MediaWiki:Gadget-ImageAnnotator.js

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

- **Firefox / Safari:** Hold *Shift* while clicking *Reload*, or press either *Ctrl-F5* or *Ctrl-R* (*⌘-R* on a Mac)
- **Google Chrome:** Press *Ctrl-Shift-R* (*⌘-Shift-R* on a Mac)
- **Internet Explorer / Edge:** Hold *Ctrl* while clicking *Refresh*, or press *Ctrl-F5*
- **Opera:** Press *Ctrl-F5*.

```
// <source lang="javascript">
/*
  ImageAnnotator v2.3.2

  ATTENTION:
  This is in the Gadget- prefix but not actually registered nor loaded as a Gadget. It is
  loaded directly by [[MediaWiki:Common.js]], raw, unminified and in the global scope.

  Image annotations. Draw rectangles onto image thumbnail displayed on image description
  page and associate them with textual descriptions that will be displayed when the mouse
  moves over the rectangles. If an image has annotations, display the rectangles. Add a
  button to create new annotations.

  Note: if an image that has annotations is overwritten by a new version, only display the
  annotations if the size of the top image matches the stored size exactly. To recover
  annotations, one will need to edit the image description page manually, adjusting image
  sizes and rectangle coordinates, or re-enter annotations.

  Author: [[User:Lupo]], June 2009 - March 2010
  License: Quadruple licensed GFDL, GPL, LGPL and Creative Commons Attribution 3.0 (CC-BY-3.0)

  Choose whichever license of these you like best :-)

  See http://commons.wikimedia.org/wiki/Help:Gadget-ImageAnnotator for documentation.
*/

// Global: importScript, importScriptURI (wiki.js)
/*jshint eqnull:true, laxbreak:true, laxcomma:true */

if (typeof ImageAnnotator === 'undefined') { // Guard against multiple inclusions

mw.loader.load( '/index.php?title=MediaWiki:LAPI.js&action=raw&ctype=text/javascript' );
mw.loader.load( '/index.php?title=MediaWiki:Tooltips.js&action=raw&ctype=text/javascript' );
mw.loader.load( '/index.php?title=MediaWiki:TextCleaner.js&action=raw&ctype=text/javascript' );
mw.loader.load( '/index.php?title=MediaWiki:UIElements.js&action=raw&ctype=text/javascript' );

(function () { // Local scope

var ImageAnnotator_config = null;

var ImageAnnotation = function () {
    this.initialize.apply(this, arguments);
};

ImageAnnotation.compare = function (a, b) {
  var result = b.area() - a.area();
  if (result !== 0) return result;
  // Just to make sure the order is complete
  return a.model.id - b.model.id;
};

ImageAnnotation.prototype = {
  // Rectangle to be displayed on image: a div with pos and size
  view: null,
  // Internal representation of the annotation
  model: null,
  // Tooltip to display the annotation
  tooltip : null,
  // Content of the tooltip
  content : null,
  // Reference to the viewer this note belongs to
  viewer: null,

  initialize: function (node, viewer, id) {
    var is_new = false;
    var view_w = 0, view_h = 0, view_x = 0, view_y = 0;
    this.viewer = viewer;
    if (LAPI.DOM.hasClass(node, IA.annotation_class)) {
      // Extract the info we need
      var x = IA.getIntItem('view_x_' + id, viewer.scope);
      var y = IA.getIntItem('view_y_' + id, viewer.scope);
      var w = IA.getIntItem('view_w_' + id, viewer.scope);
      var h = IA.getIntItem('view_h_' + id, viewer.scope);
      var html = IA.getRawItem('content_' + id, viewer.scope);
      if (x === null || y === null || w === null || h === null || html === null)
        throw new Error('Invalid note');
      if (x < 0 || x >= viewer.full_img.width || y < 0 || y >= viewer.full_img.height)
        throw new Error('Invalid note: origin invalid on note ' + id);
      if (   x + w > viewer.full_img.width + 10
          || y + h > viewer.full_img.height + 10)
      {
        throw new Error('Invalid note: size extends beyond image on note ' + id);
      }
      // Notes written by early versions may be slightly too large, whence the + 10 above. Fix this.
      if (x + w > viewer.full_img.width) w = viewer.full_img.width - x;
      if (y + h > viewer.full_img.height) h = viewer.full_img.height - y;
      view_w = Math.floor(w / viewer.factors.dx);
      view_h = Math.floor(h / viewer.factors.dy);
      view_x = Math.floor(x / viewer.factors.dx);
      view_y = Math.floor(y / viewer.factors.dy);
```

```
        this.view =
          LAPI.make(
              'div', null
            , { position: 'absolute'
               ,display: 'none'
               ,lineHeight: '0px' // IE
               ,fontSize: '0px' // IE
               ,top: '' + view_y + 'px'
               ,left: '' + view_x + 'px'
               ,width: '' + view_w + 'px'
               ,height: '' + view_h + 'px'
              }
          );
        // We'll add the view to the DOM once we've loaded all notes
        this.model =
          { id: id
           ,dimension: {x: x, y: y, w: w, h: h}
           ,wiki: ''
           ,html: html.cloneNode(true)
          };
      } else {
        is_new = true;
        this.view = node;
        this.model =
          { id: -1
           ,dimension: null
           ,wiki: ''
           ,html: null
          };
        view_w = this.view.offsetWidth - 2; // Subtract cumulated border widths
        view_h = this.view.offsetHeight - 2;
        view_x = this.view.offsetLeft;
        view_y = this.view.offsetTop;
      }
      // Enforce a minimum size of the view. Center the 6x6px square over the center of the old view.
      // If we overlap the image boundary, adjustRectangleSize will take care of it later.
      if (view_w < 6) {view_x = Math.floor(view_x + view_w / 2 - 3); view_w = 6; }
      if (view_h < 6) {view_y = Math.floor(view_y + view_h / 2 - 3); view_h = 6; }
      Object.merge(
        {
          left: '' + view_x + 'px',
          top: '' + view_y + 'px',
          width: '' + view_w + 'px',
          height: '' + view_h + 'px'
        },
        this.view.style
      );
      this.view.style.zIndex = 500; // Below tooltips
      try {
        this.view.style.border = '1px solid ' + this.viewer.outer_border;
      } catch (ex) {
        this.view.style.border = '1px solid ' + IA.outer_border;
      }
      this.view.appendChild(
        LAPI.make(
            'div', null
          , { lineHeight: '0px' // IE
             ,fontSize: '0px' // IE
             ,width: '' + Math.max(view_w - 2, 0) + 'px' // -2 to leave space for the border
             ,height: '' + Math.max(view_h - 2, 0) + 'px'
            }
        )
        // width=100% doesn't work right: inner div's border appears outside on right and bottom on FF.
      );
      try {
        this.view.firstChild.style.border = '1px solid ' + this.viewer.inner_border;
      } catch (ex) {
        this.view.firstChild.style.border = '1px solid ' + IA.inner_border;
      }
      if (is_new) viewer.adjustRectangleSize(this.view);
      // IE somehow passes through event to the view even if covered by our cover, displaying the tooltips
      // when drawing a new rectangle, which is confusing and produces a selection nightmare. Hence we just
      // display raw rectangles without any tooltips attached while drawing. Yuck.
      this.dummy = this.view.cloneNode(true);
      viewer.img_div.appendChild(this.dummy);
      if (!is_new) {
        // New notes get their tooltip only once the editor has saved, otherwise IE may try to
        // open them if the mouse moves onto the view even though there is the cover above them!
        this.setTooltip();
      }
    },

    setTooltip: function () {
      if (this.tooltip || !this.view) return; // Already set, or corrupt
      // Note: on IE, don't have tooltips appear automatically. IE doesn't do it right for transparent
      // targets and we have to show and hide them ourselves through a mousemove listener in the viewer
      // anyway. The occasional event that IE sends to the tooltip may then lead to ugly flickering.
      this.tooltip = new Tooltip
        (  this.view.firstChild
         , this.display.bind(this)
         , { activate: (LAPI.DOM.is_ie ? Tooltip.NONE : Tooltip.HOVER)
            ,deactivate: (LAPI.DOM.is_ie ? Tooltip.ESCAPE : Tooltip.LEAVE)
            ,close_button: null
            ,mode: Tooltip.MOUSE
            ,mouse_offset : {x: -5, y: -5, dx: (IA.is_rtl ? -1 : 1), dy: 1}
            ,open_delay: 0
            ,hide_delay: 0
            ,onclose: (function (tooltip, evt) {
                         if (this.view) {
                           try {
                             this.view.style.border = '1px solid ' + this.viewer.outer_border;
                           } catch (ex) {
                             this.view.style.border = '1px solid ' + IA.outer_border;
                           }
                         }
                         if (this.viewer.tip == tooltip) this.viewer.tip = null;
                         // Hide all boxes if we're outside the image. Relies on hide checking the
                         // coordinates! (Otherwise, we'd always hide...)
                         if (evt) this.viewer.hide(evt);
```

```
                           }).bind(this)
        ,onopen: (function (tooltip) {
                             if (this.view) {
                               try {
                                 this.view.style.border = '1px solid ' + this.viewer.active_border;
                               } catch (ex) {
                                 this.view.style.border = '1px solid ' + IA.active_border;
                               }
                             }
                             this.viewer.tip = tooltip;
                           }).bind(this)
       }
     , IA.tooltip_styles
    );
  },

  display: function (evt) {
    if (!this.content) {
      this.content = LAPI.make('div');
      var main = LAPI.make('div');
      this.content.appendChild(main);
      this.content.main = main;
      if (this.model.html) main.appendChild(this.model.html.cloneNode(true));
      // Make sure that the popup encompasses all floats
      this.content.appendChild(LAPI.make('div', null, { clear: 'both' }));
      if (this.viewer.may_edit) {
        this.content.button_section =
          LAPI.make(
             'div'
             ,null
             ,{ fontSize : 'smaller'
               ,textAlign: (IA.is_rtl ? 'left' : 'right')
               ,borderTop: IA.tooltip_styles.border
              }
            );
        this.content.appendChild(this.content.button_section);
        this.content.button_section.appendChild(LAPI.DOM.makeLink(
               '#'
             , ImageAnnotator.UI.get('wpImageAnnotatorEdit', true)
             , null
             , LAPI.Evt.makeListener(this, this.edit)
            )
          );
        if (ImageAnnotator_config.mayDelete()) {
          this.content.button_section.appendChild(document.createTextNode('\xa0'));
          this.content.button_section.appendChild
            (LAPI.DOM.makeLink
              (  '#'
               , ImageAnnotator.UI.get('wpImageAnnotatorDelete', true)
               , null
               , LAPI.Evt.makeListener(this, this.remove_event)
              )
            );
        }
      }
    }
    return this.content;
  },

  edit: function (evt) {
    if (IA.canEdit()) IA.editor.editNote(this);
    if (evt) return LAPI.Evt.kill(evt);
    return false;
  },

  remove_event: function (evt) {
    if (IA.canEdit()) this.remove();
    return LAPI.Evt.kill(evt);
  },

  remove: function () {
    if (!this.content) { // New note: just destroy it.
      this.destroy();
      return true;
    }
    if (!ImageAnnotator_config.mayDelete()) return false;

    // Close and remove tooltip only if edit succeeded! Where and how to display error messages?

    var reason = '';
    if (!ImageAnnotator_config.mayBypassDeletionPrompt() || !window.ImageAnnotator_noDeletionPrompt) {
      // Prompt for a removal reson
      reason = prompt(ImageAnnotator.UI.get('wpImageAnnotatorDeleteReason', true), '');
      if (reason === null) return false; // Cancelled
      reason = reason.trim();
      if (!reason.length) {
        if (!ImageAnnotator_config.emptyDeletionReasonAllowed()) return false;
      }
      // Re-show tooltip (without re-positioning it, we have no mouse coordinates here) in case
      // it was hidden because of the alert. If possible, we want the user to see the spinner.
      this.tooltip.show_now(this.tooltip);
    }


    var self = this;
    var spinnerId = 'image_annotation_delete_' + this.model.id;
    LAPI.Ajax.injectSpinner(this.content.button_section.lastChild, spinnerId);
    if (this.tooltip) this.tooltip.size_change();
    LAPI.Ajax.editPage(
        mw.config.get('wgPageName')
      , function (doc, editForm, failureFunc, revision_id) {
          try {
            if (revision_id && revision_id != mw.config.get('wgCurRevisionId'))
              throw new Error('#Page version (revision ID) mismatch: edit conflict.');

            var textbox = editForm.wpTextbox1;
            if (!textbox) throw new Error('#Server replied with invalid edit page.');
            var pagetext = textbox.value.replace(/\r\n/g, '\n');
            // Normalize different end-of-line handling. Opera and IE may use \r\n, whereas other
            // browsers just use '\n'. Note that all browsers do the right thing if a '\n' is added.
            // We normally don't care, but here we need this to make sure we don't leave extra line
            // breaks when we remove the note.

            IA.setWikitext(pagetext);

            var span = IA.findNote(pagetext, self.model.id);
            if (!span) { // Hmmm? Doesn't seem to exist
              LAPI.Ajax.removeSpinner(spinnerId);
              if (self.tooltip) self.tooltip.size_change();
              self.destroy();
              return;
            }
            var char_before = 0;
            var char_after  = 0;
            if (span.start > 0) char_before = pagetext.charCodeAt(span.start - 1);
            if (span.end < pagetext.length) char_after = pagetext.charCodeAt(span.end);
            if (   String.fromCharCode(char_before) == '\n'
                && String.fromCharCode(char_after)  == '\n')
              span.start = span.start - 1;
            pagetext = pagetext.substring(0, span.start) + pagetext.substring(span.end);
            textbox.value = pagetext;
            var summary = editForm.wpSummary;
            if (!summary)
              throw new Error('#Summary field not found. Check that edit pages have valid XHTML.');
            IA.setSummary(
                summary
              ,    ImageAnnotator.UI.get('wpImageAnnotatorRemoveSummary', true)
                || '[[MediaWiki talk:Gadget-ImageAnnotator.js|Removing image note]]$1'
              , (reason.length ? reason + ': ' : '') + self.model.wiki
            );
          } catch (ex) {
            failure (null, ex);
            return;
          }
          var edit_page = doc;
          LAPI.Ajax.submitEdit(
              editForm
            , function (request) {
                if (edit_page.isFake && (typeof edit_page.dispose === 'function'))
                  edit_page.dispose();
                var revision_id = LAPI.WP.revisionFromHtml(request.responseText);
                if (!revision_id) {
                  failureFunc (request, new Error('Revision ID not found. Please reload the page.'));
                  return;
                }
                mw.config.set('wgCurRevisionId', revision_id); // Bump revision id!!
                LAPI.Ajax.removeSpinner(spinnerId);
                if (self.tooltip) self.tooltip.size_change();
                self.destroy();
              }
            , function (request, ex) {
                if (edit_page.isFake && (typeof edit_page.dispose === 'function'))
                  edit_page.dispose();
                failureFunc (request, ex);
              }
          );
        }
      , function (request, ex) {
          // Failure. What now? TODO: Implement some kind of user feedback.
