University of Southampton

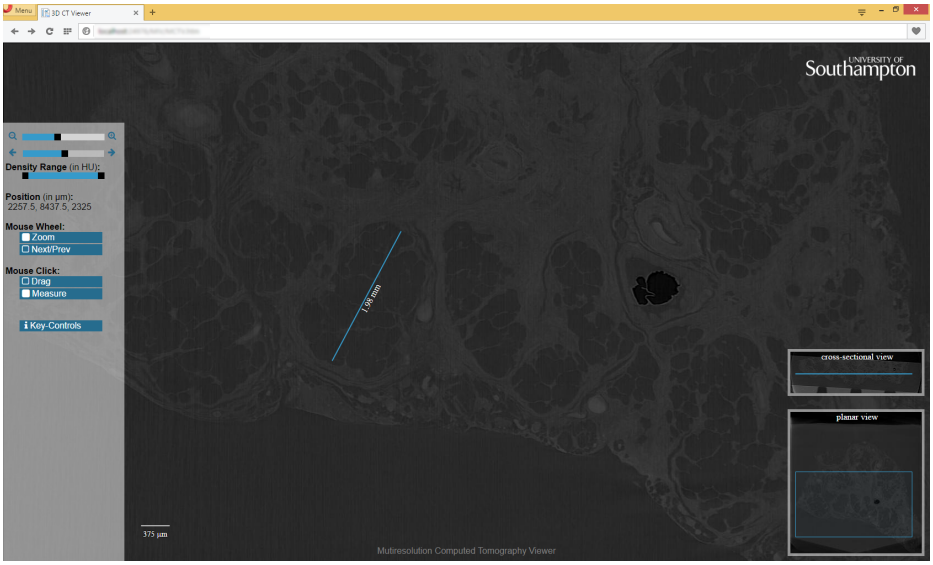
Faculty of Engineering and the Environment

Documentation of the Multiresolution Computed Tomography Viewer

(MCTV version 1)

by Lasse Wollatz

October 2016



Contents

About MCTV

The Multiresolution Computed Tomography Viewer, MCTV, allows large images from a micro-CT to be displayed over the web, whilst being easy to set up and not requiring any plug-ins to be enabled apart from JavaScript which is by default enabled on all major browsers. It allows easy navigation through a 3D image as well as modification of the threshold using JavaScript canvas.
This documentation will explain how to set up MCTV and also contains a full copy of the source code as well as some screenshots to show the expected output.

Requirements

- Webserver (e.g. Apache Tomcat or Microsoft IIS
- Webbrowser on Client (preferably HTML5/CSS3 ready) with JavaScript enabled
  - Webbrowsers tested:
    - Internet Explorer 9/10/11
    - Opera 39
    - Firefox 36/43/47
    - Android Browser (version unknown)
  - Expected Compatibility:
    - Safari 8/9/10
    - Google Chrome
    - mobile Safari (old version tested on iOS 5.1)

Preparation

Setting Up the Files

1. Place MCTV.htm, MCTV.js and MCTV.css into a directory on your server
2. Place loading.gif, temp.png and controls.png into the same directory
3. If you *don't* want to use font-awesome:
   - place next.png, prev.png, zoomin.png, zoomout.png into the same directory as the html file
   - in MCTV.css change the following lines

     ```
     .nofa{
         display: none;
     }
     ```

     to

```
.nofa{
    display: block;
}
```

- in MCTV.html delete the line loading font-awesome

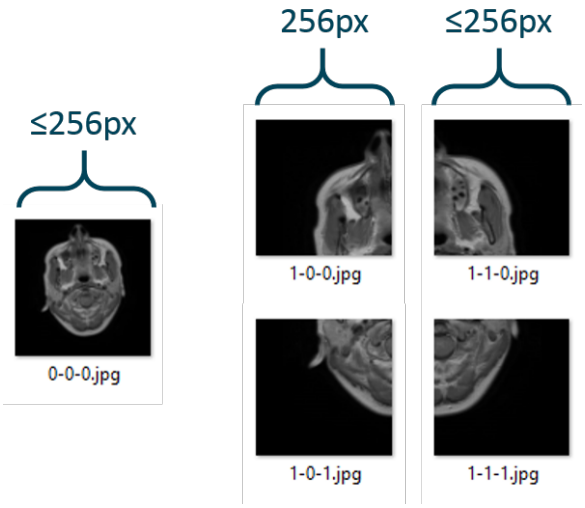
```
<link rel="stylesheet" href="http://.../css/font-awesome.min.css">
```

4. If you *do* want to use font-awesome:
   - get the font-awesome icons from `http://fontawesome.io/`
   - in MCTV.html replace the link to load your own font-awesome icons

```
<link rel="stylesheet" href="http://.../css/font-awesome.min.css">
```

Preparing the Images

1. Create Tiles
   - convert the 3D image into many 2D slices (usually already the case)
   - for each 2D slice
     - create 256x256 tiles starting from the top left
     - half the image in size and repeat until the image is $\leq$ 1 tile
     - save the images as zoom-y-x.jpg where zoom is your zoom level and x,y,z the coordinates in units of 1 tile as illustrated in this graphics:



     - create a new folder for each image

2. Create cross-sectional view
   - collate the centre row from each 2D slice into one image
   - rescale the image width to

$$\frac{\text{image width in pixel}}{2^{\text{max zoom level}}}$$

   and the image height to

$$\frac{\text{count of 2D slices}}{2^{\text{max zoom level}}}$$

- save the image as tc.jpg

3. Create JSON

- The generic structure of the JSON file is as follows

Listing 1: sample JSON file

```
{"height":2048,
 "width":1904,
 "resunit":"&mu;m",
 "res":0.4492,
 "zres":2.7670,
 "densmin":0.0,
 "densmax":1255.60,
 "slides": [
{"path": "../MCTV/exampleblock/_0000/", "height":2048, "
   width":1904},
{"path": "../MCTV/exampleblock/_0001/", "height":2048, "
   width":1904},
{"path": "../MCTV/exampleblock/_0587/", "height":2048, "
   width":1904},
{"path": "../MCTV/exampleblock/_0588/", "height":2048, "
   width":1904},
 ] }
```

- *slides* is a list of all the image slices/slides, where
  - *path* is a link to the folder containing the tiles
  - *height* is the image height
  - *width* is the image width
  - *label* (optional) is a link to a JSON containing image labels.

Listing 2: sample JSON file for labels

```
{ "labels": [
{"label": "label␣text␣displayed", "name": "optional␣
   tooltip␣of␣link", "url": "http://my_optional/url" , "
   x": "0.749", "y": "0.262"},
{"label": "UoS", "name": "University␣of␣Southampton", "
   url": "http://www.southampton.ac.uk", "x": "0.5", "y"
   : "0.5"},
{"label": "X", "x": "0.33", "y": "0.665"},
] }
```

- *height* is the height of the slides in pixel
- *width* is the width of the slides in pixel
- *resunit* is the unit to be used (defaults to mm)
- *res* is the resolution of the slice in resunit/pixel (same as the DICOM tag (0028,0030) "Pixel Spacing", note that x and y resolution must be identical!)
- *zres* is the resolution between the slices in resunit/pixel (same as the DICOM tag (0018,0088) "Spacing Between Slices")
- *densmin* is the Hounsefield unit (HU) of a black pixel

$$\text{densmin} = \text{minimum pixel value} * \text{DICOM}(0028, 1053) + \text{DICOM}(0028, 1052)$$

assuming that the minimum pixel value will be mapped to 0 when converting to 8-bit. If you use a different conversion method, adjust the formula accordingly.

- *densmax* is the Hounsefield unit (HU) of a white pixel

  densmax = maximum pixel value∗DICOM(0028, 1053)+DICOM(0028, 1052)

  assuming that the maximum pixel value will be mapped to 255 when converting to 8-bit. If you use a different conversion method, adjust the formula accordingly.

- save the file as infoJSON.txt

4. make sure that tc.jpg and infoJSON.txt are located in the same folder. It might be easiest to place the folders with the tiles as subfolders to this one.

Putting it All Together

1. Make sure all your files are located on your server
2. Make sure your server is running
3. Navigate your browser to MCTV.html on your server
4. Add `?root=../myimages/exampleblock/` to the end of the URL, specifying the path to your infoJSON.txt file
5. Your image should now be displayed!
6. You can now connect MCTV to the rest of your website by dynamically providing the root in the URL.

