```
/*
* ================================================================
*                     FileUploadWizard.js
* Script for uploading files through a dynamic questionnaire.
* This is the code to accompany [[Wikipedia:File Upload Wizard]].
* ================================================================
*/

var fuwTesting = false;
var fuwDefaultTextboxLength = 60;
var fuwDefaultTextareaWidth = '90%';
var fuwDefaultTextareaLines = 3;

// ================================================================
// Constructor function of global fuw (= File Upload Wizard) object
// ================================================================
function fuwGlobal() {

   // Loading the accompanying .css
   mw.loader.load( mw.config.get('wgServer') + mw.config.get('wgScriptPath')
+

'/index.php?title=MediaWiki:FileUploadWizard.css&action=raw&ctype=text/css',
      'text/css'  );

   // see if user is logged in, autoconfirmed, experienced etc.
   this.getUserStatus();

   fuwSetVisible('warningLoggedOut', (this.userStatus == 'anon'));
   fuwSetVisible('warningNotConfirmed', (this.userStatus ==
'notAutoconfirmed'));
   if (this.userStatus == 'anon') {
      return;
   }
   fuwSetVisible('fuwStartScriptLink', false);

   // create the form element to wrap the main ScriptForm area
   // containing input elements of Step2 and Step3
   var frm = fuwGet('fuwScriptForm');
   if (! frm) {
      frm = document.createElement('form');
      frm.id = "fuwScriptForm";
      var area = fuwGet('placeholderScriptForm');
      var parent = area.parentNode;
      parent.insertBefore(frm, area);
      parent.removeChild(area);
      frm.appendChild(area);
   }
```

```
   this.ScriptForm = frm;

   // create the TargetForm element that contains the filename
   // input box, together with hidden input controls.
   // This is the form that is actually submitted to the api.php.
   frm = fuwGet('TargetForm');
   if (! frm) {
      frm = document.createElement('form');
      frm.id = "TargetForm";
      frm.method = "post";
      frm.enctype = "multipart/form-data";
      // "enctype" doesn't work properly on IE; need "encoding" instead:
      frm.encoding = "multipart/form-data";
      // we'll submit via api.php, not index.php, mainly because that
      // allows us to use a proper edit summary different from the page
content
      frm.action = mw.config.get('wgServer') + mw.config.get('wgScriptPath')
+ '/api.php';

      // However, since api.php sends back a response page that humans won't
want to read,
      // we'll have to channel that response away and discard it. We'll use a
hidden iframe
      // for that purpose.
      // Unfortunately, it doesn't seem possible to submit file upload
content through an
      // Xmlhtml object via Ajax.

      frm.target = "TargetIFrame";
      //testing:
      //frm.target = "_blank";
      var area = fuwGet('placeholderTargetForm');
      var parent = area.parentNode;
      parent.insertBefore(frm, area);
      parent.removeChild(area);
      frm.appendChild(area);
   }
   this.TargetForm = frm;

   // For the testing version, create a third form that will display
   // the contents to be submitted, at the bottom of the page
   if (fuwTesting) {
      frm = fuwGet('fuwTestForm');
      if (! frm) {
         frm = document.createElement('form');
         frm.id = "fuwTestForm";
         var area = fuwGet('placeholderTestForm');
         var parent = area.parentNode;
         parent.insertBefore(frm, area);
         parent.removeChild(area);
         frm.appendChild(area);
```

```
      }
      this.TestForm = frm;
   }

   // objects to hold cached results during validation and processing
   this.opts = { };
   this.warn = { };

   // create the input filename box
   var filebox  = document.createElement('input');
   filebox.id   = 'file';
   filebox.name = 'file';
   filebox.type = 'file';
   filebox.size = fuwDefaultTextboxLength;
   filebox.onchange = fuwValidateFile;
   filebox.accept =
'image/png,image/jpeg,image/gif,image/svg+xml,image/tiff,image/x-xcf,applicat
ion/pdf,image/vnd.djvu,audio/ogg,video/ogg,audio/rtp-midi';
   fuwAppendInput('file', filebox);

   // create hidden controls for sending the remaining API parameters:
   fuwMakeHiddenfield('action', 'upload', 'apiAction');
   fuwMakeHiddenfield('format', 'xml', 'apiFormat');
   fuwMakeHiddenfield('filename', '', 'apiFilename');
   fuwMakeHiddenfield('text', '', 'apiText');
   fuwMakeHiddenfield('comment', '', 'apiComment');
   fuwMakeHiddenfield('token', mw.user.tokens.get('editToken'), 'apiToken');
   fuwMakeHiddenfield('ignorewarnings', 1, 'apiIgnorewarnings');
   fuwMakeHiddenfield('watch', 1, 'apiWatch');

   if (fuwTesting) {
      fuwMakeHiddenfield('title', mw.config.get('wgPageName') + "/sandbox",
'SandboxTitle');
      fuwMakeHiddenfield('token', mw.user.tokens.get('editToken'),
'SandboxToken');
      fuwMakeHiddenfield('recreate', 1, 'SandboxRecreate');
   }

   // create a hidden IFrame to send the api.php response to
   var ifr = document.createElement('iframe');
   ifr.id   = "TargetIFrame";
   ifr.name = "TargetIFrame";
   //ifr.setAttribute('style', 'float:right;width:150px;height:150px;');
   ifr.style.display = "none";
   ifr.src = "";

   fuwAppendInput('TargetIFrame', ifr);

   if (fuwTesting) {

      // create the sandbox submit button
```

```
   btn = document.createElement('input');
   btn.id = 'SandboxButton';
   btn.value = 'Sandbox';
   btn.name  = 'Sandbox';
   btn.disabled = true;
   btn.type = 'button';
   btn.style.width = '12em';
   btn.onclick = fuwSubmitSandbox;
   fuwAppendInput('SandboxButton', btn);

}

// create the real submit button
btn = document.createElement('input');
btn.id = "SubmitButton";
btn.value = "Upload";
btn.name = "Upload";
btn.disabled = true;
btn.type = "button";
btn.onclick = fuwSubmitUpload;
btn.style.width = '12em';
fuwAppendInput('SubmitButton', btn);

// create the Commons submit button
btn = document.createElement('input');
btn.id = "CommonsButton";
btn.value = "Upload on Commons";
btn.name  = "Upload_on_Commons";
btn.disabled = true;
btn.type = "button";
btn.onclick = fuwSubmitCommons;
btn.style.width = '12em';
fuwAppendInput('CommonsButton', btn);

// create reset buttons
for (i = 1; i<=2; i++) {
   btn = document.createElement('input');
   btn.id = 'ResetButton' + i;
   btn.value = "Reset form";
   btn.name  = "Reset form";
   btn.type  = "button";
   btn.onclick = fuwReset;
   btn.style.width = '12em';
   fuwAppendInput('ResetButton' + i, btn);
}

// names of radio button fields
var optionRadioButtons = {
   // top-level copyright status choice
   'FreeOrNonFree' : ['OptionFree','OptionNonFree','OptionNoGood'],
   // main subsections under OptionFree
```

```
      'FreeOptions'   : ['OptionOwnWork', 'OptionThirdParty',
'OptionFreeWebsite',
                         'OptionPDOld', 'OptionPDOther'],
      // main subsections under OptionNonFree
      'NonFreeOptions': ['OptionNFSubject','OptionNF3D','OptionNFExcerpt',
                         'OptionNFCover','OptionNFLogo','OptionNFPortrait',
                         'OptionNFMisc'],
      // response options inside warningFileExists
      'FileExistsOptions':
                        ['NoOverwrite','OverwriteSame','OverwriteDifferent'],
      // choice of evidence in OptionThirdParty subsection
      'ThirdPartyEvidenceOptions' :
                        ['ThirdPartyEvidenceOptionLink',
                         'ThirdPartyEvidenceOptionOTRS',
                         'ThirdPartyEvidenceOptionOTRSForthcoming',
                         'ThirdPartyEvidenceOptionNone'],
      // choice of PD status in OptionPDOld subsection
      'PDOldOptions'  : ['PDUS1923','PDURAA','PDFormality','PDOldOther'],
      // choice of PD status in OptionPDOther subsection
      'PDOtherOptions': ['PDOtherUSGov','PDOtherOfficial','PDOtherSimple',
                         'PDOtherOther'],
      // whether target article is wholly or only partly dedicated to
discussing non-free work:
      'NFSubjectCheck':
['NFSubjectCheckDedicated','NFSubjectCheckDiscussed'],
      'NF3DCheck'     : ['NF3DCheckDedicated','NF3DCheckDiscussed'],
      // choice about copyright status of photograph in OptionNF3D
      'NF3DOptions'   : ['NF3DOptionFree','NF3DOptionSame']
   };
   for (var group in optionRadioButtons) {
      var op = optionRadioButtons[group];
      for (i=0; i<op.length; i++) {
         fuwMakeRadiobutton(group, op[i]);
      }
   }
   this.ScriptForm.NoOverwrite.checked = true;

   // input fields that trigger special
   // onchange() event handlers for validation:
   fuwMakeTextfield('InputName', fuwValidateFilename);
   fuwMakeTextfield('NFArticle', fuwValidateNFArticle);

   // names of input fields that trigger normal
   // validation event handler
   var activeTextfields = [
      'Artist3D','Country3D',
      'Date','OwnWorkCreation','OwnWorkPublication',
      'Author','Source',
      'Permission','ThirdPartyOtherLicense',
      'ThirdPartyEvidenceLink','ThirdPartyOTRSTicket',
      'FreeWebsiteOtherLicense',
```

```
      'PDOldAuthorLifetime','Publication',
      'PDOldCountry','PDOldPermission',
      'PDOfficialPermission','PDOtherPermission',
      'NFSubjectPurpose', 'NF3DOrigDate', 'NF3DPurpose',
      'NF3DCreator',
      'NFPortraitDeceased',
      'EditSummary'
   ];
   for (i=0; i<activeTextfields.length; i++) {
      fuwMakeTextfield(activeTextfields[i]);
   }

   // names of multiline textareas
   var activeTextareas = [
      'InputDesc','NF3DPermission',
      'NFCommercial','NFPurpose','NFReplaceableText',
      'NFReplaceable','NFCommercial','NFMinimality','AnyOther'
   ];
   for (i=0; i<activeTextareas.length; i++) {
      fuwMakeTextarea(activeTextareas[i]);
   };

   var checkboxes = [

'NFCoverCheckDedicated','NFLogoCheckDedicated','NFPortraitCheckDedicated'
   ];
   for (i=0; i<checkboxes.length; i++) {
      fuwMakeCheckbox(checkboxes[i]);
   };

   var licenseLists = {
      'OwnWorkLicense' :
        // array structure as expected for input to fuwMakeSelection()
function.
        // any entry that is a two-element array will be turned into an
option
        // (first element is the value, second element is the display
string).
        // Entries that are one-element arrays will be the label of an option
group.
        // Zero-element arrays mark the end of an option group.
