);
            }
        }
        return $BK(contain);
    } else {
        return $BK((this.getSel().type == "Control") ? r.item(0) :
r.parentElement());
    }
},

saveRng: function() {
    this.savedRange = this.getRng();
    this.savedSel = this.getSel();
},

restoreRng: function() {
    if (this.savedRange) {
        this.selRng(this.savedRange, this.savedSel);
    }
},

keyDown: function(e, t) {
    this.ne.fireEvent('keyDown', this, e);

    if (e.ctrlKey) {
        this.ne.fireEvent('key', this, e);
    }
},

selected: function(e, t) {
    if (!t) {
        t = this.selElm();
    }
    if (!e.ctrlKey) {
        var selInstance = this.ne.selectedInstance;
        if (selInstance != this) {
            if (selInstance) {
                this.ne.fireEvent('blur', selInstance, t);
            }
            this.ne.selectedInstance = this;
            this.ne.fireEvent('focus', selInstance, t);
        }
        this.ne.fireEvent('selected', selInstance, t);
        this.isFocused = true;
    }
}
```

```
        this.elm.addClass('selected');
    }
    return false;
},

blur: function() {
    this.isFocused = false;
    this.elm.removeClass('selected');
},

saveContent: function() {
    if (this.copyElm || this.options.hasPanel) {
        this.ne.fireEvent('save', this);
        if (this.copyElm) {
            this.copyElm.value = this.getContent();
        } else {
            this.e.innerHTML = this.getContent();
        }
    }
},

getElm: function() {
    return this.elm;
},

getContent: function() {
    this.content = this.getElm().innerHTML;
    this.ne.fireEvent('get', this);
    return this.content;
},

setContent: function(e) {
    this.content = e;
    this.ne.fireEvent('set', this);
    this.elm.innerHTML = this.content;
},

nicCommand: function(cmd, args) {
    document.execCommand(cmd, false, args);
}
});

var nicEditorIFrameInstance = nicEditorInstance.extend({
    savedStyles: [],

    init: function() {
        var c = this.elm.innerHTML.replace(/^\\s+|\\s+$/g, '');
        this.elm.innerHTML = '';
        if (!c) {
            c = "<br />";
        }
    }
});
```

```
    }
    this.initialContent = c;

    this.elmFrame = new bkElement('iframe').setAttributes({
        'src': 'javascript:;',
        'frameBorder': 0,
        'allowTransparency': 'true',
        'scrolling': 'no'
    }).setStyle({
        height: '100px',
        width: '100%'
    }).addClass('frame').appendTo(this.elm);

    if (this.copyElm) {
        this.elmFrame.setStyle({
            width: (this.elm.offsetWidth - 4) + 'px'
        });
    }

    var styleList = ['font-size', 'font-family', 'font-weight',
'color'];
    for (var itm in styleList) {
        this.savedStyles[bkLib.camelize(itm)] = this.elm.getStyle(itm);
    }

    setTimeout(this.initFrame.closure(this), 50);
},

disable: function() {
    this.elm.innerHTML = this.getContent();
},

initFrame: function() {
    var fd = $BK(this.elmFrame.contentWindow.document);
    fd.designMode = "on";
    fd.open();
    var css = this.ne.options.externalCSS;
    fd.write('<html><head>' + ((css) ? '<link href="' + css + '"
rel="stylesheet" type="text/css" />' : '') + '</head><body
id="nicEditContent" style="margin: 0 !important; background-color:
transparent !important;">' + this.initialContent + '</body></html>');
    fd.close();
    this.frameDoc = fd;

    this.frameWin = $BK(this.elmFrame.contentWindow);
    this.frameContent =
$BK(this.frameWin.document.body).setStyle(this.savedStyles);
    this.instanceDoc = this.frameWin.document.defaultView;

    this.heightUpdate();
    this.frameDoc.addEvent('mousedown',
```

```
this.selected.closureListener(this)).addEvent('keyup',
this.heightUpdate.closureListener(this)).addEvent('keydown',
this.keyDown.closureListener(this)).addEvent('keyup',
this.selected.closure(this));
    this.ne.fireEvent('add', this);
},

getElm: function() {
    return this.frameContent;
},

setContent: function(c) {
    this.content = c;
    this.ne.fireEvent('set', this);
    this.frameContent.innerHTML = this.content;
    this.heightUpdate();
},

getSel: function() {
    return (this.frameWin) ? this.frameWin.getSelection() :
this.frameDoc.selection;
},

heightUpdate: function() {
    this.elmFrame.style.height =
Math.max(this.frameContent.offsetHeight, this.initialHeight) + 'px';
},

nicCommand: function(cmd, args) {
    this.frameDoc.execCommand(cmd, false, args);
    setTimeout(this.heightUpdate.closure(this), 100);
}

});
var nicEditorPanel = bkClass.extend({
    construct: function(e, options, nicEditor) {
        this.elm = e;
        this.options = options;
        this.ne = nicEditor;
        this.panelButtons = [];
        this.buttonList = bkExtend([], this.ne.options.buttonList);

        this.panelContain = new bkElement('DIV').setStyle({
            overflow: 'hidden',
            width: '100%',
            border: '1px solid #cccccc',
            backgroundColor: '#efefef'
        }).addClass('panelContain');
        this.panelElm = new bkElement('DIV').setStyle({
```