Screenshots

Here are some screenshots to show how MCTV looks in various browsers.

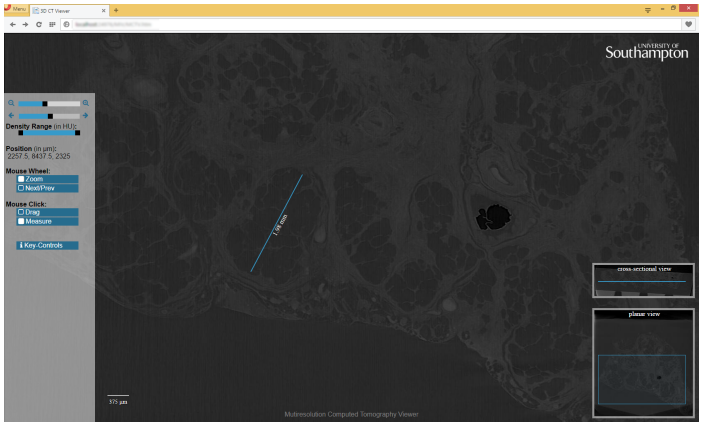


Figure 1: Screenshot of MCTV in Opera 39 - µCT image

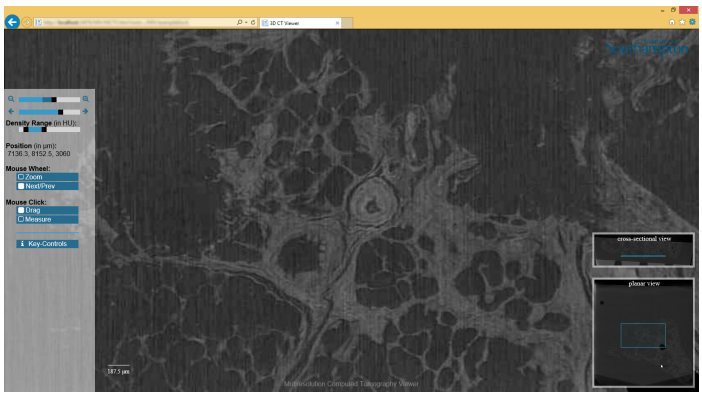


Figure 2: Screenshot of MCTV in Internet Explorer 9 - adapted density range

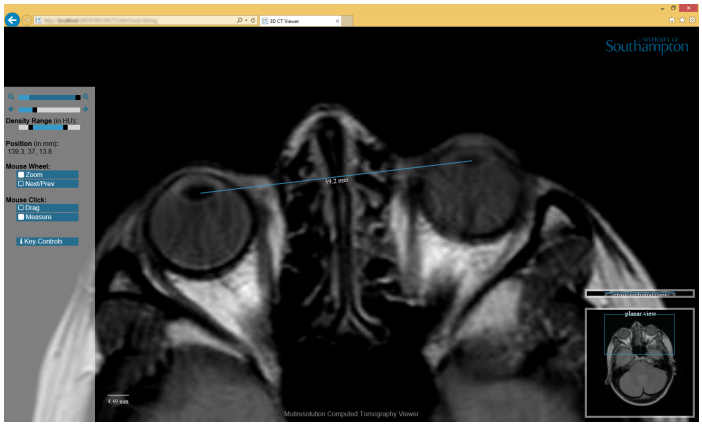


Figure 3: Screenshot of MCTV in Internet Explorer 11 - a standard CT

Figure 4: Screenshot of MCTV in Internet Explorer 11 tile mode - showing key controls

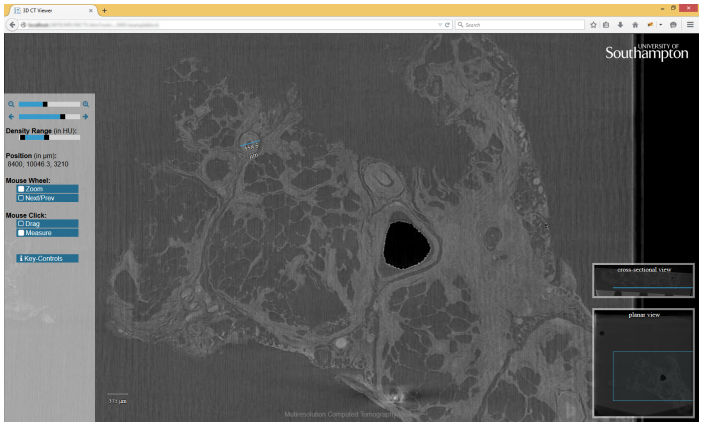


Figure 5: Screenshot of MCTV in Firefox 36

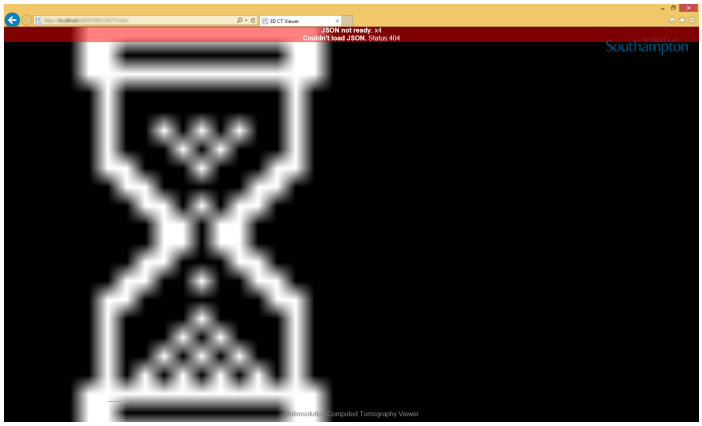


Figure 6: Screenshot of MCTV in Internet Explorer 11 - Error displayed in case of missing JSON file

Figure 7: Screenshot of MCTV in Firefox 47 - Error displayed in case of invalid JSON file

References

MCTV has first been presented in
L. Wollatz, S. J. Cox, S. J. Johnston, (2015) "Web-Based Manipulation of Multiresolution Micro-CT Images.", IEEE e-Science.
`http://dx.doi.org/10.1109/eScience.2015.42`
and further been published as
L. Wollatz, (2016) "Multiresolution Computed Tomography Viewer.", [dataset].
`http://dx.doi.org/10.5258/SOTON/400332`

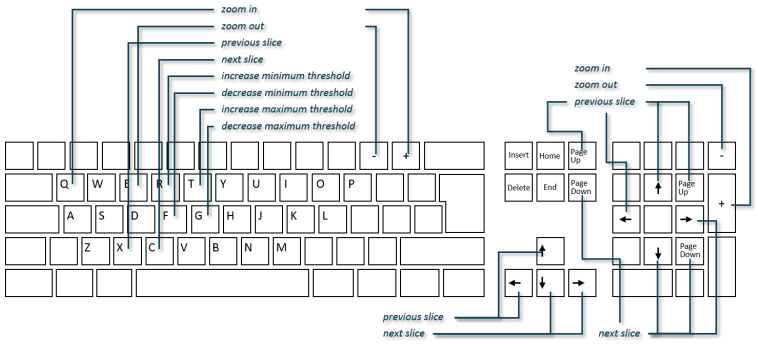
MCTV is based on the Multiresolution Image Viewer (MIV), now under the name of BrainMaps API. The latest version of the BrainMaps API can be found on `http://www.connectomes.org`.

Font Awesome (`http://fontawesome.io`) has been used for better looking icons. Even though Font Awesome can be turned off, it adds a nice feel to the page.

Source Code

Image Files

(a) loading.gif



(b) controls.png

Figure 8: Image files required for MCTV.

(a) prev.png

(b) next.png

(c) zoomin.png

(d) zoomout.png

Figure 9: Image files additionally required if not using font awesome.