        [
        ['Allow all use as long as others credit you and share it under
similar conditions'],
        ['self|GFDL|cc-by-sa-3.0|migration=redundant',
         'Creative Commons Attribution-Share Alike 3.0 + GFDL (recommended)',
         true],
        ['self|cc-by-sa-3.0',
         'Creative Commons Attribution-Share Alike 3.0'],
        [],
        ['Allow all use as long as others credit you'],
```

```
        ['self|cc-by-3.0',
         'Creative Commons Attribution 3.0'],
        [],
        ['Reserve no rights'],
        ['cc-zero',
         'CC-zero Universal Public Domain Dedication'],
        []
        ],
      'ThirdPartyLicense' :
        [
        ['', 'please select the correct license...'],
        ['Freely licensed:'],
        ['cc-by-sa', 'Creative Commons Attribution-Share Alike (cc-by-sa)'],
        ['cc-by', 'Creative Commons Attribution (cc-by)'],
        ['GFDL', 'GNU Free Documentation License (GFDL)'],
        [],
        ['No rights reserved:'],
        ['PD-author', 'Public domain'],
        [],
        ['Other (see below)'],
        []
        ],
      'FreeWebsiteLicense' :
        [
        ['', 'please select the correct license...'],
        ['Freely licensed:'],
        ['cc-by-sa', 'Creative Commons Attribution-Share Alike (cc-by-sa)'],
        ['cc-by', 'Creative Commons Attribution (cc-by)'],
        ['GFDL', 'GNU Free Documentation License (GFDL)'],
        [],
        ['No rights reserved:'],
        ['PD-author', 'Public domain'],
        [],
        ['Other (see below)'],
        []
        ],
      'USGovLicense' :
       [
       ['PD-USGov', 'US Federal Government'],
       ['PD-USGov-NASA','NASA'],
       ['PD-USGov-Military-Navy','US Navy'],
       ['PD-USGov-NOAA','US National Oceanic and Atmospheric
Administration'],
       ['PD-USGov-Military-Air_Force','US Air Force'],
       ['PD-USGov-Military-Army','US Army'],
       ['PD-USGov-CIA-WF','CIA World Factbook'],
       ['PD-USGov-USGS','United States Geological Survey']
       ],
      'IneligibleLicense' :
       [
       ['', 'please select one...'],
```

```
       ['PD-shape','Item consists solely of simple geometric shapes'],
       ['PD-text','Item consists solely of a few individual words or
letters'],
       ['PD-textlogo','Logo or similar item consisting solely of letters and
simple geometric shapes'],
       ['PD-chem','Chemical structural formula'],
       ['PD-ineligible','Other kind of item that contains no original
authorship']
       ],
      'NFSubjectLicense' :
       [
       ['', 'please select one...'],
       ['Non-free 2D art', '2-dimensional artwork (painting, drawing etc.)'],
       ['Non-free historic image', 'Historic photograph'],
       ['Non-free fair use in', 'something else (please describe in
description field on top)']
       ],
      'NF3DLicense' :
       [
       ['', 'please select one...'],
       ['Non-free architectural work', 'Architectural work'],
       ['Non-free 3D art', 'Other 3-dimensional creative work (sculpture
etc.)']
       ],
      'NFCoverLicense' :
         [
         ['', 'please select one...'],
         ['Non-free book cover', 'Cover page of a book'],
         ['Non-free album cover', 'Cover of a sound recording (album, single,
song, CD)'],
         ['Non-free game cover', 'Cover of a video/computer game'],
         ['Non-free magazine cover', 'Cover page of a magazine'],
         ['Non-free video cover', 'Cover of a video'],
         ['Non-free software cover', 'Cover of a software product'],
         ['Non-free product cover', 'Cover of some commercial product'],
         ['Non-free title-card', 'Title screen of a TV programme'],
         ['Non-free movie poster', 'Movie poster'],
         ['Non-free poster', 'Official poster of an event'],
         ['Non-free fair use in', 'something else (please describe in
description field on top)']
         ],
      'NFExcerptLicense' :
         [
         ['', 'please select one...'],
         ['Non-free television screenshot', 'Television screenshot'],
         ['Non-free film screenshot', 'Movie screenshot'],
         ['Non-free game screenshot', 'Game screenshot'],
         ['Non-free video screenshot', 'Video screenshot'],
         ['Non-free music video screenshot', 'Music video screenshot'],
         ['Non-free software screenshot', 'Software screenshot'],
         ['Non-free web screenshot', 'Website screenshot'],
```

```
         ['Non-free speech', 'Audio excerpt from a speech'],
         ['Non-free audio sample', 'Sound sample of an audio recording'],
         ['Non-free video sample', 'Sample extract from a video'],
         ['Non-free sheet music', 'Sheet music representing a musical
piece'],
         ['Non-free comic', 'Panel from a comic, graphic novel, manga etc.'],
         ['Non-free computer icon', 'Computer icon'],
         ['Non-free newspaper image', 'Page from a newspaper'],
         ['Non-free fair use in', 'something else (please describe in
description field on top)']
         ],
      'NFLogoLicense' :
         [
         ['Non-free logo', 'Logo of a company, organization etc.'],
         ['Non-free seal', 'Official seal, coat of arms etc'],
         ['Non-free symbol', 'Other official symbol']
         ],
      'NFMiscLicense' :
         [
         ['Non-free fair use in', 'something else (please describe in
description field on top)'],
         ['Non-free historic image', 'Historic photograph'],
         ['Non-free 2D art', '2-dimensional artwork (painting, drawing
etc.)'],
         ['Non-free currency', 'Depiction of currency (banknotes, coins
etc.)'],
         ['Non-free architectural work', 'Architectural work'],
         ['Non-free 3D art', 'Other 3-dimensional creative work (sculpture
etc.)'],
         ['Non-free book cover', 'Cover page of a book'],
         ['Non-free album cover', 'Cover of a sound recording(album, single,
song, CD)'],
         ['Non-free game cover', 'Cover of a video/computer game'],
         ['Non-free magazine cover', 'Cover page of a magazine'],
         ['Non-free video cover', 'Cover of a video'],
         ['Non-free software cover', 'Cover of a software product'],
         ['Non-free product cover', 'Cover of some commercial product'],
         ['Non-free title-card', 'Title screen of a TV programme'],
         ['Non-free movie poster', 'Movie poster'],
         ['Non-free poster', 'Official poster of an event'],
         ['Non-free television screenshot', 'Television screenshot'],
         ['Non-free film screenshot', 'Movie screenshot'],
         ['Non-free game screenshot', 'Game screenshot'],
         ['Non-free video screenshot', 'Video screenshot'],
         ['Non-free music video screenshot', 'Music video screenshot'],
         ['Non-free software screenshot', 'Software screenshot'],
         ['Non-free web screenshot', 'Website screenshot'],
         ['Non-free speech', 'Audio excerpt from a speech'],
         ['Non-free audio sample', 'Sound sample of an audio recording'],
         ['Non-free video sample', 'Sample extract from a video'],
         ['Non-free sheet music', 'Sheet music representing a musical
```

```
piece'],
         ['Non-free comic', 'Panel from a comic, graphic novel, manga etc.'],
         ['Non-free computer icon', 'Computer icon'],
         ['Non-free newspaper image', 'Page from a newspaper'],
         ['Non-free logo', 'Logo of a company, organization etc.'],
         ['Non-free seal', 'Official seal, coat of arms etc'],
         ['Non-free symbol', 'Other official symbol'],
         ['Non-free sports uniform', 'Sports uniform'],
         ['Non-free stamp', 'Stamp']
         ],
      'NFExtraLicense' :
         [
         ['', 'none'],
         ['Crown copyright and other governmental sources'],
         ['Non-free Crown copyright', 'UK Crown Copyright'],
         ['Non-free New Zealand Crown Copyright', 'NZ Crown Copyright'],
         ['Non-free Canadian Crown Copyright', 'Canadian Crown Copyright'],
         ['Non-free AUSPIC', 'AUSPIC (Australian Parliament image database'],
         ['Non-free Philippines government', 'Philippines government'],
         ['Non-free Finnish Defence Forces', 'Finnish Defence Forces'],
         [],
         ['Other individual sources'],
         ['Non-free Denver Public Library', 'Denver Public Library'],
         ['Non-free ESA media', 'ESA (European Space Agency)'],
         [],
         ['Possibly public domain in other countries'],
         ['Non-free Old-50', 'Author died more than 50 years ago.'],
         ['Non-free Old-70', 'Author died more than 70 years ago.'],
         [],
         ['Some permissions granted, but not completely free'],
         ['Non-free promotional', 'From promotional press kit'],
         ['Non-free with NC', 'Permission granted, but only for educational
and/or non-commercial purposes'],
         ['Non-free with ND', 'Permission granted, but no derivative works
allowed'],
         ['Non-free with permission', 'Permission granted, but only for
Wikipedia'],
         []
         ]
   };
   for (var group in licenseLists) {
      fuwMakeSelection(group, licenseLists[group]);
   }


   this.knownCommonsLicenses = {
      'self|GFDL|cc-by-sa-all|migration=redundant' : 1,
      'self|Cc-zero' : 1,
      'PD-self' : 1,
      'self|GFDL|cc-by-sa-3.0|migration=redundant' : 1,
      'self|GFDL|cc-by-3.0|migration=redundant' : 1,
```

```
      'self|cc-by-sa-3.0' : 1,
      'cc-by-sa-3.0' : 1,
      'cc-by-sa-2.5' : 1,
      'cc-by-3.0' : 1,
      'cc-by-2.5' : 1,
      'FAL' : 1,
      'PD-old-100' : 1,
      'PD-old' : 1,
      'PD-Art' : 1,
      'PD-US' : 1,
      'PD-USGov' : 1,
      'PD-USGov-NASA' : 1,
      'PD-USGov-Military-Navy' : 1,
      'PD-ineligible' : 1,
      'Attribution' : 1,
      'Copyrighted free use' : 1
   };


   // textfields that don't react directly
   // to user input and are used only for assembling stuff:
   if (fuwTesting) {
      fuwMakeTextfield('SandboxSummary', function(){void(0);});
      fuwMakeTextarea('SandboxText', function(){void(0);});
      fuwGet('SandboxSummary').disabled="disabled";
      fuwGet('SandboxText').disabled="disabled";
      fuwGet('SandboxText').rows = 12;
   }

   // set links to "_blank" target, so we don't accidentally leave the page,
   // because on some browsers that would destroy all the input the user has
already entered
   $('.fuwOutLink a').each(function() {
      this.target = '_blank';
   });

   // make main area visible
   fuwSetVisible('UploadScriptArea', true);

}
// ====================================
// end of fuwGlobal constructor function
// ====================================



function fuwRadioClick(e) {
   var ev = e || event;
   var src = ev.target || ev.srcElement;
   //alert('onclick event from ' + src + ' (' + src.value + ')');
   fuwUpdateOptions();
```

```
   return true;
}

/*
* ============================================================
* function fuwUpdateOptions
* ============================================================
* This is the onchange event handler for most of the input
* elements in the main form. It changes visibility and disabled
* status for the various sections of the input form in response
* to which options are chosen.
