MediaWiki:Gadget-Cat-a-lot.js

From Blue Gold Program Wiki
The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.

Note: After saving, you may have to bypass your browser's cache to see the changes.

- **Firefox / Safari:** Hold *Shift* while clicking *Reload*, or press either *Ctrl-F5* or *Ctrl-R* (⌘-*R* on a

Mac)
- **Google Chrome:** Press *Ctrl-Shift-R* (⌘-*Shift-R* on a Mac)
- **Internet Explorer / Edge:** Hold *Ctrl* while clicking *Refresh*, or press *Ctrl-F5*
- **Opera:** Go to *Menu → Settings* (*Opera → Preferences* on a Mac) and then to *Privacy & security → Clear browsing data → Cached images and files*.

```
/**
* Cat-a-lot
* Changes category of multiple files
*
* @rev 00:13, 10 February 2018 (UTC)
* @author Originally by Magnus Manske (2007)
* @author RegExes by Ilmari Karonen (2010)
* @author Completely rewritten by DieBuche (2010-2012)
* @author Rillke (2012-2014)
* @author Perhelion (2017)

* Requires [[MediaWiki:Gadget-SettingsManager.js]] and [[MediaWiki:Gadget-
SettingsUI.js]] (properly registered) for per-user-settings
*
* READ THIS PAGE IF YOU WANT TO TRANSLATE OR USE THIS ON ANOTHER SITE:
* http://commons.wikimedia.org/wiki/MediaWiki:Gadget-Cat-a-lot.js/translating
* <nowiki>
*/

/* global jQuery, mediaWiki */
/* eslint one-var:0, vars-on-top:0, no-underscore-dangle:0, valid-jsdoc:0,
curly:0, camelcase:0, no-useless-escape:0, no-alert:0 */ // extends:
wikimedia
/* jshint unused:true, forin:false, smarttabs:true, loopfunc:true,
browser:true */

( function ( $, mw ) {
'use strict';

var formattedNS = mw.config.get( 'wgFormattedNamespaces' ),
        ns = mw.config.get( 'wgNamespaceNumber' ),
        nsIDs = mw.config.get( 'wgNamespaceIds' ),
        userGrp = mw.config.get( 'wgUserGroups' ),
        project = mw.config.get( 'wgDBname' );

var msgs = {
// Preferences
// new: added 2012-09-19. Please translate.
// Use user language for i18n
        'cat-a-lot-watchlistpref': 'Watchlist preference concerning files
edited with Cat-a-lot',
        'cat-a-lot-watch_pref': 'According to your general preferences',
        'cat-a-lot-watch_nochange': 'Do not change watchstatus',
        'cat-a-lot-watch_watch': 'Watch pages edited with Cat-a-lot',
```

```
        'cat-a-lot-watch_unwatch': 'Remove pages while editing with Cat-a-lot
from your watchlist',
        'cat-a-lot-minorpref': 'Mark edits as minor (if you generally mark
your edits as minor, this won’t change anything)',
        'cat-a-lot-editpagespref': 'Allow categorising pages (including
categories) that are not files',
        'cat-a-lot-docleanuppref': 'Remove {{Check categories}} and other
minor cleanup',
        'cat-a-lot-uncatpref': 'Remove {{Uncategorized}}',
        'cat-a-lot-subcatcountpref': 'Sub-categories to show at most',
        'cat-a-lot-config-settings': 'Preferences',
        'cat-a-lot-buttonpref': 'Use buttons instead of text-links',
        'cat-a-lot-comment-label': 'Custom edit comment',
        'cat-a-lot-edit-question': 'Why is this change necessary?',

        // Progress
        // 'cat-a-lot-loading': 'Loading …',
        'cat-a-lot-editing': 'Editing page',
        'cat-a-lot-of': 'of ',
        'cat-a-lot-skipped-already': 'The following {{PLURAL:$1|1=page was|$1
pages were}} skipped, because the page was already in the category:',
        'cat-a-lot-skipped-not-found': 'The following {{PLURAL:$1|1=page
was|$1 pages were}} skipped, because the old category could not be found:',
        'cat-a-lot-skipped-server': 'The following {{PLURAL:$1|1=page|$1
pages}} couldn’t be changed, since there were problems connecting to the
server:',
        'cat-a-lot-all-done': 'All pages are processed.',
        'cat-a-lot-done': 'Done!', // mw.msg("Feedback-close")
        'cat-a-lot-added-cat': 'Added category $1',
        'cat-a-lot-copied-cat': 'Copied to category $1',
        'cat-a-lot-moved-cat': 'Moved to category $1',
        'cat-a-lot-removed-cat': 'Removed from category $1',
        // 'cat-a-lot-return-to-page': 'Return to page',
        // 'cat-a-lot-cat-not-found': 'Category not found.',

        // as in 17 files selected
        'cat-a-lot-files-selected': '{{PLURAL:$1|1=One file|$1 files}}
selected.',
        'cat-a-lot-pe_file': '$1 {{PLURAL:$1|page|pages}} of $2 affected',
        'cat-a-lot-parent-cat': 'Has parent-category: ',
        'cat-a-lot-sub-cat': 'Has sub-category: ',

        // Actions
        'cat-a-lot-copy': 'Copy',
        'cat-a-lot-move': 'Move',
        'cat-a-lot-add': 'Add',
        // 'cat-a-lot-remove-from-cat': 'Remove from this category',
        'cat-a-lot-overcat': 'Check over-categorization',
        'cat-a-lot-enter-name': 'Enter category name',
        'cat-a-lot-select': 'Select',
        'cat-a-lot-all': 'all',
```

```
        'cat-a-lot-none': 'none',
        // 'cat-a-lot-none-selected': 'No files selected!', 'Ooui-selectfile-
placeholder'

        // Summaries (project language):
        'cat-a-lot-pref-save-summary': 'Updating user preferences',
        'cat-a-lot-summary-add': 'Adding [[Category:$1]]',
        'cat-a-lot-summary-copy': 'Copying from [[Category:$1]] to
[[Category:$2]]',
        'cat-a-lot-summary-move': 'Moving from [[Category:$1]] to
[[Category:$2]]',
        'cat-a-lot-summary-remove': 'Removing from [[Category:$1]]',
        'cat-a-lot-prefix-summary': '',
        'cat-a-lot-using-summary': ' using [[c:Help:Cat-a-lot|Cat-a-lot]]'
};
mw.messages.set( msgs );

function msg( /* params */ ) {
        var args = Array.prototype.slice.call( arguments, 0 );
        args[ 0 ] = 'cat-a-lot-' + args[ 0 ];
        return ( args.length === 1 ) ?
                mw.message( args[ 0 ] ).plain() :
                mw.message.apply( mw.message, args ).parse();
}

// There is only one Cat-a-lot on one page
var $body, $container, $dataContainer, $searchInputContainer, $searchInput,
$resultList, $markCounter, $selections,
        $selectFiles, $selectPages, $selectNone, $selectInvert,
$settingsWrapper, $settingsLink, $head, $link, $overcat,
        commonsURL = 'https://commons.wikimedia.org/w/index.php',
        is_rtl = $( 'body' ).hasClass( 'rtl' ),
        reCat, // localized category search regexp
        non,
        r; // result file count for overcat

var CAL = mw.libs.catALot = {
        apiUrl: mw.util.wikiScript( 'api' ),
        origin: '',
        searchmode: false,
        version: '4.77',
        setHeight: 450,
        changeTag: 'Cat-a-lot',

        settings: {
        /* Any category in this category is deemed a disambiguation category;
i.e., a category that should not contain
any items, but that contains links to other categories where stuff should be
categorized. If you don't have
that concept on your wiki, set it to null. Use blanks, not underscores. */
                disambig_category: 'Disambiguation', // Commons and EnWP
```

```
                /* Any category in this category is deemed a (soft) redirect
to some other category defined by a link
* to another non-blacklisted category. If your wiki doesn't have soft
category redirects, set this to null.