          LAPI.Ajax.removeSpinner(spinnerId);
          if (self.tooltip) self.tooltip.size_change();
        }
    );

    return true;
  },

  destroy: function () {
    if (this.view) LAPI.DOM.removeNode(this.view);
    if (this.dummy) LAPI.DOM.removeNode(this.dummy);
    if (this.tooltip) this.tooltip.hide_now();
    if (this.model && this.model.id > 0 && this.viewer) this.viewer.deregister(this);
    this.model   = null;
    this.view    = null;
    this.content = null;
    this.tooltip = null;
    this.viewer  = null;
  },

  area: function () {
    if (!this.model || !this.model.dimension) return 0;
    return (this.model.dimension.w * this.model.dimension.h);
  },

  cannotEdit: function () {
    if (this.content && this.content.button_section) {
      LAPI.DOM.removeNode(this.content.button_section);
      this.content.button_section = null;
      if (this.tooltip) this.tooltip.size_change();
    }
  }

}; // end ImageAnnotation

var ImageAnnotationEditor = function () {this.initialize.apply(this, arguments);};

ImageAnnotationEditor.prototype =
{
  initialize: function () {
    var editor_width = 50;
    // Respect potential user-defined width setting
    if (   window.ImageAnnotationEditor_columns
        && !isNaN (window.ImageAnnotationEditor_columns)
        && window.ImageAnnotationEditor_columns >= 30
        && window.ImageAnnotationEditor_columns <= 100) {
      editor_width = window.ImageAnnotationEditor_columns;
    }
    this.editor =
      new LAPI.Edit(
          '' , editor_width, 6
        , { box: ImageAnnotator.UI.get('wpImageAnnotatorEditorLabel', false)
           ,preview: ImageAnnotator.UI.get('wpImageAnnotatorPreview', true).capitalizeFirst()
           ,save: ImageAnnotator.UI.get('wpImageAnnotatorSave', true).capitalizeFirst()
           ,revert: ImageAnnotator.UI.get('wpImageAnnotatorRevert', true).capitalizeFirst()
           ,cancel: ImageAnnotator.UI.get('wpImageAnnotatorCancel', true).capitalizeFirst()
           ,nullsave : ImageAnnotator_config.mayDelete()
                         ? ImageAnnotator.UI.get('wpImageAnnotatorDelete', true).capitalizeFirst()
                         : null
           ,post: ImageAnnotator.UI.get('wpImageAnnotatorCopyright', false)
          }
        , {
            onsave: this.save.bind(this)
           ,onpreview : this.onpreview.bind(this)
           ,oncancel: this.cancel.bind(this)
           ,ongettext: function (text) {
                          if (text == null) return '';
                          text = text.trim()
                                     .replace(/\{\{(\s*ImageNote(End)?\s*\|)/g, '&#x7B;&#x7B;$1')
                          ;
                          // Guard against people trying to break notes on purpose
                          if (text.length && typeof TextCleaner !== 'undefined')
                            text = TextCleaner.sanitizeWikiText(text, true);
                          return text;
                        }
          }
      );
    this.box = LAPI.make('div');
    this.box.appendChild(this.editor.getView());
    // Limit the width of the bounding box to the size of the textarea, taking into account the
    // tooltip styles. Do *not* simply append this.box or the editor view, Opera behaves strangely
    // if textboxes were ever hidden through a visibility setting! Use a second throw-away textbox
    // instead.
    var temp = LAPI.make('div', null, IA.tooltip_styles);
    temp.appendChild(LAPI.make('textarea', { cols : editor_width, rows : 6 }));
    Object.merge(
      {
        position: 'absolute',
        top: '0px',
        left: '-10000px',
        visibility: 'hidden'
      },
      temp.style
    );
    document.body.appendChild(temp);
    // Now we know how wide this textbox will be
    var box_width = temp.offsetWidth;
    LAPI.DOM.removeNode(temp);
    // Note: we need to use a tooltip with a dynamic content creator function here because
    // static content is cloned inside the Tooltip. Cloning on IE loses all attached handlers,
    // and thus the editor's controls wouldn't work anymore. (This is not a problem on FF3,
    // where cloning preserves the handlers.)
    this.tooltip = new Tooltip(
        IA.get_cover()
      , this.get_editor.bind(this)
      , { activate: Tooltip.NONE    // We'll always show it explicitly
         ,deactivate: Tooltip.ESCAPE
         ,close_button : null            // We have a cancel button anyway
         ,mode: Tooltip.FIXED
         ,anchor: Tooltip.TOP_LEFT
         ,mouse_offset : {x:10, y: 10, dx:1, dy:1} // Misuse this: fixed offset from view
         ,max_pixels: (box_width ? box_width + 20 : 0) // + 20 gives some slack
         ,z_index: 2010 // Above the cover.
         ,open_delay: 0
         ,hide_delay: 0
         ,onclose: this.close_tooltip.bind(this)
        }
      , IA.tooltip_styles
    );
    this.note = null;
    this.visible = false;
    LAPI.Evt.listenTo(this, this.tooltip.popup, IA.mouse_in,
      function (evt) {
        Array.forEach(IA.viewers, (function (viewer) {
            if (viewer != this.viewer && viewer.visible) viewer.hide();
        }).bind(this));
      }
    );
  },

  get_editor: function () {
    return this.box;
  },

  editNote: function (note) {
    var same_note = (note == this.note);
    this.note = note;
    this.viewer = this.note.viewer;

    var cover    = IA.get_cover();
    cover.style.cursor = 'auto';
    IA.show_cover();

    if (note.tooltip) note.tooltip.hide_now();

    IA.is_editing = true;
    if (note.content && !IA.wiki_read) {
      // Existing note, and we don't have the wikitext yet: go get it
      var self = this;
      LAPI.Ajax.apiGet(
          'query'
        , { prop: 'revisions'
           ,titles: mw.config.get('wgPageName')
           ,rvlimit: 1
           ,rvstartid : mw.config.get('wgCurRevisionId')
           ,rvprop: 'ids|content'
          }
        , function (request, json_result) {
            if (json_result && json_result.query && json_result.query.pages) {
              // Should have only one page here
              for (var page in json_result.query.pages) {
                var p = json_result.query.pages[page];
                if (p && p.revisions && p.revisions.length) {
                  var rev = p.revisions[0];
                  if (rev.revid == mw.config.get('wgCurRevisionId') && rev["*"] && rev["*"].length)
                    IA.setWikitext(rev["*"]);
                }
                break;
              }
            }
            // TODO: What upon a failure?
            self.open_editor(same_note, cover);
          }
        , function (request) {
            // TODO: What upon a failure?
            self.open_editor(same_note, cover);
          }
      );
    } else {
      this.open_editor(same_note, cover);
    }
  },

  open_editor: function (same_note, cover) {
    this.editor.hidePreview();
    if (!same_note || this.editor.textarea.readOnly)
      // Different note, or save error last time
      this.editor.setText(this.note.model.wiki);
    this.editor.enable
      (LAPI.Edit.SAVE + LAPI.Edit.PREVIEW + LAPI.Edit.REVERT + LAPI.Edit.CANCEL);
    this.editor.textarea.readOnly = false;
    this.editor.textarea.style.backgroundColor = 'white';
    // Set the position relative to the note's view.
    var view_pos = LAPI.Pos.position(this.note.view);
    var origin   = LAPI.Pos.position(cover);
    this.tooltip.options.fixed_offset.x =
      view_pos.x - origin.x + this.tooltip.options.mouse_offset.x;
    this.tooltip.options.fixed_offset.y =
      view_pos.y - origin.y + this.tooltip.options.mouse_offset.y;
    this.tooltip.options.fixed_offset.dx = 1;
    this.tooltip.options.fixed_offset.dy = 1;
    // Make sure mouse event listeners are removed, especially on IE.
    this.dim = { x : this.note.view.offsetLeft,  y : this.note.view.offsetTop
                ,w : this.note.view.offsetWidth, h : this.note.view.offsetHeight };
    this.viewer.setShowHideEvents(false);
    this.viewer.hide();       // Make sure notes are hidden
    this.viewer.toggle(true); // Show all note rectangles (but only the dummies)
    // Now show the editor
    this.tooltip.show_tip(null, false);
    var tpos = LAPI.Pos.position(this.editor.textarea);
    var ppos = LAPI.Pos.position(this.tooltip.popup);
    tpos = tpos.x - ppos.x;
    if (tpos + this.editor.textarea.offsetWidth > this.tooltip.popup.offsetWidth)
      this.editor.textarea.style.width = (this.tooltip.popup.offsetWidth - 2 * tpos) + 'px';
    if (LAPI.Browser.is_ie) {
      // Fixate textarea width to prevent ugly flicker on each keypress in IE6...
      this.editor.textarea.style.width = this.editor.textarea.offsetWidth + 'px';
    }
    this.visible = true;
  },

  hide_editor: function (evt) {
    if (!this.visible) return;
    this.visible = false;
    IA.is_editing = false;
    this.tooltip.hide_now(evt);
    if (evt && evt.type == 'keydown' && !this.saving) {
      // ESC pressed on new note before a save attempt
      this.cancel();
    }
    IA.hide_cover();
    this.viewer.setDefaultMsg();
    this.viewer.setShowHideEvents(true);
    this.viewer.hide();
    this.viewer.show(); // Make sure we get the real views again.
    // FIXME in Version 2.1: Unfortunately, we don't have a mouse position here, so sometimes we
    // may show the note rectangles even though the mouse is now outside the image. (It was
    // somewhere inside the editor in most cases (if an editor button was clicked), but if ESC was
    // pressed, it may actually be anywhere.)
  },

  save: function (editor) {
    var data = editor.getText();
    if (!data || !data.length) {
      // Empty text
      if (this.note.remove()) {
        this.hide_editor();
        this.cancel();
        this.note = null;
      } else {
        this.hide_editor();
        this.cancel();
      }
      return;
    } else if (data == this.note.model.wiki) {
      // Text unchanged
      this.hide_editor();
      this.cancel();
      return;
    }
    // Construct what to insert
    var dim = Object.clone(this.note.model.dimension);
    if (!dim) {
      dim = {
        x : Math.round(this.dim.x * this.viewer.factors.dx),
        y : Math.round(this.dim.y * this.viewer.factors.dy),
        w : Math.round(this.dim.w * this.viewer.factors.dx),
        h : Math.round(this.dim.h * this.viewer.factors.dy)
      };
      // Make sure everything is within bounds
      if (dim.x + dim.w > this.viewer.full_img.width) {
        if (dim.w > this.dim.w * this.viewer.factors.dx) {
          dim.w--;
          if (dim.x + dim.w > this.viewer.full_img.width) {
            if (dim.x > 0) dim.x--; else dim.w = this.viewer.full_img.width;
          }
        } else {
          // Width already was rounded down
          if (dim.x > 0) dim.x--;
        }
      }
      if (dim.y + dim.h > this.viewer.full_img.height) {
        if (dim.h > this.dim.h * this.viewer.factors.dy) {
          dim.h--;
          if (dim.y + dim.h > this.viewer.full_img.height) {
            if (dim.y > 0) dim.y--; else dim.h = this.viewer.full_img.height;
          }
        } else {
          // Height already was rounded down
          if (dim.y > 0) dim.y--;
        }
      }
      // If still too large, adjust width and height
      if (dim.x + dim.w > this.viewer.full_img.width) {
        if (this.viewer.full_img.width > dim.x) {
          dim.w = this.viewer.full_img.width - dim.x;
        } else {
          dim.x = this.viewer.full_img.width - 1;
          dim.w = 1;
        }
      }
      if (dim.y + dim.h > this.viewer.full_img.height) {
        if (this.viewer.full_img.height > dim.y) {
          dim.h = this.viewer.full_img.height - dim.y;
        } else {
          dim.y = this.viewer.full_img.height - 1;
          dim.h = 1;
        }
      }
    }
    this.to_insert =
        '{{ImageNote'
      + '|id=' + this.note.model.id
      + '|x=' + dim.x + '|y=' + dim.y + '|w=' + dim.w + '|h=' + dim.h
      + '|dimx=' + this.viewer.full_img.width
      + '|dimy=' + this.viewer.full_img.height
      + '|style=2'
      + '}}\n'
      + data + (data.endsWith('\n') ? '' : '\n')
      + '{{ImageNoteEnd|id=' + this.note.model.id + '}}';
    // Now edit the page
    var self = this;
    this.editor.busy(true);
    this.editor.enable(0); // Disable all buttons
    this.saving = true;
    LAPI.Ajax.editPage(
        mw.config.get('wgPageName')
      , function (doc, editForm, failureFunc, revision_id) {
          try {
            if (revision_id && revision_id != mw.config.get('wgCurRevisionId'))
              // Page was edited since the user loaded it.
              throw new Error('#Page version (revision ID) mismatch: edit conflict.');

            // Modify the page
            var textbox = editForm.wpTextbox1;
            if (!textbox) throw new Error('#Server replied with invalid edit page.');
            var pagetext = textbox.value;

            IA.setWikitext(pagetext);

            var span = null;
            if (self.note.content) // Otherwise it's a new note!
              span = IA.findNote(pagetext, self.note.model.id);
            if (span) { // Replace
              pagetext =
                  pagetext.substring(0, span.start)
                + self.to_insert
                + pagetext.substring(span.end)
                ;
            } else { // If not found, append
              // Try to append right after existing notes
              var lastNote = pagetext.lastIndexOf('{{ImageNoteEnd|id=');
              if (lastNote >= 0) {
                var endLastNote = pagetext.substring(lastNote).indexOf('}}');
                if (endLastNote < 0) {
                  endLastNote = pagetext.substring(lastNote).indexOf('\n');
                  if (endLastNote < 0) lastNote = -1; else lastNote += endLastNote;
                } else
                  lastNote += endLastNote + 2;
              }
              if (lastNote >= 0) {
                pagetext =
                    pagetext.substring(0, lastNote)
                  + '\n' + self.to_insert
                  + pagetext.substring(lastNote)
                  ;
              } else
                pagetext = pagetext.trimRight() + '\n' + self.to_insert;
            }
            textbox.value = pagetext;
            var summary = editForm.wpSummary;
            if (!summary)
              throw new Error('#Summary field not found. Check that edit pages have valid XHTML.');
            // If [[MediaWiki:Copyrightwarning]] is invalid XHTML, we may not have wpSummary!
            if (self.note.content != null) {
              IA.setSummary(
                  summary
                ,    ImageAnnotator.UI.get('wpImageAnnotatorChangeSummary', true)
                  || '[[MediaWiki talk:Gadget-ImageAnnotator.js|Changing image note]]$1'
                , data
              );
            } else {
              IA.setSummary(
                  summary
                ,    ImageAnnotator.UI.get('wpImageAnnotatorAddSummary', true)
                  || '[[MediaWiki talk:Gadget-ImageAnnotator.js|Přidání anotace]]$1'
                , data
              );
            }
          } catch (ex) {
            failureFunc (null, ex);
            return;
          }
          var edit_page = doc;
          LAPI.Ajax.submitEdit(
              editForm
            , function (request) {
                // After a successful submit.
                if (edit_page.isFake && (typeof edit_page.dispose === 'function'))
                  edit_page.dispose();
                // TODO: Actually, the edit got through here, so calling failureFunc on
                // inconsistencies isn't quite right. Should we reload the page?
                var id   = 'image_annotation_content_' + self.note.model.id;
                var doc  = LAPI.Ajax.getHTML(request, failureFunc, id);
                if (!doc) return;
                var html = LAPI.$ (id, doc);
                if (!html) {
                  if (doc.isFake && (typeof doc.dispose === 'function')) doc.dispose();
                  failureFunc
                    (request, new Error('#Note not found after saving. Please reload the page.'));
                  return;
                }
                var revision_id = LAPI.WP.revisionFromHtml(request.responseText);
                if (!revision_id) {
                  if (doc.isFake && (typeof doc.dispose === 'function')) doc.dispose();
                  failureFunc
                    (request, new Error('#Version inconsistency after saving. Please reload the page.'));
                  return;
                }
                mw.config.set('wgCurRevisionId', revision_id); // Bump revision id!!