HTML Code

Listing 3: code for MCTV.htm

```
1 <!DOCTYPE html>
2 <html style="height: 100%;">
3 <head>
4 <meta content="text/html;charset=utf-8" http-equiv="Content-Type">
5 <meta content="utf-8" http-equiv="encoding">
```

```
6  <meta name="author" content="Lasse␣Wollatz">
7  <meta name="date" content="2015-02-18">
8  <meta name="description" content="multi-resolution,␣tile-based␣3D␣CT␣
       image␣viewer">
9  <meta name="viewport" content="width=device-width,␣initial-scale=1">
10 <title>3D CT Viewer</title>
11 <!--style-->
12 <link rel="stylesheet" href="http://fontawesome.io/assets/font-
       awesome/css/font-awesome.min.css">
13 <link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/
       smoothness/jquery-ui.css">
14 <link rel="stylesheet" href="MCTV.css" type="text/css">
15 <!--javascript-->
16 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min
       .js"></script>
17 <script src="//code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
18 <script src="MCTV.js" type="text/javascript"></script>
19 <!-- end head-->
20 </head>
21
22 <body>
23     <noscript>
24         <blockquote>
25             <br/><br/><br/>
26             <b>JavaScript must be enabled in order for you to use the
                    Mutiresolution Computed Tomography Viewer.</b> 
27             However, it seems JavaScript is either disabled or not
                   supported by your browser. 
28             To view this page, enable JavaScript by changing your
                   browser options, and then try again.
29         </blockquote>
30     </noscript>
31     <div id="outerDiv0">
32
33         <div id="outerDiv" style="position:relative;␣height:100%;␣
               width:100%;␣border:1px␣solid␣black;␣overflow:hidden;">
34
35 <!--Reference-->
36             <div id="copy" style="text-shadow:none;␣text-align:center
                   ;␣position:fixed;␣bottom:10px;␣left:0px;␣width:100%;␣
                   z-index:2;">
37                 <a href="http://dx.doi.org/10.5258/SOTON/400332"
                       target="_blank" style="width:100%;">
38                     Mutiresolution Computed Tomography Viewer
39                 </a>
40             </div>
41
42             <div id="innerDiv" style="position:relative;␣top:0px;␣
                   left:0px;␣z-index:0;">
43 <!--Image Tiles-->
44                 <div id="imageTiles" style="position:relative;␣top:0;
                       ␣left:0;␣z-index:0;␣width:100%;">
45                     <a id="mainTileA">
46                         <img alt="busy␣loading" src="loading.gif" id=
                               "mainTile" style="position:␣absolute;␣
                               left:␣0px;␣top:␣0px;␣z-index:␣-1;␣width:␣
                               1000px;">
47                     </a>
48                 </div>
49 <!--Labels-->
```

```
50                  <div id="imageLabels" style="position:relative;␣top
                        :0;␣left:0;␣z-index:1;␣width:900000px;␣height
                        :900000px;">
51                  </div>
52              </div>
53
54  <!--Rule (during measurement)-->
55              <div id="ruler" class="ruler␣rulerlbl" style="display:
                    none;␣position:absolute;␣top:510px;␣left:20px;␣width
                    :160px;␣height:0px;"></div>
56  <!--Scale-->
57              <div style="position:fixed;␣bottom:25px;␣left:250px;␣font
                    -size:14px;␣text-align:center;">
58                  <hr id="scale" style="width:50px;">
59                  <div id="theScale"  class="rulerlbl"></div><br>
60              </div>
61  <!--University of Southampton Logo-->
62              <div style="position:fixed;␣top:25px;␣right:25px;␣width
                    :30%;␣max-width:200px;">
63                  <a href="http://www.southampton.ac.uk" target="_blank
                        ">
64                      <!--<img style="width:100%;" src="sotonlogo.gif"/
                            >-->
65                      <img style="width:100%;" class="whitelogo" src="
                            https://upload.wikimedia.org/wikipedia/
                            commons/thumb/a/a7/
                            University_of_Southampton_Logo.svg/2000px-
                            University_of_Southampton_Logo.svg.png"/>
66                  </a>
67              </div>
68          </div>
69
70  <!--Toolbox-->
71          <div id="overlay" style="top:145px;␣left:-250px;␣width:220px;
                ">
72              <br/>
73              <div id="zoom">
74                  <div class="zoom␣icon␣left">
75                      <a class="tooltip" title="Zoom␣In">
76                          <span id="zoomouticonfa" class="fa␣fa-search-
                                minus" title="Zoom␣Out"></span>
77                          <img id="zoomouticon" class="nofa" src="
                                zoomout.png" alt="Zoom␣Out"/>
78                      </a>
79                  </div>
80                  <div class="zoom␣slider" id="zoomslider"></div>
81                  <div class="zoom␣icon␣right">
82                      <a class="tooltip" title="Zoom␣In">
83                          <span id="zoominiconfa" class="fa␣fa-search-
                                plus" title="Zoom␣In"></span>
84                          <img id="zoominicon" class="nofa" src="zoomin
                                .png" alt="Zoom␣In"/>
85                      </a>
86                  </div>
87              </div><br/>
88
89              <div id="slices">
90                  <div class="slices␣icon␣left">
91                      <a id="slidePrev" class="tooltip" title="Previous
                            ␣Slice">
```

```
92                      <span id="sliceprevicon fa" class="fa fa-arrow
                            -left" title="Previous Slice"></span>
93                      <img id="sliceprevicon" class="nofa" src="
                            prev.png" alt="Previous Slice"/>
94                  </a>
95              </div>
96              <div class="slices slider" id="sliceslider"></div>
97              <div class="slices icon right">
98                  <a id="slideNext" class="tooltip" title="Next 
                        Slice">
99                      <span id="slicenexticon fa" class="fa fa-arrow
                            -right" title="Next Slice"></span>
100                     <img id="slicenexticon" class="nofa" src="
                            next.png" alt="Next Slice"/>
101                 </a>
102             </div>
103         </div><br/><br/>
104
105         <div id="thres">
106             <b> Density Range</b> (in HU)<b>:</b><br/>
107             <div class="thres slider" id="thresslider"></div>
108         </div><br/>
109
110         <div><br/></div>
111
112         <div id="coords">
113         </div><br/>
114
115         <div id="wheelMode">
116             <b id="wmlbl"> Mouse Wheel:</b><br/>
117             <div id="chkMW1div" class="optionbtnlbl">
118                 <input id="chkMW1" class="nofa" name='wheelmode'
                        type="radio" checked>
119                  <span id="chkMW1fa" class="fa fa-fw fa-
                        square"></span> Zoom
120             </div>
121             <div id="chkMW2div" class="optionbtnlbl">
122                 <input id="chkMW2" class="nofa" name='wheelmode'
                        type="radio">
123                  <span id="chkMW2fa" class="fa fa-fw fa-
                        square-o"></span> Next/Prev
124             </div>
125         </div><br/>
126
127         <div id="clickMode">
128             <b id="cmlbl"> Mouse Click:</b>
129             <div id="chkMC1div" class="optionbtnlbl">
130                 <input id="chkMC1" class="nofa" name='clickmode'
                        type="radio" checked>
131                  <span id="chkMC1fa" class="fa fa-fw fa-
                        square"></span> Drag
132             </div>
133             <div id="chkMC2div" class="optionbtnlbl">
134                 <input id="chkMC2" class="nofa" name='clickmode'
                        type="radio">
135                  <span id="chkMC2fa" class="fa fa-fw fa-
                        square-o"></span> Measure
136             </div>
137             <br/><div id="sleepRuler" class="ruler"></div>
138             <br/><!--
```

```
139                 <div id="chkMC3div" class="optionbtnlbl">
140                     <input id="chkMC3" class="nofa" name='clickmode'
                            type="radio">
141                      <span id="chkMC3fa" class="fa␣fa-fw␣fa-
                            square-o"></span> Place Label
142                 </div>-->
143             </div>
144
145             <div id="cntrlButton">
146                  <span class="fa␣fa-fw␣fa-info"></span><span
                        class="nofa">   </span> Key-
                        Controls
147             </div>
148
149             <div class="debug" id="debug"></div>
150             <div class="debug" id="debug2"></div>
151             <div class="debug" id="debug3"></div>
152         </div>
153
154 <!--Thumbnails-->
155         <div class="thumbnail" id="Thumb0" style="bottom:10px;␣right
                :10px;">
156             <div class="thumbimg" id="Thumb">
157                 <div style="position:absolute;␣width:100%;␣text-align
                        :center;"> planar view</div>
158                 <div><img id="timg" alt="thumbnail" src="temp.png"></
                        div>
159             </div>
160             <div class="thumbpos" id="Thumb2"></div>
161         </div>
162         <div class="thumbnail" id="Thumb0C" style="bottom:300px;␣
                right:10px;">
163             <div class="thumbimg" id="ThumbC">
164                 <div style="position:absolute;␣width:100%;␣text-align
                        :center;"> cross-sectional view</div>
165                 <div><img id="tcimg" alt="thumbnail" src="temp.png"><
                        /div>
166             </div>
167             <div class="thumbpos" id="Thumb2C"></div>
168         </div>
169
170     </div>
171
172 <!--Controls-->
173     <div id="controls" class="controls">
174         <div id="closeControls" class="close">
175             <span class="fa␣fa-close"></span>
176             <span class="nofa"><b>x</b></span>
177         </div>
178         <img src="controls.png"/>
179     </div>
180
181 <!--Errorbar-->
182     <div class="error" id="error">ERROR</div>
183
184 </body>
185
186 </html>
```

