Image Map Plugin With Marker

Vintage World Map CON1 Asia @ 434,106,30 CON2 Africa @ 308,176,30 CON3 North America @ 112,116,30 CON4 South America @ 186,218,30 CON5 Antarctica @ 291,334,30 CON6 Europe @ 327,111,30 CON7 Australia @ 484,242,30 map

Asia | Africa | North America | South America | Antarctica | Europe | Australia | **← CLICK HERE!** { „marker-color“: „tomatored“, „marker-width“: 21, „marker-height“: 32 }

Part of Central Park CP001 Seneca Village @ 329,132,30 CP002 Arthur Ross Pinetum @ 490,222,30 CP003 Diana Ross Playground @ 99,364,30 CP004 Summit Rock @ 152,247,213,283 map

Arthur Ross Pinetum | Diana Ross Playground | Seneca Village | Summit Rock **← CLICK HERE!** { „marker“: „imagemapping:marker.002.png“, „marker-width“: 20, „marker-height“: 32 }

Tompkins Square Park TS01 Tompkins Square Pool @ 357,180,15 TS02 Pets Playground @ 311,206,365,232,348,256,301,222 map

Tompkins Square Pool | Pets Playground **← CLICK HERE!** { „marker“: „https://upload.wikimedia.org/wikipedia/commons/f/f2/678111-map-marker-512.png“, „marker-width“: 32, „marker-height“: 32 }

About

If you change the original script.js in the plug-in **dokuwiki-plugin-imagemap**, you can add clickable markers on images. This may be very useful for maps.

A word to the author of dokuwiki-plugin-imagemap, Gerry Weißbach: Feel free to merge this code with your project. Other parts of your software are unchanged. I repaired the window.resize handler, in case your viewport grows.

Used plug-in: **wrap**

How to use?

Add to the mappings (areas) an unique identifier. Unique in the sense, that only their lower case form are used. This is the **first word** in the title of the mapping.

E.g. `[[imagemapping:start|CP01 Seneca Village @ 329,132,30]]` The identifier is: `CP01` or technically: `cp01`. The coordinates (here) `329,132` are from the original image.

You now may add spans (references) with the CSS-class `wrap_imaploc` and with the ID set to the identifier. I've used the plug-in **wrap** to perform this.

Reference wiki code (with wrap): `<wrap imaploc #CP01>Seneca Village</wrap>` / Resulting HTML-Code: `<span class="wrap_imaploc" id="cp01">Seneca Village</span>`

You may use multiple references for the same identifyer. You can place references at anyplace in the page. There is no rule to place them right behind the image mapping.

In case you click the reference, a marker is shown at the given point. Works with all three forms of mapping: circle, rectangle and polygon.

This Page Source

```
====== Image Map Plugin With Marker ======

{{map>imagemapping:part_of_central_park.png|Part of Central Park}}
[[imagemapping:start|CP01 Seneca Village @ 329,132,30]]
[[imagemapping:start|CP02 Arthur Ross Pinetum @ 490,222,30]]
[[imagemapping:start|CP03 Diana Ross Playground @ 99,364,30]]
[[imagemapping:start|CP04 Summit Rock @ 152,247,213,283]]
{{<map}}

<wrap imaploc #CP02>Arthur Ross Pinetum</wrap> |
<wrap imaploc #CP03>Diana Ross Playground</wrap> |
<wrap imaploc #CP01>Seneca Village</wrap> |
<wrap imaploc #CP04>Summit Rock</wrap>
<wrap hi>**<- CLICK HERE!**</wrap>

{{map>imagemapping:tompkins_square_park.png|Tompkins Square Park}}
[[imagemapping:start|TS01 Tompkins Square Pool @ 357,180,15]]
[[imagemapping:start|TS02 Pets Playground @
311,206,365,232,348,256,301,222]]
{{<map}}

<wrap imaploc #TS01>Tompkins Square Pool</wrap> |
<wrap imaploc #TS02>Pets Playground</wrap>
<wrap hi>**<- CLICK HERE!**</wrap>
```