                self.note.model.html = LAPI.DOM.importNode(document, html, true);
                if (doc.isFake && (typeof doc.dispose === 'function')) doc.dispose();
                self.note.model.dimension = dim; // record dimension
                self.note.model.html.style.display = '';
                self.note.model.wiki = data;
                self.editor.busy(false);
                if (self.note.content) {
                  LAPI.DOM.removeChildren(self.note.content.main);
                  self.note.content.main.appendChild(self.note.model.html);
                } else {
                  // New note.
                  self.note.display(); // Actually a misnomer. Just creates 'content'.
                  if (self.viewer.annotations.length > 1) {
                    self.viewer.annotations.sort(ImageAnnotation.compare);
                    var idxOfNote = Array.indexOf(self.viewer.annotations, self.note);
                    if (idxOfNote+1 < self.viewer.annotations.length)
                      LAPI.DOM.insertNode
                        (self.note.view, self.viewer.annotations [idxOfNote+1].view);
                  }
                }
                self.to_insert = null;
                self.saving = false;
                if (!self.note.tooltip) self.note.setTooltip();
                self.hide_editor();
                IA.is_editing = false;
                self.editor.setText(data); // In case the same note is re-opened: start new undo cycle
              }
            , function (request, ex) {
                if (edit_page.isFake && (typeof edit_page.dispose === 'function'))
                  edit_page.dispose();
                failureFunc (request, ex);
              }
          );
        }
      , function (request, ex) {
          self.editor.busy(false);
          self.saving = false;
          // TODO: How and where to display error if user closed editor through ESC (or through
          // opening another tooltip) in the meantime?
          if (!self.visible) return;
          // Change the tooltip to show the error.
          self.editor.setText(self.to_insert);
          // Error message. Use preview field for this.
          var error_msg = ImageAnnotator.UI.get('wpImageAnnotatorSaveError', false);
          var lk = getElementsByClassName(error_msg, 'span', 'wpImageAnnotatorOwnPageLink');
          if (lk && lk.length && lk[0].firstChild.nodeName.toLowerCase() === 'a') {
            lk = lk[0].firstChild;
            lk.href = mw.config.get('wgServer') + mw.config.get('wgArticlePath').replace('$1',
encodeURIComponent(mw.config.get('wgPageName'))) + '?action=edit';
          }
          if (ex) {
            var ex_msg = LAPI.formatException(ex, true);
            if (ex_msg) {
              ex_msg.style.borderBottom = '1px solid red';
              var tmp = LAPI.make('div');
              tmp.appendChild(ex_msg);
              tmp.appendChild(error_msg);
              error_msg = tmp;
            }
          }
          self.editor.setPreview(error_msg);
          self.editor.showPreview();
          self.editor.textarea.readOnly = true;
          // Force a light gray background, since IE has no visual readonly indication.
          self.editor.textarea.style.backgroundColor = '#EEEEEE';
          self.editor.enable(LAPI.Edit.CANCEL); // Disable all other buttons
        }
    );
  },

  onpreview: function (editor) {
    if (this.tooltip) this.tooltip.size_change();
  },

  cancel: function (editor) {
    if (!this.note) return;
    if (!this.note.content) {
      // No content: Cancel and remove this note!
      this.note.destroy();
      this.note = null;
    }
    if (editor) this.hide_editor();
  },

  close_tooltip: function (tooltip, evt) {
    this.hide_editor(evt);
    this.cancel();
  }

};

var ImageNotesViewer = function () {this.initialize.apply(this, arguments); };

ImageNotesViewer.prototype = {
  initialize: function (descriptor, may_edit) {
    Object.merge(descriptor, this);
    this.annotations = [];
    this.max_id      = 0;
    this.main_div    = null;
    this.msg         = null;
    this.may_edit    = may_edit;
    this.setup_done  = false;
    this.tip         = null;
    this.icon        = null;
    this.factors = {
      dx : this.full_img.width / this.thumb.width,
      dy : this.full_img.height / this.thumb.height
    };

    if (!this.isThumbnail && !this.isOther) {
      this.setup();
    } else {
      // Normalize the namespace of the realName to 'File' to account for images possibly stored at
      // a foreign repository (the Commons). Otherwise a later information load might fail because
      // the link is local and might actually be given as "Bild:Foo.jpg". If that page doesn't exist
      // locally, we want to ask at the Commons about "File:Foo.jpg". The Commons doesn't understand
      // the localized namespace names of other wikis, but the canonical namespace name 'File' works
      // also locally.
      this.realName = 'File:' + this.realName.substring(this.realName.indexOf(':') + 1);
    }
  },

  setup: function (onlyIcon) {
    this.setup_done = true;
    var name = this.realName;
    if (this.isThumbnail || this.scope == document || this.may_edit || !IA.haveAjax) {
      this.imgName  = this.realName;
      this.realName = '';
    } else {
      name = getElementsByClassName (this.scope, '*', 'wpImageAnnotatorFullName');
      this.realName = ((name && name.length) ? LAPI.DOM.getInnerText(name[0]) : '');
      this.imgName  = this.realName;
    }

    var annotations = getElementsByClassName (this.scope, 'div', IA.annotation_class);

    if (!this.may_edit && (!annotations || annotations.length === 0))
      return; // Nothing to do

    // A div inserted around the image. It ensures that everything we add is positioned properly
    // over the image, even if the browser window size changes and re-layouts occur.
    var isEnabledImage = LAPI.DOM.hasClass(this.scope, 'wpImageAnnotatorEnable');
    if (!this.isThumbnail && !this.isOther && !isEnabledImage) {
      this.img_div =
        LAPI.make('div', null, {position: 'relative', width: '' + this.thumb.width + 'px'});
      var floater =
        LAPI.make(
            'div', null
          , { cssFloat: (IA.is_rtl ? 'right' : 'left')
             ,styleFloat: (IA.is_rtl ? 'right' : 'left') // For IE...
             ,width: '' + this.thumb.width + 'px'
             ,position: 'relative' // Fixes IE layout bugs...
            }
        );
      floater.appendChild(this.img_div);
      this.img.parentNode.parentNode.insertBefore(floater, this.img.parentNode);
      this.img_div.appendChild(this.img.parentNode);
      // And now a clear:left to make the rest appear below the image, as usual.
      var breaker = LAPI.make('div', null, {clear: (IA.is_rtl ? 'right' : 'left')});
      LAPI.DOM.insertAfter(breaker, floater);
      // Remove spurious br tag.
      if (breaker.nextSibling && breaker.nextSibling.nodeName.toLowerCase() == 'br')
        LAPI.DOM.removeNode(breaker.nextSibling);
    } else if (this.isOther || isEnabledImage) {
      this.img_div =
        LAPI.make('div', null, {position: 'relative', width: '' + this.thumb.width + 'px'});
      this.img.parentNode.parentNode.insertBefore(this.img_div, this.img.parentNode);
      this.img_div.appendChild(this.img.parentNode);
      // Insert one more to have a file_div, so that we can align the message text correctly
      this.file_div = LAPI.make('div', null, {width: '' + this.thumb.width + 'px'});
      this.img_div.parentNode.insertBefore(this.file_div, this.img_div);
      this.file_div.appendChild(this.img_div);
    } else { // Thumbnail
      this.img_div =
        LAPI.make(
            'div'
          , {className: 'thumbimage'}
          , {position: 'relative', width: '' + this.thumb.width + 'px'}
        );
      this.img.parentNode.parentNode.insertBefore(this.img_div, this.img.parentNode);
      this.img.style.border = 'none';
      this.img_div.appendChild(this.img.parentNode);
    }
    if (   (this.isThumbnail || this.isOther) && !this.may_edit
        && (   onlyIcon
            || this.iconOnly
            || ImageAnnotator_config.inlineImageUsesIndicator
                 (name, this.isLocal, this.thumb, this.full_img, annotations.length, this.isThumbnail)
           )
       )
    {
      // Use an onclick handler instead of a link around the image. The link may have a default white
      // background, but we want to be sure to have transparency. The image should be an 8-bit indexed
      // PNG or a GIF and have a transparent background.
      this.icon = ImageAnnotator.UI.get('wpImageAnnotatorIndicatorIcon', false);
      if (this.icon) this.icon = this.icon.firstChild; // Skip the message container span or div
      // Guard against misconfigurations
      if (   this.icon
          && this.icon.nodeName.toLowerCase() == 'a'
          && this.icon.firstChild.nodeName.toLowerCase() == 'img'
         )
      {
        // Make sure we use the right protocol:
        var srcFixed = this.icon.firstChild.getAttribute('src', 2).replace(/^https?\:/, document.location.protocol);
        this.icon.firstChild.src = srcFixed;
        this.icon.firstChild.title = this.icon.title;
        this.icon = this.icon.firstChild;
      } else if (!this.icon || this.icon.nodeName.toLowerCase() !== 'img') {
        this.icon =
          LAPI.DOM.makeImage(
              IA.indication_icon
            , 14, 14
            , ImageAnnotator.UI.get('wpImageAnnotatorHasNotesMsg', true) || ''
          );
      }
      Object.merge(
          {position: 'absolute', zIndex: 1000, top: '0px', cursor: 'pointer'}
        , this.icon.style
      );
      this.icon.onclick = (function () { location.href = this.img.parentNode.href; }).bind(this);
      if (IA.is_rtl)
        this.icon.style.right = '0px';
      else
        this.icon.style.left = '0px';
      this.img_div.appendChild(this.icon);
      // And done. We just show the icon, no fancy event handling needed.
      return;
    }
    // Set colors
    var colors = IA.getRawItem('colors', this.scope);
    this.outer_border  =
      colors && IA.getItem('outer',  colors) || IA.outer_border;
    this.inner_border  =
      colors && IA.getItem('inner',  colors) || IA.inner_border;
    this.active_border =
      colors && IA.getItem('active', colors) || IA.active_border;
    if (annotations) {
      for (var i = 0; i < annotations.length; i++) {
        var id = annotations[i].id;
        if (id && /^image_annotation_note_(\d+)$/.test(id)) {
          id = parseInt (id.substring('image_annotation_note_'.length));
        } else
          id = null;
        if (id) {
          if (id > this.max_id) this.max_id = id;
          var w = IA.getIntItem('full_width_'  + id, this.scope);
          var h = IA.getIntItem('full_height_' + id, this.scope);
          if (   w == this.full_img.width && h == this.full_img.height
              && !Array.exists(this.annotations, function (note) { return note.model.id == id; })
             )
          {
            try {
              this.register(new ImageAnnotation(annotations[i], this, id));
            } catch (ex) {
              // Swallow.
            }
          }
        }
      }
    }
    if (this.annotations.length > 1) this.annotations.sort(ImageAnnotation.compare);
    // Add the rectangles of existing notes to the DOM now that they are sorted.
    Array.forEach(this.annotations, (function (note) {this.img_div.appendChild(note.view);}).bind(this));
    if (this.isThumbnail) {
      this.main_div = getElementsByClassName (this.file_div, 'div', 'thumbcaption');
      if (!this.main_div || this.main_div.length == 0)
        this.main_div = null;
      else
        this.main_div = this.main_div[0];
    }
    if (!this.main_div) {
      this.main_div = LAPI.make('div');
      if (IA.is_rtl) {
        this.main_div.style.direction = 'rtl';
        this.main_div.style.textAlign = 'right';
        this.main_div.className = 'rtl';
      } else {
        this.main_div.style.textAlign = 'left';
      }
      if (!this.isThumbnail && !this.isOther && !isEnabledImage) {
        LAPI.DOM.insertAfter(this.main_div, this.file_div);
      } else {
        LAPI.DOM.insertAfter(this.main_div, this.img_div);
      }
    }
    if (   !(this.isThumbnail || this.isOther)
        ||    !this.noCaption
           && !IA.hideCaptions
           && ImageAnnotator_config.displayCaptionInArticles
                (name, this.isLocal, this.thumb, this.full_img, annotations.length, this.isThumbnail)
       )
    {
      this.msg = LAPI.make('div', null, {display: 'none'});
      if (IA.is_rtl) {
        this.msg.style.direction = 'rtl';
        this.msg.className = 'rtl';
      }
      if (this.isThumbnail) this.msg.style.fontSize = '90%';
      this.main_div.appendChild(this.msg);
    }

    // Set overflow parents, if any

    var simple   = !!window.getComputedStyle;
    var checks   = (simple ? ['overflow', 'overflow-x', 'overflow-y']
                           : ['overflow', 'overflowX', 'overflowY']
                   );
    var curStyle = null;
    for (var up = this.img.parentNode.parentNode; up != document.body; up = up.parentNode) {
      curStyle = (simple ? window.getComputedStyle(up, null) : (up.currentStyle || up.style));
      // "up.style" is actually incorrect, but a best-effort fallback.
      var overflow =
        Array.any(checks, function (t) {
          var o = curStyle[t];
          return (o && o != 'visible') ? o : null;
        });
      if (overflow) {
        if (!this.overflowParents)
          this.overflowParents = [up];
        else
          this.overflowParents[this.overflowParents.length] = up;
      }
    }
    /*
    if (this.overflowParents && this.may_edit) {
      // Forbid editing if we have such a crazy layout.
      console.log('overflowParents');
      this.may_edit = false;
      IA.may_edit = false;
    }
    */
    this.show_evt = LAPI.Evt.makeListener(this, this.show);
    if (this.overflowParents || LAPI.Browser.is_ie) {
      // If we have overflowParents, also use a mousemove listener to show/hide the whole
      // view (FF doesn't send mouseout events if the visible border is still within the image, i.e.,
      // if not the whole image is visible). On IE, also use this handler to show/hide the notes
      // if we're still within the visible area of the image. IE passes through mouse_over events to
      // the img even if the mouse is within a note's rectangle. Apparently is doesn't handle
      // transparent divs correctly. As a result, the notes will pop up and disappear only when the
      // mouse crosses the border, and if one moves the mouse a little fast across the border, we
      // don't get any event at all. That's no good.
      this.move_evt = LAPI.Evt.makeListener(this, this.check_hide);
    } else
      this.hide_evt = LAPI.Evt.makeListener(this, this.hide);
    this.move_listening = false;
    this.setShowHideEvents(true);
    this.visible = false;
    this.setDefaultMsg();
  },

  cannotEdit: function () {
    if (!this.may_edit) return;
    this.may_edit = false;
    Array.forEach(this.annotations, function (note) { note.cannotEdit(); });
  },

  setShowHideEvents: function (set) {
    if (this.icon) return;
    if (set) {
      LAPI.Evt.attach(this.img, IA.mouse_in, this.show_evt);
      if (this.hide_evt) LAPI.Evt.attach(this.img, IA.mouse_out, this.hide_evt);
    } else {
      LAPI.Evt.remove(this.img, IA.mouse_in, this.show_evt);
      if (this.hide_evt) {
        LAPI.Evt.remove(this.img, IA.mouse_out, this.hide_evt);
      } else if (this.move_listening)
        this.removeMoveListener();
    }
  },

  removeMoveListener: function () {
    if (this.icon) return;
    this.move_listening = false;
    if (this.move_evt) {
      if (!LAPI.Browser.is_ie && typeof document.captureEvents === 'function')
        document.captureEvents(null);
      LAPI.Evt.remove(document, 'mousemove', this.move_evt, true);
    }
  },

  adjustRectangleSize: function (node) {
    if (this.icon) return;
    // Make sure the note boxes don't overlap the image boundary; we might get an event
    // loop otherwise if the mouse was just on that overlapped boundary, resulting in flickering.