```
        margin: '2px',
        marginTop: '0px',
        zoom: 1,
        overflow: 'hidden'
    }).addClass('panel').appendTo(this.panelContain);
    this.panelContain.appendTo(e);

    var opt = this.ne.options;
    var buttons = opt.buttons;
    for (var button in buttons) {
        this.addButton(button, opt, true);
    }
    this.reorder();
    e.noSelect();
},

addButton: function(buttonName, options, noOrder) {
    var button = options.buttons[buttonName];
    var type = null;

    if (button['type']) {
        type = typeof(window[button['type']]) === undefined ? null :
window[button['type']];
    } else {
        type = nicEditorButton;
    }
    var hasButton = bkLib.inArray(this.buttonList, buttonName);
    if (type && (hasButton || this.ne.options.fullPanel)) {
        this.panelButtons.push(new type(this.panelElm, buttonName,
options, this.ne));
        if (!hasButton) {
            this.buttonList.push(buttonName);
        }
    }
},

findButton: function(itm) {
    for (var i = 0; i < this.panelButtons.length; i++) {
        if (this.panelButtons[i].name == itm)
            return this.panelButtons[i];
    }
},

reorder: function() {
    var bl = this.buttonList;
    for (var i = 0; i < bl.length; i++) {
        var button = this.findButton(bl[i]);
        if (button) {
            this.panelElm.appendChild(button.margin);
        }
    }
}
```

```
    },  
  
    remove: function() {  
        this.elm.remove();  
    }  
});  
var nicEditorButton = bkClass.extend({  
  
    construct: function(e, buttonName, options, nicEditor) {  
        this.options = options.buttons[buttonName];  
        this.name = buttonName;  
        this.ne = nicEditor;  
        this.elm = e;  
  
        this.margin = new bkElement('DIV').setStyle({  
            'float': 'left',  
            'marginTop': '2px'  
        }).appendTo(e);  
        this.contain = new bkElement('DIV').setStyle({  
            'width': '20px',  
            'height': '20px'  
        }).addClass('buttonContain').appendTo(this.margin);  
        this.border = new bkElement('DIV').setStyle({  
            'backgroundColor': '#efefef',  
            'border': '1px solid #efefef'  
        }).appendTo(this.contain);  
        this.button = new bkElement('DIV').setStyle({  
            'width': '18px',  
            'height': '18px',  
            'overflow': 'hidden',  
            'zoom': 1,  
            'cursor': 'pointer'  
        }).addClass('button').setStyle(this.ne.getIcon(buttonName,  
options)).appendTo(this.border);  
        this.button.addEvent('mouseover',  
this.hoverOn.closure(this)).addEvent('mouseout',  
this.hoverOff.closure(this)).addEvent('mousedown',  
this.mouseClick.closure(this)).noSelect();  
  
        if (!window.opera) {  
            this.button.onmousedown = this.button.onclick =  
bkLib.cancelEvent;  
        }  
  
        nicEditor.addEvent('selected',  
this.enable.closure(this)).addEvent('blur',  
this.disable.closure(this)).addEvent('key', this.key.closure(this));  
  
        this.disable();  
        this.init();  
    }  
});
```

```
    },

    init: function() {},

    hide: function() {
        this.contain.setStyle({
            display: 'none'
        });
    },

    updateState: function() {
        if (this.isDisabled) {
            this.setBg();
        } else if (this.isHover) {
            this.setBg('hover');
        } else if (this.isActive) {
            this.setBg('active');
        } else {
            this.setBg();
        }
    },

    setBg: function(state) {
        var stateStyle;
        switch (state) {
            case 'hover':
                stateStyle = {
                    border: '1px solid #666',
                    backgroundColor: '#ddd'
                };
                break;
            case 'active':
                stateStyle = {
                    border: '1px solid #666',
                    backgroundColor: '#ccc'
                };
                break;
            default:
                stateStyle = {
                    border: '1px solid #efefef',
                    backgroundColor: '#efefef'
                };
        }
        this.border.setStyle(stateStyle).addClass('button-' + state);
    },

    checkNodes: function(e) {
        var elm = e;
        do {
            if (this.options.tags && bkLib.inArray(this.options.tags,
                elm.nodeName)) {
```

```
        this.activate();
        return true;
    }
    } while ((elm = elm.parentNode) && elm.className != "nicEdit");
    elm = $BK(e);
    while (elm.nodeType == 3) {
        elm = $BK(elm.parentNode);
    }
    if (this.options.css) {
        for (var itm in this.options.css) {
            if (elm.getStyle(itm, this.ne.selectedInstance.instanceDoc) ==
this.options.css[itm]) {
                this.activate();
                return true;
            }
        }
    }
    this.deactivate();
    return false;
},

activate: function() {
    if (!this.isDisabled) {
        this.isActive = true;
        this.updateState();
        this.ne.fireEvent('buttonActivate', this);
    }
},

deactivate: function() {
    this.isActive = false;
    this.updateState();
    if (!this.isDisabled) {
        this.ne.fireEvent('buttonDeactivate', this);
    }
},

enable: function(ins, t) {
    this.isDisabled = false;
    this.contain.setStyle({
        'opacity': 1
    }).addClass('buttonEnabled');
    this.updateState();
    if (t !== document) {
        this.checkNodes(t);
    }
},

disable: function(ins, t) {
    this.isDisabled = true;
```

```
        this.contain.setStyle({
            'opacity': 0.6
        }).removeClass('buttonEnabled');
        this.updateState();
    },

    toggleActive: function() {
        if (this.isActive) {
            this.deactivate();
        } else {
            this.activate();
        }
    },

    hoverOn: function() {
        if (!this.isDisabled) {
            this.isHover = true;
            this.updateState();
            this.ne.fireEvent("buttonOver", this);
        }
    },

    hoverOff: function() {
        this.isHover = false;
        this.updateState();
        this.ne.fireEvent("buttonOut", this);
    },

    mouseClick: function() {
        if (this.options.command) {
            this.ne.nicCommand(this.options.command,
this.options.commandArgs);
            if (!this.options.noActive) {
                this.toggleActive();
            }
        }
        this.ne.fireEvent("buttonClick", this);
    },

    key: function(nicInstance, e) {
        if (this.options.key && e.ctrlKey && String.fromCharCode(e.keyCode
|| e.charCode).toLowerCase() == this.options.key) {
            this.mouseClick();
            if (e.preventDefault) e.preventDefault();
        }
    }
});

var nicPlugin = bkClass.extend({
```

```
construct: function(nicEditor, options) {
    this.options = options;
    this.ne = nicEditor;
    this.ne.addEvent('panel', this.loadPanel.closure(this));

    this.init();
},

loadPanel: function(np) {
    var buttons = this.options.buttons;
    for (var button in buttons) {
        np.addButton(button, this.options);
    }
    np.reorder();
},

init: function() {}
});