Javascript Replacement Source (aka. the Plug-Ins „script.js“)

script.js

```
/* DOKUWIKI:include_once jquery.imagemapster.js */


// Original source code:
https://github.com/i-net-software/dokuwiki-plugin-imagemap
globalThis[Symbol.for('imap_lib')] = (function () {
  var defaults = {
      'clicked_reference_color': "#00AEEF",
```

```
      'marker_color': "#00AEEF",
      'debug': false  // false = no debug on console
    };
  var a_maps = [];
  var a_areas = [];
  var a_references = [];
  var resize_timeout = null;
  var a_last_clicked_id = [];
  var a_imap_div = [];
  var imap_div_interval = null;
  var nr_startup_intervals = 0;

  function make_svg_and_append_to_jquery_object(html, object) {
    object[0].insertAdjacentHTML(
      "beforeEnd",
      html
    );
  }  // function make_svg_and_append_to_jquery_object

  function calc_marker_pos(coords) {
    let split_coords = coords.split(",");
    var x = 0;
    var y = 0;
    switch(split_coords.length) {
      case 3:  // circle
        x = parseInt(split_coords[0]);
        y = parseInt(split_coords[1]);
        break;
      case 4:  // rectangle
        x =
Math.floor((parseInt(split_coords[0])+parseInt(split_coords[2]))/2);
        y =
Math.floor((parseInt(split_coords[1])+parseInt(split_coords[3]))/2);
        break;
      default:
        if(split_coords.length >= 6) {
          var num_coords = 0;
          for(var i = 0; i < (split_coords.length - 1); i += 2) {
            x += parseInt(split_coords[i]);
            y += parseInt(split_coords[i + 1]);
            num_coords++;
          }
          x = Math.floor(x / num_coords);
          y = Math.floor(y / num_coords);;
        }
    }
    return [ x, y ];
  }  // function calc_marker_pos

  function get_id_from_string(s) {
    const a_ids = s.split(" ");
```

```
    if (a_ids.length >= 1) {
      if (a_ids[0].length > 0) {
        return "#"+String(a_ids[0]).toLowerCase();
      }
    }
    return null;
  }

  function find_area_by_id(id, imap_index = null) {
    try {
      if (imap_index === null) {
        for(var i = 0; i < a_areas.length; i++) {
          if (a_areas[i] !== undefined) {
            for(var j = 0; j < a_areas[i].length; j++) {
              const area = a_areas[i][j];
              if (area['id'] == id) {
                return [ i, area['area'], j ];
              }
            }
          }
        }
      } else {
        if (a_areas[imap_index] !== undefined) {
          for(var j = 0; j < a_areas[imap_index].length; j++) {
            const area = a_areas[imap_index][j];
            if (area['id'] == id) {
              return [ imap_index, area['area'], j ];
            }
          }
        }
      }
    } catch(e) { console.error("EXCEPTION="+e); }
    return [ -1, null, -1 ];
  }  // function find_area_by_id

  return {
    // exported variables:
    defaults: defaults,
    a_maps: a_maps,
    a_areas: a_areas,
    a_references: a_references,
    resize_timeout: resize_timeout,
    a_last_clicked_id: a_last_clicked_id,
    a_imap_div: a_imap_div,
    imap_div_interval: imap_div_interval,
    nr_startup_intervals: nr_startup_intervals,
    // exported functions:
    make_svg_and_append_to_jquery_object:
make_svg_and_append_to_jquery_object,
    calc_marker_pos: calc_marker_pos,
```

```
    get_id_from_string: get_id_from_string,
    find_area_by_id: find_area_by_id
  };
})();


addEventListener("DOMContentLoaded", (event) => {
  (function($) {
    var _g = globalThis[Symbol.for('imap_lib')];

    $('img[usemap]').mapster({
      fillColor: 'ffffff',
      fillOpacity: 0.3,
      wrapClass: true,
      wrapCss: true,
      clickNavigate: true
    });

    function get_marker_width_and_height(id) {
      return [ $(id).width(), $(id).height() ];
    }  // function get_marker_width_and_height

    function do_marker_if_resize() {
      _g.a_imap_div.forEach((object, index) => {
        let marker_id_jquery = "#imap-marker-"+index;
        if(_g.defaults['debug']) { console.log("RESIZE::["+index+"]
LCID="+_g.a_last_clicked_id[index]); }
        if((_g.a_last_clicked_id[index] !== undefined) &&
(_g.a_last_clicked_id[index]['id'] !== undefined) &&
$(marker_id_jquery).is(":visible")) {
          let id = _g.a_last_clicked_id[index]['id'];
          let area_index = _g.a_last_clicked_id[index]['area_index'];
          if(_g.defaults['debug']) { console.log("RESIZE::["+index+"]
AREA-INDEX="+area_index); }
          try {
            let found_area = _g.a_areas[index][area_index]['area'];
            let marker_id_jquery = "#imap-marker-"+index;