    var view_x = node.offsetLeft;
    var view_y = node.offsetTop;
    var view_w = node.offsetWidth;
    var view_h = node.offsetHeight;
    if (view_x === 0) view_x = 1;
    if (view_y === 0) view_y = 1;
    if (view_x + view_w >= this.thumb.width) {
      view_w = this.thumb.width - view_x - 1;
```

```
        if (view_w <= 4) { view_w = 4; view_x = this.thumb.width - view_w - 1;}
      }
      if (view_y + view_h >= this.thumb.height) {
        view_h = this.thumb.height - view_y - 1;
        if (view_h <= 4) { view_h = 4; view_y = this.thumb.height - view_h - 1;}
      }
      // Now set position and width and height, subtracting cumulated border widths
      if (   view_x != node.offsetLeft || view_y != node.offsetTop
          || view_w != node.offsetWidth || view_h != node.offsetHeight)
      {
        node.style.top = '' + view_y + 'px';
        node.style.left = '' + view_x + 'px';
        node.style.width = '' + (view_w - 2) + 'px';
        node.style.height = '' + (view_h - 2) + 'px';
        node.firstChild.style.width = '' + (view_w - 4) + 'px';
        node.firstChild.style.height = '' + (view_h - 4) + 'px';
      }
    },

    toggle: function (dummies) {
      var i;
      if (!this.annotations || this.annotations.length === 0 || this.icon) return;
      if (dummies) {
        for (i = 0; i < this.annotations.length; i++) {
          this.annotations[i].view.style.display = 'none';
          if (this.visible && this.annotations[i].tooltip)
            this.annotations[i].tooltip.hide_now(null);
          this.annotations[i].dummy.style.display = (this.visible ? 'none' : '');
          if (!this.visible) this.adjustRectangleSize(this.annotations[i].dummy);
        }
      } else {
        for (i = 0; i < this.annotations.length; i++) {
          this.annotations[i].dummy.style.display = 'none';
          this.annotations[i].view.style.display = (this.visible ? 'none' : '');
          if (!this.visible) this.adjustRectangleSize(this.annotations[i].view);
          if (this.visible && this.annotations[i].tooltip)
            this.annotations[i].tooltip.hide_now(null);
        }
      }
      this.visible = !this.visible;
    },

    show: function (evt) {
      if (this.visible || this.icon) return;
      this.toggle(IA.is_adding || IA.is_editing);
      if (this.move_evt && !this.move_listening) {
        LAPI.Evt.attach(document, 'mousemove', this.move_evt, true);
        this.move_listening = true;
        if (!LAPI.Browser.is_ie && typeof document.captureEvents === 'function')
          document.captureEvents(Event.MOUSEMOVE);
      }
    },

    hide: function (evt) {
      if (this.icon) return true;
      if (!this.visible) {
        // Huh?
        if (this.move_listening) this.removeMoveListener();
        return true;
      }
      if (evt) {
        var mouse_pos = LAPI.Pos.mousePosition(evt);
        if (mouse_pos) {
          if (this.tip) {
            // Check whether we're within the visible note.
            if (LAPI.Pos.isWithin(this.tip.popup, mouse_pos.x, mouse_pos.y)) return true;
          }
          var is_within = true;
          var img_pos = LAPI.Pos.position(this.img);
          var rect = {
            x: img_pos.x,
            y: img_pos.y,
            r: (img_pos.x + this.img.offsetWidth),
            b: (img_pos.y + this.img.offsetHeight)
          };
          var i;
          if (this.overflowParents) {
            // We're within some elements having overflow:hidden or overflow:auto or overflow:scroll set.
            // Compute the actually visible region by intersecting the rectangle given by img_pos and
            // this.img.offsetWidth, this.img.offsetTop with the rectangles of all overflow parents.

            function intersect_rectangles (a, b) {
              if (b.x > a.r || b.r < a.x || b.y > a.b || b.b < a.y)
                return { x:0, y:0, r:0, b:0 };

              return {
                x: Math.max(a.x, b.x),
                y: Math.max(a.y, b.y),
                r: Math.min(a.r, b.r),
                b: Math.min(a.b, b.b)
              };
            }

            for (i = 0; i < this.overflowParents.length && rect.x < rect.r && rect.y < rect.b; i++) {
              img_pos   = LAPI.Pos.position(this.overflowParents[i]);
              img_pos.r = img_pos.x + this.overflowParents[i].clientWidth;
              img_pos.b = img_pos.y + this.overflowParents[i].clientHeight;
              rect = intersect_rectangles (rect, img_pos);
            }
          }

          is_within = !(   rect.x >= rect.r || rect.y >= rect.b // Empty rectangle
                        || rect.x >= mouse_pos.x || rect.r <= mouse_pos.x
                        || rect.y >= mouse_pos.y || rect.b <= mouse_pos.y
          );
          if (is_within) {
            if (LAPI.Browser.is_ie && evt.type === 'mousemove') {
              var display;
              // Loop in reverse order to properly display top rectangle's note!
            for (i = this.annotations.length - 1; i >= 0; i--) {
              display = this.annotations[i].view.style.display;
              if (   display !== 'none' && display != null
                  && LAPI.Pos.isWithin(this.annotations[i].view.firstChild, mouse_pos.x, mouse_pos.y)
                 )
              {
                if (!this.annotations[i].tooltip.visible) this.annotations[i].tooltip.show(evt);
                return true;
              }
            }
            if (this.tip) this.tip.hide_now(); // Inside the image, but not within any note rectangle
          }
          return true;
        }
      }
    }
    // Not within the image, or forced hiding (no event)
    if (this.move_listening) this.removeMoveListener();
    this.toggle(IA.is_adding || IA.is_editing);
    return true;
  },

  check_hide: function (evt) {
    if (this.icon) return true;
    if (this.visible)
      this.hide(evt);
    return true;
  },

  register: function (new_note) {
    this.annotations[this.annotations.length] = new_note;
    if (new_note.model.id > 0) {
      if (new_note.model.id > this.max_id) this.max_id = new_note.model.id;
    } else {
      new_note.model.id = ++this.max_id;
    }
  },

  deregister: function (note) {
    Array.remove(this.annotations, note);
    if (note.model.id == this.max_id) this.max_id--;
    if (this.annotations.length === 0) this.setDefaultMsg(); //If we removed the last one, clear the msg
  },

  setDefaultMsg: function () {
    if (this.annotations && this.annotations.length && this.msg) {
      LAPI.DOM.removeChildren(this.msg);
      this.msg.appendChild
        (ImageAnnotator.UI.get('wpImageAnnotatorHasNotesMsg', false));
      if (this.realName && typeof this.realName === 'string' && this.realName.length) {
        var otherPageMsg = ImageAnnotator.UI.get('wpImageAnnotatorEditNotesMsg', false);
        if (otherPageMsg) {
          var lk = otherPageMsg.getElementsByTagName('a');
          if (lk && lk.length) {
            lk = lk[0];
            lk.parentNode.replaceChild(
                LAPI.DOM.makeLink(
                    mw.config.get('wgArticlePath').replace('$1', encodeURIComponent(this.realName))
                  , this.realName
                  , this.realName
                )
              , lk
            );
            this.msg.appendChild(otherPageMsg);
          }
        }
      }
      this.msg.style.display = '';
    } else {
      if (this.msg) this.msg.style.display = 'none';
    }
    if (IA.button_div && this.may_edit) IA.button_div.style.display = '';
  }

};

var IA = {
  // This object is responsible for setting up annotations when a page is loaded. It loads all
  // annotations in the page source, and adds an "Annotate this image" button plus the support
  // for drawing rectangles onto the image if there is only one image and editing is allowed.

  haveAjax: false,

  button_div: null,
  add_button: null,

  cover: null,
  border: null,
  definer: null,

  mouse_in: (window.ActiveXObject ? 'mouseenter' : 'mouseover'),
  mouse_out: (window.ActiveXObject ? 'mouseleave' : 'mouseout'),

  annotation_class : 'image_annotation',

  // Format of notes in Wikitext. Note: there are two formats, an old one and a new one.
  // We only write the newest (last) one, but we can read also the older formats. Order is
  // important, because the old format also used the ImageNote template, but for a different
  // purpose.
  note_delim       :
  [
    { start: '<div id="image_annotation_note_$1"'
     ,end: '</div><!-- End of annotation $1-->'
     ,content_start : '<div id="image_annotation_content_$1">\n'
     ,content_end: '</div>\n<span id="image_annotation_wikitext_$1"'
    }
   ,{ start: '{{ImageNote|id=$1'
     ,end: '{{ImageNoteEnd|id=$1}}'
     ,content_start : '}}\n'
```

```
      ,content_end: '{{ImageNoteEnd|id=$1}}'
    }
  ],

  tooltip_styles : // The style for all our tooltips
   {  border: '1px solid #8888aa'
    , backgroundColor : '#ffffe0'
    , padding: '0.3em'
    , fontSize: ((mw.config.get('skin') == 'monobook' || mw.config.get('skin') == 'modern') ? '127%' : '100%')
      // Scale up to default text size
   },

  editor: null,

  wiki_read: false,
  is_rtl: false,

  move_listening : false,
  is_tracking: false,
  is_adding: false,
  is_editing: false,

  zoom_threshold : 8.0,
  zoom_factor: 4.0,

  install_attempts: 0,
  max_install_attempts : 10, // Maximum 5 seconds

  imgs_with_notes : [],
  thumbs: [],
  other_images: [],

  // Fallback
  indication_icon : '//upload.wikimedia.org/wikipedia/commons/8/8a/Gtk-dialog-info-14px.png',

  install: function (config) {
    if (typeof ImageAnnotator_disable !== 'undefined' && !!ImageAnnotator_disable) return;
    if (!config || ImageAnnotator_config != null) return;

    // Double check.
    if (!config.viewingEnabled()) return;

    var self = IA;
    ImageAnnotator_config = config;

    // Determine whether we have XmlHttp. We try to determine this here to be able to avoid
    // doing too much work.
    if (   window.XMLHttpRequest
        && typeof LAPI !== 'undefined'
        && typeof LAPI.Ajax !== 'undefined'
        && typeof LAPI.Ajax.getRequest !== 'undefined'
       )
    {
      self.haveAjax    = (LAPI.Ajax.getRequest() != null);
      self.ajaxQueried = true;
    } else {
      self.haveAjax    = true; // A pity. May occur on IE. We'll check again later on.
      self.ajaxQueried = false;
    }

    // We'll include self.haveAjax later on once more, to catch the !ajaxQueried case.
    self.may_edit = mw.config.get('wgNamespaceNumber') >= 0 && mw.config.get('wgArticleId') > 0 && self.haveAjax &&
config.editingEnabled();

    function namespaceCheck (list)
    {
      if (!list || Object.prototype.toString.call(list) !== '[object Array]') return false;
      var namespaceIds = mw.config.get('wgNamespaceIds');
      if(!namespaceIds) return false;
      var namespaceNumber = mw.config.get('wgNamespaceNumber');
      for (var i = 0; i < list.length; i++) {
        if (typeof list[i] === 'string'
            && (list[i] === '*'
                || namespaceIds[list[i].toLowerCase().replace(/ /g, '_')] === namespaceNumber
               )
           )
          return true;
      }
      return false;
    }

    self.rules = { inline: {}, thumbs: {}, shared : {} };

    // Now set the default rules. Undefined means default setting (true for show, false for icon),
    // but overrideable by per-image rules. If set, it's not overrideable by per-image rules.
    //
    if (   !self.haveAjax
        || !config.generalImagesEnabled()
        || namespaceCheck (window.ImageAnnotator_no_images || null)
       )
    {
      self.rules.inline.show = false;
      self.rules.thumbs.show = false;
      self.rules.shared.show = false;
    } else {
      if (   !self.haveAjax
          || !config.thumbsEnabled()
          || namespaceCheck(window.ImageAnnotator_no_thumbs || null)
         )
      {
        self.rules.thumbs.show = false;
      }
      if (mw.config.get('wgNamespaceNumber') == 6)
        self.rules.shared.show = true;
      else if (   !config.sharedImagesEnabled()
               || namespaceCheck(window.ImageAnnotator_no_shared || null)
              )
      {
        self.rules.shared.show = false;
    }
    if (namespaceCheck(window.ImageAnnotator_icon_images || null))
      self.rules.inline.icon = true;
    if (namespaceCheck(window.ImageAnnotator_icon_thumbs || null))
      self.rules.thumbs.icon = true;
  }

  // User rule for displaying captions on images in articles
  self.hideCaptions = namespaceCheck(window.ImageAnnotator_hide_captions || null);

  var do_images = typeof self.rules.inline.show === 'undefined' || self.rules.inline.show;

  if (do_images) {
    // Per-article switching off of note display on inline images and thumbnails
    var rules = document.getElementById('wpImageAnnotatorImageRules');
    if (rules) {
      if (rules.className.indexOf('wpImageAnnotatorNone') >= 0) {
        self.rules.inline.show = false;
        self.rules.thumbs.show = false;
        self.rules.shared.show = false;
      }
      if (   typeof self.rules.inline.show === 'undefined'
          && rules.className.indexOf('wpImageAnnotatorDisplay') >= 0
         )
      {
        self.rules.inline.show = true;
      }
      if (rules.className.indexOf('wpImageAnnotatorNoThumbDisplay') >= 0) {
        self.rules.thumbs.show = false;
      }
      if (   typeof self.rules.thumbs.show === 'undefined'
          && rules.className.indexOf('wpImageAnnotatorThumbDisplay') >= 0
         )
      {
        self.rules.thumbs.show = true;
      }
      if (rules.className.indexOf('wpImageAnnotatorInlineDisplayIcons') >= 0) {
        self.rules.inline.icon = true;
      }
      if (rules.className.indexOf('wpImageAnnotatorThumbDisplayIcons') >= 0) {
        self.rules.thumbs.icon = true;
      }
      if (rules.className.indexOf('wpImageAnnotatorOnlyLocal') >= 0)
      {
        self.rules.shared.show = false;
      }
    }
  }

  // Make sure the shared value is set
  self.rules.shared.show =
    typeof self.rules.shared.show === 'undefined' || self.rules.shared.show;

  do_images = typeof self.rules.inline.show === 'undefined' || self.rules.inline.show;
  var do_thumbs = typeof self.rules.thumbs.show === 'undefined' || self.rules.thumbs.show;

  if (do_images) {
    var bodyContent =    document.getElementById('content')          // template-based skins
                      || document.getElementById('bodyContent')      // monobook, vector
                      || document.getElementById('mw_contentholder') // modern
                      || document.getElementById('article')          // old skins
    ;
    if (bodyContent) {
      var all_imgs = bodyContent.getElementsByTagName('img');

      // This prevents traversing a page with more than 400 images
      // There are extreme cases like [[Emoji]] that high number of images can cause
      // huge lag specially on Chrome
      if (all_imgs.length > 400) {
        // purging the array, simply a hack to avoid more indention
        all_imgs = [];
      }

      for (var i = 0; i < all_imgs.length; i++) {
        // Exclude all that are in img_with_notes or in thumbs. Also exclude all in galleries.
        var up = all_imgs[i].parentNode;
        if (up.nodeName.toLowerCase() !== 'a') continue;
        up = up.parentNode;
        if ((' ' + up.className + ' ').indexOf(' wpImageAnnotatorOff ') >= 0) continue;
        if ((' ' + up.className + ' ').indexOf(' thumbinner ') >= 0) {
          if (do_thumbs) self.thumbs[self.thumbs.length] = up;
          continue;
        }
        up = up.parentNode;
        if (!up) continue;
        if ((' ' + up.className + ' ').indexOf(' wpImageAnnotatorOff ') >= 0) continue;
        if ((' ' + up.className + ' ').indexOf(' wpImageAnnotatorEnable ') >= 0) {
          self.imgs_with_notes[self.imgs_with_notes.length] = up;
          continue;
        }
        up = up.parentNode;
        if (!up) continue;
        // Other images not in galleries
        if ((' ' + up.className + ' ').indexOf(' wpImageAnnotatorOff ') >= 0) continue;
        if ((' ' + up.className + ' ').indexOf(' gallerybox ') >= 0) continue;
        if ((' ' + up.className + ' ').indexOf(' wpImageAnnotatorEnable ') >= 0) {
          self.imgs_with_notes[self.imgs_with_notes.length] = up;
          continue;
        }
        up = up.parentNode;
        if (!up) continue;
        if ((' ' + up.className + ' ').indexOf(' wpImageAnnotatorOff ') >= 0) continue;
        if ((' ' + up.className + ' ').indexOf(' gallerybox ') >= 0) continue;
        if ((' ' + up.className + ' ').indexOf(' wpImageAnnotatorEnable ') >= 0) {
          self.imgs_with_notes[self.imgs_with_notes.length] = up;
        } else {
          // Guard against other scripts adding aribtrary numbers of divs (dshuf for instance!)
          var is_other = true;
          while (up && up.nodeName.toLowerCase() == 'div' && is_other) {
              up = up.parentNode;
              if (up) is_other = (' ' + up.className + ' ').indexOf(' gallerybox ') < 0;
            }
            if (is_other) self.other_images[self.other_images.length] = all_imgs[i];
          }
        } // end loop
      }
    } else {
      self.imgs_with_notes = getElementsByClassName (document, '*', 'wpImageAnnotatorEnable');
      if (do_thumbs)
        self.thumbs = getElementsByClassName (document, 'div', 'thumbinner'); // No galleries!