/* START CONFIG */
var nicPaneOptions = {};
/* END CONFIG */

var nicEditorPane = bkClass.extend({
    construct: function(elm, nicEditor, options, openButton) {
        this.ne = nicEditor;
        this.elm = elm;
        this.pos = elm.pos();

        this.contain = new bkElement('div').setStyle({
            zIndex: '99999',
            overflow: 'hidden',
            position: 'absolute',
            left: this.pos[0] + 'px',
            top: this.pos[1] + 'px'
        });
        this.pane = new bkElement('div').setStyle({
            fontSize: '12px',
            border: '1px solid #ccc',
            overflow: 'hidden',
            padding: '4px',
            textAlign: 'left',
            backgroundColor: '#ffffc9'
        }).addClass('pane').setStyle(options).appendTo(this.contain);

        if (openButton && !openButton.options.noClose) {
```

```
        this.close = new bkElement('div').setStyle({
            'float': 'right',
            height: '16px',
            width: '16px',
            cursor: 'pointer'
        }).setStyle(this.ne.getIcon('close',
nicPaneOptions)).addEvent('mousedown',
openButton.removePane.closure(this)).appendTo(this.pane);
    }

    this.contain.noSelect().appendTo(document.body);

    this.position();
    this.init();
},

init: function() {},

position: function() {
    if (this.ne.nicPanel) {
        var panelElm = this.ne.nicPanel.elm;
        var panelPos = panelElm.pos();
        var newLeft = panelPos[0] + parseInt(panelElm.getStyle('width'),
10) - (parseInt(this.pane.getStyle('width'), 10) + 8);
        if (newLeft < this.pos[0]) {
            this.contain.setStyle({
                left: newLeft + 'px'
            });
        }
    }
},

toggle: function() {
    this.isVisible = !this.isVisible;
    this.contain.setStyle({
        display: ((this.isVisible) ? 'block' : 'none')
    });
},

remove: function() {
    if (this.contain) {
        this.contain.remove();
        this.contain = null;
    }
},

append: function(c) {
    c.appendTo(this.pane);
},

setContent: function(c) {
```

```
        this.pane.setContent(c);
    }

});

var nicEditorAdvancedButton = nicEditorButton.extend({

    init: function() {
        this.ne.addEvent('selected',
this.removePane.closure(this)).addEvent('blur',
this.removePane.closure(this));
    },

    mouseClick: function() {
        if (!this.isDisabled) {
            if (this.pane && this.pane.pane) {
                this.removePane();
            } else {
                this.pane = new nicEditorPane(this.contain, this.ne, {
                    width: (this.width || '270px'),
                    backgroundColor: '#fff'
                }, this);
                this.addPane();
                this.ne.selectedInstance.saveRng();
            }
        }
    },

    addForm: function(f, elm) {
        this.form = new bkElement('form').addEvent('submit',
this.submit.closureListener(this));
        this.pane.append(this.form);
        this.inputs = {};

        for (var itm in f) {
            var field = f[itm];
            var val = '';
            if (elm) {
                val = elm.getAttribute(itm);
            }
            if (!val) {
                val = field['value'] || '';
            }
            var type = f[itm].type;

            if (type == 'title') {
                new bkElement('div').setContent(field.txt).setStyle({
                    fontSize: '14px',
```

```
        fontWeight: 'bold',
        padding: '0px',
        margin: '2px 0'
    }).appendTo(this.form);
} else {
    var contain = new bkElement('div').setStyle({
        overflow: 'hidden',
        clear: 'both'
    }).appendTo(this.form);
    if (field.txt) {
        new bkElement('label').setAttributes({
            'for': itm
        }).setContent(field.txt).setStyle({
            margin: '2px 4px',
            fontSize: '13px',
            width: '50px',
            lineHeight: '20px',
            textAlign: 'right',
            'float': 'left'
        }).appendTo(contain);
    }

    switch (type) {
    case 'text':
        this.inputs[itm] = new bkElement('input').setAttributes({
            id: itm,
            'value': val,
            'type': 'text'
        }).setStyle({
            margin: '2px 0',
            fontSize: '13px',
            'float': 'left',
            height: '20px',
            border: '1px solid #ccc',
            overflow: 'hidden'
        }).setStyle(field.style).appendTo(contain);
        break;
    case 'select':
        this.inputs[itm] = new bkElement('select').setAttributes({
            id: itm
        }).setStyle({
            border: '1px solid #ccc',
            'float': 'left',
            margin: '2px 0'
        }).appendTo(contain);
        for (var opt in field.options) {
            var o = new bkElement('option').setAttributes({
                value: opt,
                selected: (opt == val) ? 'selected' : ''
            }).setContent(field.options[opt]).appendTo(this.inputs[itm]);
        }
    }
}
```