JavaScript

Listing 4: code for MCTV.js

```
/**** Multiresolution Computed Tomography Viewer (MCTV) ****
 *Contributions:
 * Further enhancements and cleaning of code                    by Lasse Wollatz; October, 2016; L.Wollatz@soton.ac.uk
 *    - improved error handling
 *    - error bar added for feedback
 *    - automatic adjustment of units added
 *    - outsourced styles into css
 *    - improved styles and backward compatibility
 * Bug fixes, 3D mods, threshold and enhancements written     by Lasse Wollatz; February, 2015; L.Wollatz@soton.ac.uk
 *
    Publication: L. Wollatz, S. J. Cox, and S. J. Johnston, (2015) Web-Based Manipulation of Multiresolution Micro-CT Images. e-Science 2015.
 *    - added tile loading
 *    - corrected placement of tiles
 *    - optimized tile loading to reduce delay
 *    - threshold modifications functionality added
 *    - zoom "underlay" to improve user experience
 *    - non-tiled zoom added
 *    - measuring functionality added
 *    - resolution and CT data loading from JSON
 *    - improved styles
 *    - added animations
 *    - sliders added for more feedback and improved navigation
 *    - cross-sectional thumbnail added
 *    - display and event handling of thumbnails corrected and enhanced
 * iPhone/iPad modifications written                            by Matthew K. Lindley; August 25, 2010
 * base code (MIV)                                              by Shawn Mikula; 2007; brainmaps@gmail.com.
 *
    Publication: S. Mikula, I. Trotts, J. M. Stone, and E. G. Jones, (2007) Internet-Enabled High-Resolution Brain Mapping and Virtual Microscopy. NeuroImage 35 (1): 9-15.
 *                                                                (Updated version of his script available at http://www.connectomes.org.)
 *    - main html structure
 *    - tile placement
 *    - basic zoom
 *    - basic navigation
 *
 *You are free to use this software for non-commercial
 *use only, and only if proper credit is clearly visibly
 *given wherever the code is used.
 ************************************************************/


var imgtype = ".jpg";           //filetype of image files, can be jpg,png,bmp should also work with gif NOT tif or dicom!
var tileSize = 256;             //size of a tile in pixel
/*variables that can be set through the url*/
//root                          //path to infoJson.txt -> requires
```

```
    either root or JSON!
43 //JSON                          //or directly specify the full path
    -> requires either root or JSON!
44 //start                         //number of first slice to display ->
     defaults to 1/3 of image stack
45 //vX                            //X position to center as fraction of
     the image width -> defaults to 0.5
46 //vY                            //Y position to center as fraction of
     the image height -> defaults to 0.5
47 //vT                            //initial zoom level -> defaults to
    maximum normal zoom
48 //coords                        //0 if no position information should
     be displayed -> defaults to 1
49 //width                         //declare image width if not defined
    in JSON
50 //height                        //declare image height if not defined
     in JSON
51 //res                           //declare image resolution if not
    defined in JSON
52 //zres                          //declare image z-resolution if not
    defined in JSON
53 //resunits                      //declare image resolution units if
    not defined in JSON
54 /*JSON*/
55 var JSONout;                    //content of JSON file
56 var getJSON = false;            //has JSON been requested?
57 var loadedJSON = false;         //has JSON file been loaded?
58 var JSON;                       //full path of JSON file including
    filename
59 var rootpath;                   //path where to check for infoJSON.
    txt
60 var labelspath;                 //full path for labels JSON file for
    current tile
61 /*positioning and sizing*/
62 var viewportWidth = 2000;       //width of viewable area in browser
63 var viewportHeight = 1000;      //height of viewable area in browser
64 var innerDiv;                   //innerDiv html element
65 var mTop;                       //current top position of image
    relative to viewport
66 var mLeft;                      //current left position of image
    relative to viewport
67 var gImageWidth, gImageHeight;  //width and height of image
68 var width, height;              //width and height of specific slice
    images
69 /*resolution/dimension*/
70 var resunits = "&mu;m";         //units of the resolution factors
71 var res = 1.0;                  //size of 1 pixel (resunits/px)
72 var zres = 1.0;                 //thickness of 1 slice (resunits/
    slice)
73 var JSONnum;                    //number of images (z dimension size)
74 /*zoom*/
75 var xtrazoomMax = 5;            //maximal additional zoom beyond
    actual size of the image
76 var gTierCount;                 //number of zoom levels due to tiling
77 var zoom = 0;                   //current zoom level
78 var xtrazoom = 1;               //current extra zoom beyond actual
    size of image
79 /*threshold*/
80 var thresLower;                 //Lower Threshold
81 var thresUpper;                 //Upper Threshold
```

```
82  var densmin = -1000;              //HU of minimum value (0)
83  var densmax = 1000;               //HU of maximum value (255)
84  /*mouse movement*/
85  var clickmode = 0;                //mode for mouse click/ drag
86  var lasteventX = 0;               //X coordiante from last known mouse
        movement
87  var lasteventY = 0;               //Y coordiante from last known mouse
        movement
88  var dragStartTop;                 //Y coordiante from mouse down event
89  var dragStartLeft;                //X coordiante from mouse down event
90  var dragging = false;             //if mouse-drag and mouse-click set
        to pan
91  var measuring = false;            //if mouse-drag and mouse-click set
        to measure
92  /*mouse wheel*/
93  var wheelmode = 0;                //mode for mousewheel movement
94  var wheelobs = 0;                 //count of mousewheel operations
95  /*touch*/
96  var touchIdentifier;
97  var gestureScale = 1;
98  /*code for checking performance*/
99  // try {
100 //  var Tstart = performance.now();
101 // } catch (err) {
102 //  var Tstart = new Date().getTime();
103 // }
104 ///try {
105 //  var Tend = performance.now();
106 // } catch (err) {
107 //  var Tend = new Date().getTime();
108 // }
109 // try {
110 //  var TendL = performance.now();
111 // } catch (err) {
112 //  var TendL = new Date().getTime();
113 // }
114 // try {
115 //   var TendR = performance.now();
116 // } catch (err) {
117 //   var TendR = new Date().getTime();
118 // }
119 /*others*/
120 var path = '/';                   //path to specific slice images
121 var imgpath = '';                 //path to specific tile
122 var slidePointer = 0;             //current image number (z dimension
        position)
123 var start;                        //first slice to load
124 var coords = 1;                   //if coordinates are displayed
125 var ActiveTask;                   //taskid of currently active task
126 var isControl = false;            //boolean if key controls are being
        displayed
127 var spaddingX = 20;               //horizontal artificial padding for
        slider tables
128
129
130 /*** BASICS ***
131  * logError(errstr)
132  * removeError(errstr)
133  * isNumeric(n)
134  * engUnit(number,unit)
```

```
135  * stripPx(value)
136  * getVar(name)
137  */
138 function logError(errstr) {
139 /*adds an error to the list of errors
140  *if Error already exists, a counter is used to save space
141  */
142     var a, b, c;
143     var remStr ="";
144     var ecntr = 2;
145     try {
146         var allErrStr = document.getElementById('error').innerHTML;
147         a = allErrStr.toLowerCase().indexOf(errstr.toLowerCase());
148         if (a < 0) {
149             document.getElementById('error').innerHTML += errstr + "<
                br/>";
150         } else {
151             b = allErrStr.toLowerCase().indexOf("<br", a);
152             var remStr = allErrStr.substring(a, b + 5);
153             c = allErrStr.toLowerCase().indexOf("␣x", b - 11);
154             if (c > 0 && c < b) {