    }
    if (   mw.config.get('wgNamespaceNumber') == 6
        || (self.imgs_with_notes.length)
        || (self.thumbs.length)
        || (self.other_images.length)
       )
    {
      // Publish parts of config.
      ImageAnnotator.UI  = config.UI;
      self.outer_border  = config.outer_border;
      self.inner_border  = config.inner_border;
      self.active_border = config.active_border;
      self.new_border    = config.new_border;
      self.wait_for_required_libraries();
    }
  },

  wait_for_required_libraries: function () {
    if (typeof Tooltip == 'undefined' || typeof LAPI == 'undefined') {
      if (IA.install_attempts++ < IA.max_install_attempts) {
        setTimeout(IA.wait_for_required_libraries, 500); // 0.5 sec.
      }
      return;
    }
    if (LAPI.Browser.is_opera && !LAPI.Browser.is_opera_ge_9) return; // Opera 8 has severe problems
    // Get the UI. We're likely to need it if we made it to here.
    IA.setup_ui();
    IA.setup();
  },

  setup: function () {
    var self = IA;
    self.imgs = [];

    // Catch both native RTL and "faked" RTL through [[MediaWiki:Rtl.js]]
    self.is_rtl       =
         LAPI.DOM.hasClass(document.body, 'rtl')
      || (   LAPI.DOM.currentStyle // Paranoia: added recently, not everyone might have it
          && LAPI.DOM.currentStyle(document.body, 'direction') == 'rtl'
         )
    ;

    var stylepath = mw.config.get('stylepath') || '/skin';

    // Use this to temporarily display an image off-screen to get its dimensions
    var testImgDiv =
      LAPI.make('div', null,
        {  display: 'none', position: 'absolute', width: '300px'
         , overflow: 'hidden', overflowX: 'hidden', left: '-10000px'
        }
      );
    document.body.insertBefore(testImgDiv, document.body.firstChild);

    function img_check (img, is_other)
    {
      var srcW = parseInt (img.getAttribute('width', 2), 10);
      var srcH = parseInt (img.getAttribute('height', 2), 10);
      // Don't do anything on extremely small previews. We need some minimum extent to be able to place
      // rectangles after all...
      if (!srcW || !srcH || srcW < 20 || srcH < 20) return null;
      // For non-thumbnail images, the size limit is larger.
      if (is_other && (srcW < 60 || srcH < 60)) return null;
      var w = img.clientWidth;  // Don't use offsetWidth, thumbnails may have a boundary...
      var h = img.clientHeight;
      // If the image is currently hidden, its clientWidth and clientHeight are not set. Try
      // harder to get the true width and height:
      if ((!w || !h) && img.style.display != 'none') {
        var copied = img.cloneNode(true);
        copied.style.display = '';
        testImgDiv.appendChild(copied);
        testImgDiv.style.display = '';
        w = copied.clientWidth;
        h = copied.clientHeight;
        testImgDiv.style.display = 'none';
        LAPI.DOM.removeNode(copied);
      }
      // Quit if the image wasn't loaded properly for some reason:
      if (w != srcW || h != srcH) return null;
      // Exclude system images
      if (img.src.contains(stylepath)) return null;
      // Only if within a link
      if (img.parentNode.nodeName.toLowerCase() != 'a') return null;
      if (is_other) {
        // Only if the img-within-link construction is within some element that may contain a div!
        if (/^(p|span)$/i.test(img.parentNode.parentNode.nodeName)) {
          // Special case: a paragraph may contain only inline elements, but we want to be able to handle
          // files in single paragraphs. Maybe we need to properly split the paragraph and wrap the image
          // in a div, but for now we assume that all browsers can handle a div within a paragraph or
          // a span in a meaningful way, even if that is not really allowed.
        } else if
(!/^(object|applet|map|fieldset|noscript|iframe|body|div|li|dd|blockquote|center|ins|del|button|th|td|form)$/i.test(img.parentNode.pa
rentNode.nodeName))
          return null;
      }
      // Exclude any that are within an image note!
      var up = img.parentNode.parentNode;
      while (up != document.body) {
        if (LAPI.DOM.hasClass(up, IA.annotation_class)) return null;
        up = up.parentNode;
      }
      return {width: w, height: h};
    }

    function setup_one (scope) {
      var file_div = scope;
      var is_thumb =
          scope != document
       && scope.nodeName.toLowerCase() == 'div'
       && LAPI.DOM.hasClass(scope, 'thumbinner')
      ;
      var is_other = scope.nodeName.toLowerCase() == 'img';
      if (is_other && self.imgs.length && scope == self.imgs[0]) return null;
      if (scope == document) {
        file_div = LAPI.$('file');
      } else if (!is_thumb && !is_other) {
        file_div = getElementsByClassName(scope, 'div', 'wpImageAnnotatorFile');
        if (!file_div || file_div.length != 1) return null;
        file_div = file_div[0];
      }
      if (!file_div) return null;
      var img = null;
      if (scope == document) {
        img = LAPI.WP.getPreviewImage(mw.config.get('wgTitle'));
        // TIFFs may be multi-paged: allow only for single-page TIFFs
        if ( document.pageselector ) img = null;
      } else if (is_other) {
        img = scope;
      } else {
        img = file_div.getElementsByTagName('img');
        if (!img || img.length === 0) return null;
        img = img[0];
      }
      if (!img) return null;
      var dim = img_check (img, is_other);
      if (!dim) return null;
      // Conditionally exclude shared images.
      if (   scope != document
          && !self.rules.shared.show
          && ImageAnnotator_config.imageIsFromSharedRepository(img.src)
         )
        return null;
      var name = null;
      if (scope == document) {
        name = mw.config.get('wgPageName');
      } else {
        name = LAPI.WP.pageFromLink(img.parentNode);
        if (!name) return null;
        name = name.replace(/ /g, '_');
        if (is_thumb || is_other) {
          var img_src = decodeURIComponent(img.getAttribute('src', 2)).replace(/ /g, '_');
          // img_src should have a component "/name" in there somewhere
          var colon = name.indexOf(':');
          if (colon <= 0) return null;
          var img_name = name.substring(colon + 1);

          //if (img_src.search(new RegExp('/' + img_name.escapeRE() + '(/.*)?$')) < 0 ) // oprava jmarti pro $wgThumbnailScriptPath
          if (img_src.search(new RegExp('/' + img_name.escapeRE() + '(/.*)?$')) < 0 && img_src.search(new RegExp('/thumb\.php\\?f=' +
img_name.escapeRE() + '&width=[0-9]+$') ) < 0)
            return null;
          // If the link is not going to file namespace, we won't find the full size later on and
          // thus we won't do anything with it.
        }
      }
      if (name.search(/\.(jpe?g|png|gif|svg|tiff?)$/i) < 0) return null; // Only PNG, JPE?G, GIF, SVG, and TIFF?
      // Finally check for wpImageAnnotatorControl
      var icon_only = false;
      var no_caption = false;
      if (is_thumb || is_other) {
        var up = img.parentNode.parentNode;
        // Three levels is sufficient: thumbinner-thumb-control, or floatnone-center-control, or direct
        for (var i = 0; ++i <= 3 && up; up = up.parentNode) {
          if (LAPI.DOM.hasClass(up, 'wpImageAnnotatorControl')) {
            if (LAPI.DOM.hasClass(up, 'wpImageAnnotatorOff')) return null;
            if (LAPI.DOM.hasClass(up, 'wpImageAnnotatorIconOnly')) icon_only = true;
            if (LAPI.DOM.hasClass(up, 'wpImageAnnotatorCaptionOff')) no_caption = true;
            break;
          }
        }
      }
      return { scope: scope
              ,file_div: file_div
              ,img: img
              ,realName: name
              ,isThumbnail: is_thumb
              ,isOther: is_other
              ,thumb: {width: dim.width, height: dim.height}
              ,iconOnly: icon_only
              ,noCaption: no_caption
             };
    }

    function setup_images (list)
    {
      Array.forEach(list,
        function (elem) {
          var desc = setup_one (elem);
          if (desc) self.imgs[self.imgs.length] = desc;
        }
      );
    }

    if (mw.config.get('wgNamespaceNumber') == 6) {
      setup_images ([document]);
      self.may_edit = self.may_edit && (self.imgs.length == 1);
      setup_images (self.imgs_with_notes);
    } else {
      setup_images (self.imgs_with_notes);
      self.may_edit = self.may_edit && (self.imgs.length == 1);
    }

    self.may_edit = self.may_edit && location.href.search(/[?&]oldid=/) < 0;

    if (self.haveAjax) {
      setup_images (self.thumbs);
      setup_images (self.other_images);
    }

    // Remove the test div
    LAPI.DOM.removeNode(testImgDiv);

    if (self.imgs.length === 0) return;

    // We get the UI texts in parallel, but wait for them at the beginning of complete_setup, where we
    // need them. This has in particular a benefit if we do have to query for the file sizes below.

    if (self.imgs.length == 1 && self.imgs[0].scope == document && !self.haveAjax) {
      // Try to get the full size without Ajax.
      self.imgs[0].full_img = LAPI.WP.fullImageSizeFromPage();
      if (self.imgs[0].full_img.width > 0 && self.imgs[0].full_img.height > 0) {
        self.setup_step_two();
        return;
      }
    }

    // Get the full sizes of all the images. If more than 50, make several calls. (The API has limits.)
    // Also avoid using Ajax on IE6...

    var cache = {};
    var names = [];

    Array.forEach(self.imgs, function (img, idx) {
      if (cache[img.realName]) {
        cache[img.realName][cache[img.realName].length] = idx;
      } else {
        cache[img.realName] = [idx];
        names[names.length] = img.realName;
      }
    });

    var to_do = names.length;
    var done  = 0;

    function check_done (length)
    {
      done += length;
      if (done >= names.length) {
        if (typeof ImageAnnotator.info_callbacks !== 'undefined') ImageAnnotator.info_callbacks = null;
        self.setup_step_two();
      }
    }

    function make_calls (execute_call, url_limit)
    {
      function build_titles (from, length, url_limit)
      {
        var done = 0;
        var text = '';
        for (var i = from; i < from + length; i++) {
          var new_text = names[i];
          if (url_limit) {
            new_text = encodeURIComponent(new_text);
            if (text.length && (text.length + new_text.length + 1 > url_limit)) break;
          }
          text += (text.length ? '|' : '') + new_text;
          done++;
        }
        return {text: text, n: done};
      }

      var start = 0, chunk = 0, params;
      while (to_do > 0) {
        params = build_titles (start, Math.min(50, to_do), url_limit);
        execute_call (params.n, params.text);
        to_do -= params.n;
        start += params.n;
      }
    }

    function set_info (json)
    {
      try {
        if (json && json.query && json.query.pages) {
          function get_size (info) {
            if (!info.imageinfo || info.imageinfo.length === 0) return;
            var title = info.title.replace(/ /g, '_');
            var indices = cache[title];
            if (!indices) return;
            Array.forEach(
                indices
              , function (i) {
                  self.imgs[i].full_img = { width : info.imageinfo[0].width
                                           ,height: info.imageinfo[0].height};
                  self.imgs[i].has_page = (typeof info.missing === 'undefined');
                  self.imgs[i].isLocal  = !info.imagerepository || info.imagerepository == 'local';
                  if (i != 0 || !self.may_edit || !info.protection || mw.config.get('wgNamespaceNumber') != 6) return;
                  // Care about the protection settings
                  var protection = Array.any(info.protection, function (e) {
                    return (e.type == 'edit' ? e : null);
                  });
                  self.may_edit =
                       !protection
                    || (mw.config.get('wgUserGroups') && mw.config.get('wgUserGroups').join(' ').contains(protection.level))
                  ;
                }
            );
          }
          for (var page in json.query.pages) {
            get_size (json.query.pages[page]);
          }
        } // end if
      } catch (ex) {
      }
    }

    if ((!window.XMLHttpRequest && !!window.ActiveXObject) || !self.haveAjax) {
      // IE has a stupid security setting asking whether ActiveX should be allowed. We avoid that
      // prompt by using getScript instead of parseWikitext in this case.
      ImageAnnotator.info_callbacks = [];
      var template = mw.config.get('wgServer') + mw.config.get('wgScriptPath') + '/api.php?action=query&format=json'
                   + '&prop=info|imageinfo&inprop=protection&iiprop=size'
                   + '&titles=&callback=ImageAnnotator.info_callbacks[].callback';
      if (template.startsWith('//')) template = document.location.protocol + template; // Avoid protocol-relative URIs (IE7 bug)
      make_calls (
          function (length, titles) {
            var idx = ImageAnnotator.info_callbacks.length;
            ImageAnnotator.info_callbacks[idx] = {
              callback: function (json) {
                set_info (json);
                ImageAnnotator.info_callbacks[idx].done = true;
                if (ImageAnnotator.info_callbacks[idx].script) {
                  LAPI.DOM.removeNode(ImageAnnotator.info_callbacks[idx].script);
                  ImageAnnotator.info_callbacks[idx].script = null;
                }
                check_done (length);
              },
              done: false
            };
            ImageAnnotator.info_callbacks[idx].script =
              IA.getScript(
                  template.replace('info_callbacks[].callback', 'info_callbacks[' + idx + '].callback')
                          .replace('&titles=&', '&titles=' + titles + '&')
                , true // No local caching!