```
        break;
    case 'content':
        this.inputs[itm] = new bkElement('textarea').setAttributes({
            id: itm
        }).setStyle({
            border: '1px solid #ccc',
            'float': 'left'
        }).setStyle(field.style).appendTo(contain);
        this.inputs[itm].value = val;
    }
}
new bkElement('input').setAttributes({
    'type': 'submit'
}).setStyle({
    backgroundColor: '#efefef',
    border: '1px solid #ccc',
    margin: '3px 0',
    'float': 'left',
    'clear': 'both'
}).appendTo(this.form);
this.form.onsubmit = bkLib.cancelEvent;
},

submit: function() {},

findElm: function(tag, attr, val) {
    var list =
this.ne.selectedInstance.getElm().getElementsByTagName(tag);
    for (var i = 0; i < list.length; i++) {
        if (list[i].getAttribute(attr) == val) {
            return $BK(list[i]);
        }
    }
},

removePane: function() {
    if (this.pane) {
        this.pane.remove();
        this.pane = null;
        this.ne.selectedInstance.restoreRng();
    }
}
});

var nicButtonTips = bkClass.extend({
    construct: function(nicEditor) {
        this.ne = nicEditor;
        nicEditor.addEvent('buttonOver',
```

```
this.show.closure(this)).addEvent('buttonOut',
this.hide.closure(this));

},

show: function(button) {
    this.timer = setTimeout(this.create.closure(this, button), 400);
},

create: function(button) {
    this.timer = null;
    if (!this.pane) {
        this.pane = new nicEditorPane(button.button, this.ne, {
            fontSize: '12px',
            marginTop: '5px'
        });
        this.pane.setContent(button.options.name);
    }
},

hide: function(button) {
    if (this.timer) {
        clearTimeout(this.timer);
    }
    if (this.pane) {
        this.pane = this.pane.remove();
    }
}
});
nicEditors.registerPlugin(nicButtonTips);

/* START CONFIG */
var nicSelectOptions = {
    buttons: {
        'fontSize': {
            name: __('Select Font Size'),
            type: 'nicEditorFontSizeSelect',
            command: 'fontsize'
        },
        'fontFamily': {
            name: __('Select Font Family'),
            type: 'nicEditorFontFamilySelect',
            command: 'fontname'
        },
        'fontFormat': {
            name: __('Select Font Format'),
            type: 'nicEditorFontFormatSelect',
            command: 'formatBlock'
        }
    }
}
```

```
    }  
};  
/* END CONFIG */  
var nicEditorSelect = bkClass.extend({  
  
    construct: function(e, buttonName, options, nicEditor) {  
        this.options = options.buttons[buttonName];  
        this.elm = e;  
        this.ne = nicEditor;  
        this.name = buttonName;  
        this.selOptions = [];  
  
        this.margin = new bkElement('div').setStyle({  
            'float': 'left',  
            margin: '2px 1px 0 1px'  
        }).appendTo(this.elm);  
        this.contain = new bkElement('div').setStyle({  
            width: '90px',  
            height: '20px',  
            cursor: 'pointer',  
            overflow: 'hidden'  
        }).addClass('selectContain').addEvent('click',  
this.toggle.closure(this)).appendTo(this.margin);  
        this.items = new bkElement('div').setStyle({  
            overflow: 'hidden',  
            zoom: 1,  
            border: '1px solid #ccc',  
            paddingLeft: '3px',  
            backgroundColor: '#fff'  
        }).appendTo(this.contain);  
        this.control = new bkElement('div').setStyle({  
            overflow: 'hidden',  
            'float': 'right',  
            height: '18px',  
            width: '16px'  
        }).addClass('selectControl').setStyle(this.ne.getIcon('arrow',  
options)).appendTo(this.items);  
        this.txt = new bkElement('div').setStyle({  
            overflow: 'hidden',  
            'float': 'left',  
            width: '66px',  
            height: '14px',  
            marginTop: '1px',  
            fontFamily: 'sans-serif',  
            textAlign: 'center',  
            fontSize: '12px'  
        }).addClass('selectTxt').appendTo(this.items);  
  
        if (!window.opera) {  
            this.contain.onmousedown = this.control.onmousedown =
```

```
this.txt.onmousedown = bkLib.cancelEvent;
}

this.margin.noSelect();

this.ne.addEvent('selected',
this.enable.closure(this)).addEvent('blur',
this.disable.closure(this));

this.disable();
this.init();
},

disable: function() {
    this.isDisabled = true;
    this.close();
    this.contain.setStyle({
        opacity: 0.6
    });
},

enable: function(t) {
    this.isDisabled = false;
    this.close();
    this.contain.setStyle({
        opacity: 1
    });
},

setDisplay: function(txt) {
    this.txt.setContent(txt);
},

toggle: function() {
    if (!this.isDisabled) {
        if (this.pane) {
            this.close();
        } else {
            this.open();
        }
    }
},

open: function() {
    this.pane = new nicEditorPane(this.items, this.ne, {
        width: '88px',
        padding: '0px',
        borderTop: 0,
        borderLeft: '1px solid #ccc',
        borderRight: '1px solid #ccc',
        borderBottom: '0px',
```

```
        backgroundColor: '#fff'
    });

    for (var i = 0; i < this.selOptions.length; i++) {
        var opt = this.selOptions[i];
        var itmContain = new bkElement('div').setStyle({
            overflow: 'hidden',
            borderBottom: '1px solid #ccc',
            width: '88px',
            textAlign: 'left',
            cursor: 'pointer'
        });
        var itm = new bkElement('div').setStyle({
            padding: '0px 4px'
        }).setContent(opt[1]).appendTo(itmContain).noSelect();
        itm.addEvent('click', this.update.closure(this,
opt[0])).addEvent('mouseover', this.over.closure(this,
itm)).addEvent('mouseout', this.out.closure(this,
itm)).setAttributes('id', opt[0]);
        this.pane.append(itmContain);
        if (!window.opera) {
            itm.onmousedown = bkLib.cancelEvent;
        }
    }
},

close: function() {
    if (this.pane) {
        this.pane = this.pane.remove();
    }
},

over: function(opt) {
    opt.setStyle({
        backgroundColor: '#ccc'
    });
},