* If a soft-redirected category contains more than one link to another non-
blacklisted category, it's considered
* a disambiguation category instead. */
                redir_category: 'Category redirects'

        },

        init: function () {
                // Prevent historical double marker (maybe remove in future)
                if ( /Cat-?a-?lot/i.test( msgs[ 'cat-a-lot-pref-save-summary'
] ) ) { mw.messages.set( { 'cat-a-lot-prefix-summary': '', 'cat-a-lot-using-
summary': '' } ); } else {
                        mw.messages.set( {
                                'cat-a-lot-pref-save-summary': msgs[ 'cat-a-
lot-prefix-summary' ] + msgs[ 'cat-a-lot-pref-save-summary' ] + msgs[ 'cat-a-
lot-using-summary' ]
                        } );
                }

                // TODO: better extern project support for possible change-
tag? (needs currently change after init)
                if ( project === 'commonswiki' ) { mw.messages.set( { 'cat-a-
lot-using-summary': '' } ); } else { // Reset
                        this.changeTag = '';
                        this.settings.redir_category = '';
                }

                this._initSettings();
                $body = $( document.body );
                $container = $( '<div>' )
                        .attr( 'id', 'cat_a_lot' )
                        .appendTo( $body );
                $dataContainer = $( '<div>' )
                        .attr( 'id', 'cat_a_lot_data' )
                        .appendTo( $container );
                $searchInputContainer = $( '<div>' )
                        .appendTo( $dataContainer );
                $searchInput = $( '<input>', {
                        id: 'cat_a_lot_searchcatname',
                        placeholder: msg( 'enter-name' ),
                        type: 'text'
                } )
                        .appendTo( $searchInputContainer );
                $resultList = $( '<div>' )
                        .attr( 'id', 'cat_a_lot_category_list' )
                        .appendTo( $dataContainer );
                $markCounter = $( '<div>' )
```

```
                        .attr( 'id', 'cat_a_lot_mark_counter' )
                        .appendTo( $dataContainer );
                $selections = $( '<div>' )
                        .attr( 'id', 'cat_a_lot_selections' )
                        .text( msg( 'select' ) + ':' )
                        .appendTo( $dataContainer );
                $settingsWrapper = $( '<div>' )
                        .attr( 'id', 'cat_a_lot_settings' )
                        .appendTo( $dataContainer );
                $settingsLink = $( '<a>', {
                        id: 'cat_a_lot_config_settings',
                        title: 'Version ' + this.version,
                        text: msg( 'config-settings' )
                } )
                        .appendTo( $settingsWrapper );
                $head = $( '<div>' )
                        .attr( 'id', 'cat_a_lot_head' )
                        .appendTo( $container );
                $link = $( '<a>' )
                        .attr( 'id', 'cat_a_lot_toggle' )
                        .text( 'Cat-a-lot' )
                        .appendTo( $head );
                $settingsWrapper.append( $( '<a>', {
                        href: commonsURL +
'?title=Special:MyLanguage/Help:Gadget-Cat-a-lot',
                        target: '_blank',
                        style: 'float:right',
                        title: ( $( '#n-help a' ).attr( 'title' ) || '' ) + '
(v. ' + this.version + ')'
                } ).text( '?' ) );
                $container.one( 'mouseover', function () { // Try load on
demand earliest as possible
                        mw.loader.load( [ 'jquery.ui'] );
                } );

                if ( this.origin && !non ) {
                        $overcat = $( '<a>' )
                                .attr( 'id', 'cat_a_lot_overcat' )
                                .html( msg( 'overcat' ) )
                                .on( 'click', function ( e ) {
                                        CAL.getOverCat( e );
                                } )
                                .insertBefore( $selections );
                }

                if ( ( mw.util.getParamValue( 'withJS' ) ===
'MediaWiki:Gadget-Cat-a-lot.js' &&
                        !mw.util.getParamValue( 'withCSS' ) ) ||
                        mw.loader.getState( 'ext.gadget.Cat-a-lot' ) ===
'registered' ) {
                        mw.loader.load( mw.config.get( 'wgServer' ) +
```

```
'/w/index.php?title=MediaWiki:Gadget-Cat-a-
lot.css&action=raw&ctype=text/css', 'text/css' );
                        // importStylesheet( 'MediaWiki:Gadget-Cat-a-lot.css'
);
                }

                reCat = new RegExp( '^\\s*' + CAL.localizedRegex( 14,
'Category' ) + ':', '' );

                $searchInput.on( 'keypress', function ( e ) {
                        if ( e.which === 13 ) {
                                CAL.updateCats( $.trim( $( this
).val().replace( /[\u200E\u200F\u202A-\u202E]/g, '' ) ) );
                                mw.cookie.set( 'catAlot', CAL.currentCategory
);
                        }
                } )
                        .on( 'input keyup', function () {
                                var oldVal = this.value,
                                        newVal = oldVal.replace( reCat, '' );
                                if ( newVal !== oldVal ) { this.value =
newVal; }

                                if ( !newVal ) { mw.cookie.set( 'catAlot',
null ); }
                        } );

                function initAutocomplete() {
                        if ( CAL.autoCompleteIsEnabled ) { return; }

                        CAL.autoCompleteIsEnabled = true;

                        if ( !$searchInput.val() && mw.cookie &&
mw.cookie.get( 'catAlot' ) ) { $searchInput.val( mw.cookie.get( 'catAlot' )
); }

                        $searchInput.autocomplete( {
                                source: function ( request, response ) {
                                        CAL.doAPICall( {
                                                action: 'opensearch',
                                                search: request.term,
                                                redirects: 'resolve',
                                                namespace: 14
                                        }, function ( data ) {
                                                if ( data[ 1 ] ) {
                                                        response( $( data[ 1
] )
                                                                .map(
function ( index, item ) {
return item.replace( reCat, '' );
                                                                } ) );
```

```
                                        }

                                } );
                        },
                        open: function () {
                                $( '.ui-autocomplete' )
                                        .position( {
                                                my: is_rtl ? 'left
bottom' : 'right bottom',
                                                at: is_rtl ? 'left
top' : 'right top',
                                                of: $searchInput
                                        } );
                        },
                        appendTo: '#cat_a_lot'
                } );
        }
        $( '<a>' )
        // .attr( 'id', 'cat_a_lot_select_all' )
                .text( msg( 'all' ) )
                .on( 'click', function () {
                        CAL.toggleAll( true );
                } )
                .appendTo( $selections.append( ' ' ) );
        if ( this.settings.editpages ) {
                $selectFiles = $( '<a>' )
                        .on( 'click', function () {
                                CAL.toggleAll( 'files' );
                        } );
                $selectPages = $( '<a>' )
                        .on( 'click', function () {
                                CAL.toggleAll( 'pages' );
                        } );
                $selections.append( $( '<span>' ).hide().append( [ '
/ ', $selectFiles, ' / ', $selectPages ] ) );
        }
        $selectNone = $( '<a>' )
        // .attr( 'id', 'cat_a_lot_select_none' )
                .text( msg( 'none' ) )
                .on( 'click', function () {
                        CAL.toggleAll( false );
                } );
        $selectInvert = $( '<a>' )
                .on( 'click', function () {
                        CAL.toggleAll( null );
                } );
        $selections.append( [ ' • ', $selectNone, ' • ',
$selectInvert,
                $( '<div>' ).append( [
                        $( '<label>' )
                                .attr( {
```

```
                                                'for': 'cat_a_lot_comment',
                                                style: 'line-
height:1.5em;vertical-align:bottom'
                                        } )
                                        .text( msg( 'comment-label' ) ),
                                $( '<input>' )
                                        .attr( {
                                                id: 'cat_a_lot_comment',
                                                type: 'checkbox'
                                        } )
                        ] )
                ] );

                $link
                        .on( 'click', function () {
                                $( this ).toggleClass( 'cat_a_lot_enabled' );
                                // Load autocomplete on demand
                                mw.loader.using( 'jquery.ui',
initAutocomplete );

                                if ( !CAL.executed ) {
                                        $.when( mw.loader.using( [
                                                'jquery.ui',
                                                'jquery.ui',
                                                'jquery.ui',
                                                'mediawiki.api',
                                                'mediawiki.jqueryMsg'
                                        ] ), $.ready )
                                                .then( function () {
                                                        return new
mw.Api().loadMessagesIfMissing( [
                                                                'Cancel',
'Categorytree-not-found',
                                                                //
'Checkuser-all',
                                                                // 'Code-
field-select',
                                                                // 'Export-
addcat',
                                                                'Filerevert-
submit',
                                                                'Mobile-
frontend-return-to-page',
                                                                'Ooui-
selectfile-placeholder',
                                                                //
'Visualeditor-clipboard-copy',
                                                                'Wikieditor-
loading',
                                                                'Prefs-
files',
```

```
                                                                'Categories',
                                                                'Checkbox-
invert',
'Centralnotice-remove', // 'Ooui-item-remove'
'Apifeatureusage-warnings'
                                                                ] );
                                                        } ).then( function () {
                                                                CAL.run();
                                                        } );
                                        } else { CAL.run(); }
                                } );
                        $settingsLink
                                .on( 'click', CAL.manageSettings );