155                 ecntr = parseFloat(allErrStr.substring(c + 2, b));
156                 ecntr += 1;
157             }
158             allErrStr = allErrStr.replace(remStr, errstr + "␣x" +
                ecntr + "<br/>");
159             document.getElementById('error').innerHTML = allErrStr;
160         }
161     } catch (err) {}
162 }
163
164 function removeError(errstr) {
165 /*removes an error from the list of errors
166  *if Error existed multiple times, all entrys are removed
167  *only the first part of the error message needs to be provided,
168  *  useful if part of the error message is variable
169  */
170     try {
171         var allErrStr = document.getElementById('error').innerHTML;
172         if(allErrStr != ""){
173             var a = allErrStr.indexOf(errstr);
174             if (a >= 0) {
175                 var b = allErrStr.indexOf("<br", a + 1);
176                 var remStr = allErrStr.substring(a, b + 5);
177                 allErrStr = allErrStr.replace(remStr, "");
178                 document.getElementById('error').innerHTML =
                    allErrStr;
179             }
180         }
181     } catch (err) {
182         logError("Failed␣to␣remove␣Error:␣" + err);
183     }
184 }
185
186 function isNumeric(n) {
187 /*checks if value is a number
188  */
189     return !isNaN(parseFloat(n)) && isFinite(n);
190 }
191
```

```
192 function engUnit(number,unit){
193 /*converts number and unit into human readable string
194  *unit has to be a metric unit for length as a string (nm to km)
195  *any other unit will default to meter
196  *unit will be changed if number is smaller than 0.1 or has more than
         3 digits.
197  */
198     var value = parseFloat(number);
199     var unitlevel = 0;
200     if (Math.abs(value) >= 0.1 && Math.abs(value) < 100){
201         return Math.round(value*100)/100 + "␣" + unit;
202     }
203     //detect zeros of unit
204     if (unit === "nm"){
205         unitlevel = -9;
206     }else if (unit === "&mu;m"){
207         unitlevel = -6;
208     }else if (unit === "mm"){
209         unitlevel = -3;
210     }else if (unit === "cm"){
211         unitlevel = -2;
212     }else if (unit === "dm"){
213         unitlevel = -1;
214     }else if (unit === "km"){
215         unitlevel = 3;
216     }
217     //get to next best unit
218     while(Math.abs(value) < 0.1 && unitlevel <= 3){
219         value = value*10;
220         unitlevel -= 1;
221     }
222     while(Math.abs(value) >= 100  && unitlevel >= -9){
223         value = value/10;
224         unitlevel += 1;
225     }
226     if (unitlevel == 2 || unitlevel == -4 || unitlevel == -7){
227         value = value/10;
228         unitlevel += 1;
229     }else if (unitlevel == 1 || unitlevel == -5 || unitlevel == -8){
230         value = value*10;
231         unitlevel -= 1;
232     }
233     //return string representation
234     if (unitlevel == -9){
235         unit = "nm";
236     }else if (unitlevel == -6){
237         unit = "&mu;m";
238     }else if (unitlevel == -3){
239         unit = "mm";
240     }else if (unitlevel == -2){
241         unit = "cm";
242     }else if (unitlevel == -1){
243         unit = "dm";
244     }else if (unitlevel == 0){
245         unit = "m";
246     }else if (unitlevel == 3){
247         unit = "km";
248     }
249     return Math.round(value*100)/100 + "␣" + unit;
250 }
```

```
251
252 function stripPx(value) {
253 /*converts pixel string into float
254  * therefore '128px' -> 128.0
255  */
256     if (value === "") {
257         return 0;
258     }
259     return parseFloat(value.substring(0, value.length - 2));
260 }
261
262 function getVar(name) {
263 /*reads a variable from the URL
264  */
265     var get_string = document.location.search;
266     var return_value = '';
267     var name_index = 0;
268     var end_of_value = -1;
269     var value = '';
270     do {
271         name_index = get_string.indexOf(name + '=');
272         if (name_index !== -1) {
273             get_string = get_string.substr(name_index + name.length +
                    1, get_string.length - name_index);
274             end_of_value = get_string.indexOf('&');
275             if (end_of_value !== -1) {
276                 value = get_string.substr(0, end_of_value);
277             } else {
278                 value = get_string;
279             }
280             if (return_value === '' || value === '') {
281                 return_value += value;
282             } else {
283                 return_value += ',␣' + value;
284             }
285         }
286     } while (name_index !== -1);
287     var space = return_value.indexOf('+');
288     while (space !== -1) {
289         return_value = return_value.substr(0, space) + '␣' +
               return_value.substr(space + 1, return_value.length);
290         space = return_value.indexOf('+');
291     }
292     return (return_value);
293 }
294 /*** END BASICS ***/
295
296
297 /*** TILE HANDLER ***
298  * getVisibleTiles()
299  * createTile(pCol, pRow, tileSize, tileName, staticPath)
300  * displayTile(pCol, pRow, tileName)
301  * checkTiles(isForced)
302  * refreshTiles()
303  * refreshUnderlay()
304  */
305 function getVisibleTiles() {
306 /*get x and y of all tiles that are in the visible area
307  *units of x and y are in tiles (starting at 0)
308  *Note: upper bound is not defined here but only within the
```

```
        checkTiles function
309  *      => wouldn't it be more sensible to check it here?
310  */
311     innerDiv = document.getElementById("innerDiv");
312     //need to get an absolute function here!
313     var mapX = stripPx(innerDiv.style.left);
314     var mapY = stripPx(innerDiv.style.top);
315     //changed from abs function to -1*.
316     //this way only the tiles on screen +-neighbours tiles are loaded
            .
317     var neighbours = 1; //minimum required is 1!
318     var startX = -1 * Math.floor(mapX / (tileSize * xtrazoom)) -
            neighbours;
319     var startY = -1 * Math.floor(mapY / (tileSize * xtrazoom)) -
            neighbours;
320     //+neighbours and not +neighbours+1, as the start of the image is
             recorded...
321     var tilesX = Math.ceil(viewportWidth / (tileSize * xtrazoom)) +
            neighbours;
322     var tilesY = Math.ceil(viewportHeight / (tileSize * xtrazoom)) +
            neighbours;
323     var visibleTileArray = [];
324     var counter = 0;
325     var x, y;
326     for (x = startX; x < (tilesX + startX); x++) {
327         for (y = startY; y < (tilesY + startY); y++) {
328             //need to add upper bound here (number of available tiles
                    in each direction is???)
329             if (x >= 0 && y >= 0) {
330                 visibleTileArray[counter++] = [x, y];
331             }
332         }
333     }
334     return visibleTileArray;
335 }
336
337 function createTile(pCol, pRow, tileSize, tileName, staticPath) {
338 /*function called if tile is not yet loaded
339  */
340     // console.timeStamp(tileName + " Start");
341     var image = document.createElement("img");
342     image.src = staticPath + tileName;
343     image.setAttribute("id", tileName);
344     image.style.opacity = 0;
345     image.style.zIndex = -2;
346     //start threshold
347     //needed to place in onload to avoid threshold images before they
             are loaded
348     image.onload = function () {
349         var i;
350         var imgstr;
351         var v = 0;
352         var imgid = "";
353         // try {
354         //   TendL = performance.now();
355         // } catch (err) {
356         //   TendL = new Date().getTime();
357         // }
358         // var time = TendL - Tstart;
359         // document.getElementById('debug2').innerHTML = 'LT: ' +
```

```
            time;
360         var c = document.createElement("canvas");
361         var thresLowerVal = parseInt(255 * (thresLower - densmin) / (
               densmax - densmin), 10);
362         var thresUpperVal = parseInt(255 * (thresUpper - densmin) / (
               densmax - densmin), 10);
363         c.width = image.width;
364         c.height = image.height;
365         try{
366             var ctx = c.getContext('2d');
367             ctx.drawImage(image, 0, 0);
368             var idata = ctx.getImageData(0, 0, c.width, c.height);
369             var d = idata.data;
370             for (i = 0; i < d.length; i += 4) {