out: function(opt) {
    opt.setStyle({
        backgroundColor: '#fff'
    });
},

add: function(k, v) {
    this.selOptions.push(new Array(k, v));
},

update: function(elm) {
```

```
        this.ne.nicCommand(this.options.command, elm);
        this.close();
    }
});

var nicEditorFontSizeSelect = nicEditorSelect.extend({
    sel: {
        1: '1&nbsp;(8pt)',
        2: '2&nbsp;(10pt)',
        3: '3&nbsp;(12pt)',
        4: '4&nbsp;(14pt)',
        5: '5&nbsp;(18pt)',
        6: '6&nbsp;(24pt)'
    },
    init: function() {
        this.setDisplay('Font&nbsp;Size...');
        for (var itm in this.sel) {
            this.add(itm, '<font size="' + itm + '">' + this.sel[itm] +
'</font>');
        }
    }
});

var nicEditorFontFamilySelect = nicEditorSelect.extend({
    sel: {
        'arial': 'Arial',
        'comic sans ms': 'Comic Sans',
        'courier new': 'Courier New',
        'georgia': 'Georgia',
        'helvetica': 'Helvetica',
        'impact': 'Impact',
        'times new roman': 'Times',
        'trebuchet ms': 'Trebuchet',
        'verdana': 'Verdana'
    },
    init: function() {
        this.setDisplay('Font&nbsp;Family...');
        for (var itm in this.sel) {
            this.add(itm, '<font face="' + itm + '">' + this.sel[itm] +
'</font>');
        }
    }
});

var nicEditorFontFormatSelect = nicEditorSelect.extend({
    sel: {
        'p': 'Paragraph',
        'pre': 'Pre',
        'h6': 'Heading&nbsp;6',
        'h5': 'Heading&nbsp;5',
```

```
'h4': 'Heading&nbsp;4',
'h3': 'Heading&nbsp;3',
'h2': 'Heading&nbsp;2',
'h1': 'Heading&nbsp;1'
},

init: function() {
    this.setDisplay('Font&nbsp;Format...');
    for (var itm in this.sel) {
        var tag = itm.toUpperCase();
        this.add('<' + tag + '>', '<' + itm + ' style="padding: 0px; margin: 0px;">' + this.sel[itm] + '</' + tag + '>');
    }
}
});

nicEditors.registerPlugin(nicPlugin, nicSelectOptions);

/* START CONFIG */
var nicLinkOptions = {
    buttons: {
        'link': {
            name: 'Add Link',
            type: 'nicLinkButton',
            tags: ['A']
        },
        'unlink': {
            name: 'Remove Link',
            command: 'unlink',
            noActive: true
        }
    }
};
/* END CONFIG */

var nicLinkButton = nicEditorAdvancedButton.extend({
    addPane: function() {
        this.ln = this.ne.selectedInstance.selElm().parentTag('A');
        this.addForm({
            '': {
                type: 'title',
                txt: 'Add/Edit Link'
            },
            'href': {
                type: 'text',
                txt: 'URL',
                value: 'http://',
                style: {
```

```
        width: '150px'
    },
    'title': {
        type: 'text',
        txt: 'Title'
    },
    'target': {
        type: 'select',
        txt: 'Open In',
        options: {
            '': 'Current Window',
            '_blank': 'New Window'
        },
        style: {
            width: '100px'
        }
    }
}, this.ln);
},

submit: function(e) {
    var url = this.inputs['href'].value;
    if (url === "http://" || url === "") {
        alert("You must enter a URL to Create a Link");
        return false;
    }
    this.removePane();

    if (!this.ln) {
        var tmp = 'javascript:nicTemp();';
        this.ne.nicCommand("createlink", tmp);
        this.ln = this.findElm('A', 'href', tmp);
        // set the link text to the title or the url if there is no text
        selected
        if (this.ln.innerHTML == tmp) {
            this.ln.innerHTML = this.inputs['title'].value || url;
        }
    }
    if (this.ln) {
        var oldTitle = this.ln.title;
        this.ln.setAttributes({
            href: this.inputs['href'].value,
            title: this.inputs['title'].value,
            target:
this.inputs['target'].options[this.inputs['target'].selectedIndex].valu
e
        });
        // set the link text to the title or the url if the old text was
        the old title
        if (this.ln.innerHTML == oldTitle) {
```

```
        this.ln.innerHTML = this.inputs['title'].value ||  
this.inputs['href'].value;  
    }  
}  
});  
  
nicEditors.registerPlugin(nicPlugin, nicLinkOptions);  
  
/* START CONFIG */  
var nicColorOptions = {  
    buttons: {  
        'forecolor': {  
            name: __('Change Text Color'),  
            type: 'nicEditorColorButton',  
            noClose: true  
        },  
        'bgcolor': {  
            name: __('Change Background Color'),  
            type: 'nicEditorBgColorButton',  
            noClose: true  
        }  
    }  
};  
/* END CONFIG */  
  
var nicEditorColorButton = nicEditorAdvancedButton.extend({  
    addPane: function() {  
        var colorList = {  
            0: '00',  
            1: '33',  
            2: '66',  
            3: '99',  
            4: 'CC',  
            5: 'FF'  
        };  
        var colorItems = new bkElement('DIV').setStyle({  
            width: '270px'  
        });  
  
        for (var r in colorList) {  
            for (var b in colorList) {  
                for (var g in colorList) {  
                    var colorCode = '#' + colorList[r] + colorList[g] +  
colorList[b];  
  
                    var colorSquare = new bkElement('DIV').setStyle({  
                        'cursor': 'pointer',  

```