                        this.localCatName = formattedNS[ 14 ] + ':';
                        mw.loader.using( 'mediawiki.cookie', function () { // Let
catAlot stay open
                                var val = mw.cookie.get( 'catAlotO' );
                                if ( val && Number( val ) === ns ) { $link.click(); }
                        }
                        );
                },

                getOverCat: function ( e ) {
                        var files = [];
                        r = 0; // result counter
                        if ( e ) {
                                e.preventDefault();
                                this.files = this.getMarkedLabels(); // .toArray()
not working
                                for ( var f = 0; f < this.files.length; f++ ) {
files.push( this.files[ f ] ); }

                        }
                        if ( !files.length || !( files instanceof Array ) ) { return
alert( mw.msg( 'Ooui-selectfile-placeholder' ) ); }
                        this.files = files;
                        mw.loader.using( [ 'jquery.spinner' ], function () {
                                $markCounter.injectSpinner( 'overcat' );
                                CAL.getFileCats();
                        } );
                },

                getFileCats: function () {
                        var aLen = this.files.length;
                        var bLen = this.selectedLabels.length;
                        var file = this.files[ aLen - 1 ][ 0 ];
                        $overcat.text( '…' + aLen + '\/' + bLen );
                        if ( file ) {
                                this.doAPICall( {
                                        prop: 'categories',
                                        titles: file
```

```
                        }, this.checkFileCats
                        );
                }

        },

        checkFileCats: function ( data ) {
                var cc = 0; // current cat counter;
                var file = CAL.files.pop();
                if ( data.query && data.query.pages ) {
                        $.each( data.query.pages, function ( id, page ) {
                                if ( page.categories ) {
                                        var target = file[ 1 ].removeClass(
'cat_a_lot_selected' );
                                        $.each( page.categories, function (
c, cat ) {
                                                var title =
cat.title.replace( reCat, '' ),
                                                        color = 'orange',
                                                        mark = function (
kind ) { // kind of category
                                                        // TODO: store data
to use this for special remove function
                                                                if ( kind ===
'sub' ) { color = 'green'; }
                                                                var border =
'3px dotted ';
                                                                if (
$.inArray( title, CAL[ kind + 'Cats' ] ) !== -1 ) {
                                                                        cc++;
target = target.parents( '.gallerybox' );
target = target[ 0 ] ? target : file[ 1 ];
target.css( {
border: border + color
                                                                }
).prop( 'title', msg( kind + '-cat' ) + title );
                                                                color
= 'red';
return false;
                                                                }
                                                        };
                                                mark( 'sub' );
                                                return mark( 'parent' );
                                        } );
                                        if ( cc ) { r++; }
                                }
                        } );
                } else { mw.log( 'Api-fail', file, data ); }
                if ( CAL.files[ 0 ] ) { return setTimeout( function () {
CAL.getFileCats(); }, 100 ); } // Api has bad performance here, so we can get
only each file separately
```

```
                $overcat.text( msg( 'pe_file', r, CAL.selectedLabels.length )
);
                $.removeSpinner( 'overcat' );
        },

        findAllLabels: function ( searchmode ) {
        // It's possible to allow any kind of pages as well but what happens
if you click on "select all" and don't expect it
                switch ( searchmode ) {
                        case 'search':
                                this.labels = this.labels.add( $(
'table.searchResultImage' ).find( 'tr>td:eq(1)' ) );
                                if ( this.settings.editpages ) { this.labels
= this.labels.add( 'div.mw-search-result-heading' ); }

                                break;
                        case 'category':
                                this.findAllLabels( 'gallery' );
                                this.labels = this.labels.add( $( '#mw-
category-media' ).find( 'li[class!="gallerybox"]' ) );
                                if ( this.settings.editpages ) {
                                        this.pageLabels = $( '#mw-pages, #mw-
subcategories' ).find( 'li' );
                                        // this.files = this.labels;
                                        this.labels = this.labels.add(
this.pageLabels );
                                }
                                break;
                        case 'contribs':
                                this.labels = this.labels.add( $( 'ul.mw-
contributions-list li' ) );
                                // FIXME: Filter if !this.settings.editpages
                                break;
                        case 'prefix':
                                this.labels = this.labels.add( $( 'ul.mw-
prefixindex-list li' ) );
                                break;
                        case 'listfiles':
                        // this.labels = this.labels.add( $(
'table.listfiles>tbody>tr' ).find( 'td:eq(1)' ) );
                                this.labels = this.labels.add( $(
'.TablePager_col_img_name' ) );
                                break;
                        case 'gallery':
                        // this.labels = this.labels.add( '.gallerybox' ); //
TODO incombatible with GalleryDetails
                                this.labels = this.labels.add( '.gallerytext'
);
                                break;
                }
        },
```

```
        getTitleFromLink: function ( $a ) {
                try {
                        return decodeURIComponent( $a.attr( 'href' ) )
                                .match( /wiki\/(.+?)(?:#.+)?$/ )[ 1
].replace( /_/g, ' ' );
                } catch ( ex ) {
                        return '';
                }
        },

        /**
*  @brief Get title from selected pages
*  @return [array] touple of page title and $object
*/
        getMarkedLabels: function () {
                this.selectedLabels = this.labels.filter(
'.cat_a_lot_selected:visible' );
                return this.selectedLabels.map( function () {
                        var label = $( this ), file = label.find(
'a[title][class$="title"]' );
                        file = file.length ? file : label.find( 'a[title]' );
                        var title = file.attr( 'title' ) ||
CAL.getTitleFromLink( file ) ||
CAL.getTitleFromLink( label.find( 'a' ) ) ||
CAL.getTitleFromLink( label.parent().find( 'a' ) ); // TODO needs
optimization
                        if ( title.indexOf( formattedNS[ 2 ] + ':' ) ) {
return [ [ title, label ] ]; }
                } );
        },

        updateSelectionCounter: function () {
                this.selectedLabels = this.labels.filter(
'.cat_a_lot_selected:visible' );
                var first = $markCounter.is( ':hidden' );
                $markCounter
                        .html( msg( 'files-selected',
this.selectedLabels.length ) )
                        .show();
                if ( first && !$dataContainer.is( ':hidden' ) ) { //
Workaround to fix position glitch
                        first = $markCounter.innerHeight();
                        $container
                                .offset( { top: $container.offset().top -
first } )
                                .height( $container.height() + first );
                        $( window ).on( 'beforeunload', function () {
                                if ( CAL.labels.filter(
'.cat_a_lot_selected:visible' )[ 0 ] ) { return 'You have pages selected!'; }
// There is a default message in the browser
                        } );
```

```
                }
        },

        makeClickable: function () {
                this.labels = $();
                this.pageLabels = $(); // only for distinct all selections
                this.findAllLabels( this.searchmode );
                this.labels.catALotShiftClick( function () {
                        CAL.updateSelectionCounter();
                } )
                        .addClass( 'cat_a_lot_label' );
        },

        toggleAll: function ( select ) {
                if ( typeof select === 'string' && this.pageLabels[ 0 ] ) {
                        this.pageLabels.toggleClass( 'cat_a_lot_selected',
true );
                        if ( select === 'files' ) // pages get deselected
                        { this.labels.toggleClass( 'cat_a_lot_selected' ); }
                } else {
                // invert / none / all
                        this.labels.toggleClass( 'cat_a_lot_selected', select
);
                }
                this.updateSelectionCounter();
        },

        getSubCats: function () {
                var data = {
                        list: 'categorymembers',
                        cmtype: 'subcat',
                        cmlimit: this.settings.subcatcount,
                        cmtitle: 'Category:' + this.currentCategory
                };

                this.doAPICall( data, function ( result ) {
                        var cats = result.query.categorymembers;
                        CAL.subCats = [];
                        for ( var i = 0; i < cats.length; i++ ) {
CAL.subCats.push( cats[ i ].title.replace( /^[^:]+:/, '' ) ); }

                        CAL.catCounter++;
                        if ( CAL.catCounter === 2 ) { CAL.showCategoryList();
}

                } );
        },

        getParentCats: function () {
                var data = {
                        prop: 'categories',
```

```
                        titles: 'Category:' + this.currentCategory
                };
                this.doAPICall( data, function ( result ) {
                        CAL.parentCats = [];
                        var cats,
                                pages = result.query.pages,
                                table = $( '<table>' );

                        if ( pages[ -1 ] && pages[ -1 ].missing === '' ) {
                                $resultList.html( '<span
id="cat_a_lot_no_found">' + mw.msg( 'Categorytree-not-found',
this.currentCategory ) + '</span>' );
                                document.body.style.cursor = 'auto';
                                CAL.createCatLinks( '→', [
CAL.currentCategory ], table );
                                $resultList.append( table );
                                return;
                        }
                        // there should be only one, but we don't know its ID
                        for ( var id in pages ) { cats = pages[ id
].categories || []; }

                        for ( var i = 0; i < cats.length; i++ ) {
CAL.parentCats.push( cats[ i ].title.replace( /^[^:]+:/, '' ) ); }

                        CAL.catCounter++;
                        if ( CAL.catCounter === 2 ) { CAL.showCategoryList();
}

                } );
        },

        localizedRegex: function ( namespaceNumber, fallback ) {
        // Copied from HotCat, thanks Lupo.