            let coords = found_area.attr("coords");
            let xy = _g.calc_marker_pos(coords);
            let wh = get_marker_width_and_height(marker_id_jquery);
            $(marker_id_jquery).css({ top: xy[1] - wh[1] + 3, left:
xy[0] - (wh[0] / 2) });
          } catch(e) { console.error("EXCEPTION="+e); }
        }
      });
    }  // do_marker_if_resize

    $(window).resize(function() {
      if (_g.resize_timeout != null) { clearTimeout(_g.resize_timeout);
}
      _g.resize_timeout = setTimeout(function() {
```

```
        $('img[usemap]').each(function() {
          let parent = $(this.offsetParent);
          let parentparent = $(parent).parent();
          //if (_g.defaults['debug']) {
          //  console.log("PARENTPARENT
TAG="+parentparent.prop("tagName")+" ID="+parentparent.attr("id"));
          //  console.log("PARENT TAG="+parent.prop("tagName")+"
ID="+parent.attr("id"));
          //  console.log("THIS TAG="+$(this).prop("tagName")+"
ID="+$(this).attr("id"));
          //}
          // limit image width to naturalWidth:
          $(this).mapster('resize', ($(this)[0].naturalWidth <
parentparent.width()) ? $(this)[0].naturalWidth :
parentparent.width());
        });
        do_marker_if_resize();
      }, 100);
    });

    _g.imap_div_interval = setInterval(function() {
      if(_g.defaults['debug']) { console.log("START IMAGEMAPPING
MARKER"); }
      _g.nr_startup_intervals++;
      if (_g.nr_startup_intervals >= 50) {
        // stop after 5 s searching:
        clearInterval(_g.imap_div_interval);
        if(_g.defaults['debug']) { console.log("GIVE UP IMAGEMAPPING
SEARCH"); }
        return;
      }
      // find container and maps:
      var imap_index = 0;
      $(".imap, map").each(function(index, object) {
        let tag = $(this).prop("tagName").toLowerCase();
        if ((tag == "div") && ($(this).hasClass("imap"))) {
          if(_g.defaults['debug']) { console.log("["+imap_index+"] THIS
TAG="+$(this).prop("tagName")+" ID="+$(this).attr("id")); }
          _g.a_imap_div[imap_index] = $(this);
        } else {
          if ((tag == "map") && (_g.a_imap_div[imap_index] !==
undefined) && (_g.a_maps[imap_index] === undefined)) {
            _g.a_maps[imap_index] = $(this);
            if(_g.defaults['debug']) { console.log("["+imap_index+"]
THIS TAG="+$(this).prop("tagName")+" ID="+$(this).attr("id")+"
NAME="+$(this).attr("name")); }
            imap_index++;
          }
        }
      });
```

```
      if ((_g.a_imap_div[0] !== undefined) && (_g.a_imap_div[0] !==
null)) {
        // clear search interval:
        clearInterval(_g.imap_div_interval);
        _g.a_imap_div.forEach((object, index) => {
          // set z-index for images:
          if(_g.defaults['debug']) { console.log("["+index+"] Z-INDEX
THIS TAG="+object.prop("tagName")+" ID="+object.attr("id")+"
NAME="+object.attr("name")); }
          object.find("img").css("z-index", "0");
          // create marker:
          let marker_id = "imap-marker-"+index;
          let svg_marker = '<svg version="1.1" id="'+marker_id+'"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" viewBox="0 0
365 560" enable-background="new 0 0 365 560" xml:space="preserve"> ' +
            '<g> ' +
            '  <path class="svg-marker-path" fill="#00AEEF"
d="M182.9,551.7c0,0.1,0.2,0.3,0.2,0.3S358.3,283,358.3,194.6c0-130.1-88.
8-186.7-175.4-186.9 ' +
            '
C96.3,7.9,7.5,64.5,7.5,194.6c0,88.4,175.3,357.4,175.3,357.4S182.9,551.7
,182.9,551.7z M122.2,187.2c0-33.6,27.2-60.8,60.8-60.8 ' +
            '
c33.6,0,60.8,27.2,60.8,60.8S216.5,248,182.9,248C149.4,248,122.2,220.8,1
22.2,187.2z"/> ' +
            '</g> ' +
            '</svg>';
          _g.make_svg_and_append_to_jquery_object(svg_marker, object);
          let marker_id_jquery = "#"+marker_id;
          $(marker_id_jquery).css("position", "absolute");
          $(marker_id_jquery).css("z-index", 1000);
          $(marker_id_jquery).css("width", 21);
          $(marker_id_jquery).css("height", 32);
          $(marker_id_jquery).css("top", -32);
          $(marker_id_jquery).css("left", -32);
          $(".svg-marker-path", $(marker_id_jquery)).attr("style",
"fill:"+_g.defaults['marker_color']);
          $(marker_id_jquery).hide();
          if(_g.defaults['debug']) { console.log("["+index+"] ADD
MARKER ID='"+marker_id+"'"); }
          // collect areas:
          if (_g.a_maps[index] !== undefined) {
            _g.a_maps[index].find("area").each(function(area_index,
object) {