```
        'height': '15px',
        'float': 'left'
    }).appendTo(colorItems);
    var colorBorder = new bkElement('DIV').setStyle({
        border: '2px solid ' + colorCode
    }).appendTo(colorSquare);
    var colorInner = new bkElement('DIV').setStyle({
        backgroundColor: colorCode,
        overflow: 'hidden',
        width: '11px',
        height: '11px'
    }).addEvent('click', this.colorSelect.closure(this,
colorCode)).addEvent('mouseover', this.on.closure(this,
colorBorder)).addEvent('mouseout', this.off.closure(this, colorBorder,
colorCode)).appendTo(colorBorder);

        if (!window.opera) {
            colorSquare.onmousedown = colorInner.onmousedown =
bkLib.cancelEvent;
        }

    }
}
}
this.pane.append(colorItems.noSelect());
},

colorSelect: function(c) {
    this.ne.nicCommand('foreColor', c);
    this.removePane();
},

on: function(colorBorder) {
    colorBorder.setStyle({
        border: '2px solid #000'
    });
},

off: function(colorBorder, colorCode) {
    colorBorder.setStyle({
        border: '2px solid ' + colorCode
    });
}
});

var nicEditorBgColorButton = nicEditorColorButton.extend({
    colorSelect: function(c) {
        this.ne.nicCommand('hiliteColor', c);
        this.removePane();
    }
});
```

```
nicEditors.registerPlugin(nicPlugin, nicColorOptions);

/* START CONFIG */
var nicImageOptions = {
  buttons: {
    'image': {
      name: 'Add Image',
      type: 'nicImageButton',
      tags: ['IMG']
    }
  }
};
/* END CONFIG */

var nicImageButton = nicEditorAdvancedButton.extend({
  addPane: function() {
    this.im = this.ne.selectedInstance.selElm().parentTag('IMG');
    this.addForm({
      '': {
        type: 'title',
        txt: 'Add/Edit Image'
      },
      'src': {
        type: 'text',
        txt: 'URL',
        'value': 'http://',
        style: {
          width: '150px'
        }
      },
      'alt': {
        type: 'text',
        txt: 'Alt Text',
        style: {
          width: '100px'
        }
      },
      'align': {
        type: 'select',
        txt: 'Align',
        options: {
          none: 'Default',
          'left': 'Left',
          'right': 'Right'
        }
      }
    })
  }
});
```

```
    }, this.im);
  },

  submit: function(e) {
    var src = this.inputs['src'].value;
    if (src === "" || src === "http://") {
      alert("You must enter a Image URL to insert");
      return false;
    }
    this.removePane();

    if (!this.im) {
      var tmp = 'javascript:nicImTemp();';
      this.ne.nicCommand("insertImage", tmp);
      this.im = this.findElm('IMG', 'src', tmp);
    }
    if (this.im) {
      this.im.setAttributes({
        src: this.inputs['src'].value,
        alt: this.inputs['alt'].value,
        align: this.inputs['align'].value
      });
    }
  }
});

nicEditors.registerPlugin(nicPlugin, nicImageOptions);

/* START CONFIG */
var nicSaveOptions = {
  buttons: {
    'save': {
      name: __('Save this content'),
      type: 'nicEditorSaveButton'
    }
  }
};
/* END CONFIG */

var nicEditorSaveButton = nicEditorButton.extend({
  init: function() {
    if (!this.ne.options.onSave) {
      this.margin.setStyle({
        'display': 'none'
      });
    }
  },
  mouseClick: function() {
```

```
        var onSave = this.ne.options.onSave;
        var selectedInstance = this.ne.selectedInstance;
        onSave(selectedInstance.getContent(), selectedInstance.elm.id,
selectedInstance);
    }
});

nicEditors.registerPlugin(nicPlugin, nicSaveOptions);

/* START CONFIG */
var nicUploadOptions = {
    buttons: {
        'upload': {
            name: 'Upload Image',
            type: 'nicUploadButton'
        }
    }
};

/* END CONFIG */

var nicUploadButton = nicEditorAdvancedButton.extend({
    nicURI: 'http://files.nicedit.com/',

    addPane: function() {
        this.im = this.ne.selectedInstance.selElm().parentTag('IMG');
        this.myID = Math.round(Math.random() * Math.pow(10, 15));
        this.requestInterval = 1000;
        this.uri = this.ne.options.uploadURI || this.nicURI;
        nicUploadButton.lastPlugin = this;

        this.myFrame = new bkElement('iframe').setAttributes({
            width: '100%',
            height: '100px',
            frameBorder: 0,
            scrolling: 'no'
        }).setStyle({
            border: 0
        }).appendTo(this.pane.pane);
        this.progressWrapper = new bkElement('div').setStyle({
            display: 'none',
            width: '100%',
            height: '20px',
            border: '1px solid #ccc'
        }).appendTo(this.pane.pane);
        this.progress = new bkElement('div').setStyle({
            width: '0%',
            height: '20px',
```

```
        backgroundColor: '#ccc'
    }).setContent('&nbsp;').appendTo(this.progressWrapper);

    setTimeout(this.addForm.closure(this), 50);
},

addForm: function() {
    var myDoc = this.myDoc = this.myFrame.contentWindow.document;
    myDoc.open();
    myDoc.write("<html><body>");
    myDoc.write('<form method="post" action="' + this.uri + '?id=' +
this.myID + '" enctype="multipart/form-data">');
    myDoc.write('<input type="hidden" name="APC_UPLOAD_PROGRESS"
value="' + this.myID + '" />');
    if (this.uri == this.nicURI) {
        myDoc.write('<div style="position: absolute; margin-left:
160px;"><div style="float: left; margin-left: 5px; font-
size: 10px;">Hosted by<br /><a href="http://www.imageshack.us/"
target="_blank">ImageShack</a></div></div>');
    }
    myDoc.write('<div style="font-size: 14px; font-weight: bold;
padding-top: 5px;">Insert an Image</div>');
    myDoc.write('<input name="nicImage" type="file" style="margin-top:
10px;" />');
    myDoc.write('</form>');
    myDoc.write("</body></html>");
    myDoc.close();

    this.myBody = myDoc.body;

    this.myForm = $BK(this.myBody.getElementsByTagName('form')[0]);
    this.myInput =
$BK(this.myBody.getElementsByTagName('input')[1]).addEvent('change',
this.startUpload.closure(this));
    this.myStatus = new bkElement('div', this.myDoc).setStyle({
        textAlign: 'center',
        fontSize: '14px'
    }).appendTo(this.myBody);
},

startUpload: function() {
    this.myForm.setStyle({
        display: 'none'
    });
    this.myStatus.setContent('<strong>Uploading...</strong><br />Please
wait');
    this.myForm.submit();
    setTimeout(this.makeRequest.closure(this), this.requestInterval);
```