                var wikiTextBlank = '[\\t _\\xA0\\u1680\\u180E\\u2000-
\\u200A\\u2028\\u2029\\u202F\\u205F\\u3000]+';
                var wikiTextBlankRE = new RegExp( wikiTextBlank, 'g' );
                var createRegexStr = function ( name ) {
                        if ( !name || !name.length ) { return ''; }

                        var regexName = '';
                        for ( var i = 0; i < name.length; i++ ) {
                                var ii = name[ i ];
                                var ll = ii.toLowerCase();
                                var ul = ii.toUpperCase();
                                regexName += ( ll === ul ) ? ii : '[' + ll +
ul + ']';
                        }
                        return regexName.replace( /([\\\^\$\.\?\*\+\(\)])/g,
'\\$1' )
                                .replace( wikiTextBlankRE, wikiTextBlank );
```

```
                };

                fallback = fallback.toLowerCase();
                var canonical = formattedNS[ namespaceNumber ].toLowerCase();
                var RegexString = createRegexStr( canonical );
                if ( fallback && canonical !== fallback ) { RegexString +=
'|' + createRegexStr( fallback ); }

                for ( var catName in nsIDs ) { if ( typeof catName ===
'string' && catName.toLowerCase() !== canonical && catName.toLowerCase() !==
fallback && nsIDs[ catName ] === namespaceNumber ) { RegexString += '|' +
createRegexStr( catName ); } }

                return ( '(?:' + RegexString + ')' );
        },

        regexCatBuilder: function ( category ) {
                var catname = this.localizedRegex( 14, 'Category' );

                // Build a regexp string for matching the given category:
                // trim leading/trailing whitespace and underscores
                category = category.replace( /^[\s_]+|[\s_]+$/g, '' );

                // escape regexp metacharacters (= any ASCII punctuation
except _)
                category = mw.util.escapeRegExp( category );

                // any sequence of spaces and underscores should match any
other
                category = category.replace( /[\s_]+/g, '[\\s_]+' );

                // Make the first character case-insensitive:
                var first = category.substr( 0, 1 );
                if ( first.toUpperCase() !== first.toLowerCase() ) { category
= '[' + first.toUpperCase() + first.toLowerCase() + ']' + category.substr( 1
); }

                // Compile it into a RegExp that matches MediaWiki category
syntax (yeah, it looks ugly):
                // XXX: the first capturing parens are assumed to match the
sortkey, if present, including the | but excluding the ]]
                return new RegExp( '\\[\\[[\\s_]*' + catname +
'[\\s_]*:[\\s_]*' + category +
'[\\s_]*(\\|[^\\]]*(?:\\][^\\]]+)*)?\\]\\]\\s*', 'g' );
        },

        getContent: function ( page, targetcat, mode ) {
                if ( !this.cancelled ) {
                        this.doAPICall( {
                                curtimestamp: 1,
                                // meta: 'tokens',
```

```
                                prop: 'revisions',
                                rvprop: 'content|timestamp',
                                titles: page[ 0 ]
                        }, function ( result ) {
                                CAL.editCategories( result, page, targetcat,
mode );
                        } );
                }

        },

        getTargetCat: function ( pages, targetcat, mode ) {
                if ( !this.cancelled ) {
                        this.doAPICall( {
                                meta: 'tokens',
                                prop: 'categories|categoryinfo',
                                titles: 'Category:' + targetcat
                        }, function ( result ) {
                                if ( !result || !result.query ) { return; }
                                CAL.edittoken =
result.query.tokens.csrftoken;
                                result = CAL._getPageQuery( result );
                                CAL.checkTargetCat( result );
                                for ( var i = 0; i < pages.length; i++ ) {
CAL.getContent( pages[ i ], targetcat, mode ); }

                        } );
                }

        },

        checkTargetCat: function ( page ) {
                var is_dab = false; // disambiguation
                var is_redir = typeof page.redirect === 'string'; // Hard
redirect?
                if ( typeof page.missing === 'string' ) { return alert(
mw.msg( 'Apifeatureusage-warnings', mw.msg( 'Categorytree-not-found',
page.title ) ) ); }
                var cats = page.categories;
                this.is_hidden = page.categoryinfo && typeof
page.categoryinfo.hidden === 'string';

                if ( !is_redir && cats && ( CAL.disambig_category ||
CAL.redir_category ) ) {
                        for ( var c = 0; c < cats.length; c++ ) {
                                var cat = cats[ c ].title;
                                if ( cat ) { // Strip namespace prefix
                                        cat = cat.substring( cat.indexOf( ':'
) + 1 ).replace( /_/g, ' ' );
                                        if ( cat === CAL.disambig_category )
{
```

```
                                                is_dab = true; break;
                                        } else if ( cat ===
CAL.redir_category ) {
                                                is_redir = true; break;
                                        }
                                }
                        }
                }

                if ( !is_redir && !is_dab ) { return; }
                alert( mw.msg( 'Apifeatureusage-warnings', page.title + ' is
a ' + CAL.disambig_category ) );
        },

        // Remove {{Uncategorized}} (also with comment). No need to replace
it with anything.
        removeUncat: function ( text ) {
                return ( this.settings.uncat ? text.replace(
/\{\{\s*[Uu]ncategorized\s*[^}]*\}\}\s*(<!--.*?-->\s*)?/, '' ) : text );
        },

        doCleanup: function ( text ) {
                return ( this.settings.docleanup ? text.replace(
/\{\{\s*[Cc]heck categories\s*(\|?.*?)\}\}/, '' ) : text );
        },

        editCategories: function ( result, file, targetcat, mode ) {
                if ( !result || !result.query ) {
                // Happens on unstable wifi connections..
                        this.connectionError.push( file[ 0 ] );
                        this.updateCounter();
                        return;
                }
                var otext,
                        timestamp,
                        page = CAL._getPageQuery( result );
                if ( page.ns === 2 ) { return; }
                var id = page.revisions[ 0 ],
                        catNS = this.localCatName; // canonical cat-name

                this.starttimestamp = result.curtimestamp;
                otext = id[ '*' ];
                timestamp = id.timestamp;

                var sourcecat = this.origin;
                // Check if that file is already in that category
                if ( mode !== 'remove' && this.regexCatBuilder( targetcat
).test( otext ) ) {
                        // If the new cat is already there, just remove the
old one
                        if ( mode === 'move' ) {
```

```
                                mode = 'remove';
                                targetcat = sourcecat;
                        } else {
                                this.alreadyThere.push( file[ 0 ] );
                                this.updateCounter();
                                return;
                        }
                }

                // Text modification (following 3 functions are partialy
taken from HotCat)
                var wikiTextBlankOrBidi = '[\\t _\\xA0\\u1680\\u180E\\u2000-
\\u200B\\u200E\\u200F\\u2028-\\u202F\\u205F\\u3000]*';
                // Whitespace regexp for handling whitespace between link
components. Including the horizontal tab, but not \n\r\f\v:
                // a link must be on one single line.
                // MediaWiki also removes Unicode bidi override characters in
page titles (and namespace names) completely.
                // This is *not* handled, as it would require us to allow any
of [\u200E\u200F\u202A-\u202E] between any two
                // characters inside a category link. It _could_ be done
though... We _do_ handle strange spaces, including the
                // zero-width space \u200B, and bidi overrides between the
components of a category link (adjacent to the colon,
                // or adjacent to and inside of "[[" and "]]").
                var findCatsRE = new RegExp( '\\[\\[' + wikiTextBlankOrBidi +
this.localizedRegex( 14, 'Category' ) + wikiTextBlankOrBidi +
':[^\\]]+\\]\\]', 'g' );

                function replaceByBlanks( match ) {
                        return match.replace( /(\s|\S)/g, ' ' ); // /./
doesn't match linebreaks. /(\s|\S)/ does.