371                 var r = d[i];
372                 v = parseInt(255 * (r - thresLowerVal) / (
                       thresUpperVal - thresLowerVal), 10);
373                 v = (v >= 255) ? 255 : v;
374                 v = (v <= 0) ? 0 : v;
375                 d[i] = d[i + 1] = d[i + 2] = v;
376                 d[i + 3] = 255;
377             }
378             ctx.putImageData(idata, 0, 0);
379             imgstr = c.toDataURL("image/png");
380             removeError("Failed to apply threshold");
381         }catch(err){
382             logError("Failed to apply threshold: "+err);
383             imgstr = image.src;
384         }
385         //reset the onload function, to avoid recursive threshold
386         image.onload = function () {
387             image.style.zIndex = 0;
388             image.style.height = (tileSize * xtrazoom) + "px";
389             image.style.display = "block";
390             image.style.width = "auto";
391             imgid = tileName.replace(".", "\\.");
392             $("#" + imgid + "").finish();
393             $("#" + imgid + "").animate({
394                 opacity : 1
395             }, 600);
396             // console.timeStamp(tileName + " End");
397             // try {
398             //   Tend = performance.now();
399             // }  catch (err) {
400             //   Tend = new Date().getTime();
401             // }
402             // var time = Tend - Tstart;
403             // document.getElementById('debug').innerHTML = 'ET: ' +
                   time;
404         };
405         image.src = imgstr;
406     };
407     //end threshold
408     var brighness = 50;
409     image.style.position = "absolute";
410     image.style.left = (pCol * tileSize * xtrazoom) + "px";
411     image.style.top = (pRow * tileSize * xtrazoom) + "px";
412     var imageTiles = document.getElementById("imageTiles");
413     imageTiles.appendChild(image);
414
```

```
415 }
416
417 function displayTile(pCol, pRow, tileName) {
418 /*called if tile is loaded already but needs to be updated
419  */
420     var image = document.getElementById(tileName);
421     var imgid = tileName.replace(".", "\\.");
422     $("#" + imgid + "").finish();
423     $("#" + imgid + "").animate({
424         opacity : 0
425     }, 2);
426     image.style.zIndex = -2;
427     image.style.height = "";
428     image.src = imgpath + tileName;
429     //start threshold
430     image.onload = function () {
431         var i;
432         var r, v;
433         var imgstr;
434         // try {
435         //   TendL = performance.now();
436         // } catch (err) {
437         //  TendL = new Date().getTime();
438         // }
439         // var time = TendL - Tstart;
440         // document.getElementById('debug2').innerHTML = 'LT: ' +
                time;
441         var c = document.createElement("canvas");
442         c.width = image.width;
443         c.height = image.height;
444         var thresLowerVal = parseInt(255 * (thresLower - densmin) / (
                densmax - densmin), 10);
445         var thresUpperVal = parseInt(255 * (thresUpper - densmin) / (
                densmax - densmin), 10);
446         try{
447             var ctx = c.getContext('2d');
448             ctx.drawImage(image, 0, 0);
449             var idata = ctx.getImageData(0, 0, c.width, c.height);
450             var d = idata.data;
451             for (i = 0; i < d.length; i += 4) {
452                 r = d[i];
453                 v = parseInt(255 * (r - thresLowerVal) / (
                        thresUpperVal - thresLowerVal), 10);
454                 v = (v >= 255) ? 255 : v;
455                 v = (v <= 0) ? 0 : v;
456                 d[i] = d[i + 1] = d[i + 2] = v;
457                 d[i + 3] = 255;
458             }
459             ctx.putImageData(idata, 0, 0);
460             imgstr = c.toDataURL("image/png");
461             removeError("Failed to apply threshold");
462         }catch(err){
463             logError("Failed to apply threshold: "+err);
464             imgstr = image.src;
465         }
466         //reset the onload function, to avoid recursive threshold
467         image.onload = function () {
468             image.style.height = (tileSize * xtrazoom) + "px";
469             image.style.display = "block";
470             image.style.width = "auto";
```

```
471             image.style.zIndex = 0;
472             imgid = tileName.replace(".", "\\.");
473             $("#" + imgid + "").stop();
474             $("#" + imgid + "").animate({
475                 opacity : 1
476             }, 600);
477             // try {
478             //   Tend = performance.now();
479             // } catch (err) {
480             //   Tend = new Date().getTime();
481             // }
482             // var time = Tend - Tstart;
483             // document.getElementById('debug').innerHTML = 'ET: ' +
                    time;
484         };
485         image.src = imgstr;
486
487     };
488     //end threshold
489 }
490
491 function checkTiles(isForced) {
492 /*goes through the tiles and updates them as necessary.
493  *Updates every tile, if isForced
494  */
495     // console.timeStamp("checkTiles Start");
496     // document.getElementById('debug').innerHTML = "";
497     //static path stuff didn't work - but shouldn't be necessary any
            longer
498     var staticPath = imgpath.substring(0);
499     innerDiv = document.getElementById("innerDiv");
500     var imageTiles = document.getElementById("imageTiles");
501     var visibleTiles = getVisibleTiles();
502     var tileArray = visibleTiles[0];
503     var visibleTilesMap = {};
504     var gTileCountWidth = new Array();
505     var gTileCountHeight = new Array();
506     var tempWidth = gImageWidth;
507     var tempHeight = gImageHeight;
508
509     var divider = Math.pow(2, (gTierCount - zoom - 1)) / xtrazoom;
510     var j;
511     var i = 0;
512     while (i < visibleTiles.length) {
513         tileArray = visibleTiles[i];
514         gTileCountWidth = new Array();
515         gTileCountHeight = new Array();
516         tempWidth = gImageWidth;
517         tempHeight = gImageHeight;
518         divider = 2;
519         //do I need to compute this for all zoom levels here?
520         for (j = gTierCount - 1; j >= 0; j--) {
521             gTileCountWidth[j] = Math.floor(tempWidth / (tileSize));
522             if (tempWidth % (tileSize)) {
523                 gTileCountWidth[j]++;
524             }
525             gTileCountHeight[j] = Math.floor(tempHeight / (tileSize))
                    ;
526             if (tempHeight % (tileSize)) {
527                 gTileCountHeight[j]++;
```

```
528             }
529             tempWidth = Math.floor(gImageWidth / divider);
530             tempHeight = Math.floor(gImageHeight / divider);
531             divider *= 2;
532             if (tempWidth % 2){tempWidth++;}
533             if (tempHeight % 2){tempHeight++;}
534         }
535 
536         moveThumb2();
537         var pCol = tileArray[0];
538         var pRow = tileArray[1];
539         var tier = zoom;
540 
541         //why do I check this here? else -> repeat image which is
                already loaded?
542         if (pCol < gTileCountWidth[zoom] && pRow < gTileCountHeight[
                zoom]) {
543             var tileName = zoom + "-" + pCol + "-" + pRow + imgtype;
544 
545             visibleTilesMap[tileName] = true;
546             var img = document.getElementById(tileName);
547             if (!img) {
548                 //try and catch should no longer be necessary
549                 try {
550                     createTile(pCol, pRow, tileSize, tileName,
                            staticPath);
551                     removeError("<b>Failed to create tile " +
                            tileName + ":</b>");
552                 } catch (err) {
553                     var image = document.createElement("img");
554                     image.src = staticPath + tileName;
555                     image.setAttribute("id", tileName);
556                     image.style.position = "absolute";
557                     image.style.left = (pCol * tileSize * xtrazoom) +
                             "px";
558                     image.style.top = (pRow * tileSize * xtrazoom) +
                            "px";