```
    },

    makeRequest: function() {
        if (this.pane && this.pane.pane) {
            nicUploadButton.lastPlugin = this;
            var s = new bkElement('script').setAttributes({
                type: 'text/javascript',
                src: this.uri + '?check=' + this.myID + '&rand=' +
Math.round(Math.random() * Math.pow(10, 15))
            }).addEvent('load', function() {
                s.parentNode.removeChild(s);
            }).appendTo(document.getElementsByTagName('head')[0]);
            if (this.requestInterval) {
                setTimeout(this.makeRequest.closure(this),
this.requestInterval);
            }
        }
    },

    setProgress: function(percent) {
        this.progressWrapper.setStyle({
            display: 'block'
        });
        this.progress.setStyle({
            width: percent + '%'
        });
    },

    update: function(o) {
        if (o === false) {
            this.progressWrapper.setStyle({
                display: 'none'
            });
        } else if (o.url) {
            this.setProgress(100);
            this.requestInterval = false;

            if (!this.im) {
                this.ne.selectedInstance.restoreRng();
                var tmp = 'javascript:nicImTemp();';
                this.ne.nicCommand("insertImage", tmp);
                this.im = this.findElm('IMG', 'src', tmp);
            }
            var w = parseInt(this.ne.selectedInstance.elm.getStyle('width'),
10);
            if (this.im) {
                this.im.setAttributes({
                    src: o.url,
                    width: (w && o.width) ? Math.min(w, o.width) : ''
                });
            }
        }
    }
}
```

```
    }

    this.removePane();
  } else if (o.error) {
    this.requestInterval = false;
    this.setProgress(100);
    alert("There was an error uploading your image (" + o.error +
    ").");
    this.removePane();
  } else if (o.noprogress) {
    this.progressWrapper.setStyle({
      display: 'none'
    });
    if (this.uri.indexOf('http:') == -1 ||
    this.uri.indexOf(window.location.host) != -1) {
      this.requestInterval = false;
    }
  } else {
    this.setProgress(Math.round((o.current / o.total) * 75));
    if (o.interval) {
      this.requestInterval = o.interval;
    }
  }
}

});

nicUploadButton.statusCb = function(o) {
  nicUploadButton.lastPlugin.update(o);
};

nicEditors.registerPlugin(nicPlugin, nicUploadOptions);

var nicXHTML = bkClass.extend({
  stripAttributes: ['_moz_dirty', '_moz_resizing', '_extended'],
  noShort: ['style', 'title', 'script', 'textarea', 'a'],
  cssReplace: {
    'font-weight:bold;': 'strong',
    'font-style:italic;': 'em'
  },
  sizes: {
    1: 'xx-small',
    2: 'x-small',
    3: 'small',
    4: 'medium',
    5: 'large',
    6: 'x-large'
  },
},
```

```
construct: function(nicEditor) {
    this.ne = nicEditor;
    if (this.ne.options.xhtml) {
        nicEditor.addEvent('get', this.cleanup.closure(this));
    }
},

cleanup: function(ni) {
    var node = ni.getElm();
    var xhtml = this.toXHTML(node);
    ni.content = xhtml;
},

toXHTML: function(n, r, d) {
    var txt = '';
    var attrTxt = '';
    var cssTxt = '';
    var nType = n.nodeType;
    var nName = n.nodeName.toLowerCase();
    var nChild = n.hasChildNodes && n.hasChildNodes();
    var extraNodes = [];

    switch (nType) {
    case 1:
        var nAttributes = n.attributes;

        switch (nName) {
        case 'b':
            nName = 'strong';
            break;
        case 'i':
            nName = 'em';
            break;
        case 'font':
            nName = 'span';
            break;
        }

        if (r) {
            for (var i = 0; i < nAttributes.length; i++) {
                var attr = nAttributes[i];

                var attributeName = attr.nodeName.toLowerCase();
                var attributeValue = attr.nodeValue;

                if (!attr.specified || !attributeValue ||
                    bkLib.inArray(this.stripAttributes, attributeName) ||
                    typeof(attributeValue) == "function") {
                    continue;
                }
            }
        }
    }
}
```

```

        switch (attributeName) {
        case 'style':
            var css = attributeValue.replace(/ /g, "");
            for (var itm in this.cssReplace) {
                if (css.indexOf(itm) != -1) {
                    extraNodes.push(this.cssReplace[itm]);
                    css = css.replace(itm, '');
                }
            }
            cssTxt += css;
            attributeValue = "";
            break;
        case 'class':
            attributeValue = attributeValue.replace("Apple-style-span",
""");
            break;
        case 'size':
            cssTxt += "font-size:" + this.sizes[attributeValue] + ';;';
            attributeValue = "";
            break;
        }

        if (attributeValue) {
            attrTxt += ' ' + attributeName + '=' + attributeValue +
""';
        }
    }

    if (cssTxt) {
        attrTxt += ' style=' + cssTxt + '""';
    }

    for (var j = 0; j < extraNodes.length; j++) {
        txt += '<' + extraNodes[j] + '>';
    }

    if (attrTxt === "" && nName == "span") {
        r = false;
    }
    if (r) {
        txt += '<' + nName;
        if (nName != 'br') {
            txt += attrTxt;
        }
    }
}

if (!nChild && !bkLib.inArray(this.noShort, attributeName)) {
    if (r) {
        txt += ' />';
    }
}

```