*/
function fuwUpdateOptions() {

   var fuw = window.fuw;
   var warn = fuw.warn;
   var opts = fuw.opts = { };
   opts.InputFilename = $('#TargetForm input#file').val();

   var widgets = fuw.ScriptForm.elements;
   for (i = 0; i < widgets.length; i++) {
      var w = widgets[i];
      if (w.type == "radio") {
         var nm = w.name;
         var id = w.id;
         var vl = w.checked && !w.disabled && fuwIsVisible(w);
         opts[id] = vl;
         if (vl) opts[nm] = id;
      }
      else {
         var id = w.id;
         var active = !w.disabled && fuwIsVisible(w);
         if (active) {
            var value = ((type == 'checkbox') ? w.checked : w.value);
            opts[id] = value;
         }
      }
   };
   opts.MainOption = opts.FreeOptions || opts.NonFreeOptions;

   // some parts of the input form are re-used across sections
   // and must be moved into the currently active input section:

   // minimality section is shared between all NF sections
   fuwMove('NFMinimalitySection', 'detailsNFSubject', (opts.OptionNFSubject))
||
   fuwMove('NFMinimalitySection', 'detailsNF3D', (opts.OptionNF3D)) ||
   fuwMove('NFMinimalitySection', 'detailsNFExcerpt', (opts.OptionNFExcerpt))
||
   fuwMove('NFMinimalitySection', 'detailsNFCover', (opts.OptionNFCover)) ||
   fuwMove('NFMinimalitySection', 'detailsNFLogo', (opts.OptionNFLogo)) ||
```

```
   fuwMove('NFMinimalitySection', 'detailsNFPortrait',
(opts.OptionNFPortrait)) ||
   fuwMove('NFMinimalitySection', 'detailsNFMisc', true);

   // AnyOtherInfo section is shared between all
   fuwMove('AnyOtherInfo', 'detailsOwnWork', opts.OptionOwnWork) ||
   fuwMove('AnyOtherInfo', 'detailsThirdParty', opts.OptionThirdParty) ||
   fuwMove('AnyOtherInfo', 'detailsFreeWebsite', opts.OptionFreeWebsite) ||
   fuwMove('AnyOtherInfo', 'detailsPDOld', opts.OptionPDOld) ||
   fuwMove('AnyOtherInfo', 'detailsPDOther', opts.OptionPDOther) ||
   fuwMove('AnyOtherInfo', 'detailsNFSubject', opts.OptionNFSubject) ||
   fuwMove('AnyOtherInfo', 'detailsNF3D', opts.OptionNF3D) ||
   fuwMove('AnyOtherInfo', 'detailsNFExcerpt', opts.OptionNFExcerpt) ||
   fuwMove('AnyOtherInfo', 'detailsNFCover', opts.OptionNFCover) ||
   fuwMove('AnyOtherInfo', 'detailsNFLogo', opts.OptionNFLogo) ||
   fuwMove('AnyOtherInfo', 'detailsNFPortrait', opts.OptionNFPortrait) ||
   fuwMove('AnyOtherInfo', 'detailsNFMisc', opts.OptionNFMisc);

   // author input field is shared between all sections except "Own Work".
   // (will serve for the immediate/photographic author, in those cases where
there
   // are two author fields)
   fuwMove('Author', 'placeholderFreeWebsiteAuthor',
(opts.OptionFreeWebsite)) ||
   fuwMove('Author', 'placeholderPDOldAuthor', (opts.OptionPDOld)) ||
   fuwMove('Author', 'placeholderPDOtherAuthor', (opts.OptionPDOther)) ||
   fuwMove('Author', 'placeholderNFSubjectAuthor', (opts.OptionNFSubject)) ||
   fuwMove('Author', 'placeholderNF3DAuthor', (opts.OptionNF3D)) ||
   fuwMove('Author', 'placeholderNFExcerptAuthor', (opts.OptionNFExcerpt)) ||
   fuwMove('Author', 'placeholderNFCoverAuthor', (opts.OptionNFCover)) ||
   fuwMove('Author', 'placeholderNFPortraitAuthor', (opts.OptionNFPortrait))
||
   fuwMove('Author', 'placeholderNFMiscAuthor', (opts.OptionNFMisc)) ||
   fuwMove('Author', 'placeholderAuthor', true);

   // source input field is shared between all sections except "Own Work".
   // (will serve for immediate/web source, in those cases where there are
two
   // source fields involved)
   fuwMove('Source', 'placeholderFreeWebsiteSource',
(opts.OptionFreeWebsite)) ||
   fuwMove('Source', 'placeholderPDOldSource', (opts.OptionPDOld)) ||
   fuwMove('Source', 'placeholderPDOtherSource', (opts.OptionPDOther)) ||
   fuwMove('Source', 'placeholderNFSubjectSource', (opts.OptionNFSubject)) ||
   fuwMove('Source', 'placeholderNF3DSource', (opts.OptionNF3D)) ||
   fuwMove('Source', 'placeholderNFExcerptSource', (opts.OptionNFExcerpt)) ||
   fuwMove('Source', 'placeholderNFCoverSource', (opts.OptionNFCover)) ||
   fuwMove('Source', 'placeholderNFLogoSource', (opts.OptionNFLogo)) ||
   fuwMove('Source', 'placeholderNFPortraitSource', (opts.OptionNFPortrait))
||
   fuwMove('Source', 'placeholderNFMiscSource', (opts.OptionNFMisc)) ||
```

```
   fuwMove('Source', 'placeholderSource', true);

   // date input field is shared between all sections except "Logo", which
doesn't need it.
   // will serve for derived/photographic date in the case of 3D items
   fuwMove('Date', 'placeholderFreeWebsiteDate', (opts.OptionFreeWebsite)) ||
   fuwMove('Date', 'placeholderThirdPartyDate', (opts.OptionThirdParty)) ||
   fuwMove('Date', 'placeholderPDOldDate', (opts.OptionPDOld)) ||
   fuwMove('Date', 'placeholderPDOtherDate', (opts.OptionPDOther)) ||
   fuwMove('Date', 'placeholderNFSubjectDate', (opts.OptionNFSubject)) ||
   fuwMove('Date', 'placeholderNF3DDate', (opts.OptionNF3D)) ||
   fuwMove('Date', 'placeholderNFExcerptDate', (opts.OptionNFExcerpt)) ||
   fuwMove('Date', 'placeholderNFCoverDate', (opts.OptionNFCover)) ||
   fuwMove('Date', 'placeholderNFPortraitDate', (opts.OptionNFPortrait)) ||
   fuwMove('Date', 'placeholderNFMiscDate', (opts.OptionNFMisc)) ||
   fuwMove('Date', 'placeholderDate', true);

   // permission field is shared between ThirdParty and FreeWebsite sections
   fuwMove('Permission', 'placeholderFreeWebsitePermission',
(opts.OptionFreeWebsite)) ||
   fuwMove('Permission', 'placeholderPermission', true);

   // publication field is shared between PDOld, NFPortrait and NFMisc
   fuwMove('Publication', 'placeholderNFPortraitPublication',
(opts.OptionNFPortrait)) ||
   fuwMove('Publication', 'placeholderNFMiscPublication',
(opts.OptionNFMisc)) ||
   fuwMove('Publication', 'placeholderPublication', true);

   // Purpose, Commercial, Replaceable and ReplaceableText FUR fields are
shared
   // between some but not all of the non-free sections
   fuwMove('NFPurpose', 'placeholderNFExcerptPurpose',
(opts.OptionNFExcerpt)) ||
   fuwMove('NFPurpose', 'placeholderNFPurpose');
   fuwMove('NFCommercial', 'placeholderNFPortraitCommercial',
(opts.OptionNFPortrait)) ||
   fuwMove('NFCommercial', 'placeholderNFCommercial');
   fuwMove('NFReplaceable', 'placeholderNFPortraitReplaceable',
(opts.OptionNFPortrait)) ||
   fuwMove('NFReplaceable', 'placeholderNFReplaceable');
   fuwMove('NFReplaceableText', 'placeholderNFExcerptReplaceable',
(opts.OptionNFExcerpt)) ||
   fuwMove('NFReplaceableText', 'placeholderNFReplaceableText', true);

   // submit button goes to Step1 if user has chosen a plain overwrite of an
existing file,
   // and to the active section of Step3 if otherwise
   fuwMove('fuwSubmit', 'UploadScriptStep1', (warn.ImageExists &&
opts.OverwriteSame)) ||
   fuwMove('fuwSubmit', 'detailsOwnWork', opts.OptionOwnWork) ||
```

```
   fuwMove('fuwSubmit', 'detailsThirdParty', opts.OptionThirdParty) ||
   fuwMove('fuwSubmit', 'detailsFreeWebsite', opts.OptionFreeWebsite) ||
   fuwMove('fuwSubmit', 'detailsPDOld', opts.OptionPDOld) ||
   fuwMove('fuwSubmit', 'detailsPDOther', opts.OptionPDOther) ||
   fuwMove('fuwSubmit', 'detailsNFSubject', opts.OptionNFSubject) ||
   fuwMove('fuwSubmit', 'detailsNF3D', opts.OptionNF3D) ||
   fuwMove('fuwSubmit', 'detailsNFExcerpt', opts.OptionNFExcerpt) ||
   fuwMove('fuwSubmit', 'detailsNFCover', opts.OptionNFCover) ||
   fuwMove('fuwSubmit', 'detailsNFLogo', opts.OptionNFLogo) ||
   fuwMove('fuwSubmit', 'detailsNFPortrait', opts.OptionNFPortrait) ||
   fuwMove('fuwSubmit', 'fuwSubmitHost', true);


   // Show and hide warnings:

   // filename-related warnings:
   fuwSetVisible('warningIllegalChars', warn.IllegalChars);
   fuwSetVisible('warningBadFilename',  warn.BadFilename);
   fuwSetVisible('warningImageOnCommons', warn.ImageOnCommons);
   fuwSetVisible('warningImageExists', warn.ImageExists);
   fuwMove('warningImageThumb', 'warningImageOnCommons', warn.ImageOnCommons,
true) ||
   fuwMove('warningImageThumb', 'warningImageExists', true, true);


   // notices related to the top-level options:
   fuwSetVisible('warningWhyNotCommons', opts.OptionFree);
   fuwSetVisible('warningNF', opts.OptionNonFree);
   fuwSetVisible('warningNoGood', opts.OptionNoGood);

   // warnings related to non-free "used in" article
   fuwSetVisible('warningNFArticleNotFound', warn.NFArticleNotFound);
   fuwSetVisible('warningNFArticleNotMainspace', warn.NFArticleNotMainspace);
   fuwSetVisible('warningUserspaceDraft', warn.UserspaceDraft);
   fuwSetVisible('warningNFArticleDab', warn.NFArticleDab);
   fuwSetVisible('NFArticleOK', warn.NFArticleOK);

   // warnings depending on user status:
   if (fuw.userStatus.match(/problem|newbie|notAutoconfirmed/)) {
      fuwSetVisible('warningFreeWebsite', opts.OptionFreeWebsite);
      fuwSetVisible('warningOwnWork', opts.OptionOwnWork);
      fuwSetVisible('warningPDOther', opts.OptionPDOther);
      fuwSetVisible('warningNFSubject', opts.OptionNFSubject);
   }

   // hide main sections in case of intended plain overwrite:
   fuwSetVisible('UploadScriptStep2', !(warn.ImageExists &&
opts.OverwriteSame));
   fuwSetVisible('UploadScriptStep3', !(warn.ImageExists &&
opts.OverwriteSame));
```

```
   // show/hide top-level options
   fuwSetVisible('detailsFreeStatus', opts.OptionFree);
   fuwSetVisible('sendToCommons', opts.OptionFree);

   // show/hide details sections
   fuwSetVisible('detailsNFArticle', opts.OptionNonFree);
   fuwSetVisible('detailsNFWorkType', opts.OptionNonFree);
   fuwSetVisible('detailsOwnWork', opts.OptionOwnWork);
   fuwSetVisible('detailsThirdParty', opts.OptionThirdParty);
   fuwSetVisible('detailsFreeWebsite', opts.OptionFreeWebsite);
   fuwSetVisible('detailsPDOld', opts.OptionPDOld);
   fuwSetVisible('detailsPDOther', opts.OptionPDOther);
   fuwSetVisible('detailsNFSubject', opts.OptionNFSubject);
   fuwSetVisible('detailsNF3D', opts.OptionNF3D);
   fuwSetVisible('detailsNFExcerpt', opts.OptionNFExcerpt);
   fuwSetVisible('detailsNFCover', opts.OptionNFCover);
   fuwSetVisible('detailsNFLogo', opts.OptionNFLogo);
   fuwSetVisible('detailsNFPortrait', opts.OptionNFPortrait);
   fuwSetVisible('detailsNFMisc', opts.OptionNFMisc);

   fuwSetVisible('EditSummaryDiv', opts.OverwriteSame ||
opts.OverwriteDifferent);

   // set enabled/disabled
   // It might be useful to adapt this to be more liberal about
   // the order of input, at least for experienced users.