                }

                function find_insertionpoint( wikitext ) {
                        var copiedtext = wikitext
                                .replace( /<!--(\s|\S)*?-->/g,
replaceByBlanks )
                                .replace( /<nowiki>(\s|\S)*?<\/nowiki>/g,
replaceByBlanks );
                        // Search in copiedtext to avoid that we insert
inside an HTML comment or a nowiki "element".
                        var index = -1;
                        findCatsRE.lastIndex = 0;
                        while ( findCatsRE.exec( copiedtext ) !== null ) {
index = findCatsRE.lastIndex; }

                        return index;
                }

                /**
```

```
*  @brief Adds the new Category by searching the right insert point,
*         if there is text after the category section
*  @param [string] wikitext
*  @param [string] toAdd
*  @return Return wikitext
*/
                function addCategory( wikitext, toAdd ) {
                        if ( toAdd && toAdd[ 0 ] ) {
                        // TODO: support sort key
                                var cat_point = find_insertionpoint( wikitext
); // Position of last category
                                var newcatstring = '[[' + catNS + toAdd +
']]';
                                if ( cat_point > -1 ) {
                                        var suffix = wikitext.substring(
cat_point );
                                        wikitext = wikitext.substring( 0,
cat_point ) + ( cat_point ? '\n' : '' ) + newcatstring;
                                        if ( suffix[ 0 ] && suffix.substr( 0,
1 ) !== '\n' ) { wikitext += '\n'; }
                                        wikitext += suffix;
                                } else {
                                        if ( wikitext[ 0 ] &&
wikitext.substr( wikitext.length - 1, 1 ) !== '\n' ) { wikitext += '\n'; }

                                        wikitext += ( wikitext[ 0 ] ? '\n' :
'' ) + newcatstring;
                                }
                        }
                        return wikitext;
                }
                // End HotCat functions

                var text = otext,
                        arr = is_rtl ? '\u2190' : '\u2192', // left and right
arrows. Don't use ← and → in the code.
                        sumCmt, // summary comment
                        sumCmtShort;
                // Fix text
                switch ( mode ) {
                        case 'add':
                                text = addCategory( text, targetcat );
                                sumCmt = msg( 'summary-add' ).replace(
/\$1/g, targetcat );
                                sumCmtShort = '+[[' + catNS + targetcat +
']]';
                                break;
                        case 'copy':
                                text = text.replace( this.regexCatBuilder(
sourcecat ), '[[' + catNS + sourcecat + '$1]]\n[[' + catNS + targetcat +
'$1]]\n' );
```

```
                                sumCmt = msg( 'summary-copy' ).replace(
/\$1/g, sourcecat ).replace( /\$2/g, targetcat );
                                sumCmtShort = '+[[' + catNS + sourcecat +
']]' + arr + '[[' + catNS + targetcat + ']]';
                                // If category is added through template:
                                if ( otext === text ) { text = addCategory(
text, targetcat ); }

                                break;
                        case 'move':
                                text = text.replace( this.regexCatBuilder(
sourcecat ), '[[' + catNS + targetcat + '$1]]\n' );
                                sumCmt = msg( 'summary-move' ).replace(
/\$1/g, sourcecat ).replace( /\$2/g, targetcat );
                                sumCmtShort = '±[[' + catNS + sourcecat +
']]' + arr + '[[' + catNS + targetcat + ']]';
                                break;
                        case 'remove':
                                text = text.replace( this.regexCatBuilder(
targetcat ), '' );
                                sumCmt = msg( 'summary-remove' ).replace(
/\$1/g, targetcat );
                                sumCmtShort = '-[[' + catNS + targetcat +
']]';
                                break;
                }

                if ( text === otext ) {
                        this.notFound.push( file[ 0 ] );
                        this.updateCounter();
                        return;
                }
                otext = text;

                // Remove {{uncat}} after we checked whether we changed the
text successfully.
                // Otherwise we might fail to do the changes, but still
replace {{uncat}}
                if ( mode !== 'remove' && ( !non || userGrp.indexOf(
'autoconfirmed' ) > -1 ) ) {
                        if ( !this.is_hidden ) {
                                text = this.removeUncat( text );
                                if ( text.length !== otext.length ) { sumCmt
+= '; ' + msg( 'uncatpref' ); }
                        }
                        text = this.doCleanup( text );
                }

                sumCmt += this.summary ? ' ' + this.summary : '';

                var preM = msg( 'prefix-summary' );
```

```
                var usgM = msg( 'using-summary' );
                // Try shorten summary
                if ( preM || usgM )        {
                        sumCmt = ( sumCmt.length > 250 - preM.length -
usgM.length ) ?
                                sumCmt + ' (CatAlot)' : preM + sumCmt + usgM;
                }

                if ( sumCmt.length > 254 ) // Try short summary
                { sumCmt = sumCmtShort; }

                var data = {
                        action: 'edit',
                        assert: 'user',
                        summary: sumCmt,
                        title: file[ 0 ],
                        text: text,
                        bot: true,
                        starttimestamp: this.starttimestamp,
                        basetimestamp: timestamp,
                        watchlist: this.settings.watchlist,
                        minor: this.settings.minor,
                        tags: this.changeTag,
                        token: this.edittoken
                };

                this.doAPICall( data, function ( r ) {
                        delete CAL.XHR[ file[ 0 ] ];
                        return CAL.updateUndoCounter( r );
                } );
                this.markAsDone( file[ 1 ], mode, targetcat );
        },

        markAsDone: function ( label, mode, targetcat ) {
                mode = ( function ( m ) {
                        switch ( m ) {
                                case 'add': return 'added-cat';
                                case 'copy': return 'copied-cat';
                                case 'move': return 'moved-cat';
                                case 'remove': return 'removed-cat';
                        }
                }( mode ) );
                label.addClass( 'cat_a_lot_markAsDone' ).append( '<br>' +
msg( mode, targetcat ) );
        },

        updateUndoCounter: function ( r ) {
                this.updateCounter();
                if ( !r.edit || r.edit.result !== 'Success' ) { return; }
                r = r.edit;
```

```
                this.undoList.push( {
                        title: r.title,
                        id: r.newrevid,
                        timestamp: r.newtimestamp
                } );
        },

        updateCounter: function () {
                this.counterCurrent++;
                if ( this.counterCurrent > this.counterNeeded ) {
this.displayResult(); } else { this.domCounter.text( this.counterCurrent ); }
        },

        displayResult: function () {
                document.body.style.cursor = 'auto';
                $.removeSpinner( 'fb-dialog' );
                this.progressDialog.parent()
                        .addClass( 'cat_a_lot_done' )
                        .find( '.ui-dialog-buttonpane button span' ).eq( 0 )
                        .text( mw.msg( 'Mobile-frontend-return-to-page' ) );
                var rep = this.domCounter.parent()
                        .height( 'auto' )
                        .html( '<h3>' + msg( 'done' ) + '</h3>' )
                        .append( msg( 'all-done' ) + '<br>' );
                if ( this.alreadyThere.length ) {
                        rep.append( '<h5>' + msg( 'skipped-already',
this.alreadyThere.length ) + '</h5>' )
                                .append( this.alreadyThere.join( '<br>' ) );
                }

                if ( this.notFound.length ) {
                        rep.append( '<h5>' + msg( 'skipped-not-found',
this.notFound.length ) + '</h5>' )
                                .append( this.notFound.join( '<br>' ) );
                }

                if ( this.connectionError.length ) {
                        rep.append( '<h5>' + msg( 'skipped-server',
this.connectionError.length ) + '</h5>' )
                                .append( this.connectionError.join( '<br>' )
);
                }

        },

        /**
*  @brief set parameters for API call,
*       convert targetcat to string, get selected pages/files
*  @param [dom object] targetcat with data
*  @param [string] mode action
*  @return Return API call getTargetCat with pages
```

```
*/
        doSomething: function ( targetcat, mode ) {
                var pages = this.getMarkedLabels();
                if ( !pages.length ) { return alert( mw.msg( 'Ooui-
selectfile-placeholder' ) ); }
                targetcat = $( targetcat ).closest( 'tr' ).data( 'cat' );

                this.notFound = [];
                this.alreadyThere = [];
                this.connectionError = [];
                this.counterCurrent = 1;
                this.counterNeeded = pages.length;
                this.undoList = [];
                this.XHR = {};
                this.cancelled = 0;
                this.summary = '';

                if ( $( '#cat_a_lot_comment' ).prop( 'checked' ) ) {
this.summary = window.prompt( msg( 'edit-question' ), '' ); } // TODO custom
pre-value
                if ( this.summary !== null ) {
                        mw.loader.using( [ 'jquery.ui', 'jquery.spinner',
'mediawiki.util' ], function () {
                                CAL.showProgress();
                                CAL.getTargetCat( pages, targetcat, mode );
                        } );
                }

        },

        doAPICall: function ( params, callback ) {
                params = $.extend( {
                        action: 'query',
                        format: 'json'
                }, params );

                var i = 0,
                        apiUrl = this.apiUrl,
                        doCall,
                        handleError = function ( jqXHR, textStatus,
errorThrown ) {
                                mw.log( 'Error: ', jqXHR, textStatus,
errorThrown );
                                if ( i < 4 ) {
                                        window.setTimeout( doCall, 300 );