559                     image.style.zIndex = 0;
560                     imageTiles.appendChild(image);
561                     logError("<b>Failed to create tile " + tileName +
                             ":</b> " + err);
562                 }
563             } else if (isForced) {
564                 displayTile(pCol, pRow, tileName);
565             }
566         }
567         i++;
568 
569     }
570 
571     var imgs = imageTiles.getElementsByTagName("img");
572     for (i = 0; i < imgs.length; i++) {
573         var id = imgs[i].getAttribute("id");
574         if (!visibleTilesMap[id] && id !== "mainTile") {
575             imageTiles.removeChild(imgs[i]);
576             i--;
577         }
578     }
579 
580     // try {
```

```
581     //  TendR = performance.now();
582     // } catch (err) {
583     //  TendR = new Date().getTime();
584     // }
585     // var time = TendR - Tstart;
586     // document.getElementById('debug3').innerHTML = 'TTR: ' + time;
587 }
588 
589 function refreshTiles() {
590 /*remove all tiles so that a reload is forced.
591  */
592     //refresh main tile
593     var mainTile = document.getElementById("mainTile");
594     var divider = Math.pow(2, (gTierCount - zoom - 1)) / xtrazoom;
595     var MTwidth = (gImageWidth / divider) + "px";
596     $("#mainTile").finish();
597     $("#mainTile").animate({
598         width : MTwidth
599     }, 300, "swing");
600     //remove other tiles
601     var imageTiles = document.getElementById("imageTiles");
602     var imgs = imageTiles.getElementsByTagName("img");
603     while (imgs.length > 1) {
604         if (imgs[0].id != "mainTile") {
605             imageTiles.removeChild(imgs[0]);
606         } else {
607             imageTiles.removeChild(imgs[1]);
608         }
609     }
610 }
611 
612 function refreshUnderlay() {
613 /*update the underlying (bad-resolution) image
614  */
615     var temp = document.createElement("img");
616     var mainTile = document.getElementById("mainTile");
617     var timg = document.getElementById('timg');
618     //start threshold
619     temp.onload = function () {
620         var i;
621         var r, v;
622         try{
623             var c = document.createElement("canvas");
624             c.width = timg.width;
625             c.height = timg.height;
626             var thresLowerVal = parseInt(255 * (thresLower - densmin)
                    / (densmax - densmin), 10);
627             var thresUpperVal = parseInt(255 * (thresUpper - densmin)
                    / (densmax - densmin), 10);
628             var ctx = c.getContext('2d');
629             ctx.drawImage(timg, 0, 0);
630             var idata = ctx.getImageData(0, 0, c.width, c.height);
631             var d = idata.data;
632             for (i = 0; i < d.length; i += 4) {
633                 r = d[i];
634                 v = parseInt(255 * (r - thresLowerVal) / (
                        thresUpperVal - thresLowerVal), 10);
635                 v = (v >= 255) ? 255 : v;
636                 v = (v <= 0) ? 0 : v;
637                 d[i] = d[i + 1] = d[i + 2] = v;
```

```
638                 d[i + 3] = 255;
639             }
640             ctx.putImageData(idata, 0, 0);
641             var imgstr = c.toDataURL("image/png");
642             mainTile.onload = function () {
643                 var divider = Math.pow(2, (gTierCount - zoom - 1)) /
                        xtrazoom;
644                 mainTile.style.width = gImageWidth / divider + "px";
645             };
646             mainTile.src = imgstr;
647             removeError("Failed␣to␣apply␣threshold:");
648         }catch(err){
649             //failed to apply filter to image
650             logError("Failed␣to␣apply␣threshold:␣"+err);
651             mainTile.src = timg.src;
652         }
653     };
654     //end threshold
655     temp.src = imgpath + '0-0-0' + imgtype;
656 }
657 /*** END TILE HANDLER ***/
658
659
660 /*** ZOOM ***
661  * zoomIn()
662  * zoomOut()
663  * updateZoom()
664  * clickZoom(event)
665  */
666 function zoomIn() {
667 /*zooms in 1 level
668  */
669     var IDtop, IDleft;
670     if (zoom !== gTierCount - 1) {
671         //normal zoom through tiles
672         $("#innerDiv").finish();
673         innerDiv = document.getElementById("innerDiv");
674         mTop = stripPx(innerDiv.style.top);
675         mLeft = stripPx(innerDiv.style.left);
676         IDtop = 2 * mTop - viewportHeight / 2 + 'px';
677         IDleft = 2 * mLeft - viewportWidth / 2 + 'px';
678         zoom = zoom + 1;
679         $("#innerDiv").animate({
680             top : IDtop,
681             left : IDleft
682         }, {
683             duration : 300,
684             complete : function () {
685                 checkTiles(1);
686             }
687         });
688         refreshTiles();
689         var imageLabels = document.getElementById("imageLabels");
690         var divs = imageLabels.getElementsByTagName("div");
691         for (var $i = 0; $i < divs.length; $i++) {
692             var Ltemp = "L" + $i;
693             $("#" + Ltemp + "").finish();
694             IDtop = 2 * stripPx(document.getElementById(Ltemp).style.
                    top) + 'px';
695             IDleft = 2 * stripPx(document.getElementById(Ltemp).style
```

```
                .left) + 'px';
696             $("#" + Ltemp + "").animate({
697                 top : IDtop,
698                 left : IDleft
699             }, 300, "swing");
700         }
701         clickMode1()
702     } else if (xtrazoom < xtrazoomMax) {
703         //extra zoom beyond image resolution
704         zoomdif = xtrazoom;
705         xtrazoom = xtrazoom + 1;
706         zoomdif = xtrazoom / zoomdif;
707         $("#innerDiv").finish();
708         innerDiv = document.getElementById("innerDiv");
709         mTop = stripPx(innerDiv.style.top);
710         mLeft = stripPx(innerDiv.style.left);
711         IDtop = 0.5 * (1 - zoomdif) * viewportHeight + zoomdif * mTop
                + 'px';
712         IDleft = 0.5 * (1 - zoomdif) * viewportWidth + zoomdif *
               mLeft + 'px';
713         $("#innerDiv").animate({
714             top : IDtop,
715             left : IDleft
716         }, {
717             duration : 300,
718             complete : function () {
719                 checkTiles(0);
720             }
721         });
722         var imageLabels = document.getElementById("imageLabels");
723         var divs = imageLabels.getElementsByTagName("div");
724         for (var $i = 0; $i < divs.length; $i++) {
725             var Ltemp = "L" + $i;
726             $("#" + Ltemp + "").finish();
727             IDtop = zoomdif * stripPx(document.getElementById(Ltemp).
                   style.top) + 'px';
728             IDleft = zoomdif * stripPx(document.getElementById(Ltemp)
                   .style.left) + 'px';
729             $("#" + Ltemp + "").animate({
730                 top : IDtop,
731                 left : IDleft
732             }, 300, "swing");
733         }
734         clickMode1()
735         refreshTiles();
736         //checkTiles(0);
737     }
738     updateZoom();
739 }
740
741 function zoomOut() {
742 /*zooms out 1 level
743  */
744     if (xtrazoom > 1) {
745         zoomdif = xtrazoom;
746         xtrazoom = xtrazoom - 1;
747         zoomdif = xtrazoom / zoomdif;
748         $("#innerDiv").finish();
749         innerDiv = document.getElementById("innerDiv");
750         mTop = stripPx(innerDiv.style.top);
```

```
751         mLeft = stripPx(innerDiv.style.left);
752         var IDtop = 0.5 * (1 - zoomdif) * viewportHeight + zoomdif *
                mTop + 'px';
753         var IDleft = 0.5 * (1 - zoomdif) * viewportWidth + zoomdif *
                mLeft + 'px';
754         $("#innerDiv").animate({
755             top : IDtop,
756             left : IDleft
757         }, {
758             duration : 300,
759             complete : function () {
760                 checkTiles(0);
761             }
762         });
763         var imageLabels = document.getElementById("imageLabels");
764         var divs = imageLabels.getElementsByTagName("div");
765         for (var $i = 0; $i < divs.length; $i++) {
766             var Ltemp = "L" + $i;
767             $("#" + Ltemp + "").finish();
768             IDtop = zoomdif * stripPx(document.getElementById(Ltemp).
                    style.top) + 'px';