   //fuwSetEnabled('Artist3D', opts.PD3D);
   //fuwSetEnabled('Country3D', opts.FOP3D);
   fuwSetEnabled('ThirdPartyEvidenceLink',
opts.ThirdPartyEvidenceOptionLink);
   fuwSetEnabled('ThirdPartyOTRSTicket', opts.ThirdPartyEvidenceOptionOTRS);
   fuwSetEnabled('NFSubjectPurpose', opts.NFSubjectCheckDiscussed);
   fuwSetEnabled('NF3DPurpose', opts.NF3DCheckDiscussed);
   fuwSetEnabled('NF3DPermission', opts.NF3DOptionFree);
   fuwSetEnabled('USGovLicense', opts.PDOtherUSGov);
   fuwSetEnabled('PDOfficialPermission', opts.PDOtherOfficial);
   fuwSetEnabled('IneligibleLicense', opts.PDOtherSimple);
   fuwSetEnabled('PDOtherPermission', opts.PDOtherOther);
   fuwSetEnabled('AnyOther', true);

   // need to re-collect the remaining (non-radiobutton) input into the opts
object again,
   // preparing for validation:
   for (i = 0; i < widgets.length; i++) {
      var w = widgets[i];
      var type = w.type;

      if (type != "radio") {
         var id = w.id;
         var active = !w.disabled && fuwIsVisible(w);
```

```
         if (active) {
            var value = ((type == 'checkbox') ? w.checked : w.value);
            opts[id] = value;
         }
      }
   };

   // final step of validation: check if input is sufficient for
   // setting the submit buttons active
   var valid = fuw.validateInput();
   var validForCommons = valid && opts.OptionFree && !(opts.OverwriteSame ||
opts.OverwriteDifferent)
         && !opts.ThirdPartyEvidenceOptionNone;
   fuwSetVisible('sendToCommons', opts.OptionFree);
   fuwSetEnabled('CommonsButton', validForCommons);
   fuwGet('fuwSubmitText').innerHTML = opts.OptionFree ?
         ("<b>No</b>, I want to upload this file here on this wiki only.<br/>"
+
          "<small>This way it can be used only on the English Wikipedia.
However, somebody " +
          "else might still decide to copy it to Commons or use it elsewhere
later. If you " +
          "do not want your file to be copied to Commons and deleted locally,
consider adding " +
          "{{tl|Keep local}}.</small>") :
         "Upload this file.";
   fuwGet('SubmitButton').value = validForCommons ? "Upload locally" :
"Upload";
   fuwSetEnabled('EditSummary', true);
   fuwSetEnabled('SubmitButton', valid && (fuw.userStatus !=
'notAutoconfirmed'));
   if (fuwTesting) {
      fuwSetEnabled('SandboxButton', valid);
   }

   // if we're in testing mode, update the Sandbox display fields
   // after each input change. In normal mode, collectInput() will
   // only be needed on submit.
   if (fuwTesting) {
      fuw.collectInput();
      fuw.formatOutput(false);
      fuwSetVisible('placeholderTestForm', true);
   }
}

// ============================================================
// methods of the global fuw object
// ============================================================

// ============================================================
// report validation status of filename information
```

```
// ============================================================
// This is called from within fuw.validateInput(), i.e. every
// time anything in the whole form is changed. It only reports
// results that have previously been cached in the opts and warn
// objects. The actual checking is done in the event handler
// of the file input boxes.
fuwGlobal.prototype.hasValidFilename = function() {
   var opts = this.opts;
   var warn = this.warn;
   var valid =
      opts.InputName &&
      opts.InputFilename &&
      !warn.BadFilename &&
      !warn.ImageOnCommons &&
      // if image exists on enwiki, accept only if user has confirmed
overwrite:
      !(warn.ImageExists && !(opts.OverwriteSame ||
opts.OverwriteDifferent));
   //alert("HasValidFilename: " + valid);
   return valid;
};

// ============================================================
// validation status for common input elements for all free
// options
// ============================================================
fuwGlobal.prototype.hasValidCommonFreeInput = function() {
   var opts = this.opts;
   var warn = this.warn;
   var valid = opts.InputDesc;
   //alert("HasValidCommonFreeInput: " + valid);
   return valid;
};
// ============================================================
// validation status for common input elements for all non-free
// options
// ============================================================
fuwGlobal.prototype.hasValidCommonNFInput = function() {
   var opts = this.opts;
   var warn = this.warn;
   var valid =
      opts.OptionNonFree &&
      opts.InputDesc &&
      opts.NFArticle &&
      opts.Source &&
      opts.NFMinimality &&
      !warn.NFArticleNotFound &&
      !warn.NFArticleNotMainspace &&
      !warn.NFArticleDab;
   //alert("hasValidCommonNFInput: " + valid);
   return valid;
```

```
};
// ============================================================
// Main validation routine. Modify this to tweak which fields
// are to be considered obligatory for each case group
// ============================================================
fuwGlobal.prototype.validateInput = function() {
   var opts = this.opts;
   var warn = this.warn;
   var valid = (
      this.hasValidFilename()
      &&
      (! (opts.OverwriteDifferent && ! opts.EditSummary))
      &&
      (
       ( // overwriting is okay if there is an edit summary
        opts.OverwriteSame && opts.EditSummary
       )
       ||
       ( // free options
         this.hasValidCommonFreeInput() &&
         (
          (opts.OptionOwnWork &&
           opts.Date &&
           opts.OwnWorkLicense)
          ||
          (opts.OptionThirdParty &&
           opts.Author &&
           opts.Source &&
           opts.Permission &&
           (opts.ThirdPartyOtherLicense || opts.ThirdPartyLicense) &&
           ((opts.ThirdPartyEvidenceOptionLink && opts.ThirdPartyEvidenceLink)
||
            opts.ThirdPartyEvidenceOptionOTRS ||
            opts.ThirdPartyEvidenceOptionOTRSForthcoming ||
            opts.ThirdPartyEvidenceOptionNone))
          ||
          (opts.OptionFreeWebsite &&
           opts.Author &&
           opts.Source &&
           (opts.FreeWebsiteOtherLicense || opts.FreeWebsiteLicense) &&
           opts.Permission)
          ||
          (opts.OptionPDOld &&
           opts.Author &&
           opts.PDOldAuthorLifetime &&
           opts.Publication &&
           opts.Date &&
           opts.Source &&
           opts.PDOldOptions &&
           (! (opts.PDOldOther && ! opts.PDOldPermission))
          ||
```

```
   (opts.OptionPDOther &&
    opts.Author &&
    opts.Source &&
    ((opts.PDOtherUSGov && opts.USGovLicense) ||
     (opts.PDOtherOfficial && opts.PDOfficialPermission) ||
     (opts.PDOtherSimple && opts.IneligibleLicense) ||
     (opts.PDOtherOther && opts.PDOtherPermission)))
  )
 )
) // end of free options
||
( // non-free options
  this.hasValidCommonNFInput() &&
  (
   (opts.OptionNFSubject &&
    opts.NFSubjectLicense &&
    opts.Author &&
    (opts.NFSubjectCheckDedicated ||
    (opts.NFSubjectCheckDiscussed && opts.NFSubjectPurpose)))
   ||
   (opts.OptionNF3D &&
    opts.NF3DLicense &&
    opts.NF3DCreator &&
    opts.Author &&
    (opts.NF3DOptionSame ||
    (opts.NF3DOptionFree || opts.NF3DPermission)) &&
    (opts.NF3DCheckDedicated ||
     (opts.NF3DCheckDiscussed && opts.NF3DPurpose)))
   ||
   (opts.OptionNFExcerpt &&
    opts.NFExcerptLicense &&
    opts.Author &&
    opts.NFPurpose)
   ||
   (opts.OptionNFCover &&
    opts.NFCoverLicense &&
    opts.Author &&
    opts.NFCoverCheckDedicated
   )
   ||
   (opts.OptionNFLogo &&
    opts.NFLogoLicense &&
    opts.NFLogoCheckDedicated
   )
   ||
   (opts.OptionNFPortrait &&
    opts.Publication &&
    opts.NFPortraitDeceased &&
    opts.Author &&
    opts.NFPortraitCheckDedicated &&
    opts.NFReplaceable &&
```

```
            opts.NFCommercial)
           ||
           (opts.OptionNFMisc &&
            opts.NFMiscLicense &&
            opts.Author &&
            opts.Publication &&
            opts.NFPurpose &&
            opts.NFReplaceableText &&
            opts.NFReplaceable &&
            opts.NFCommercial)
         )
       ) // end of non-free options
      )
   );
   return valid;
};

// ============================================================
// return which template name will be used as the main
// description template
// ============================================================
fuwGlobal.prototype.getDescriptionTemplateName = function() {
   // standard "Information" template for free files:
   if (this.opts.OptionFree) return "Information";
   // experimental new version of fair-use rationale template,
   // designed to fit the fields used in the wizard
   else if (this.opts.OptionNonFree) return "Non-free use rationale 2";
   return undefined;
};

// ============================================================
// get the license tag code from the appropriate input element
// ============================================================

fuwGlobal.prototype.getStandardLicense = function() {
   var opts = this.opts;

}

fuwGlobal.prototype.getLicense = function() {
   var opts = this.opts;
      // ThirdParty and FreeWebsite have alternative input fields
      // for manual entry of other licenses:
   var license = {};
   if (opts.PDOtherOther || opts.PDOldOther) {
      license.special = opts.PDOtherOther ? opts.PDOtherPermission :
opts.PDOldPermission;
      if (! (license.special.match(/^\s*\{\{.+\}\}\s*$/))) {
         license.special = '{{PD-because|' + license.special + '}}';
      }
   }
```

```
   else {
      license.special =
         opts.ThirdPartyOtherLicense ||
         opts.FreeWebsiteOtherLicense ||
         (opts.PDOtherOfficial ? ('{{PD-because|official item legally exempt
from copyright in its country of origin}}') : null) ||
         (opts.OptionNFPortrait ? ('{{Non-free biog-pic|' + opts.NFArticle +
'}}') : null);
   }
   if (! license.special) {
      // standard, non-parametrized tag licenses from dropdownbox.
      var simpleLicense = (opts.OptionOwnWork ? opts.OwnWorkLicense : null)
||
          (opts.OptionThirdParty ? opts.ThirdPartyLicense : null) ||
          (opts.OptionFreeWebsite ? opts.FreeWebsiteLicense : null) ||
          (opts.OptionNFSubject ? opts.NFSubjectLicense : null) ||
          (opts.OptionNF3D ? opts.NF3DLicense : null) ||
          (opts.OptionNFExcerpt ? opts.NFExcerptLicense : null) ||
          (opts.OptionNFCover ? opts.NFCoverLicense : null) ||
          (opts.OptionNFLogo ? opts.NFLogoLicense : null) ||
          (opts.OptionNFMisc ? opts.NFMiscLicense : null) ||
          (opts.PDOtherUSGov ? opts.USGovLicense : null) ||
          (opts.PDOtherSimple ? opts.IneligibleLicense : null) ||
          (opts.PDUS1923 ? 'PD-US-1923' : null) ||
          (opts.PDURAA   ? 'PD-URAA' : null) ||
          (opts.PDFormality ? 'PD-US' : null);

       // "PD-author" needs parameter, at least on Commons
       if (simpleLicense == 'PD-author') {
          license.special = '{{PD-author|' + opts.Author + '}}';
       }
       else if (this.knownCommonsLicenses[simpleLicense]) {
       // make sure we send only those licenses as "standard" licenses
       // that exist in the Commons license dropdown box
          license.standard = simpleLicense;
       }
       else {
          license.special = '{{' + simpleLicense + '}}';
       }
   }
   return license;
};

function fuwSubst(template) {
   return '{{subst:' + template + '}}';
}

// ================================================================
// Produce code for local tracking categories
// ================================================================
fuwGlobal.prototype.getTrackingCategory = function() {
```

```
   var opts = this.opts;
   var cat = "";
   if (opts.OptionFreeWebsite) { cat = "Files from freely licensed external
sources"; }
   else if (opts.OptionThirdParty) { cat = "Files licensed by third parties";
}
   else if (opts.PDOtherOther || opts.PDOldOther) { cat = "Files with
non-standard public domain statements"; }
   else if (opts.OptionNFSubject || opts.OptionNF3D) { cat = "Non-free files
uploaded as object of commentary"; }
   if (cat) {
      cat = "\n{{Category ordered by date|" + cat + "|" +
      fuwSubst("CURRENTYEAR") + "|" + fuwSubst("CURRENTMONTH") + "|" +
fuwSubst("CURRENTDAY2") + "}}";
   }
   return cat;
};

// ================================================================
// Get or create an edit summary for the upload
// ================================================================
// Note: if we work with the api.php interface, we can have separate
// data for the edit summary and the description page, which is far
// better than the way the index.php interface does it.