              );
            // We do bypass the local JavaScript cache of importScriptURI, but on IE, we still may
            // get the script from the browser's cache, and if that happens, IE may execute the
            // script (and call the callback) synchronously before the assignment is done. Clean
            // up in that case.
            if (   ImageAnnotator.info_callbacks && ImageAnnotator.info_callbacks[idx]
                && ImageAnnotator.info_callbacks[idx].done && ImageAnnotator.info_callbacks[idx].script
            ) {
              LAPI.DOM.removeNode(ImageAnnotator.info_callbacks[idx].script);
              ImageAnnotator.info_callbacks[idx].script = null;
            }
          }
        , (LAPI.Browser.is_ie ? 1950 : 4000) - template.length // Some slack for caching parameters
      );
    } else {
      make_calls (
        function (length, titles) {
          LAPI.Ajax.apiGet(
              'query'
            , { titles : titles
               ,prop: 'info|imageinfo'
               ,inprop : 'protection'
               ,iiprop : 'size'
              }
            , function (request, json_result) {
                set_info (json_result);
                check_done (length);
              }
            , function () {check_done (length);}
          );
        }
      );
    } // end if can use Ajax
  },

  setup_ui: function () {
    // Complete the UI object we've gotten from config.

    ImageAnnotator.UI.ready      = false;
    ImageAnnotator.UI.repo       = null;
    ImageAnnotator.UI.needs_plea = false;

    var readyEvent = [];

    ImageAnnotator.UI.fireReadyEvent = function () {
      if (ImageAnnotator.UI.ready) return; // Already fired, nothing to do.
      ImageAnnotator.UI.ready = true;
      // Call all registered handlers, and clear the array.
      Array.forEach( readyEvent , function (f, idx) {
        try {f ();} catch (ex) {}
        readyEvent[idx] = null;
      });
      readyEvent = null;
    };

    ImageAnnotator.UI.addReadyEventHandler = function (f) {
      if (ImageAnnotator.UI.ready) {
        f (); // Already fired: call directly
      } else {
        readyEvent[readyEvent.length] = f;
      }
    };

    ImageAnnotator.UI.setup = function () {
      if (ImageAnnotator.UI.repo) return;
      var self = ImageAnnotator.UI;
      var node = LAPI.make('div', null, { display: 'none' });
      document.body.appendChild(node);
      if (typeof UIElements === 'undefined') {
        self.basic = true;
        self.repo = {};
        for (var item in self.defaults) {
          node.innerHTML = self.defaults[item];
          self.repo[item] = node.firstChild;
          LAPI.DOM.removeChildren(node);
        }
      } else {
        self.basic = false;
        self.repo = UIElements.emptyRepository(self.defaultLanguage);
        for (var item in self.defaults) {
          node.innerHTML = self.defaults[item];
          UIElements.setEntry(item, self.repo, node.firstChild);
          LAPI.DOM.removeChildren(node);
        }
        UIElements.load('wpImageAnnotatorTexts', null, null, self.repo);
      }
      LAPI.DOM.removeNode(node);
    };

    ImageAnnotator.UI.get = function (id, basic, no_plea) {
      var self = ImageAnnotator.UI;
      if (!self.repo) self.setup();
      var result   = null;
      var add_plea = false;
      if (self.basic) {
        result = self.repo[id];
      } else {
        result = UIElements.getEntry(id, self.repo, mw.config.get('wgUserLanguage'), null);
        add_plea = !result;
        if (!result) result = UIElements.getEntry(id, self.repo);
      }
      self.needs_plea = add_plea;
      if (!result) return null; // Hmmm... what happened here? We normally have defaults...
      if (basic) return LAPI.DOM.getInnerText(result).trim();
      result = result.cloneNode(true);
      if (mw.config.get('wgServer').contains('/commons') && add_plea && !no_plea) {
        // Add a translation plea.
        if (result.nodeName.toLowerCase() == 'div') {
          result.appendChild(self.get_plea());
        } else {
          var span = LAPI.make('span');
          span.appendChild(result);
          span.appendChild(self.get_plea());
          result = span;
        }
      }
      return result;
    };

    ImageAnnotator.UI.get_plea = function () {
      var self = ImageAnnotator.UI;
      var translate = self.get('wpTranslate', false, true) || 'translate';
      var span = LAPI.make('small');
      span.appendChild(document.createTextNode('\xa0('));
      span.appendChild(
        LAPI.DOM.makeLink(
              mw.config.get('wgServer') + mw.config.get('wgScript') + '?title=MediaWiki_talk:ImageAnnotatorTexts'
            + '&action=edit&section=new&withJS=MediaWiki:ImageAnnotatorTranslator.js'
            + '&language=' + mw.config.get('wgUserLanguage')
          , translate
          , (typeof translate === 'string' ? translate : LAPI.DOM.getInnerText(translate).trim())
        )
      );
      span.appendChild(document.createTextNode(')'));
      return span;
    };

    ImageAnnotator.UI.init = function (html_text_or_json) {
      var text;
      if (typeof html_text_or_json === 'string')
        text = html_text_or_json;
      else if (   typeof html_text_or_json !== 'undefined'
               && typeof html_text_or_json.parse !== 'undefined'
               && typeof html_text_or_json.parse.text !== 'undefined'
               && typeof html_text_or_json.parse.text['*'] !== 'undefined'
            )
        text = html_text_or_json.parse.text['*'];
      else
        text = null;

      if (!text) {
        ImageAnnotator.UI.fireReadyEvent();
        return;
      }

      var node = LAPI.make('div', null, {display: 'none'});
      document.body.appendChild(node);
      try {
        node.innerHTML = text;
      } catch (ex) {
        LAPI.DOM.removeNode(node);
        node = null;
        // Swallow. We'll just work with the default UI
      }
      if (node && !ImageAnnotator.UI.repo) ImageAnnotator.UI.setup();
      ImageAnnotator.UI.fireReadyEvent();
    };

    var ui_page = '{{MediaWiki:ImageAnnotatorTexts'
                + (mw.config.get('wgUserLanguage') != mw.config.get('wgContentLanguage') ? '|lang=' + mw.config.get('wgUserLanguage')
: '')
                + '|live=1}}';

    function get_ui_no_ajax ()
    {
      var url =
          mw.config.get('wgServer') + mw.config.get('wgScriptPath') + '/api.php?action=parse&pst&text='
        + encodeURIComponent(ui_page) + '&title=API&prop=text&format=json'
        + '&callback=ImageAnnotator.UI.init&maxage=14400&smaxage=14400'
      ;
      // Result cached for 4 hours. How to properly handle an error? It appears there's no way to catch
      // that on IE. (On FF, we could use an onerror handler on the script tag, but on FF, we use Ajax
      // anyway.)
      IA.getScript(url, true); // No local caching!
    }

    function get_ui ()
    {
      IA.haveAjax    = (LAPI.Ajax.getRequest() != null);
      IA.ajaxQueried = true;

      // Works only with Ajax (but then, most of this script doesn't work without).
      // Check what this does to load times... If lots of people used this, it might be better to
      // have the UI texts included in some footer as we did on Special:Upload. True, everybody
      // would get the texts, even people not using this, but the texts are small anyway...
      if (!IA.haveAjax) {
        get_ui_no_ajax (); // Fallback.
        return;
      }

      LAPI.Ajax.parseWikitext(
          ui_page
        , ImageAnnotator.UI.init
        , ImageAnnotator.UI.fireReadyEvent
        , false
        , null
        , "API" // A fixed string to enable caching at all.
        , 14400 // 4 hour caching.
      );
    } // end get_ui

    if (!window.XMLHttpRequest && !!window.ActiveXObject) {
      // IE has a stupid security setting asking whether ActiveX should be allowed. We avoid that
      // prompt by using getScript instead of parseWikitext in this case. The disadvantage
      // is that we don't do anything if this fails for some reason.
      get_ui_no_ajax ();
    } else {
      get_ui ();
    }
  },

  setup_step_two: function () {
    var self = IA;

    // Throw out any images for which we miss either the thumbnail or the full image size.
    // Also throws out thumbnails that are larger than the full image.
    self.imgs = Array.select(self.imgs, function (elem, idx) {
        var result =
             elem.thumb.width > 0 && elem.thumb.height > 0
          && typeof elem.full_img !== 'undefined'
          && elem.full_img.width > 0 && elem.full_img.height > 0
          && elem.full_img.width >= elem.thumb.width
          && elem.full_img.height >= elem.thumb.height
        ;
        if (self.may_edit && idx === 0 && !result) self.may_edit = false;
        return result;
      });

    if (self.imgs.length === 0) return;

    ImageAnnotator.UI.addReadyEventHandler(IA.complete_setup);
  },

  complete_setup: function () {
    // We can be sure to have the UI here because this is called only when the ready event of the
    // UI object is fired.
    var self = IA;

    // Check edit permissions
    if (self.may_edit && mw.config.get('wgRestrictionEdit')) {
      self.may_edit =
        (   (mw.config.get('wgRestrictionEdit').length === 0 || mw.config.get('wgUserGroups') && mw.config.get('wgUserGroups').join('
').contains('sysop'))
         || (   mw.config.get('wgRestrictionEdit').length === 1 && mw.config.get('wgRestrictionEdit')[0] === 'autoconfirmed'
             && mw.config.get('wgUserGroups') && mw.config.get('wgUserGroups').join(' ').contains('confirmed') // confirmed &
autoconfirmed
            )
        );
    }

    if (self.may_edit) {
      // Check whether the image is local. Don't allow editing if the file is remote.
      var sharedUpload = getElementsByClassName (document, 'div', 'sharedUploadNotice');
      self.may_edit = (!sharedUpload || sharedUpload.length === 0);
    }

    if (self.may_edit && mw.config.get('wgNamespaceNumber') != 6) {
      // Only allow edits if the stored page name matches the current one.
      var img_page_name =
        getElementsByClassName (self.imgs[0].scope, '*', 'wpImageAnnotatorPageName');
      if (img_page_name && img_page_name.length)
        img_page_name = LAPI.DOM.getInnerText(img_page_name[0]);
      else
        img_page_name = '';
      self.may_edit = (img_page_name.replace(/ /g, '_') == mw.config.get('wgTitle').replace(/ /g, '_'));
    }

    if (self.may_edit && self.ajaxQueried) self.may_edit = self.haveAjax;

    // Now create viewers for all images
    self.viewers = new Array (self.imgs.length);
    for (var i = 0; i < self.imgs.length; i++) {
      self.viewers[i] = new ImageNotesViewer (self.imgs[i], i === 0 && self.may_edit);
      //console.log('may edit inside loop: ' + self.may_edit);
    }

    //console.log('may edit after loop: ' + self.may_edit);

    if (self.may_edit) {

    // Respect user override for zoom, if any
    self.zoom_threshold = ImageAnnotator_config.zoom_threshold;
    if (   typeof window.ImageAnnotator_zoom_threshold !== 'undefined'
        && !isNaN (window.ImageAnnotator_zoom_threshold)
        && window.ImageAnnotator_zoom_threshold >= 0.0
       )
    {
      // If somebody sets it to a nonsensical high value, that's his or her problem: there won't be any
      // zooming.
      self.zoom_threshold = window.ImageAnnotator_zoom_threshold;
    }
    // Adapt zoom threshold for small thumbnails or images with a very lopsided width/height ratio,
    // but only if we *can* zoom at least twice
    if (   self.viewers[0].full_img.width > 300
        && Math.min(self.viewers[0].factors.dx, self.viewers[0].factors.dy) >= 2.0
       )
    {
      if (   self.viewers[0].thumb.width < 400
          || self.viewers[0].thumb.width / self.viewers[0].thumb.height > 2.0
          || self.viewers[0].thumb.height / self.viewers[0].thumb.width > 2.0
         )
      {
        self.zoom_threshold = 0; // Force zooming
      }
    }

    self.editor = new ImageAnnotationEditor ();

    function track (evt) {
      evt = evt || window.event;
      if (self.is_adding) self.update_zoom(evt);
      if (!self.is_tracking) return LAPI.Evt.kill(evt);
      var mouse_pos = LAPI.Pos.mousePosition(evt);
      if (!LAPI.Pos.isWithin(self.cover, mouse_pos.x, mouse_pos.y)) return;
      var origin    = LAPI.Pos.position(self.cover);
      // Make mouse pos relative to cover
      mouse_pos.x = mouse_pos.x - origin.x;
      mouse_pos.y = mouse_pos.y - origin.y;
      if (mouse_pos.x >= self.base_x) {
        self.definer.style.width = '' + (mouse_pos.x - self.base_x) + 'px';
        self.definer.style.left  = '' + self.base_x + 'px';
      } else {
        self.definer.style.width = '' + (self.base_x - mouse_pos.x) + 'px';
        self.definer.style.left  = '' + mouse_pos.x + 'px';
      }
      if (mouse_pos.y >= self.base_y) {
        self.definer.style.height = '' + (mouse_pos.y - self.base_y) + 'px';
        self.definer.style.top    = '' + self.base_y + 'px';
      } else {
        self.definer.style.height = '' + (self.base_y - mouse_pos.y) + 'px';
        self.definer.style.top    = '' + mouse_pos.y + 'px';
      }
      return LAPI.Evt.kill(evt);
    }

    function pause (evt) {
      LAPI.Evt.remove(document, 'mousemove', track, true);
      if (!LAPI.Browser.is_ie && typeof document.captureEvents === 'function')
        document.captureEvents(null);
      self.move_listening = false;
    }

    function resume (evt) {
      // captureEvents is actually deprecated, but I haven't succeeded to make this work with
      // addEventListener only.
      if ((self.is_tracking || self.is_adding) && !self.move_listening) {
        if (!LAPI.Browser.is_ie && typeof document.captureEvents === 'function')
          document.captureEvents(Event.MOUSEMOVE);
        LAPI.Evt.attach(document, 'mousemove', track, true);
        self.move_listening = true;
      }
    }

    function stop_tracking (evt) {
      evt = evt || window.event;
      // Check that we're within the image. Note: this check can fail only on IE >= 7, on other
      // browsers, we attach the handler on self.cover and thus we don't even get events outside
      // that range.
      var mouse_pos = LAPI.Pos.mousePosition(evt);
      if (!LAPI.Pos.isWithin(self.cover, mouse_pos.x, mouse_pos.y)) return;
      if (self.is_tracking) {
        self.is_tracking = false;
        self.is_adding = false;
        // Done.
        pause ();
        if (LAPI.Browser.is_ie) {
          //Trust Microsoft to get everything wrong!