```
    }
  } else {
    if (r) {
      txt += '>';
    }

    for (var k = 0; k < n.childNodes.length; k++) {
      var results = this.toXHTML(n.childNodes[k], true, true);
      if (results) {
        txt += results;
      }
    }
  }

  if (r && nChild) {
    txt += '</' + nName + '>';
  }

  for (var l = 0; l < extraNodes.length; l++) {
    txt += '</' + extraNodes[l] + '>';
  }

  break;
case 3:
  //if(n.nodeValue != '\n') {
  txt += n.nodeValue;
  //}
  break;
}

return txt;
}
});
nicEditors.registerPlugin(nicXHTML);

var nicBBCode = bkClass.extend({
  construct: function(nicEditor) {
    this.ne = nicEditor;
    if (this.ne.options.bbCode) {
      nicEditor.addEvent('get', this.bbGet.closure(this));
      nicEditor.addEvent('set', this.bbSet.closure(this));

      var loadedPlugins = this.ne.loadedPlugins;
      for (var itm in loadedPlugins) {
        if (loadedPlugins[itm].toXHTML) {
          this.xhtml = loadedPlugins[itm];
        }
      }
    }
  }
});
```

```

    }
  },

  bbGet: function(ni) {
    var xhtml = this.xhtml.toXHTML(ni.getElm());
    ni.content = this.toBBCode(xhtml);
  },

  bbSet: function(ni) {
    ni.content = this.fromBBCode(ni.content);
  },

  toBBCode: function(xhtml) {
    function rp(r, m) {
      xhtml = xhtml.replace(r, m);
    }

    rp(/\n/gi, "");
    rp(/<strong>(.*?)</strong>/gi, "[b]$1[/b]");
    rp(/<em>(.*?)</em>/gi, "[i]$1[/i]");
    rp(/<span.*?style="text-decoration:underline;">(.*?)</span>/gi,
"[u]$1[/u]");
    rp(/<ul>(.*?)</ul>/gi, "[list]$1[/list]");
    rp(/<li>(.*?)</li>/gi, "[*]$1[/*]");
    rp(/<ol>(.*?)</ol>/gi, "[list=1]$1[/list]");
    rp(/<img.*?src="(.*?)".*?>/gi, "[img]$1[/img]");
    rp(/<a.*?href="(.*?)".*?>(.*?)</a>/gi, "[url=$1]$2[/url]");
    rp(/<br.*?>/gi, "\n");
    rp(/<.*?>.*?<\/.*?>/gi, "");

    return xhtml;
  },

  fromBBCode: function(bbCode) {
    function rp(r, m) {
      bbCode = bbCode.replace(r, m);
    }

    rp(/\[b\](.*?)\[\/b\]/gi, "<strong>$1</strong>");
    rp(/\[i\](.*?)\[\/i\]/gi, "<em>$1</em>");
    rp(/\[u\](.*?)\[\/u\]/gi, "<span style=\"text-
decoration:underline;\">$1</span>");
    rp(/\[list\](.*?)\[\/list\]/gi, "<ul>$1</ul>");
    rp(/\[list=1\](.*?)\[\/list\]/gi, "<ol>$1</ol>");
    rp(/\[*\](.*?)\[\/*\]/gi, "<li>$1</li>");
    rp(/\[img\](.*?)\[\/img\]/gi, "<img src=\""$1\" />");
    rp(/\[url=(.*?)\](.*?)\[\/url\]/gi, "<a href=\""$1\">$2</a>");
    rp(/\n/gi, "<br />");
    //rp(/\[.*?\](.*?)\[\/.*?\]/gi, "$1");

    return bbCode;
  }
}

```

```
}

});
nicEditors.registerPlugin(nicBBCode);

nicEditor = nicEditor.extend({
  floatingPanel: function() {
    this.floating = new bkElement('DIV').setStyle({
      position: 'absolute',
      top: '-1000px'
    }).appendTo(document.body);
    this.addEvent('focus',
this.reposition.closure(this)).addEvent('blur',
this.hide.closure(this));
    this.setPanel(this.floating);
  },

  reposition: function() {
    var e = this.selectedInstance.e;
    this.floating.setStyle({
      width: (parseInt(e.getStyle('width'), 10) || e.clientWidth) +
'px'
    });
    var top = e.offsetTop - this.floating.offsetHeight;
    if (top < 0) {
      top = e.offsetTop + e.offsetHeight;
    }

    this.floating.setStyle({
      top: top + 'px',
      left: e.offsetLeft + 'px',
      display: 'block'
    });
  },

  hide: function() {
    this.floating.setStyle({
      top: '-1000px'
    });
  }
});

/* START CONFIG */
var nicCodeOptions = {
  buttons: {
```

```
        'xhtml': {
            name: 'Edit HTML',
            type: 'nicCodeButton'
        }
    }

};
/* END CONFIG */

var nicCodeButton = nicEditorAdvancedButton.extend({
    width: '350px',

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

    submit: function(e) {
        var code = this.inputs['code'].value;
        this.ne.selectedInstance.setContent(code);
        this.removePane();
    }
});

nicEditors.registerPlugin(nicPlugin, nicCodeOptions);
```

From:
<https://wwoss.direct.quickconnect.to/> - worldwide open-source software

Permanent link:
https://wwoss.direct.quickconnect.to/doku.php?id=software:development:web:docs:web:wysiwyg:nicedit_nicedit_js

Last update: 2026/01/07 15:42