// TO DO: need to actually define an input element for a manually
// entered edit summary. Must be obligatory when overwriting files.
// In other cases we'll use an automatic edit summary.
// ================================================================
fuwGlobal.prototype.getEditSummary = function() {
   var opts = this.opts;
   return (
      (opts.EditSummary ? (opts.EditSummary + ' ([[' +
mw.config.get('wgPageName') + '|File Upload Wizard]])') : null)||
      ("Uploading " +
       (
        (opts.OptionOwnWork ? 'a self-made file ' : false) ||
        (opts.OptionThirdParty ? 'a free file from somebody else ' : false)
||
        (opts.OptionFreeWebsite ? 'a file from a free published source ' :
false) ||
        (opts.OptionPDOld       ? 'an old public-domain work ' : false) ||
        (opts.OptionPDOther     ? 'a public-domain item ' : false) ||
        (opts.OptionNFSubject   ? 'a non-free work, as object of commentary '
: false) ||
        (opts.OptionNF3D        ? 'a depiction of a non-free 3D artwork ' :
false) ||
        (opts.OptionNFExcerpt   ? 'an excerpt from a non-free work ' : false)
||
        (opts.OptionNFCover     ? 'a piece of non-free cover art ' : false)
||
        (opts.OptionNFLogo      ? 'a non-free logo ' : false) ||
```

```
        (opts.OptionNFPortrait    ? 'a non-free historic portrait ' : false)
||
        (opts.OptionNFMisc      ? 'a non-free file ' : "")
       )
       +
       ("using [[" + mw.config.get('wgPageName') + "|File Upload Wizard]]")
      ));
};


function fuwPackInfo(text, forCommons) {
   if (forCommons) {
      // reformat wikilinks embedded in description fields to adapt them for
Commons
      text = text.replace(/\[\[([^\]]+)\]\]/g,
         function(str, p1, offset, s) {

            // mark File links as local
            if (p1.match(/^:(File|Image):/)) {
               return "[[:en" + p1 + "]]";
            }
            // leave prefixed links unchanged:
            else if (p1.match(/^:[\w\-]+:/)) {
               return str;
            }
            // if the link is piped, add a prefix only
            else if (p1.match(/.+\|/)) {
               return "[[:en:" + p1 + "]]";
            }
            // introduce a pipe
            else {
               return "[[:en:" + p1 + "|" + p1 + "]]";
            }
         }
      );
      return "{{en|" + text + "}}";
   } else return text;
}

// ================================================================
// This is the main method called by the event handler for the
// (experimental) submit button. Its main task is to collect the
// input into a single string of wikitext for the description page.
// ================================================================
fuwGlobal.prototype.collectInput = function() {
   var opts = this.opts;

   // object representing template fields for filling in
   // the description template. Pre-loaded with some
   // standard settings:
   var descFields = this.descFields = {
```

```
      'Description' : opts.InputDesc,
      'Author'      : opts.Author,
      'Date'        : opts.Date,
      'Source'      : opts.Source
   };
   // "other information" (outside the template)
   this.otherInfo = null;

   if (opts.OptionNonFree) {
      descFields.Article = opts.NFArticle;
   }

   // add/modify option-specific fields:
   switch (opts.MainOption) {
      case 'OptionOwnWork':

         // use standard "source" field for optional "how created?" and
         // "previously published" input fields.
         descFields.Source = fuwAppendLines([
            (opts.OwnWorkCreation || "{{own}}"),
            "<br/>\n",
            fuwSurroundString("'''Previously published:''' ",
opts.OwnWorkPublication)]);
         var username = mw.user.name();
         descFields.Author = '[[User:' + username + '|' + username + ']]';
         break;

      case 'OptionThirdParty':

         // use standard "permission" field for a compilation of the
         // "permission" input field and the various "evidence" options
         var evidence = (
            opts.ThirdPartyEvidenceOptionLink ?
               ("The license statement can be found online at: " +
opts.ThirdPartyEvidenceLink) :
               (opts.ThirdPartyEvidenceOptionOTRS ?
               ("The license agreement has been forwarded to OTRS." +
                  fuwSurroundString(" Ticket: ", opts.ThirdPartyOTRSTicket) +
"{{OTRS pending}}") :
               (opts.ThirdPartyEvidenceOptionOTRSForthcoming ?
               "The license agreement will be forwarded to OTRS shortly.
{{OTRS pending}}" :
               (opts.ThirdPartyEvidenceOptionNone ?
               "Will be provided on request." : null))));
         descFields.Permission = fuwAppendLines([
            opts.ThirdPartyPermission,
            "<br/>\n",
            fuwSurroundString("'''Evidence:''' ", evidence)]);
         break;

      case 'OptionFreeWebsite':
```

```
         descFields.Permission = opts.Permission;
         break;

      case 'OptionPDOld':
         // add "lifetime" input to "author" field
         descFields.Author = fuwAppendLines([
            opts.Author,
            "<br/>\n",
            fuwSurroundString("(Life time: ", opts.PDOldAuthorLifetime, ")")
         ]);

         // combine original and direct source into standard "source" field:
         descFields.Source = fuwAppendLines([
            fuwSurroundString("'''Original publication''': ",
opts.Publication),
            "<br/>\n",
            fuwSurroundString("'''Immediate source''': ", opts.Source)
         ]);

         // no standard tag available for "lack-of-registration" PD-US. Need
         // to put this into the "permission" field
         if (opts.PDFormality)
            descFields.Permission =
               "Copyright expired because the work was published without a
copyright " +
               "notice and/or without the necessary copyright registration.";

         // add optional "explanation" input to "permission" field
         if (opts.PDOldPermission) {
            descFields.Permission = fuwAppendLines([
               descFields.Permission,
               "\n\n",
               opts.PDOldPermission
            ]);
         }
         break;

      case 'OptionPDOther':
         // Need "permission" field in case of "official item" option
         if (opts.PDOtherOfficial)
            descFields.Permission = opts.PDOfficialPermission;
         break;

      case 'OptionNFSubject':
         // most FUR elements can be automatically provided:
         descFields.Purpose = (
            opts.NFSubjectCheckDedicated ?
             ("For visual identification of the object of the article. " +
              "The article as a whole is dedicated specifically to a
discussion of this work.") :
            (opts.NFSubjectCheckDiscussed ?
```

```
             ("To support encyclopedic discussion of this work in this
article. " +
              "The illustration is specifically needed to support the
following point(s): " +
              "<br/>\n" + opts.NFSubjectPurpose) : null)
         );
         // I hate FURs filled with trivial/predictable/redundant verbiage,
         // so we'll just cut it short. And don't anybody dare complain that
         // that's not a valid FUR.
         descFields.Replaceability = "n.a.";
         descFields.Commercial     = "n.a.";
         break;

      case 'OptionNF3D':
         // complex case: we need to assemble attribution and FUR both for
the
         // original 3D work and for the photographic depiction. Both might
be
         // non-free.
         descFields.Author = fuwAppendLines([
            fuwSurroundString("'''Original work:''' ", opts.NF3DCreator),
            "<br/>\n",
            fuwSurroundString("'''Depiction:''' ", opts.Author)
         ]);
         descFields.Date = fuwAppendLines([
            fuwSurroundString("'''Original work:''' ", opts.NF3DOrigDate),
            "<br/>\n",
            fuwSurroundString("'''Depiction:''' ", opts.Date)
         ]);
         descFields.Purpose = (
            opts.NF3DCheckDedicated ?
             ("For visual identification of the object of the article. " +
              "The article as a whole is dedicated specifically to a
discussion of this work.") :
            (opts.NF3DCheckDiscussed ?
             ("To support encyclopedic discussion of this work in this
article. " +
              "The illustration is specifically needed to support the
following point(s): " +
              "<br/>\n" + opts.NF3DPurpose) : null)
         );
         descFields.Replaceability = "n.a.";
         descFields.Commercial    = "n.a.";
         descFields["Other information"] = (
            opts.NF3DOptionSame ?
            ("The image was created and published by the same author who also
" +
             "holds the rights to the original object, and no alternative
depiction " +
             "could be suitably created.") :
            ("The author of the image has released the photographic work
```

```
under a " +
             "free license, or it is in the public domain: " +
opts.NF3DPermission)
         );
         break;

      case 'OptionNFExcerpt':
         // FURs for screenshots etc. don't normally need to bother
         // about replaceability (with free images) and with commercial role,
         // but do need to bother about purpose and about replaceability with
text.
         descFields.Purpose        = opts.NFPurpose;
         descFields.Replaceability_text = opts.NFReplaceableText;
         descFields.Replaceability = "n.a.";
         descFields.Commercial     = "n.a.";
         break;

      case 'OptionNFCover':
         // cover art gets standard rationales.
         descFields.Purpose =
            "to serve as the primary means of visual identification " +
            "at the top of the article dedicated to the work in question.";
         descFields.Replaceability = "n.a.";
         descFields.Commercial     = "n.a.";
         break;

      case 'OptionNFLogo':
         // logos get standard rationales.