              if (_g.a_areas[index] === undefined) {
                _g.a_areas[index] = [];
              }
              const area_id =
_g.get_id_from_string($(this).attr("title"));
              _g.a_areas[index].push({ 'id': area_id, 'area': $(this)
```

```
});
              if(_g.defaults['debug']) { console.log("["+index+"] ADD
AREA #"+area_index+" ID='"+area_id+"'"); }
            });
          }
        });
        // search for references:
        $(".wrap_imaploc").each(function(index, object) {
          try {
            var id = "";
            if ($(this).attr("id").length > 0) {
              id = "#"+$(this).attr("id").toLowerCase();
            } else {
              const html = $(this).html();
              const a_ids = html.split(" ");
              id = "#"+String(a_ids[0]).toLowerCase();
            }
            if (id.length > 0) {
              if(_g.defaults['debug']) { console.log("FOUND WRAP-
IMAPLOC ID='"+id+"'"); }
              // find corresponding area
              let ia = _g.find_area_by_id(id);
              let imap_index = ia[0];
              let found_area = ia[1];
              let area_index = ia[2];
              if (found_area != null) {
                if ($(id).length) {
                  if (_g.a_references[imap_index] === undefined) {
                    _g.a_references[imap_index] = [];
                  }
                  _g.a_references[imap_index].push({ 'id': id,
'imap_index': imap_index, 'area_index': area_index, 'reference':
$(this) });
                  if(_g.defaults['debug']) { console.log("FOUND AREA
FOR WRAP-IMAPLOC ID='"+id+"' AREA IMAP-INDEX="+imap_index+" AREA-
INDEX="+area_index); }
                  $(this).bind("click", { 'area': found_area, 'id': id,
'imap_index': imap_index, 'area_index': area_index }, function(e) {
                    var data = e.data;
                    let imap_index = data['imap_index'];
                    let marker_id_jquery = "#imap-marker-"+imap_index;
                    let area_index = data['area_index'];
                    let id = data['id'];
                    if(_g.defaults['debug']) { console.log("CLICK
ID='"+id+"' AREA IMAP-INDEX="+imap_index+" AREA-INDEX="+area_index); }
                    if ((_g.a_last_clicked_id[imap_index] !==
undefined) && (_g.a_last_clicked_id[imap_index]['id'] == data['id']) &&
$(marker_id_jquery).is(":visible")) {
                      // hide marker
                      $(marker_id_jquery).hide();
```

```
                        _g.a_references[imap_index].forEach((object,
index) => {
                          if (object['id'] == id) {
                            object['reference'].css('font-weight',
'normal');
                            object['reference'].css('color', '');
                          }
                        });
                        $(this).css('font-weight', 'normal');
                        $(this).css('color', '');
                      } else {
                        // show marker
                        let coords = data['area'].attr("coords");
                        let xy = _g.calc_marker_pos(coords);
                        let wh =
get_marker_width_and_height(marker_id_jquery);
                        $(marker_id_jquery).css({ top: xy[1] - wh[1] + 3,
left: xy[0] - (wh[0] / 2) });
                        $(marker_id_jquery).show();
                        _g.a_references[imap_index].forEach((object,
index) => {
                          if (object['id'] == id) {
                            object['reference'].css('font-weight',
'bold');
                            object['reference'].css('color',
_g.defaults['clicked_reference_color']);
                          } else {
                            object['reference'].css('font-weight',
'normal');
                            object['reference'].css('color', '');
                          }
                        });
                      }
                      if(_g.defaults['debug']) {
console.log("["+imap_index+"] LAST CLICKED ID='"+data['id']+" MARKER-
ID='"+marker_id_jquery+"'"); }
                      _g.a_last_clicked_id[imap_index] = { 'imap_index':
imap_index, 'area_index': area_index, 'id': data['id'] };
                    });
                    $(this).css('cursor', 'pointer');
                  }
                }
              }
            } catch(e) { console.error("EXCEPTION="+e); }
          });
        }
      }, 100);
    })(jQuery);
});

function addBtnActionImagemap(btn, props, edid) {
```

```
  // Not implemented yet
}
```