                                        i++;
                                } else if ( params.title ) {
                                        this.connectionError.push(
params.title );
                                        this.updateCounter();
                                        return;
```

```
                        }
                };
        doCall = function () {
                var xhr = $.ajax( {
                        url: apiUrl,
                        cache: false,
                        dataType: 'json',
                        data: params,
                        type: 'POST',
                        success: callback,
                        error: handleError
                } );

                if ( params.action === 'edit' && !CAL.cancelled ) {
CAL.XHR[ params.title ] = xhr; }
        };
        doCall();
    },

    createCatLinks: function ( symbol, list, table ) {
        list.sort();
        var button = ( this.settings.button && mw.loader.getState(
'jquery.ui' ) === 'ready' ) ? 1 : 0;
        for ( var c = 0; c < list.length; c++ ) {
                var $tr = $( '<tr>' ),
                        $link = $( '<a>', {
                                href: mw.util.getUrl(
CAL.localCatName + list[ c ] ),
                                text: list[ c ]
                        } ),
                        $buttons = [];
                $tr.data( 'cat', list[ c ] );
                $link.on( 'click', function ( e ) {
                        if ( !e.ctrlKey ) {
                                e.preventDefault();
                                CAL.updateCats( $( this ).closest(
'tr' ).data( 'cat' ) );
                        }
                } );

                $tr.append( $( '<td>' ).text( symbol ) )
                        .append( $( '<td>' ).append( $link ) );

                $buttons.push( $( '<a>' )
                        .text( mw.msg( 'Centralnotice-remove' ) )
                        .on( 'click', function () {
                                CAL.doSomething( this, 'remove' );
                        } )
                        .addClass( 'cat_a_lot_move' )
                );
                if ( button ) {
```

```
                                $buttons.slice( -1 )[ 0 ].button( {
                                        icons: { primary: 'ui-icon-
minusthick' },
                                        showLabel: false,
                                        text: false
                                } );
                        }

                        if ( this.origin ) {
                        // Can't move to source category
                                if ( list[ c ] !== this.origin ) {
                                        $buttons.push( $( '<a>' )
                                                .text( msg( 'move' ) )
                                                .on( 'click', function () {
                                                        CAL.doSomething(
this, 'move' );
                                                } )
                                                .addClass( 'cat_a_lot_move' )
                                        );
                                        if ( button ) {
                                                $buttons.slice( -1 )[ 0
].button( {
                                                        icons: { primary:
'ui-icon-arrowthick-1-e' },
                                                        showLabel: false,
                                                        text: false
                                                } );
                                        }

                                        $buttons.push( $( '<a>' )
                                                .text( msg( 'copy' ) )
                                                .on( 'click', function () {
                                                        CAL.doSomething(
this, 'copy' );
                                                } )
                                                .addClass( 'cat_a_lot_action'
)
                                        );
                                        if ( button ) {
                                                $buttons.slice( -1 )[ 0
].button( {
                                                        icons: { primary:
'ui-icon-plusthick' },
                                                        showLabel: false,
                                                        text: false
                                                } );
                                        }

                                }
                        } else {
                                $buttons.push( $( '<a>' )
```

```
                                        .text( msg( 'add' ) )
                                        .on( 'click', function () {
                                                CAL.doSomething( this, 'add'
);
                                        } )
                                        .addClass( 'cat_a_lot_action' )
                                );
                                if ( button ) {
                                        $buttons.slice( -1 )[ 0 ].button( {
                                                icons: { primary: 'ui-icon-
plusthick' },
                                                showLabel: false,
                                                text: false
                                        } );
                                }

                        }
                        // TODO CSS may extern
                        var css = button ? { fontSize: '.6em', margin: '0',
width: '2.5em' } : {};
                        for ( var b = 0; b < $buttons.length; b++ ) {
$tr.append( $( '<td>' ).append( $buttons[ b ].css( css ) ) ); }

                        table.append( $tr );
                }
        },

        getCategoryList: function () {
                this.catCounter = 0;
                this.getParentCats();
                this.getSubCats();
        },

        _getPageQuery: function ( data ) {
        // There should be only one, but we don't know its ID
                if ( data && data.query && data.query.pages ) {
                        data = data.query.pages;
                        for ( var p in data ) { return data[ p ]; }
                }
        },

        /**
*  @brief takes this.currentCategory if redir_category is configured
** Cat pages with more than one cat link are still not supported for sure
*  @return soft redirected cat
*/
        solveSoftRedirect: function () {
                this.doAPICall( {
                        prop: 'links', // TODO: For more accuracy the
revisions could be checked
                        titles: 'Category:' + this.currentCategory,
```

```
                        // 'rvprop': 'content',
                        // 'pllimit': 'max',
                        plnamespace: 14
                }, function ( page ) {
                        page = CAL._getPageQuery( page );
                        if ( page ) {
                                var lks = page.links;
                                if ( lks && lks.length === 1 && lks[ 0
].title ) {
                                        CAL.currentCategory = lks[ 0
].title.replace( reCat, '' );
                                        $searchInput.val( CAL.currentCategory
);
                                        return CAL.getCategoryList();
                                } else {
                                // TODO? better translatable warning message:
"Please solve the category soft redirect manually!"
                                        $resultList.html( '<span
id="cat_a_lot_no_found">' + mw.msg( 'Apifeatureusage-warnings', mw.msg(
'Categorytree-not-found', CAL.currentCategory ) ) + '</span>' );
                                }
                        }
                } );
        },

        showCategoryList: function () {
                if ( this.settings.redir_category &&
this.settings.redir_category === this.parentCats[ 0 ] ) { return
this.solveSoftRedirect(); }

                var table = $( '<table>' );

                this.createCatLinks( '↑', this.parentCats, table );
                this.createCatLinks( '→', [ this.currentCategory ], table );
                // Show on soft-redirect
                if ( $searchInput.val() === this.currentCategory &&
this.origin !== this.currentCategory ) { this.createCatLinks( '→', [
this.origin ], table ); }
                this.createCatLinks( '↓', this.subCats, table );

                $resultList.empty();
                $resultList.append( table );

                document.body.style.cursor = 'auto';

                // Reset width
                $container.width( '' );
                $container.height( '' );
                $container.width( Math.min( table.width() * 1.1 + 15, $(
window ).width() - 10 ) );
```

```
                $resultList.css( {
                        maxHeight: Math.min( this.setHeight, $( window
).height() - $container.position().top - $settingsLink.outerHeight() -
$selections.outerHeight() - 15 ),
                        height: ''
                } );
                table.width( '100%' );
                $container.height( Math.min( $container.height(),
$head.offset().top - $container.offset().top + 10 ) );
                $container.offset( { left: $( window ).width() -
$container.outerWidth() } ); // Fix overlap
        },

        updateCats: function ( newcat ) {
                document.body.style.cursor = 'wait';
                this.currentCategory = newcat;
                $resultList.html( '<div class="cat_a_lot_loading">' + mw.msg(
'Wikieditor-loading' ) + '</div>' );
                this.getCategoryList();
        },

        doUndo: function () {
                this.cancelled = 0;
                this.doAbort();
                if ( !this.undoList.length ) { return; }

                $( '.cat_a_lot_feedback' ).removeClass( 'cat_a_lot_done' );
                this.counterNeeded = this.undoList.length;
                this.counterCurrent = 1;

                document.body.style.cursor = 'wait';

                var query = {
                        action: 'edit',
                        user: mw.config.get( 'wgUserName' ),
                        bot: true,
                        minor: this.settings.minor,
                        starttimestamp: this.starttimestamp,
                        watchlist: this.settings.watchlist,
                        tags: this.changeTag,
                        token: this.edittoken
                };
                for ( var i = 0; i < this.undoList.length; i++ ) {
                        var uID = this.undoList[ i ];
                        query.title = uID.title;
                        query.undo = uID.id;
                        query.basetimestamp = uID.timestamp;
                        this.doAPICall( query, function ( r ) {
                        // TODO: Add "details" to progressbar?