         descFields.Purpose =
            "to serve as the primary means of visual identification " +
            "at the top of the article dedicated to the entity in question.";
         descFields.Replaceability = "n.a.";
         descFields.Commercial     = "n.a.";
         break;

      case 'OptionNFPortrait':
         // as with other historic photographs, it is useful to have both
         // original publication and direct source
         descFields.Source = fuwAppendLines([
            fuwSurroundString("'''Original publication''': ",
opts.Publication),
            "<br/>\n",
            fuwSurroundString("'''Immediate source''': ", opts.Source)
         ]);
         descFields.Purpose =
            "for visual identification of the person in question, " +
            "at the top of his/her biographical article";
         descFields.Replaceability = opts.NFReplaceable;
         descFields.Commercial = opts.NFCommercial;
         descFields['Other information'] =
            "The subject of the photograph has been deceased since: " +
```

```
opts.NFPortraitDeceased;
         break;

      case 'OptionNFMisc':
         descFields.Source = fuwAppendLines([
            fuwSurroundString(
               "'''Original publication''': ",
               opts.Publication,
               "<br/>\n'''Immediate source:''' "),
            "",
            opts.Source
         ]);
         descFields.Purpose = opts.NFPurpose;
         descFields.Replaceability = opts.NFReplaceable;
         descFields.Replaceability_text = opts.NFReplaceable_text;
         descFields.Commercial = opts.NFCommercial;
         break;
   };

   if (opts.OptionNonFree) {
      // common stuff for all non-free files:

      // Minimality field (same for all NF options):
      descFields.Minimality = opts.NFMinimality;

      // append optional "extra license" selector and "AnyOther" fields
      // to "Other information" field:
      descFields['Other information'] = fuwAppendLines([
         descFields['Other information'],
         "<br/>\n",
         fuwSurroundString('{{', opts.NFExtraLicense, '}}'),
         "<br/>\n",
         opts.AnyOther
      ]);
   }
   else {
      // common stuff for all free files:
      descFields.Other_versions = ''
      this.otherInfo = fuwAppendLines([this.otherInfo, "\n\n",
opts.AnyOther]);

   }

};

fuwGlobal.prototype.formatOutput = function(forCommons) {
   var baseForm = this.ScriptForm;
   var targetForm = this.TargetForm;
   if (fuwTesting) {
      var testForm   = this.TestForm;
   }
```

```
   var opts = this.opts;
   var otherInfo = this.otherInfo;
   var descFields = this.descFields;

   var summary = "{{" + this.getDescriptionTemplateName();

   // assemble all fields into the wikitext of the description page:
   var fieldOrder = [
      'Source', 'Date', 'Author', 'Permission', 'Other_versions',
      'Article', 'Purpose', 'Replaceability', 'Replaceability_text',
      'Minimality', 'Commercial', 'Other information'
   ];
   summary += "\n|Description = " + fuwPackInfo(descFields['Description'],
forCommons);
   for (var i = 0; i < fieldOrder.length; i++) {
      if (descFields[fieldOrder[i]]) {
         summary += "\n|" + fieldOrder[i] + " = " +
descFields[fieldOrder[i]];
      }
   }
   summary += "\n}}\n";
   if (otherInfo) {
      summary += "\n;Other information:\n" + fuwPackInfo(otherInfo,
forCommons) + "\n";
   }

   var editSummary = this.getEditSummary();

   var license = this.getLicense();

   if (forCommons) {
      // pack our description info into an url pointing to the
      // standard Commons Special:Upload
      // with pre-loaded description fields

      summary = fuwSubst("Upload marker added by en.wp UW") + "\n" + summary;
      summary = summary.replace(/\{\{OTRS pending\}\}/g, fuwSubst("OP"));

      if (license.special) {
         // manually format the whole description page including the license
tag, if it
         // isn't one of the bare standard licenses in the dropdown box.
Otherwise,
         // submit description summary and license as two separate url
parameters.
         summary = summary + "\n\n" + license.special;
      }
      return (fuwGetCommonsURL() +
         "?title=Special:Upload" +
         "&wpUploadDescription=" +
         encodeURIComponent(summary) +
```

```
         (license.standard ?
          ("&wpLicense=" + encodeURIComponent(license.standard)) : '') +
         "&wpDestFile=" +
         encodeURIComponent(opts.InputName));
   }
   else {
      // pack all description into a single "text" parameter to be submitted
      // to the local api.php upload.
      summary = "==Summary==\n" +
         summary +
         "\n==Licensing==\n" +
         (license.standard ? ("{{" + license.standard + "}}") :
license.special) +
         this.getTrackingCategory();

      if (fuwTesting) {
         // Testing mode: show our data in the dummy form
         // at the bottom of the page.
         fuwGet('placeholderSandboxFilename').innerHTML = opts.InputName;
         this.TestForm.SandboxSummary.value = editSummary;
         this.TestForm.SandboxText.value = summary;
         fuwSetVisible('placeholderTestForm', true);
      }
      // write output parameters into target form
      // I can't believe IE7 is too stupid to simply understand
"this.TargetForm.filename.value".
      ($('#TargetForm [name="filename"]')[0]).value = opts.InputName;
      ($('#TargetForm [name="text"]'    )[0]).value = summary;
      ($('#TargetForm [name="comment"]' )[0]).value = editSummary;
      ($('#TargetForm [name="token"]'   )[0]).value =
mw.user.tokens.get('editToken');

   }

};

function fuwHasUserGroup(group) {
   // workaround because old IE versions don't have array.indexOf :-(
   for (i = 0; i < wgUserGroups.length; i++) {
      if (wgUserGroups[i] == group) {
         return true;
      }
   }
   return false
}

fuwGlobal.prototype.getUserStatus = function() {
   // function to determine the experience status and userrights of the
current user:
   // 'anon': not logged in; can't use script.
   // 'notAutoconfirmed': can't use local upload, but may use script to
```

```
prepare upload for Commons
   // 'newbie': autoconfirmed but editcount < 100
   //    (may be used in future to adapt instructions more to newbie needs)
   // 'problem': autoconfirmed but has 3 or more image-related warnings or
deletion notifications among recent user talk entries
   //    (may be used in future to produce more strongly worded instructions)
   // 'autoconfirmed': regular user
   // 'sysop'

   if (wgUserName) {
      if (fuwHasUserGroup('sysop')) {
         this.userStatus = 'sysop';
      }
      else if (fuwHasUserGroup('autoconfirmed') ||
fuwHasUserGroup('confirmed')) {
         this.userStatus = 'autoconfirmed';
         $.ajax({
            url     : mw.util.wikiScript( 'api' ),
            type    : 'GET',
            dataType: 'xml',
            traditional : true,
            data:   {
                     format: 'xml',
                     action: 'query',
                     meta  : 'userinfo',
                     uiprop: 'editcount',
                     prop  : 'revisions',
                     titles: 'User talk' + wgUserName,
                     rvprop: 'comment|user',
                     rvlimit: 30
                    },
            success: function(data) {
            // callback func
               var fuw = window.fuw;
               if (data) {
                  var ui = data.getElementsByTagName('userinfo');
                  if (ui) {
                     var editcount = ui[0].getAttribute('editcount');
                     if (editcount < 100) {
                        fuw.userStatus = 'newbie';
                     }
                  }
                  var revs = data.getElementsByTagName('rev');
                  var countWarn = 0;
                  for (i = 0; i < revs.length; i++) {
                     var rev = revs[i];
                     var usr = rev.getAttribute('user');
                     var cmt = rev.getAttribute('comment');
                     if ((usr == 'ImageTaggingBot') ||
                         (cmt.search(/(tagging for deletion of
\[\[File)|(Uploading files missing)|(File (source and )?copyright licensing
```

```
problem)|(Speedy deletion nomination of \[\[File)|(Notification: listing at
\[\[possibly unfree files)/) >= 0)) {
                        countWarn += 1;
                     }
                  }
                  if (countWarn >= 3) {
                     fuw.userStatus = 'problem';
                  }
               }
            }
         });
      }
      else {
         this.userStatus = 'notAutoconfirmed';
      }
   }
   else {
      this.userStatus = 'anon';
   }
};

// ===================================================================
// Convenience function for getting the regular index.php
// interface of Commons. Not very elegant.
// ===================================================================
function fuwGetCommonsURL() {
   if (document.URL.match(/^https:/))
      return "https://commons.wikimedia.orghttp://wiki.waze.hu/w/index.php";
   else
      return "http://commons.wikimedia.orghttp://wiki.waze.hu/w/index.php";
}

// ===================================================================
// functions for building form elements
// ===================================================================
fuwMakeRadiobutton = function(group, option, checked, event) {
   // Stupid IE7 doesn't get "value" attribute unless it's created in this
convoluted way.
   // Annoying.
   var node = $('<input type="radio" id="' + option + '" name="' + group + '"
value="' + option + '"></input>')[0];
   if (checked) node.checked = true;
   node.onclick = event || fuwRadioClick;
   node.onclick = event || fuwRadioClick;
   fuwAppendInput(option, node);
};
fuwMakeTextfield = function(label, event) {
   var node  = document.createElement('input');
   node.type = 'text';
   node.name = label;
   node.size = fuwDefaultTextboxLength;
```

```
   node.onchange = event || fuwUpdateOptions;
   // only for testing:
   //node.value = label;
   fuwAppendInput(label, node);
};
fuwMakeTextarea = function(label, event) {
   var node  = document.createElement('textarea');
   node.name = label;
   node.rows = fuwDefaultTextareaLines;
   node.style.width = fuwDefaultTextareaWidth;
   node.onchange = event || fuwUpdateOptions;
   //only for testing:
   //node.innerHTML = label;
   fuwAppendInput(label, node);
};
fuwMakeCheckbox = function(label, checked, event) {
   var node  = document.createElement('input');
   node.name = label;
   node.type = 'checkbox';
   //only for testing:
   //node.title= label;
   node.checked = checked;
   node.onchange = event || fuwUpdateOptions;
   fuwAppendInput(label, node);
}
fuwMakeHiddenfield = function(name, value, id) {
   var node   = document.createElement('input');
   node.name  = name;
   node.type  = 'hidden';
   node.value = value;
   fuwAppendInput((id || name), node);
};
fuwMakeAnchor = function(label, href, content) {
   var node   = document.createElement('a');
   node.name  = label;
   node.target= "_blank";
   node.href  = href;
   node.innerHTML = content;
   fuwAppendInput(label, node);
};
fuwMakeSelection = function(name, values) {
   var root = document.createElement('select');
   var current = root;
   try {
      for (i=0; i<values.length; i++) {
         var line = values[i];
         var entry;
         if (line.length == 0) {
            current = root;
         }
         else if (line.length == 1) {
```

```
            entry = document.createElement('optgroup');
            entry.setAttribute('label', line[0]);
            root.appendChild(entry);
            current = entry;
         }
         else {
            entry = document.createElement('option');
            entry.setAttribute('value', line[0]);
            entry.setAttribute('title', '{{' + line[0] + '}}');
            entry.innerHTML = line[1];
            if (line.length > 2) {
               entry.setAttribute('selected', 'selected');
            }
            current.appendChild(entry);
         }
      }
   } catch (e) { alert("Name: " + name + ", i=" + i); }
   root.name = name;
   root.onchange = fuwUpdateOptions;
   fuwAppendInput(name, root);
};
function fuwMakeWikilink(place, target, redlink, display) {

   place = fuwGet(place);
   var id = place.id;
   var anchor;
   if (place.tagName == 'A') {
      anchor = place;
   }
   else {
      anchor = document.createElement('a');
      place.appendChild(anchor);
   }
   anchor.href = mw.util.wikiGetlink(target);
   anchor.title = target;
   anchor.innerHTML = target;
   anchor.className = (redlink ? 'new' : null);
}

function fuwAppendInput(label, content) {
   // append a newly created input element to an existing
   // span element marked as id="placeholderXYZ"
   var node = fuwGet('placeholder' + label);
   var old  = fuwGet(label);
   if (old) {
      old.parentNode.removeChild(old);
   }
   content.id = content.id || label;
   if (node) {
      while (node.hasChildNodes()) {
         node.removeChild(node.firstChild);
```

```
      }
      node.appendChild(content);
   }
}

// =====================================================
// move an element away from its current position
// and append it to a target element if condition is true
// =====================================================
function fuwMove(mv, tg, condition, toStart) {
   if (condition) {
      move   = fuwGet(mv);
      target = fuwGet(tg);
      if (move && target) {
         var parent = move.parentNode;
         if (! (target===parent)) {
            parent.removeChild(move);
            if (toStart) {
               target.insertBefore(move, target.firstChild);
            }
            else {
               target.appendChild(move);
            }
         }
      }
      else {
         alert("Can't find elements: move=" + mv + "(" + move + "), target="
+ tg + "(" + target + ")");
      }
   }
   return condition;
}

// =====================================================
// make an element visible/invisible
// =====================================================
function fuwSetVisible(tg, condition) {
   target = fuwGet(tg);
   if (target) {
      if (condition) {
         $(target).show();
      }
      else {
         $(target).hide();
      }
   }
   else {
      alert("Element not found: " +  ($.isDomElement(tg) ? tg.id : tg));
   }
}
```

```
// ================================================
// set enabled/disabled status for an element and/or
// all input controls contained in it.