          LAPI.Evt.remove(document, 'mouseup', stop_tracking);
        } else {
          LAPI.Evt.remove(self.cover, 'mouseup', stop_tracking);
        }
        LAPI.Evt.remove(window, 'blur', pause);
        LAPI.Evt.remove(window, 'focus', resume);
        self.cover.style.cursor = 'auto';
        LAPI.DOM.removeNode(self.border);
        LAPI.Evt.remove(self.cover, self.mouse_in, self.update_zoom_evt);
        LAPI.Evt.remove(self.cover, self.mouse_out, self.hide_zoom_evt);
        self.hide_zoom();
        self.viewers[0].hide(); // Hide all existing boxes
        if (!self.definer || self.definer.offsetWidth <= 0 || self.definer.offsetHeight <= 0) {
          // Nothing: just remove the definer:
          if (self.definer) LAPI.DOM.removeNode(self.definer);
          // Re-attach event handlers
          self.viewers[0].setShowHideEvents(true);
          self.hide_cover();
          self.viewers[0].setDefaultMsg();
          // And make sure we get the real view again
          self.viewers[0].show();
        } else {
          // We have a div with some extent: remove event capturing and create a new annotation
      var new_note = new ImageAnnotation (self.definer, self.viewers[0], -1);
      self.viewers[0].register(new_note);
      self.editor.editNote(new_note);
    }
    self.definer = null;
  }
  if (evt) return LAPI.Evt.kill(evt);
  return false;
}

function start_tracking (evt) {
  if (!self.is_tracking) {
    self.is_tracking = true;
    evt = evt || window.event;
    // Set the position, size 1
    var mouse_pos = LAPI.Pos.mousePosition(evt);
    var origin    = LAPI.Pos.position(self.cover);
    self.base_x = mouse_pos.x - origin.x;
    self.base_y = mouse_pos.y - origin.y;
    Object.merge(
        { left: '' + self.base_x + 'px'
         ,top: '' + self.base_y + 'px'
         ,width: '0px'
         ,height : '0px'
         ,display: ''
        }
      , self.definer.style
    );
    // Set mouse handlers
    LAPI.Evt.remove(self.cover, 'mousedown', start_tracking);
    if (LAPI.Browser.is_ie) {
      LAPI.Evt.attach(document, 'mouseup', stop_tracking); // Doesn't work properly on self.cover...
    } else {
      LAPI.Evt.attach(self.cover, 'mouseup', stop_tracking);
    }
    resume ();
    LAPI.Evt.attach(window, 'blur', pause);
    LAPI.Evt.attach(window, 'focus', resume);
  }
  if (evt) return LAPI.Evt.kill(evt);
  return false;
}

function add_new (evt) {
  if (!self.canEdit()) return;

  self.editor.hide_editor();
  Tooltips.close();
  var cover = self.get_cover();
  cover.style.cursor = 'crosshair';
  self.definer =
    LAPI.make(
        'div', null
       ,{ border: '1px solid ' + IA.new_border
         ,display: 'none'
         ,position: 'absolute'
         ,top: '0px'
         ,left: '0px'
         ,width: '0px'
         ,height: '0px'
         ,padding: '0'
         ,lineHeight : '0px' // IE needs this, even though there are no lines within
         ,fontSize: '0px' // IE
         ,zIndex: cover.style.zIndex - 2 // Below the mouse capture div
        }
    );
  self.viewers[0].img_div.appendChild(self.definer);
  // Enter mouse-tracking mode to define extent of view. Mouse cursor is outside of image,
  // hence none of our tooltips are visible.
  self.viewers[0].img_div.appendChild(self.border);
  self.show_cover();
  self.is_tracking = false;
  self.is_adding   = true;
  LAPI.Evt.attach(cover, 'mousedown', start_tracking);
  resume ();
  self.button_div.style.display = 'none';
  // Remove the event listeners on the image: IE sometimes invokes them even when the image is covered
  self.viewers[0].setShowHideEvents(false);
  self.viewers[0].hide();       // Make sure notes are hidden
  self.viewers[0].toggle(true); // Show all note rectangles (but only the dummies)
  self.update_zoom_evt = LAPI.Evt.makeListener(self, self.update_zoom);
  self.hide_zoom_evt = LAPI.Evt.makeListener(self, self.hide_zoom);
  self.show_zoom();
  LAPI.Evt.attach(cover, self.mouse_in, self.update_zoom_evt);
  LAPI.Evt.attach(cover, self.mouse_out, self.hide_zoom_evt);
  LAPI.DOM.removeChildren(self.viewers[0].msg);
  self.viewers[0].msg.appendChild
    (ImageAnnotator.UI.get('wpImageAnnotatorDrawRectMsg', false));
  self.viewers[0].msg.style.display = '';
}

self.button_div = LAPI.make('div');
self.viewers[0].main_div.appendChild(self.button_div);
self.add_button =
  LAPI.DOM.makeButton(
      'ImageAnnotationAddButton'
    , ImageAnnotator.UI.get('wpImageAnnotatorAddButtonText', true)
    , add_new
);
var add_plea = ImageAnnotator.UI.needs_plea;
self.button_div.appendChild(self.add_button);
self.help_link = self.createHelpLink();
if (self.help_link) {
  self.button_div.appendChild(document.createTextNode('\xa0'));
  self.button_div.appendChild(self.help_link);
}
if (add_plea && mw.config.get('wgServer').contains('/commons'))
  self.button_div.appendChild(ImageAnnotator.UI.get_plea());
```

```
    } // end if may_edit

    // Get the file description pages of thumbnails. Figure out for which viewers we need to do this.
    var cache       = {};
    var get_local   = [];
    var get_foreign = [];
    Array.forEach(self.viewers, function (viewer, idx) {
      if (viewer.setup_done || viewer.isLocal && !viewer.has_page) return;
      // Handle only images that either are foreign or local and do have a page.
      if (cache[viewer.realName]) {
        cache[viewer.realName][cache[viewer.realName].length] = idx;
      } else {
        cache[viewer.realName] = [idx];
        if (!viewer.has_page) {
          get_foreign[get_foreign.length] = viewer.realName;
        } else {
          get_local[get_local.length] = viewer.realName;
        }
      }
    });

    if (get_local.length === 0 && get_foreign.length === 0) return;

    // Now we have unique page names in the cache and in to_get. Go get the corresponding file
    // description pages. We make a series of simultaneous asynchronous calls to avoid hitting
    // API limits and to keep the URL length below the limit for the foreign_repo calls.

    function make_calls (list, execute_call, url_limit)
    {
      function composer (list, from, length, url_limit)
      {
        function compose (list, from, length, url_limit)
        {
          var text = '';
          var done = 0;
          for (var i = from; i < from + length; i++) {
            var new_text =
                '<div class="wpImageAnnotatorInlineImageWrapper" style="display:none;">'
              + '<span class="image_annotation_inline_name">' + list[i] + '</span>'
              + '{{:' + list[i] + '}}' // Leading dot to avoid getting links to the full images if we hit a parser limit
              + '</div>'
            ;
            if (url_limit) {
              new_text = encodeURIComponent(new_text);
              if (text.length && (text.length + new_text.length > url_limit)) break;
            }
            text = text + new_text;
            done++;
            // Additionally, limit the number of image pages to get: these can be large, and the server
            // may refuse to actually do the transclusions but may in that case include the full images
            // in the result, which would make us load the full images, which is desastrous if there are
            // many thumbs to large images on the page.
            if (done == 5) break;
          }
          return {text: text, n: done};
        }

        var param = compose (list, from, length, url_limit);
        execute_call (param.text);
        return param.n;
      }

      var start = 0, chunk = 0, to_do = list.length;
      while (to_do > 0) {
        chunk = composer (list, start, Math.min(50, to_do), url_limit);
        to_do -= chunk;
        start += chunk;
      }
    }

    var divRE         = /(<\s*div\b)|(<\/\s*div\s*>)/ig;
    var blockStart    = '<div class="wpImageAnnotatorInlineImageWrapper"';
    var inlineNameEnd = '</span>';
    var noteStart     = '<div id="image_annotation_note_';
    var noteControlRE = /<div\s*class="(wpImageAnnotatorInlinedRules|image_annotation_colors")(\S|\s)*?\/div>/g;

    // Our parse request returns the full html of the description pages' contents, including any
    // license or information or other templates that we don't care about, and which may contain
    // additional images we'd rather not load when we add this (X)HTML to the DOM. Therefore, we
    // strip out everything but the notes.
    function strip_noise (html) {
      var result = '';
      var m;
      // First, get rid of HTML comments and scripts
      html = html.replace(/<\!--(.|\s)*?--\>/g, '').replace(/<script(.|\s)*?\/script>/g, '');
      var i = html.indexOf(blockStart, idx), idx = 0, l = html.length;
      // Now collect pages
      while (idx < l && i >= idx) {
        var j = html.indexOf(inlineNameEnd, i+blockStart.length);
        if (j < i+blockStart.length) break;
        result += html.substring(i, j+inlineNameEnd.length);
        idx = j+inlineNameEnd.length;
        // Now collect all image image notes for that page
        var note_begin = 0, next_block = html.indexOf(blockStart, idx);
        // Do we have image note control or color templates?
        j = idx;
        for (;;) {
          noteControlRE.lastIndex = j;
          m = noteControlRE.exec(html);
          if (!m || next_block >= idx && m.index > next_block) break;
          result += m[0];
          j = m.index + m[0].length;
        }
        // Collect notes
        for (;;) {
          note_begin = html.indexOf(noteStart, idx);
          // Check whether next wrapper comes first
          if (note_begin < idx || (next_block >= idx && note_begin > next_block)) break;
          // Start parsing nested <div> and </div>, from note_begin on. Just ignore any other tags.
      var level = 1, k = note_begin + noteStart.length;
      while (level > 0 && k < l) {
        divRE.lastIndex = k;
        m = divRE.exec(html);
        if (!m || m.length < 2) {
          k = l; // Nothing found at all?
        } else {
          if (m[1]) {
            level++; k = m.index + m[1].length; // divStart found first
          } else if (m.length > 2 && m[2]) {
            level--; k = m.index + m[2].length; // divEnd found first
          } else {
            k = l; // Huh?
          }
        }
      } // end loop for nested divs
      result += html.substring(note_begin, k);
      while (level-- > 0) result += '</div>'; // Missing ends.
      idx = k;
    } // end loop collecting notes
    result += '</div>'; // Close the wrapper
    i = next_block;
  } // end loop collecting pages
  return result;
}

function setup_thumb_viewers (html_text)
{
  var node = LAPI.make('div', null, {display: 'none'});
  document.body.appendChild(node);
  try {
    node.innerHTML = strip_noise (html_text);
    var pages = getElementsByClassName (node, 'div', 'wpImageAnnotatorInlineImageWrapper');
    for (var i = 0; pages && i < pages.length; i++) {
      var notes = getElementsByClassName (pages[i], 'div', self.annotation_class);
      if (!notes || notes.length === 0) {
        continue;
      }
      var page = self.getItem('inline_name', pages[i]);
      if (!page) continue;
      page = page.replace(/ /g, '_');
      var viewers = cache[page] || cache['File:' + page.substring(page.indexOf(':') + 1)];
      if (!viewers || viewers.length === 0) {
        continue;
      }
      // Update rules.
      var rules = getElementsByClassName (pages[i], 'div', 'wpImageAnnotatorInlinedRules');
      var local_rules = {
        inline: Object.clone(IA.rules.inline),
        thumbs: Object.clone(IA.rules.thumbs)
      };
      if (rules && rules.length) {
        rules = rules[0];
        if (   typeof local_rules.inline.show === 'undefined'
            && LAPI.DOM.hasClass(rules, 'wpImageAnnotatorNoInlineDisplay')
           )
        {
          local_rules.inline.show = false;
        }
        if (   typeof local_rules.inline.icon === 'undefined'
            && LAPI.DOM.hasClass(rules, 'wpImageAnnotatorInlineDisplayIcon')
           )
        {
          local_rules.inline.icon = true;
        }
        if (   typeof local_rules.thumbs.show === 'undefined'
            && LAPI.DOM.hasClass(rules, 'wpImageAnnotatorNoThumbs')
           )
        {
          local_rules.thumbs.show = false;
        }
        if (   typeof local_rules.thumbs.icon === 'undefined'
            && LAPI.DOM.hasClass(rules, 'wpImageAnnotatorThumbDisplayIcon')
           )
        {
          local_rules.thumbs.icon = true;
        }
      }
      // Make sure all are set
      local_rules.inline.show =
        typeof local_rules.inline.show === 'undefined' || local_rules.inline.show;
      local_rules.thumbs.show =
        typeof local_rules.thumbs.show === 'undefined' || local_rules.thumbs.show;
      local_rules.inline.icon =
        typeof local_rules.inline.icon !== 'undefined' && local_rules.inline.icon;
      local_rules.thumbs.icon =
        typeof local_rules.thumbs.icon !== 'undefined' && local_rules.thumbs.icon;
      if (!local_rules.inline.show) continue;
      // Now use pages[i] as a scope shared by all the viewers using it. Since we clone note
      // contents for note display, this works. Otherwise, we'd have to copy the notes into
      // each viewer's scope.
      document.body.appendChild(pages[i]); // Move it out of 'node'
      // Set viewers' scopes and finish their setup.
      Array.forEach(viewers, function (v) {
        if (!self.viewers[v].isThumbnail || local_rules.thumbs.show) {
          self.viewers[v].scope = pages[i];
          self.viewers[v].setup(   self.viewers[v].isThumbnail && local_rules.thumbs.icon
                                || self.viewers[v].isOther && local_rules.inline.icon);
        }
      });
    }
  } catch (ex) {}
  LAPI.DOM.removeNode(node);
}

ImageAnnotator.script_callbacks = [];

function make_script_calls (list, api)
{
      var template = api + '?action=parse&pst&text=&prop=text&format=json'
                   + '&maxage=1800&smaxage=1800&uselang=' + mw.config.get('wgUserLanguage') //see bugzilla 22764
                   + '&callback=ImageAnnotator.script_callbacks[].callback';
      if (template.startsWith('//')) template = document.location.protocol + template; // Avoid protocol-relative URIs (IE7 bug)
      make_calls (
          list
        , function (text) {
            var idx = ImageAnnotator.script_callbacks.length;
            ImageAnnotator.script_callbacks[idx] =
              { callback: function (json) {
                            if (json && json.parse && json.parse.text && json.parse.text['*']) {
                              setup_thumb_viewers (json.parse.text['*']);
                            }
                            ImageAnnotator.script_callbacks[idx].done = true;
                            if (ImageAnnotator.script_callbacks[idx].script) {
                              LAPI.DOM.removeNode(ImageAnnotator.script_callbacks[idx].script);
                              ImageAnnotator.script_callbacks[idx].script = null;
                            }
                          }
                ,done: false
              };
            ImageAnnotator.script_callbacks[idx].script =
              IA.getScript(
                  template.replace('script_callbacks[].callback', 'script_callbacks[' + idx + '].callback')
                          .replace('&text=&', '&text=' + text + '&')
                , true // No local caching!
              );
            if (   ImageAnnotator.script_callbacks && ImageAnnotator.script_callbacks[idx]
                && ImageAnnotator.script_callbacks[idx].done && ImageAnnotator.script_callbacks[idx].script)
            {
              LAPI.DOM.removeNode(ImageAnnotator.script_callbacks[idx].script);
              ImageAnnotator.script_callbacks[idx].script = null;
            }
          }
        , (LAPI.DOM.is_ie ? 1950 : 4000) - template.length // Some slack for caching parameters
      );
    }

    if ((!window.XMLHttpRequest && !!window.ActiveXObject) || !self.haveAjax) {
      make_script_calls(get_local, mw.config.get('wgServer') + mw.config.get('wgScriptPath') + '/api.php');
    } else {
      make_calls(
          get_local
        , function (text) {
            LAPI.Ajax.parseWikitext(
                text
              , function (html_text) {if (html_text) setup_thumb_viewers(html_text);}
              , function () {}
              , false
              , null
              , 'API' // Fixed string to enable caching at all
              , 1800  // 30 minutes caching.
