Edittools.js

Aptean Ltd

Contents

1 MediaWiki:Edittools.js

//

```
	/*
		EditTools support: add a selector, change into true buttons, enable for all text input fields
		The special characters to insert are defined at [[MediaWiki:Edittools]].
	*/

	// Globals: getElementsByClassName, hookEvent, addEvent (from wikibits.js)

	if ( load_edittools == true ) { // Legacy. Do we really need this?
		if ( typeof( EditTools_set_focus ) == 'undefined' ) {
			var EditTools_set_focus = true;
		}

		if ( typeof( EditTools_set_focus_initially ) == 'undefined' ) {
			var EditTools_set_focus_initially = EditTools_set_focus;
		}

		if ( typeof( EditTools_initial_subset ) == 'undefined' ) {
			var EditTools_initial_subset = 0;
		}

		var EditTools = {
			createSelector: function() {
				var spec = document.getElementById( 'specialchars' );
				if ( !spec ) {
					return;
				}
				var sb = getElementsByClassName( spec, 'p', 'specialbasic' );
				if ( sb.length <= 1 ) {
					return; // Only care if there is more than one
				}

				var sel = document.createElement( 'select' );
				sel.style.display = 'inline';
				// sel.setAttribute( 'onchange', 'EditTools.chooseCharSubset( selectedIndex, true )
				// Apparently, this doesn't work on IE6. Use an explicit event handling function ir
				sel.onchange = EditTools.handleOnchange;

				var initial = EditTools_initial_subset;
				if ( isNaN( initial ) || initial < 0 || initial >= sb.length ) {
					initial = 0;
				}

				for ( var i = 0; i < sb.length; i++ ) {
					var o = document.createElement( 'option' );
					// Ugh. We have encoded Unicode characters in the names...
					var id = sb[i].id.replace( /.([0-9A-F][0-9A-F])/g, '%$1' ).replace( /_/g, '
					if ( i == initial ) {
						o.selected = 'selected';
					}
					o.appendChild( document.createTextNode( decodeURIComponent( id ) ) );
					sel.appendChild( o );
				}

				spec.insertBefore( sel, spec.firstChild );

				EditTools.chooseCharSubset(
					initial,
					( wgAction != 'submit' ) &&
					EditTools_set_focus_initially &&
					( wgCanonicalNamespace != 'Special' || wgCanonicalSpecialPageName != 'Uploa
				);
			},

			handleOnchange: function( evt ) {
				var e    = evt || window.event;      // W3C, IE
				var node = e.target || e.srcElement; // W3C, IE

				EditTools.chooseCharSubset( node.selectedIndex, true );
				return true;
			},

			chooseCharSubset: function( selected, set_focus ) {
				var sb = getElementsByClassName( document.getElementById( 'specialchars' ), 'p', 's
```


```
		EditTools.makeButtons( sb[selected] );
		for ( var i = 0; i < sb.length; i++ ) {
			sb[i].style.display = i == selected ? 'inline' : 'none';
		}
		if ( set_focus && EditTools_set_focus ) {
			var txtarea = EditTools.getTextArea();
			if ( txtarea ) {
				txtarea.focus();
			}
		}
	},

	fixateWidth: function() {
		var edit_bar = document.getElementById( 'specialchars' );
		if ( !edit_bar ) {
			return;
		}
		// Try to fixate the width of that bar, otherwise IE6 may make it wider, which will
		// a table re-layout on the upload form resulting in the right column extending bey
		// right edge of the window.
		edit_bar.setAttribute( 'width', '' + ( edit_bar.clientWidth || edit_bar.offsetWidth
		edit_bar.style.maxWidth = '' + ( edit_bar.clientWidth || edit_bar.offsetWidth ) + '
		// If we're inside a table, fixate the containing table cell, too.
		var parent = edit_bar.parentNode;
		while ( parent && parent != document.body && parent.nodeName.toLowerCase() != 'td'
			parent = parent.parentNode;
		}
		if ( parent && parent != document.body ) {
			parent.setAttribute( 'width', '' + ( parent.clientWidth || parent.offsetWid
			parent.style.maxWidth = '' + ( parent.clientWidth || parent.offsetWidth ) +
		}
	},

	makeButtons: function( section ) {
		var edit_bar = section || document.getElementById( 'specialchars' );
		if ( !edit_bar ) {
			return;
		}
		var links = edit_bar.getElementsByTagName( 'a' );
		// 'links' is a *live* collection!
		var b = null;
		while ( links.length ) {
			b = document.createElement( 'input' );
			b.type = 'button';
			b.style.fontSize = '0.9em';
			b.style.paddingLeft = '1px';
			b.style.paddingRight = '1px';
			b.style.marginLeft = '1px';
			b.onclick = links[0].onclick;
			b.value = links[0].firstChild.data;
			var parent = links[0].parentNode;
			parent.replaceChild( b, links[0] ); // This removes links[0] from links!
			b.blur(); // IE6 insists on marking some buttons as having the focus...
			var margin_added = false;
			// Remove text nodes (nodeType == Node.TEXT_NODE, but IE6 doesn't know that
			// Insert some spacing where desired.
			while ( b.nextSibling && b.nextSibling.nodeType == 3 ) {
				if ( !margin_added && b.nextSibling.data.search( /\S/ ) >= 0 ) {
					b.style.marginRight = '4px';
					margin_added = true;
				}
				parent.removeChild( b.nextSibling );
			}
		}
	},

	enableForAllFields: function() {
		if ( typeof( insertTags ) != 'function' ) {
			return;
		}
		// insertTags from the site-wide /skins-1.5/common/edit.js just inserts in the firs
		// textarea in the document. Evidently, that's not good if we have multiple textare
		// My first idea was to simply add a hidden textarea as the first one, and redefine
		// insertTags such that it copied first the last active textareas contents over to
		// field, set the cursor or selection there, let the standard insertTags do its thi
		// then copy the hidden field's text, cursor position and selection back to the cur
		// active field. Unfortunately, that is just as complex as simply copying the whole
		// from wikibits to here and let it work on the right text field in the first place
```


```
		var texts = document.getElementsByTagName( 'textarea' );
		for ( var i = 0; i < texts.length; i++ ) {
			$( texts[i] ).focus( EditTools.registerTextField );
		}