                        // $resultList.append( [mw.msg('Filerevert-submit') +
" done " + r.edit.title, '<br>' ] );
```

```
                                if ( r && r.edit ) { mw.log( 'Revert done',
r.edit.title ); }
                                CAL.updateCounter();
                        } );
                }
        },

        doAbort: function () {
                for ( var t in this.XHR ) { this.XHR[ t ].abort(); }

                if ( this.cancelled ) { // still not for undo
                        this.progressDialog.remove();
                        this.toggleAll( false );
                        $head.last().show();
                }
                this.cancelled = 1;
        },

        showProgress: function () {
                document.body.style.cursor = 'wait';
                this.progressDialog = $( '<div>' )
                        .html( ' ' + msg( 'editing' ) + ' <span
id="cat_a_lot_current">' + CAL.counterCurrent + '</span> ' + msg( 'of' ) +
CAL.counterNeeded )
                        .prepend( $.createSpinner( { id: 'fb-dialog', size:
'large' } ) )
                        .dialog( {
                                width: 450,
                                height: 180,
                                minHeight: 90,
                                modal: true,
                                resizable: false,
                                draggable: false,
                                // closeOnEscape: true,
                                dialogClass: 'cat_a_lot_feedback',
                                buttons: [ {
                                        text: mw.msg( 'Cancel' ), // Stops
all actions
                                        click: function () {
                                                $( this ).dialog( 'close' );
                                        }
                                } ],
                                close: function () {
                                        CAL.cancelled = 1;
                                        CAL.doAbort();
                                        $( this ).remove();
                                },
                                open: function ( event, ui ) { // Workaround
modify
                                        ui = $( this ).parent();
                                        ui.find( '.ui-dialog-titlebar'
```

```
).hide();
                                        ui.find( '.ui-dialog-buttonpane.ui-
widget-content' )
                                                .removeClass( 'ui-widget-
content' );
                                /* .find( 'span' ).css( { fontSize: '90%' }
)*/
                                }
                        } );
                if ( $head.children().length < 3 ) {
                        $( '<span>' )
                                .css( {
                                        'float': 'right',
                                        fontSize: '75%'
                                } )
                                .append( [ '[ ',
                                        $( '<a>', { title: 'Revert all last
done edits' } ) // TODO i18n
                                                .on( 'click', function () {
                                                        if ( window.confirm(
mw.msg( 'Apifeatureusage-warnings', this.title + '⁉' ) ) ) {
                                                                CAL.doUndo();
                                                                $( this
).parent().remove();
                                                        }
                                                        return false;
                                                } )
                                                .addClass( 'new' )
                                                .text( mw.msg( 'Filerevert-
submit' ) ),
                                        ' ]'
                                ] ).insertAfter( $link );
                }

                this.domCounter = $( '#cat_a_lot_current' );
        },

        minimize: function ( e ) {
                CAL.top = Math.max( 0, $container.position().top );
                CAL.height = $container.height();
                $dataContainer.hide();
                $container.animate( {
                        height: $head.height(),
                        top: $( window ).height() - $head.height() * 1.4
                }, function () {
                        $( e.target ).one( 'click', CAL.maximize );
                } );
        },

        maximize: function ( e ) {
                $dataContainer.show();
```

```
                $container.animate( {
                        top: CAL.top,
                        height: CAL.height
                }, function () {
                        $( e.target ).one( 'click', CAL.minimize );
                } );
        },

        run: function () {
                if ( $( '.cat_a_lot_enabled' )[ 0 ] ) {
                        this.makeClickable();
                        if ( !this.executed ) { // only once
                                $selectInvert.text( mw.msg( 'Checkbox-invert'
) );
                                if ( this.settings.editpages &&
this.pageLabels[ 0 ] ) {
                                        $selectFiles.text( mw.msg( 'Prefs-
files' ) );
                                        $selectPages.text( mw.msg(
'Categories' ) ).parent().show();
                                }
                                $link.after( $( '<a>' )
                                        .text( '–' )
                                        .css( { fontWeight: 'bold',
marginLeft: '.7em' } )
                                        .one( 'click', this.minimize )
                                );
                        }
                        $dataContainer.show();
                        $container.one( 'mouseover', function () {
                                $( this )
                                        .resizable( {
                                                handles: 'n',
                                                alsoResize:
'#cat_a_lot_category_list',
                                                resize: function () {
                                                        $resultList
                                                                .css( {
maxHeight: '',
width: ''
                                                                } );
                                                },
                                                start: function ( e, ui ) {
// Otherwise box get static if sametime resize with draggable
                                                        ui.helper.css( {
                                                                top:
ui.helper.offset().top - $( window ).scrollTop(),
                                                                position:
'fixed'
                                                        } );
                                                },
```

```
                                                stop: function () {
                                                        CAL.setHeight =
$resultList.height();
                                                }
                                        } )
                                        .draggable( {
                                                cursor: 'move',
                                                start: function ( e, ui ) {
                                                        ui.helper.on(
'click.prevent',
                                                                function ( e
) { e.preventDefault(); }
                                                        );
                                                        ui.helper.css(
'height', ui.helper.height() );
                                                },
                                                stop: function ( e, ui ) {
                                                        setTimeout(
                                                                function () {
ui.helper.off( 'click.prevent' );
                                                                }, 300
                                                        );
                                                }
                                        } )
                                        .one( 'mousedown', function () {
                                                $container.height(
$container.height() ); // Workaround to calculate
                                        } );
                                $resultList
                                        .css( { maxHeight: 450 } );
                        } );
                        this.updateCats( this.origin || 'Images' );

                        $link.html( $( '<span>' )
                                .text( '×' )
                                .css( { font: 'bold 2em monospace',
lineHeight: '.75em' } )
                        );
                        $link.next().show();
                        if ( this.cancelled ) { $head.last().show(); }
                        mw.cookie.set( 'catAlotO', ns ); // Let stay open on
new window
                } else { // Reset
                        $dataContainer.hide();
                        $container
                                .draggable( 'destroy' )
                                .resizable( 'destroy' )
                                .removeAttr( 'style' );
                        // Unbind click handlers
                        this.labels.off( 'click.catALot' );
                        this.setHeight = 450;
```

```
                        $link.text( 'Cat-a-lot' )
                                .nextAll().hide();
                        this.executed = 1;
                        mw.cookie.set( 'catAlotO', null );
                }
        },

        manageSettings: function () {
                mw.loader.using( [ 'ext.gadget.SettingsManager',
'ext.gadget.SettingsUI', 'jquery.ui' ], CAL._manageSettings );
        },

        _manageSettings: function () {
                mw.libs.SettingsUI( CAL.defaults, 'Cat-a-lot' )
                        .show()
                        .done( function ( s, verbose, loc, settingsOut, $dlg
) {
                                var mustRestart = false,
                                        _restart = function () {
                                                if ( !mustRestart ) { return;
}
                                                $container.remove();
                                                CAL.labels.off(
'click.catALot' );
                                                CAL.init();
                                        },
                                        _saveToJS = function () {
                                                var opt =
mw.libs.settingsManager.option( {
                                                                optionName:
'catALotPrefs',
                                                                value:
CAL.settings,
encloseSignature: 'catALot',
                                                                encloseBlock:
'////////// Cat-a-lot user preferences //////////\n',
triggerSaveAt: /Cat.?A.?Lot/i,
                                                                editSummary:
msg( 'pref-save-summary' )
                                                        } ),
                                                        oldHeight =
$dlg.height(),
                                                        $prog = $( '<div>' );

                                                $dlg.css( 'height', oldHeight
)
                                                        .html( '' );
                                                $prog.css( {
                                                        height: Math.round(
oldHeight / 8 ),