// ================================================
function fuwSetEnabled(tg, condition) {
   target = fuwGet(tg);
   try {
      var elements =
(target.tagName.match(/^(input|textarea|select|button|a)$/i) ?
         [target] :
         $('#' + target.id + ' *'));
      for (i = 0; i<elements.length; i++) {
         if
(elements[i].tagName.match(/^(input|textarea|select|button|a)$/i)) {
            elements[i].disabled = (condition ? null : "disabled");
         }
      }
   } catch (e) { alert("Element not found: " + ($.isDomElement(tg) ? tg.id :
tg)); }
}

// ================================================
// convenience function to check whether a given
// element is currenly visible. Needs to check display
// property of the element and its ancestors
// ================================================
function fuwIsVisible(el) {
   element = fuwGet(el);
   if (!element) return false;
   el = element.id;

   var visible = true;
   while (! (element === document.body)) {
      if (element.style.display == "none") {
         visible = false;
         break;
      }
      element = element.parentNode;
   }
   return visible;
}

// ================================================
// cleanup filename
// ================================================
function fuwCleanFilename() {
   var nameBox = window.fuw.ScriptForm.InputName;
   var oldname = name = $.trim(nameBox.value);

   if (name) {
      // strip accidentally added [[   ]] or [[:  ]] brackets
```

```
      name = name.replace(/(^\[\[:?)|(\]\]$)/g, "");
      // strip accidentally added "File:" prefix
      name = name.replace(/^(File|Image):/, "");
      // replace underscores with spaces
      name = name.replace(/_/g, " ");
      // uppercase first letter
      name = $.ucFirst(name);
   }
   if (oldname != name) {
      nameBox.value = name;
   }
   // always return true so the next validation step will proceed:
   return true;
}


// ================================================
// check filename for technically illegal
// characters, trying to fix them automatically
// ================================================
function fuwCheckLegalFilename() {
   var nameBox = window.fuw.ScriptForm.InputName;
   var oldname = name = $.trim(nameBox.value);

   if (name) {
      // resolve accidentally entered html entities and URI-encoded %XX
character codes
      name = name.replace(/\&[a-z]+;/g, fuwHtmlEntityDecode);
      name = name.replace(/(\%[A-F0-9]{2,2})/g, decodeURI);
      // remove illegal characters # < > [ ] | { } /:
      // using a best guess for an acceptable replacement
      name = name.replace(/[<\[\{]/g, "(");
      name = name.replace(/[>\]\}]/g, ")");
      name = name.replace(/[#:\|]/g,  ",");
      name = name.replace(/\//g, "-");
      // remove sequences of tildes
      name = name.replace(/\~{3,}/g, "---");
      // remove initial slash
      name = name.replace(/^\//, "");
   }

   if (oldname != name) {
      window.fuw.warn.IllegalChars = true;
      nameBox.value = name;
      return false;
   }
   else {
      window.fuw.warn.IllegalChars = false;
      return true;
   }
}
```

```
function fuwHtmlEntityDecode(str) {
   // hack to translate accidentally entered html entity code
   // into actual characters
   var ta=document.createElement('textarea');
   ta.innerHTML=str.replace(/</g,'<').replace(/>/g,'>');
   return ta.value;
}

// ========================================================
// Check against various common patterns of poorly chosen
// filenames (too short / too generic)
// ========================================================
function fuwCheckPoorFilename() {
   var nameBox = window.fuw.ScriptForm.InputName;
   var name = $.trim(nameBox.value);
        name =
name.replace(/\.(png|gif|jpg|jpeg|xcf|pdf|mid|ogg|ogv|svg|djvu|tiff|tif|oga)$
/i, "");

   // name should be at least 10 characters long, excluding file type
extension
   var tooShort = (name.length < 10);

   // common generic filename patterns:
   // IMG......jpg
   // Image....jpg
   // DSC......jpg
   // Picture......jpg
   // Pic.........jpg
   // anything that has fewer than 3 alphabetic letters and then just numbers
   var pattern = /^(img|image|dsc|picture|pic)?(\\s*|\\_*|[a-z]{,3})?\\d+$/i;
   var auto = name.match(pattern);

   window.fuw.warn.BadFilename = (tooShort || auto);
   return !tooShort && !auto;
}

// ========================================================
// check if file extensions match between local filename
// and target filename input box. Automatically append
// appropriate extension to target filename if they don't.
// ========================================================
function fuwCheckFileExtension() {
   var nameBox = window.fuw.ScriptForm.InputName;
   var name = $.trim(nameBox.value);
   var fileBox = window.fuw.TargetForm.file;
   var file = fileBox.value;

   // cancel check if no filename has been provided yet
   if (!file || !name) return true;
```

```
   var extensions =
/.+\.(png|gif|jpg|jpeg|xcf|pdf|mid|ogg|ogv|svg|djvu|tiff|tif|oga)$/i;
   var mimetypes = {
      "png"  : "image/png",
      "gif"  : "image/gif",
      "jpg"  : "image/jpeg",
      "jpeg" : "image/jpeg",
      "xcf"  : "image/x-xcf",
      "pdf"  : "application/pdf",
      "mid"  : "audio/rtp-midi",
      "ogg"  : "audio/ogg",
      "ogv"  : "video/ogg",
      "svg"  : "image/svg+xml",
      "djvu" : "image/vnd.djvu",
      "tiff" : "image/tiff",
      "tif"  : "image/tiff",
      "oga"  : "video/ogg"
   };

   var found = extensions.exec(file);
   var fileExt = found ? found[1].toLowerCase() : "";
   found = extensions.exec(name);
   var nameExt = found ? found[1].toLowerCase() : "";
   var mime = mimetypes[fileExt];

   if (fileExt && mime && (mimetypes[nameExt] != mime)) {
      nameBox.value = name.replace(/\.?$/, ('.' + fileExt));
   }
   return true;
}

// ============================================================
// Check if a file under the chosen name already exists,
// either locally or on Commons.
// Store results in the fuw.warn object, so warnings will
// be displayed on the next fuwUpdateOptions() call
// ============================================================
function fuwCheckFileExists() {
   // this is an asynchronous AJAX function.
   // results won't yet be present when this function returns.

   var nameBox = window.fuw.ScriptForm.InputName;
   var name = $.trim(nameBox.value);

   // using the jQuery wrapper for the Ajax functionality:
   $.ajax({
      url     : mw.util.wikiScript( 'api' ),
      type    : 'GET',
      dataType: 'xml',
      traditional : true,
      data:   {
```

```
               format: 'xml',
               action: 'query',
               titles: 'File:' + name,
               prop  : 'imageinfo',
               iiprop: 'url|user',
               iiurlwidth: 120
              },
      success: function(resp) {
      // callback function, called when API query has succeeded:
         // see if the request has returned info from an existing image:
         var foundlist = resp.getElementsByTagName('ii');
         var exists = (foundlist.length >= 1);
         var isCommons = false;
         if (exists) {

            // extract description data from http response.
            // see
https://www.mediawiki.org/wiki/API:Properties#imageinfo_.2F_ii
            // for structure of API response
            var foundImg = foundlist[0];
            isCommons =
(foundImg.parentNode.parentNode.getAttribute('imagerepository')=='shared');

            // need this data for creating our own image thumb link
            var width = foundImg.getAttribute('thumbwidth');
            var height = foundImg.getAttribute('thumbheight');
            var thumbURL = foundImg.getAttribute('thumburl');
            var lastUser = foundImg.getAttribute('user');
            var descURL = foundImg.getAttribute('descriptionurl');

            // API returns link to local description page even for Commons
images.
            // However, we want a direct link to Commons.
            if (isCommons) {
               descURL = descURL.replace(/en\.wikipedia\.org/,
"commons.wikimedia.org");
               descURL =
descURL.replace(/\/\/secure\.wikimedia\.org\/wikipedia\/en/,
"commons.wikimedia.org");
            }

            // build the image info into the warning section of our page:
            thumbDiv = fuwGet('warningImageThumb');
            if (thumbDiv) {

               // make all links point to description page:
               var thumbA = thumbDiv.getElementsByTagName('a');
               for (i = 0; i<thumbA.length; i++) {
                  thumbA[i].setAttribute('href', descURL);
               }
               // insert the image itself:
```

```
                var thumbImg = thumbDiv.getElementsByTagName('img');
                if (thumbImg.length > 0) {
                   thumbImg = thumbImg[0];
                   thumbImg.setAttribute('src', thumbURL);
                   thumbImg.setAttribute('width', width);
                   thumbImg.setAttribute('height', height);
                }
                // insert the name of the last uploader:
                var thumbSpan = fuwGet('existingImageUploader');
                // TO DO: turn this into a proper link
                if (thumbSpan) thumbSpan.innerHTML = lastUser;
             }


          }
          warn = window.fuw.warn;
          warn.ImageOnCommons = exists && isCommons;
          warn.ImageExists    = exists && !isCommons;

          fuwUpdateOptions();
       }
    });
}

// ============================================================
// onchange event handler for the local filename box
// ============================================================
fuwValidateFile = function() {
   fuwCheckFileExtension();
   fuwUpdateOptions();
}

// ============================================================
// onchange event handler for the name input box
// ============================================================
fuwValidateFilename = function() {
   fuwCleanFilename();
   if (
      fuwCheckLegalFilename() &&
      fuwCheckPoorFilename() &&
      fuwCheckFileExtension()) {
      // after fuwCheckFileExists(),
      // fuwUpdateOptions will be triggered
      // by the callback function after Ajax completion
      fuwCheckFileExists();
   }
   else {
      // if there's been no Ajax call.
      fuwUpdateOptions();
   }
};
```

```
// ============================================================
// function fuwValidateNFArticle()
// ============================================================
// This is the validation routine for the obligatory
// article-to-be-used-in field for non-free files. It queries
// api.php about the target article through an Ajax call.
// It will store error info in the fuw.warn object,
// triggering the following error on the next updateOptions():
// * warningNFArticleNotFound : target page doesn't exist.
// * warningNFArticleNotMainspace : target is not an article.
// * warningNFArticleDab : target is a disambiguation page.
// Redirects will automatically be substituted.