            );
          }
      );
    }

    // Can't use Ajax for foreign repo, might violate single-origin policy (e.g. from wikisource.org
    // to wikimedia.org). Attention, here we must care about the URL length! IE has a limit of 2083
    // character (2048 in the path part), and servers also may impose limits (on the WMF servers,
    // the limit appears to be 8kB).
    make_script_calls (get_foreign, ImageAnnotator_config.sharedRepositoryAPI());
  },

  show_zoom: function () {
    var self = IA;
    if (   (   self.viewers[0].factors.dx < self.zoom_threshold
            && self.viewers[0].factors.dy < self.zoom_threshold
           )
        || Math.max(self.viewers[0].factors.dx, self.viewers[0].factors.dy) < 2.0
       )
    {
      // Below zoom threshold, or full image not even twice the size of the preview
      return;
    }
    if (!self.zoom) {
      self.zoom =
        LAPI.make(
            'div'
          , {id : 'image_annotator_zoom'}
          , { overflow: 'hidden'
             ,width: '200px'
             ,height: '200px'
             ,position: 'absolute'
             ,display: 'none'
             ,top: '0px'
             ,left: '0px'
             ,border: '2px solid #666666'
             ,backgroundColor : 'white'
             ,zIndex: 2050 // On top of everything
            }
        );
      var src = self.viewers[0].img.getAttribute('src', 2);
      // Adjust zoom_factor
      if (self.zoom_factor > self.viewers[0].factors.dx || self.zoom_factor > self.viewers[0].factors.dy)
        self.zoom_factor = Math.min(self.viewers[0].factors.dx, self.viewers[0].factors.dy);
      self.zoom.appendChild(LAPI.make('div', null, {position : 'relative'}));
      // Calculate zoom size and source link
      var zoom_width  = Math.floor(self.viewers[0].thumb.width * self.zoom_factor);
      var zoom_height = Math.floor(self.viewers[0].thumb.height * self.zoom_factor);
      var zoom_src   = null;
      // For SVGs, always use a scaled PNG for the zoom.
      if (zoom_width > 0.9 * self.viewers[0].full_img.width && src.search(/\.svg\.png$/i) < 0) {
        // If the thumb we'd be loading was within about 80% of the full image size, we may just as
        // well get the full image instead of a scaled version.
        self.zoom_factor = Math.min(self.viewers[0].factors.dx, self.viewers[0].factors.dy);
        zoom_width       = self.viewers[0].full_img.width;
        zoom_height      = self.viewers[0].full_img.height;
      }
      // Construct the initial zoomed image. We need to clone; if we create a completely
      // new DOM node ourselves, it may not work on IE6...
      var zoomed = self.viewers[0].img.cloneNode(true);
      zoomed.width = '' + zoom_width;
      zoomed.height = '' + zoom_height;
      Object.merge({position: 'absolute', top: '0px',left: '0px'}, zoomed.style);
      self.zoom.firstChild.appendChild(zoomed);
      // Crosshair
      self.zoom.firstChild.appendChild(
        LAPI.make(
            'div', null
          , { width: '1px'
             ,height: '200px'
             ,borderLeft : '1px solid red'
             ,position: 'absolute'
             ,top: '0px'
             ,left: '100px'
            }
        )
      );
      self.zoom.firstChild.appendChild(
        LAPI.make(
            'div', null
          , { width: '200px'
             ,height: '1px'
             ,borderTop : '1px solid red'
             ,position: 'absolute'
             ,top: '100px'
             ,left: '0px'
            }
        )
      );
      document.body.appendChild(self.zoom);
      LAPI.DOM.loadImage(
          self.viewers[0].imgName
        , src
        , zoom_width
        , zoom_height
        , ImageAnnotator_config.thumbnailsGeneratedAutomatically()
        , function (img) {
            // Replace the image in self.zoom by self.zoom_loader, making sure we keep the offsets
            img.style.position = 'absolute';
            img.style.top = self.zoom.firstChild.firstChild.style.top;
            img.style.left = self.zoom.firstChild.firstChild.style.left;
            img.style.display = '';
            self.zoom.firstChild.replaceChild(img, self.zoom.firstChild.firstChild);
          }
      );
    }
    self.zoom.style.display = 'none'; // Will be shown in update
  },

  update_zoom: function (evt) {
    if (!evt) return; // We need an event to calculate positions!
    var self = IA;
    if (!self.zoom) return;
    var mouse_pos = LAPI.Pos.mousePosition(evt);
    var origin    = LAPI.Pos.position(self.cover);
    if (!LAPI.Pos.isWithin(self.cover, mouse_pos.x, mouse_pos.y)) {
      IA.hide_zoom();
      return;
    }
    var dx = mouse_pos.x - origin.x;
    var dy = mouse_pos.y - origin.y;
    // dx, dy is the offset within the preview image. Align the zoom image accordingly.
    var top  = - dy * self.zoom_factor + 100;
    var left = - dx * self.zoom_factor + 100;
    self.zoom.firstChild.firstChild.style.top  = '' + top  + 'px';
    self.zoom.firstChild.firstChild.style.left = '' + left + 'px';
    self.zoom.style.top  = mouse_pos.y + 10 + 'px'; // Right below the mouse pointer
    // Horizontally keep it in view.
    var x = (self.is_rtl ? mouse_pos.x - 10 : mouse_pos.x + 10);
    if (x < 0) x = 0;
    self.zoom.style.left = x + 'px';
    self.zoom.style.display = '';
    // Now that we have offsetWidth, correct the position.
    if (self.is_rtl) {
      x = mouse_pos.x - 10 - self.zoom.offsetWidth;
      if (x < 0) x = 0;
    } else {
      var off  = LAPI.Pos.scrollOffset();
      var view = LAPI.Pos.viewport();
      if (x + self.zoom.offsetWidth > off.x + view.x) x = off.x + view.x - self.zoom.offsetWidth;
      if (x < 0) x = 0;
    }
    self.zoom.style.left = x + 'px';
  },

  hide_zoom: function (evt) {
    if (!IA.zoom) return;
    if (evt) {
      var mouse_pos = LAPI.Pos.mousePosition(evt);
      if (LAPI.Pos.isWithin(IA.cover, mouse_pos.x, mouse_pos.y)) return;
    }
    IA.zoom.style.display = 'none';
  },

  createHelpLink: function () {
    var msg = ImageAnnotator.UI.get('wpImageAnnotatorHelp', false, true);
    if (!msg || !msg.lastChild) return null;
    // Make sure we use the right protocol for all images:
    var imgs = msg.getElementsByTagName('img');
    var text;
    var tgt;
    if (imgs) {
      for (var i = 0; i < imgs.length; i++) {
        var srcFixed = imgs[i].getAttribute('src', 2).replace(/^https?\:/, document.location.protocol);
        imgs[i].src = srcFixed;
      }
    }
    if (   msg.childNodes.length == 1
        && msg.firstChild.nodeName.toLowerCase() == 'a'
        && !LAPI.DOM.hasClass(msg.firstChild, 'image')
       ) {
      msg.firstChild.id = 'ImageAnnotationHelpButton';
      return msg.firstChild; // Single link
    }
    // Otherwise, it's either a sequence of up to three images, or a span, followed by a
    // link.
    tgt = msg.lastChild;
    if (tgt.nodeName.toLowerCase() != 'a')
      tgt = mw.config.get('wgServer') + mw.config.get('wgArticlePath').replace('$1', 'Help:Gadget-ImageAnnotator');
    else
      tgt = tgt.href;

    function make_handler (tgt) {
      var target = tgt;
      return function (evt) {
               var e = evt || window.event;
               location.href = target;
               if (e) return LAPI.Evt.kill(e);
               return false;
             };
    }

    imgs = msg.getElementsByTagName('img');

    if (!imgs || !imgs.length) {
      // We're supposed to have a spans giving the button text
      text = msg.firstChild;
      if (text.nodeName.toLowerCase() === 'span')
        text = LAPI.DOM.getInnerText(text);
      else
        text = 'Help';
      return LAPI.DOM.makeButton(
                 'ImageAnnotationHelpButton'
               , text
               , make_handler(tgt)
             );
    } else {
      return Buttons.makeButton(imgs, 'ImageAnnotationHelpButton', make_handler(tgt));
    }
  },

  get_cover: function () {
    var self = IA;
    var shim;
    if (!self.cover) {
      var pos = { position : 'absolute'
                 ,left: '0px'
                 ,top: '0px'
                 ,width: self.viewers[0].thumb.width + 'px'
                 ,height: self.viewers[0].thumb.height + 'px'
                };
      self.cover = LAPI.make('div', null, pos);
      self.border = self.cover.cloneNode(false);
      Object.merge(
          {border: '3px solid green', top: '-3px', left: '-3px'}, self.border.style);
      self.cover.style.zIndex   = 2000;       // Above the tooltips
      if (LAPI.Browser.is_ie) {
        shim = LAPI.make('iframe', {frameBorder: 0, tabIndex: -1}, pos);
        shim.style.filter   = 'alpha(Opacity=0)'; // Ensure transparency
        // Unfortunately, IE6/SP2 has a "security setting" called "Binary and script
        // behaviors". If that is disabled, filters don't work, and our iframe would
        // appear as a white rectangle. Fix this by first placing the iframe just above
        // image (to block that windowed control) and then placing *another div* just
        // above that shim having the image as its background image.
        var imgZ = self.viewers[0].img.style.zIndex;
        if (isNaN (imgZ)) imgZ = 10; // Arbitrary, positive, > 1, < 500
        shim.style.zIndex   = imgZ + 1;
        self.ieFix = shim;
        // And now the bgImage div...
        shim = LAPI.make('div', null, pos);
        Object.merge(
            { top: '0px'
             ,backgroundImage: 'url(' + self.viewers[0].img.src + ')'
             ,zIndex: imgZ + 2
            }
          , shim.style
        );
        self.ieFix2 = shim;
      }
      if (LAPI.Browser.is_opera) {
        // It appears that events just pass through completely transparent divs on Opera.
        // Hence we have to ensure that these events are killed even if our cover doesn't
        // handle them.
        shim = LAPI.make('div', null, pos);
        shim.style.zIndex = self.cover.style.zIndex - 1;
        LAPI.Evt.attach(shim, 'mousemove',
          function (evt) {return LAPI.Evt.kill(evt || window.event);});
        LAPI.Evt.attach(shim, 'mousedown',
          function (evt) {return LAPI.Evt.kill(evt || window.event);});
        LAPI.Evt.attach(shim, 'mouseup',
          function (evt) {return LAPI.Evt.kill(evt || window.event);});
        shim.style.cursor = 'default';
        self.eventFix = shim;
      }
      self.cover_visible = false;
    }
    return self.cover;
  },

  show_cover: function () {
    var self = IA;
    if (self.cover && !self.cover_visible) {
      if (self.ieFix) {
        self.viewers[0].img_div.appendChild(self.ieFix);
        self.viewers[0].img_div.appendChild(self.ieFix2);
      }
      if (self.eventFix) self.viewers[0].img_div.appendChild(self.eventFix);
      self.viewers[0].img_div.appendChild(self.cover);
      self.cover_visible = true;
    }
  },

  hide_cover: function () {
    var self = IA;
    if (self.cover && self.cover_visible) {
      if (self.ieFix) {
        LAPI.DOM.removeNode(self.ieFix);
        LAPI.DOM.removeNode(self.ieFix2);
      }
      if (self.eventFix) LAPI.DOM.removeNode(self.eventFix);
      LAPI.DOM.removeNode(self.cover);
      self.cover_visible = false;
    }
  },

  getRawItem: function (what, scope) {
    var node = null;
    if (!scope || scope == document) {
      node = LAPI.$ ('image_annotation_' + what);
    } else {
      node = getElementsByClassName (scope, '*', 'image_annotation_' + what);
      if (node && node.length) node = node[0]; else node = null;
    }
    return node;
  },

  getItem: function (what, scope) {
    var node = IA.getRawItem(what, scope);
    if (!node) return null;
    return LAPI.DOM.getInnerText(node).trim();
  },

  getIntItem: function (what, scope) {
    var x = IA.getItem(what, scope);
    if (x !== null) x = parseInt (x, 10);
    return x;
  },

  findNote: function (text, id) {
    function find (text, id, delim) {
      var start = delim.start.replace('$1', id);
      var start_match = text.indexOf(start);
      if (start_match < 0) return null;
      var end = delim.end.replace('$1', id);
      var end_match = text.indexOf(end);
      if (end_match < start_match + start.length) return null;
      return {start: start_match, end: end_match + end.length};
    }

    var result = null;
    for (var i=0; i < IA.note_delim.length && !result; i++) {
      result = find (text, id, IA.note_delim[i]);
    }
    return result;
  },

  setWikitext: function (pagetext) {
    var self = IA;
    if (self.wiki_read) return;
    Array.forEach(self.viewers[0].annotations, function (note) {
      if (note.model.id >= 0) {
        var span = self.findNote(pagetext, note.model.id);
        if (!span) return;
        // Now extract the wikitext
        var code = pagetext.substring(span.start, span.end);
        for (var i = 0; i < self.note_delim.length; i++) {
          var start = self.note_delim[i].content_start.replace('$1', note.model.id);
          var end   = self.note_delim[i].content_end.replace('$1', note.model.id);
          var j = code.indexOf(start);
          var k = code.indexOf(end);
          if (j >= 0 && k >= 0 && k >= j + start.length) {
            note.model.wiki = code.substring(j + start.length, k).trim();
            return;
          }
        }
      }
    });
    self.wiki_read = true;
  },

  setSummary: function (summary, initial_text, note_text) {
    if (initial_text.contains('$1')) {
      var max = (summary.maxlength || 200) - initial_text.length;
      if (note_text)
        initial_text =
          initial_text.replace('$1', ': ' + note_text.replace('\n', ' ').substring(0, max));
      else
        initial_text = initial_text.replace('$1', '');
    }
    summary.value = initial_text;
  },

  getScript: function (url, bypass_local_cache, bypass_caches) {
    // Don't use LAPI here, it may not yet be available
    if (bypass_caches) {
      url += ((url.indexOf('?') >= 0) ? '&' : '?') + 'dummyTimestamp=' + (new Date()).getTime();
    }
    // Avoid protocol-relative URIs (IE7 bug)
    if (url.length >= 2 && url.substring(0, 2) === '//') url = document.location.protocol + url;
    if (bypass_local_cache) {
      var s = document.createElement('script');
      s.setAttribute('src', url);
      s.setAttribute('type', 'text/javascript');
      document.getElementsByTagName('head')[0].appendChild(s);
      return s;
    } else {
      return mw.loader.load( url );
    }
  },

  canEdit: function () {
    var self = IA;
    if (self.may_edit) {
      if (!self.ajaxQueried) {
        self.haveAjax = (LAPI.Ajax.getRequest() != null);
        self.ajaxQueried = true;
        self.may_edit = self.haveAjax;
        if (!self.may_edit && self.button_div) {
          LAPI.DOM.removeChildren(self.button_div);
          self.button_div.appendChild
            (ImageAnnotator.UI.get('wpImageAnnotatorCannotEditMsg', false));
          self.viewers[0].msg.style.display = '';
          self.viewers[0].cannotEdit();
        }
      }
    }
    return self.may_edit;
  }

}; // end IA

// Backwards compatibility
function getElementsByClassName (scope, tag, className) {
    if (window.jQuery) {
        return jQuery(scope).find(((!tag || tag === '*') ? '' : tag) + '.' + className);
    } else {
        // For non-WMF wikis that might not have jQuery (yet), use the wikibits.js getElementsByClassName
        return getElementsByClassName(scope, tag, className);
    }
}

window.ImageAnnotator = {
  install: function (config) { IA.install(config); }
};


// Start it. Bypass caches; but allow for 4 hours client-side caching. Small file.
IA.getScript(
   mw.config.get('wgScript') + '?title=MediaWiki:ImageAnnotatorConfig.js&action=raw&ctype=text/javascript'
 , true // No local caching!
);



})(); // end local scope

} // end if (guard against double inclusions)

// </source>
```