                                                        'margin-top':
```

```
Math.round( ( 7 * oldHeight ) / 16 )
                                                        } )
                                                                .appendTo( $dlg );

                                                        $dlg.parent()
                                                                .find( '.ui-dialog-
buttonpane button' )
                                                                .button( 'option',
'disabled', true );

                                                        opt.save()
                                                                .done( function (
text, progress ) {
$prog.progressbar( {
value: progress
                                                                        } );
$prog.fadeOut( function () {
$dlg.dialog( 'close' );
_restart();
                                                                        } );
                                                                } )
                                                                .progress( function (
text, progress ) {
$prog.progressbar( {
value: progress
                                                                        } );
                                                                // TODO: Add
"details" to progressbar
                                                                } )
                                                                .fail( function (
text ) {
$prog.addClass( 'ui-state-error' );
                                                                        $dlg.prepend(
$( '<p>' )
.text( text ) );
                                                                } );
                                                        };
                                                $.each( settingsOut, function ( n, v ) {
                                                        if ( v.forcerestart && CAL.settings[
v.name ] !== v.value ) { mustRestart = true; }
                                                        CAL.settings[ v.name ] =
CAL.catALotPrefs[ v.name ] = v.value;
                                                } );
                                                switch ( loc ) {
                                                        case 'page':
                                                                $dlg.dialog( 'close' );
                                                                _restart();
                                                                break;
                                                        case 'account-publicly':
                                                                _saveToJS();
                                                                break;
```

```
                                }
                        } );
        },

        _initSettings: function () {
                if ( this.settings.watchlist ) { return; }
                this.catALotPrefs = window.catALotPrefs || {};
                for ( var i = 0; i < this.defaults.length; i++ ) {
                        var v = this.defaults[ i ];
                        v.value = this.settings[ v.name ] = (
this.catALotPrefs[ v.name ] || v['default'] );
                        v.label = msg( v.label_i18n );
                        if ( v.select_i18n ) {
                                v.select = {};
                                $.each( v.select_i18n, function ( i18nk, val
) {
                                        v.select[ msg( i18nk ) ] = val;
                                } );
                        }
                }
        },
        /* eslint-disable camelcase */
        defaults: [ {
                name: 'watchlist',
                'default': 'preferences',
                label_i18n: 'watchlistpref',
                select_i18n: {
                        watch_pref: 'preferences',
                        watch_nochange: 'nochange',
                        watch_watch: 'watch',
                        watch_unwatch: 'unwatch'
                }
        }, {
                name: 'minor',
                'default': false,
                label_i18n: 'minorpref'
        }, {
                name: 'editpages',
                'default': project !== 'commonswiki', // on Commons false
                label_i18n: 'editpagespref',
                forcerestart: true
        }, {
                name: 'docleanup',
                'default': false,
                label_i18n: 'docleanuppref'
        }, {
                name: 'subcatcount',
                'default': 50,
                min: 5,
                max: 500,
                label_i18n: 'subcatcountpref',
```

```
                forcerestart: true
        }, {
                name: 'uncat',
                'default': project === 'commonswiki', // on Commons true
                label_i18n: 'uncatpref'
        }, {
                name: 'button',
                'default': true,
                label_i18n: 'buttonpref'
        } ]
/* eslint-enable camelcase */
};

// The gadget is not immediately needed, so let the page load normally
window.setTimeout( function () {
        non = mw.config.get( 'wgUserName' );
        if ( non ) {
                if ( mw.config.get( 'wgRelevantUserName' ) === non ) { non =
0; } else {
                        $.each( [ 'sysop', 'filemover', 'editor',
'rollbacker', 'patroller', 'autopatrolled', 'image-reviewer', 'reviewer',
'extendedconfirmed' ], function ( i, v ) {
                                non = $.inArray( v, userGrp ) === -1;
                                return non;
                        } );
                }
        } else { non = 1; }

        switch ( ns ) {
                case 14:
                        CAL.searchmode = 'category';
                        CAL.origin = mw.config.get( 'wgTitle' );
                        break;
                case -1:
                        CAL.searchmode = {
                        // list of accepted special page names mapped to
search mode names
                                Contributions: 'contribs',
                                Listfiles: non ? null : 'listfiles',
                                Prefixindex: non ? null : 'prefix',
                                Search: 'search',
                                Uncategorizedimages: 'gallery'
                        }[ mw.config.get( 'wgCanonicalSpecialPageName' ) ];
                        break;
                case 2:
                case 0:
                        CAL.searchmode = 'gallery';
                        var parents = $( '#mw-normal-catlinks ul' ).find(
'a[title]' ), n;
                        parents.each( function ( i ) {
                                if ( new RegExp( mw.config.get( 'wgTitle' ),
```

```
'i' ).test( $( this ).text() ) ) {
                                        n = i;
                                        return false;
                                }
                        } );
                        CAL.origin = parents.eq( n || 0 ).text();
        }

        if ( CAL.searchmode ) {
                var loadingLocalizations = 1;
                var loadLocalization = function ( lang, cb ) {
                        loadingLocalizations++;
                        switch ( lang ) {
                                case 'zh-hk':
                                case 'zh-mo':
                                case 'zh-tw':
                                        lang = 'zh-hant';
                                        break;
                                case 'zh':
                                case 'zh-cn':
                                case 'zh-my':
                                case 'zh-sg':
                                        lang = 'zh-hans';
                                        break;
                        }

                        $.ajax( {
                                url: commonsURL,
                                dataType: 'script',
                                data: {
                                        title: 'MediaWiki:Gadget-Cat-a-
lot.js/' + lang,
                                        action: 'raw',
                                        ctype: 'text/javascript',
                                        // Allow caching for 28 days
                                        maxage: 2419200,
                                        smaxage: 2419200
                                },
                                cache: true,
                                success: cb,
                                error: cb
                        } );
                };
                var maybeLaunch = function () {
                        loadingLocalizations--;
                        function init() {
                                $( function () {
                                        CAL.init();
                                } );
                        }
                        if ( !loadingLocalizations ) { mw.loader.using( [
```

```
'user' ], init, init ); }
                };

                var userlang = mw.config.get( 'wgUserLanguage' ),
                        contlang = mw.config.get( 'wgContentLanguage' );
                if ( userlang !== 'en' ) { loadLocalization( userlang,
maybeLaunch ); }
                if ( $.inArray( contlang, [ 'en', userlang ] ) === -1 ) {
loadLocalization( contlang, maybeLaunch ); }
                maybeLaunch();
        }
}, 400 );

/**
 * When clicking a cat-a-lot label with Shift pressed, select all labels
between the current and last-clicked one.
 */
$.fn.catALotShiftClick = function ( cb ) {
        var prevCheckbox = null,
                $box = this;
        // When our boxes are clicked..
        $box.on( 'click.catALot', function ( e ) {
        // Prevent following the link and text selection
                if ( !e.ctrlKey ) { e.preventDefault(); }
                // Highlight last selected
                $( '#cat_a_lot_last_selected' )
                        .removeAttr( 'id' );
                var $thisControl = $( e.target ),
                        method;
                if ( !$thisControl.hasClass( 'cat_a_lot_label' ) ) {
$thisControl = $thisControl.parents( '.cat_a_lot_label' ); }

                $thisControl.attr( 'id', 'cat_a_lot_last_selected' )
                        .toggleClass( 'cat_a_lot_selected' );
                // And one has been clicked before…
                if ( prevCheckbox !== null && e.shiftKey ) {
                        method = $thisControl.hasClass( 'cat_a_lot_selected'
) ? 'addClass' : 'removeClass';
                        // Check or uncheck this one and all in-between
checkboxes
                        $box.slice(
                                Math.min( $box.index( prevCheckbox ),
$box.index( $thisControl ) ),
                                Math.max( $box.index( prevCheckbox ),
$box.index( $thisControl ) ) + 1
                        )[ method ]( 'cat_a_lot_selected' );
                }
                // Either way, update the prevCheckbox variable to the one
clicked now
                prevCheckbox = $thisControl;
                if ( $.isFunction( cb ) ) { cb(); }
```

```
        } );
        return $box;
};

}( jQuery, mediaWiki ) );
// </nowiki>
```

Innovation Fund

Namespaces

- Message
- Discussion

Variants

Blue Gold Program Wiki

The wiki version of the Lessons Learnt Report of the Blue Gold program, documents the experiences of a technical assistance (TA) team working in a development project implemented by the Bangladesh Water Development Board (BWDB) and the Department of Agricultural Extension (DAE) over an eight+ year period from March 2013 to December 2021. The wiki lessons learnt report (LLR) is intended to complement the BWDB and DAE project completion reports (PCRs), with the aim of recording lessons learnt for use in the design and implementation of future interventions in the coastal zone.