// ============================================================
fuwValidateNFArticle = function() {

   var nameBox = window.fuw.ScriptForm.NFArticle;
   oldname = name = nameBox.value;

   // cleanup article name:
   // automatically fix accidentally added [[ ... ]] and
   // regularize underscores
   name = $.trim(name);
   name = name.replace(/(^\[\[)|(\]\]$)/g, "");
   // automatically fix article names entered as full urls:
   name = name.replace(/^https?:\/\/en\.wikipedia\.org\/wiki\//, "");
   name =
name.replace(/^https?:\/\/en\.wikipedia\.org\/w\/index\.php\?title=/, "");
   name = name.replace(/_/g, " ");
   if (name != oldname) nameBox.value = name;

   // do nothing more if field was blank
   if (!name) return;

   // using the jQuery wrapper for the Ajax functionality:
   $.ajax({
      url     : mw.util.wikiScript( 'api' ),
      type    : 'GET',
      dataType: 'xml',
      traditional : true,
      data:   {
               format: 'xml',
               action: 'query',
               titles: name,
               prop  : 'info|categories|links'
              },
      success: function(resp) {
      // callback function, called when API query has succeeded:
         var errorType = 0;
         var pg = resp.getElementsByTagName('page')[0];
         var title = pg.getAttribute('title');
         var target = title;
```

```
         if (pg.getAttribute('missing') != null) {
            // no page found under this title.
            errorType = 1;
         }
         else {
            var userspace = false;
            var ns = pg.getAttribute('ns');
            var rd = pg.getAttribute('redirect');
            if (ns != 0) {
               // not a mainspace page!
               errorType = 2;

               // try to detect if the target might be a user space draft:
               if (title.match(new RegExp("User( talk)?:" +
mw.config.get('wgUserName')))) {
                  userspace = true;
               }
            }
            else if (rd != null) {
               // redirect page
               // API returns an empty redirect="" attribute if
               // the page is a redirect
               var targets = pg.getElementsByTagName('pl');
               for (i=0; i<targets.length; i++) {
                  var link = targets[i];
                  if (link.getAttribute('ns')==0) {
                     target = link.getAttribute('title');
                     errorType = 3;
                     break;
                  }
               }
            }
            else {
               // check for disambiguation categories
               var cats = pg.getElementsByTagName('cl');
               for (i=0; i<cats.length; i++) {
                  var cat = cats[i];
                  if (cat.getAttribute('title') == "Category:All
disambiguation pages") {
                     errorType = 4;
                     break;
                  }
               }
            }
         }
         warn = window.fuw.warn;
         warn.NFArticleNotFound = (errorType==1);
         warn.NFArticleNotMainspace = (errorType==2);
         warn.UserspaceDraft = ((errorType==2) && userspace);
         warn.NFArticleDab = (errorType==4);
         warn.NFArticleOK  = (errorType==0);
```

```
         // fix links in error messages:
         if (warn.NFArticleNotFound) {

fuwMakeWikilink(fuwGet('warningNFArticleNotFound').getElementsByTagName('A')[
0], target, true);
         }
         else if (warn.NFArticleNotMainspace) {

fuwMakeWikilink(fuwGet('warningNFArticleNotMainspace').getElementsByTagName('
A')[0], target);
         }
         else if (warn.NFArticleDab) {

fuwMakeWikilink(fuwGet('warningNFArticleDab').getElementsByTagName('A')[0],
target);
         }
         else if (warn.NFArticleOK) {

fuwMakeWikilink(fuwGet('NFArticleOK').getElementsByTagName('A')[0], target);
         }

         if (errorType==3) {
            // automatically replace title with redirect target
            window.fuw.ScriptForm.NFArticle.value = target;
            // need to recursively call validation again now
            //if (confirm(name + " is a redirect. Follow it to " + target +
"?")) {
               fuwValidateNFArticle();
            //}
         }
         else {
            fuwUpdateOptions();
         }
      }
   });
};

// ================================================
// manually reload script (just for testing)
// ================================================
function fuwReload() {
   mw.loader.load( 'http://localhost/script/uploadscript.js' );
   fuwReset();
}

// ================================================
// reset forms
// TO DO: add a button that actually triggers this.
// ================================================
function fuwReset() {
   var forms = mw.util.$content[0].getElementsByTagName('form');
```

```
   for (i = 0; i < forms.length; i++) {
      forms[i].reset();
      window.fuw.warn = { };
      window.fuw.opts = { };
   }
   fuwSetVisible('UploadScriptArea', true);
   fuwSetVisible('fuwSuccess', false);
   fuwSetVisible('fuwWaiting', false);
   fuwUpdateOptions();
}

// ================================================
// convenience functions for string handling
// ================================================
function fuwAppendLines(parts) {
   // assemble a string from an array of strings.
   // treat every second element as a conditional
   // separator that will be included only if
   // surrounding elements are non-empty.
   var build = "";
   for (var i = 0; i < parts.length; i += 2) {
      if (parts[i]) {
         if (build) build += parts[i - 1];
         build += parts[i];
      }
   }
   return build;
}
function fuwSurroundString(prefix, content, suffix) {
   // put a prefix and a suffix on a string,
   // if the input string is non-empty.
   if (content)
      return (prefix ? prefix : "") + content + (suffix ? suffix : "");
   else return "";
}

// ========================================================
// convenience function for accessing the contents of the
// dummy TargetIFrame
// ========================================================
function fuwGetDocumentFromIFrame(iframe) {
   var doc = (iframe.contentDocument ? iframe.contentDocument :
iframe.contentWindow.document);
   if (doc.XMLDocument) {
      doc = doc.XMLDocument;
   }
   return doc;
}
// ========================================================
// event handler for the dummy TargetIFrame's onload event.
// TO DO: expand stub to add real notification of success,
```

```
// link to new file page, instructions about how to include
// file in articles, etc.
// ======================================================
function fuwUploadCompleted() {
   var doc = fuwGetDocumentFromIFrame(fuwGet('TargetIFrame'));
   if (doc) {
      //alert(doc);
      fuwSetVisible('successThumb', false);

      var fuw = window.fuw;
      var name = fuw.opts.InputName;

      var uploads = doc.getElementsByTagName('upload');
      var success = false;
      for (i = uploads.length-1; i>=0; i--) {
         if (uploads[i].getAttribute('result') == 'Success') {
            success = true;
            // need to get the real resulting filename here; might be
different from the requested one in some cases.
            name = uploads[i].getAttribute('filename');
            break;
         }
      }
      if (success) {

         // need another ajax call to check the file is actually there,
         // and to retrieve its direct thumb img url:
         $.ajax({
            url     : mw.util.wikiScript( 'api' ),
            type    : 'GET',
            dataType: 'xml',
            traditional : true,
            data:   {
                     format: 'xml',
                     action: 'query',
                     titles: 'File:' + name,
                     prop  : 'imageinfo',
                     iiprop: 'url',
                     iiurlwidth: 120
                    },
            success: function(resp) {
               // callback function, called when API query has succeeded:
               // see if the request has returned info from an existing image:

               var foundImg = resp.getElementsByTagName('ii')[0];
               if (foundImg) {

                  // need this data for creating our own image thumb link
                  var width = foundImg.getAttribute('thumbwidth');
                  var height = foundImg.getAttribute('thumbheight');
                  var thumbURL = foundImg.getAttribute('thumburl');
```

```
                var lastUser = foundImg.getAttribute('user');
                var descURL = foundImg.getAttribute('descriptionurl');

                // build the thumbnail in the success message:
                thumbDiv = fuwGet('successThumb');

                // make link point to description page:
                var thumbA = thumbDiv.getElementsByTagName('a')[0];
                thumbA.href = descURL;

                // insert the image itself:
                var thumbImg = thumbDiv.getElementsByTagName('img')[0];
                thumbImg.setAttribute('src', thumbURL);
                thumbImg.setAttribute('width', width);
                thumbImg.setAttribute('height', height);

                fuwSetVisible(thumbDiv, true);
             }
          }
       });
       // moving this elsewhere, as long as this callback isn't working:
       //fuwMakeWikilink(
       //   fuwGet('fuwSuccessLink').getElementsByTagName('a')[0],
       //   'File:' + name);
       fuwGet('placeholderExFilename1').innerHTML = name;
       fuwGet('placeholderExFilename2').innerHTML = name;
       fuwSetVisible('fuwSuccess', true);
       fuwSetVisible('fuwWaiting', false);
    }
    else {
       var err = doc.getElementsByTagName('error');
       if (err) {
          var info = err[0].getAttribute('info');
          var details = err[0].getElementsByTagName('detail');
          var add = "";
          for (i = 0; i < details.length; i++) {
             if (add.length > 0) add += ", ";
             add += details[i].textContent;
          }
          if (add) {
             info = info + " (" + add + ")";
          }
          alert("Upload failed: " + info);
       }
       else {
          alert("Unknown error: upload may have failed.");
       }
    }
 }
}
```

```
// ============================================================
// clean out dummy IFrame before submitting a request
// ============================================================
function fuwResetTargetIFrame() {
   var doc = fuwGetDocumentFromIFrame(fuwGet('TargetIFrame'));
   if (doc) {
      while (doc.hasChildNodes()) {
         doc.removeChild(doc.firstChild);
      }
   }
}

// ============================================================
// Event handler for the real submit button
// ============================================================
function fuwSubmitUpload() {

   var fuw = window.fuw;
   var frm = fuw.TargetForm;

   fuw.collectInput();
   fuw.formatOutput(false);

   // we will use the iframe's onload event to trigger a function that
   // adds success notification etc.
   fuwResetTargetIFrame();
   var ifr = fuwGet('TargetIFrame');
   if (ifr.attachEvent) {
      // workaround for IE, according to
      //
http://www.nczonline.net/blog/2009/09/15/iframes-onload-and-documentdomain/
      ifr.attachEvent("onload", fuwUploadCompleted);
   }
   else {
      // all other browsers
      ifr.onload = fuwUploadCompleted;
   }

   if (fuwTesting) {
      fuwSetVisible('placeholderTestForm', false);
   }
   fuwSetVisible('UploadScriptArea', false);

   // moved here temporarily, for as long as the success callback isn't
working:
   fuwMakeWikilink(
     fuwGet('fuwSuccessLink').getElementsByTagName('a')[0], 'File:' +
fuw.opts.InputName);

   fuwSetVisible('fuwWaiting', true);
```

```
   frm.submit();
   var opts = window.fuw.opts;
   // the API won't overwrite the description page text while overwriting
   // a file, which is really, really, really annoying and stupid.
   // So in the opts.OverwriteDifferent scenario, we need to edit
   // the description page through a separate ajax call. Dang.
   if (opts.OverwriteDifferent) {
      $.ajax({
         url   : mw.util.wikiScript('api'),
         type  : 'POST',
         dataType : 'xml',
         data  : {
                  format : 'xml',
                  action : 'edit',
                  title  : 'File:' + opts.InputName,
                  token  : mw.user.tokens.get('editToken'),
                  summary : opts.EditSummary,
                  text   : ($('#TargetForm .[name="text"]')[0]).value

                 }
      });
   }
}

// ========================================================
// Event handler for the Commons submit button
// ========================================================
function fuwSubmitCommons() {
   var fuw = window.fuw;
   fuw.collectInput();
   var url = fuw.formatOutput(true);
   alert("You will now be redirected to Commons. \nPlease use the Commons
upload form to add categories to your file description, and then complete the
upload.");
   window.location = url;
}

// ========================================================
// Event handler for the test submit button
// (write description string to sandbox only)
// ========================================================
function fuwSubmitSandbox() {
   var frm = window.fuw.TestForm;
   $.ajax({
      url     : mw.util.wikiScript( 'api' ),
      type    : 'POST',
      dataType: 'xml',
      data:   {
               format: 'xml',
               action: 'edit',
               title : mw.config.get('wgPageName') + "/sandbox",
```

```
                token : mw.user.tokens.get('editToken'),
                recreate : 1,
                summary  : frm.SandboxSummary.value,
                text     : frm.SandboxText.value
               },
      success: function(resp) {
         alert("Sandbox page edited!");
      }
   });
}



// ============================================================
// convenience wrapper function to replace calls to
// document.getElementById()
// to avoid browser incompatibility
// ============================================================
function fuwGet(target) {
   if ($.isDomElement(target)) return target;
   else {
      var found = $('#' + target);
      if (found) return found[0];
   }
   return undefined;
}

// ============================================================
// onload hook function, loading this script
// ============================================================
addOnloadHook(function() {
   if (fuwGet('UploadScriptArea')) {
      window.fuw = new fuwGlobal();
      fuwUpdateOptions();
   }
});
```