		// While we're at it, also enable it for input fields
		texts = document.getElementsByTagName( 'input' );
		for ( var i = 0; i < texts.length; i++ ) {
			if ( texts[i].type == 'text' ) {
				$( texts[i] ).focus( EditTools.registerTextField );
			}
		}
		insertTags = EditTools.insertTags; // Redefine the global insertTags
},

last_active_textfield: null,

registerTextField: function( evt ) {
		var e = evt || window.event;
		var node = e.target || e.srcElement;
		if ( !node ) {
			return;
		}
		EditTools.last_active_textfield = node.id;
		return true;
},

getTextArea: function() {
		var txtarea = null;
		if ( EditTools.last_active_textfield && EditTools.last_active_textfield != '' )
			txtarea = document.getElementById( EditTools.last_active_textfield );
			if ( !txtarea ) {
				// Fallback option: old behaviour
				if ( document.editform ) {
					txtarea = document.editform.wpTextbox1;
				} else {
					// Some alternate form? Take the first one we can find
					txtarea = document.getElementsByTagName( 'textarea' );
					if ( txtarea.length > 0 ) {
						txtarea = txtarea[0];
					} else {
						txtarea = null;
					}
				}
			}
		}
		return txtarea;
},

insertTags: function( tagOpen, tagClose, sampleText ) {
		/* Usability initiative compatibility */
		if ( typeof $j != 'undefined' && typeof $j.fn.textSelection != 'undefined' ) {
			$j( '#wpTextbox1' ).textSelection(
				'encapsulateSelection', { 'pre': tagOpen, 'peri': sampleText, 'post
			);
			return;
		}
		var txtarea = EditTools.getTextArea ();
		if ( !txtarea ) {
			return;
		}
		var selText, isSample = false;

		function checkSelectedText() {
			if ( !selText ) {
				selText = sampleText;
				isSample = true;
			} else if ( selText.charAt( selText.length - 1 ) == ' ' ) { // Exclude endi
				selText = selText.substring( 0, selText.length - 1 );
				tagClose += ' ';
			}
		}

		if ( document.selection && document.selection.createRange ) { // IE/Opera
			// Save window scroll position
			var winScroll = 0;
			if ( document.documentElement && document.documentElement.scrollTop ) {
				winScroll = document.documentElement.scrollTop;
			} else if ( document.body ) {
				winScroll = document.body.scrollTop;
```


```
                                }
                                // Get current selection
                                txtarea.focus();
                                var range = document.selection.createRange();
                                selText = range.text;
                                // Insert tags
                                checkSelectedText ();
                                range.text = tagOpen + selText + tagClose;
                                // Mark sample text as selected
                                if ( isSample && range.moveStart ) {
                                        if ( window.opera ) {
                                                tagClose = tagClose.replace( /\n/g, '' );
                                        }
                                        range.moveStart( 'character', - tagClose.length - selText.length );
                                        range.moveEnd( 'character', - tagClose.length );
                                }
                                range.select();
                                // Restore window scroll position
                                if ( document.documentElement && document.documentElement.scrollTop ) {
                                        document.documentElement.scrollTop = winScroll;
                                } else if ( document.body ) {
                                        document.body.scrollTop = winScroll;
                                }
                        } else if ( txtarea.selectionStart || txtarea.selectionStart == '0' ) { // Mozilla
                                // Save textarea scroll position
                                var textScroll = txtarea.scrollTop;
                                // Get current selection
                                txtarea.focus();
                                var startPos = txtarea.selectionStart;
                                var endPos   = txtarea.selectionEnd;
                                selText = txtarea.value.substring( startPos, endPos );
                                // Insert tags
                                checkSelectedText();
                                txtarea.value = txtarea.value.substring( 0, startPos ) +
                                                tagOpen + selText + tagClose +
                                                txtarea.value.substring( endPos );
                                // Set new selection
                                if ( isSample ) {
                                        txtarea.selectionStart = startPos + tagOpen.length;
                                        txtarea.selectionEnd = startPos + tagOpen.length + selText.length;
                                } else {
                                        txtarea.selectionStart = startPos + tagOpen.length + selText.length
                                        txtarea.selectionEnd = txtarea.selectionStart;
                                }
                                // Restore textarea scroll position
                                txtarea.scrollTop = textScroll;
                        }
                }, // end insertTags

                setup: function() {
                        EditTools.fixateWidth();
                        EditTools.createSelector();
                        EditTools.enableForAllFields();
                }

        } // end EditTools

        // Do not use addOnloadHook; it runs *before* the onload event fires. At that time, onclick or
        // onfocus handlers may not yet be set up properly.
        hookEvent( 'load', EditTools.setup );

} // end if

//
```