769             IDleft = zoomdif * stripPx(document.getElementById(Ltemp)
                    .style.left) + 'px';
770             $("#" + Ltemp + "").animate({
771                 top : IDtop,
772                 left : IDleft
773             }, 300, "swing");
774         }
775         clickMode1()
776         refreshTiles();
777         //checkTiles(0);
778     } else if (zoom != 0) {
779         $("#innerDiv").finish();
780         var innerDiv = document.getElementById("innerDiv");
781         mTop = stripPx(innerDiv.style.top);
782         mLeft = stripPx(innerDiv.style.left);
783         var IDtop = mTop / 2 + viewportHeight / 4 + 'px';
784         var IDleft = mLeft / 2 + viewportWidth / 4 + 'px';
785         $("#innerDiv").animate({
786             top : IDtop,
787             left : IDleft
788         }, {
789             duration : 300,
790             complete : function () {
791                 checkTiles(1);
792             }
793         });
794         zoom = zoom - 1;
795         refreshTiles();
796         var imageLabels = document.getElementById('imageLabels');
797         var divs = imageLabels.getElementsByTagName("div");
798         for (var $i = 0; $i < divs.length; $i++) {
799             var Ltemp = "L" + $i;
800             $("#" + Ltemp + "").finish();
801             IDtop = .5 * stripPx(document.getElementById(Ltemp).style
                    .top) + 'px';
802             IDleft = .5 * stripPx(document.getElementById(Ltemp).
                    style.left) + 'px';
803             $("#" + Ltemp + "").animate({
804                 top : IDtop,
```

```
805                 left : IDleft
806             }, 300, "swing");
807         }
808         clickMode1()
809         //checkTiles(1);
810     }
811     updateZoom();
812 
813 }
814 
815 function updateZoom() {
816 /*updates the zoom slider
817  */
818     var zoomsliderdiv;
819     var zwidth = 150;
820     var zheight = 10;
821     var zpaddingY = 5;
822     var zpaddingX = 20;
823     var activewidth = Math.max(Math.round((zwidth * (zoom) / (
        gTierCount + xtrazoomMax - 2)) - 5), 0);
824     var active2width = Math.max(Math.round((zwidth * (xtrazoom - 1) /
         (gTierCount + xtrazoomMax - 2))), 0);
825     var inactivewidth = Math.max(Math.round((zwidth - zwidth * (zoom
        + xtrazoom - 1) / (gTierCount + xtrazoomMax - 2)) - 5), 0);
826 
827     //Create table for zoom slider
828     var zoomstr = '<table␣id="zoomslidertable">';
829     zoomstr += '<tr␣height="' + zheight + 'px">';
830     zoomstr += '<td␣width="' + zpaddingX + 'px"␣height="' + zheight +
         'px"></td>';
831     if (activewidth) {
832         zoomstr += '<td␣class="slider␣active"␣width="' + activewidth
            + 'px"␣height="' + zheight + 'px">';
833         zoomstr += '</td>';
834     } else {
835         inactivewidth -= 5;
836     }
837     if (active2width) {
838         zoomstr += '<td␣class="slider␣active2"␣width="' +
            active2width + 'px"␣height="' + zheight + 'px">';
839         zoomstr += '</td>';
840     }
841     zoomstr += '<td␣class="slider␣knob␣tooltip"␣width="' + 10 + 'px"␣
        height="' + zheight + 'px"␣title="' + (zoom + xtrazoom - 1) +
         "/" + (gTierCount + xtrazoomMax - 2) + '"></td>';
842     if (inactivewidth) {
843         zoomstr += '<td␣class="slider␣inactive"␣width="' +
            inactivewidth + 'px"␣height="' + zheight + 'px">';
844         zoomstr += '</td>';
845     }
846     zoomstr += '<td␣width="' + zpaddingX + 'px"␣height="' + zheight +
         'px"></td></tr></table>';
847 
848     zoomsliderdiv = document.getElementById('zoomslider');
849     zoomsliderdiv.innerHTML = zoomstr
850     zoomsliderdiv.onmouseup = clickZoom;
851 
852     //update the small ruler shown at the bottom left
853     document.getElementById('theScale').innerHTML = engUnit((Math.pow
        (2, gTierCount - zoom - 1) / (xtrazoom)) * res * 50,resunits)
```

```
        ;
}

function clickZoom(event) {
/*move to position clicked on zoom slider
 */
    if (event) {
        xThumb = event.clientX;
        yThumb = event.clientY;
        var sliderLeft = document.getElementById("zoomslider").
           getBoundingClientRect().left;
        var sliderWidth = document.getElementById("zoomslider").
           getBoundingClientRect().right - sliderLeft;
        var perc = Math.min(Math.abs(xThumb - sliderLeft - spaddingX)
            / (sliderWidth - 2 * spaddingX), 1);
        totzoom = Math.round(0 + (gTierCount + xtrazoomMax - 2) *
           perc);
        newzoom = Math.max(Math.min(gTierCount - 1, totzoom), 0);
        newxtrazoom = Math.min(Math.max(1, totzoom + 1 - newzoom),
           xtrazoomMax);
        totzoom = newzoom + newxtrazoom - 1;
        oldzoom = zoom + xtrazoom - 1;
        if (oldzoom < totzoom) {
            for (j = oldzoom; j < totzoom; j++) {
                zoomIn();
            }
        } else if (oldzoom > totzoom) {
            for (j = oldzoom; j > totzoom; j--) {
                zoomOut();
            }
        }
    }
}
/*** END ZOOM ***/


/*** DENSITY RANGE (THRESHOLD) ***
 * updateThreshold()
 * clickThreshold(event)
 */
function updateThreshold() {
/*refreshes the Threshold slider
 */
    var thressliderdiv;
    var totmin = Math.round(densmin);
    var totmax = Math.round(densmax);
    var swidth = 145;

    var inactive1width = Math.max(Math.round((swidth * (thresLower -
       totmin) / (totmax - totmin)) - 5), 0);
    var inactive2width = Math.max(Math.round((swidth * (totmax -
       thresUpper) / (totmax - totmin)) - 5), 0);
    var activewidth = Math.max(Math.round((swidth - swidth * (
       thresLower - totmin + totmax - thresUpper) / (totmax - totmin
       )) - 10), 0);

    //Creating table for threshold double slider
    var thresstr = '<table␣id="thresslidertable"><tr><td␣width="' +
       spaddingX + 'px"></td>';
    if (inactive1width) {
```

```
903         thresstr += '<td␣class="slider␣inactive"␣width="' +
                inactive1width + 'px"␣height="1px">';
904         thresstr += '</td>';
905     }
906     thresstr += '<td␣class="slider␣knob␣tooltip"␣width="10px"␣␣height
            ="10px"␣title="' + thresLower + '"></td>';
907     if (activewidth) {
908         thresstr += '<td␣class="slider␣active"␣␣title="[' +
                thresLower + "," + thresUpper + ']"␣width="' +
                activewidth + 'px"␣height="1px">';
909         thresstr += '</td>';
910     }
911     thresstr += '<td␣class="slider␣knob␣tooltip"␣width="10px"␣␣height
            ="10px"␣title="' + thresUpper + '"></td>';
912     if (inactive2width) {
913         thresstr += '<td␣class="slider␣inactive"␣width="' +
                inactive2width + 'px"␣height="1px">';
914         thresstr += '</td>';
915     }
916     thresstr += '<td␣width="' + spaddingX + 'px"></td>';
917     thresstr += '</tr></table>';
918
919     thressliderdiv = document.getElementById('thresslider');
920     thressliderdiv.innerHTML = thresstr;
921     thressliderdiv.onmouseup = clickThreshold;
922 }
923
924 function clickThreshold(event) {
925 /*apply clicked position on threshold slider
926  */
927     if (event) {
928         xThumb = event.clientX;
929         yThumb = event.clientY;
930         var sliderLeft = document.getElementById("thresslider").
                getBoundingClientRect().left;
931         var sliderWidth = document.getElementById("thresslider").
                getBoundingClientRect().right - sliderLeft;
932         var totmin = Math.round(densmin);
933         var totmax = Math.round(densmax);
934         var perc = Math.min(Math.abs(xThumb - sliderLeft - spaddingX)
                / (sliderWidth - 2 * spaddingX), 1);
935         perc = Math.max(perc, 0);
936         var newthres = Math.round(totmin + (totmax - totmin) * perc);
937         deltamin = Math.abs(newthres - thresLower);
938         deltamax = Math.abs(newthres - thresUpper);
939         if (deltamin < deltamax) {
940             thresLower = newthres
941         } else {
942             thresUpper = newthres
943         }
944         refreshUnderlay()
945         updateThreshold()
946         checkTiles(1)
947     }
948 }
949 /*** END DENSITY RANGE (THRESHOLD) ***/
950
951 /*** SLICES ***
952  * updateSlice()
953  * clickSlice(event)
```

```
954   * sliceNext(delta,scaled)
955   * slicePrev(delta,scaled)
956   * sliceNextDef()
957   * slicePrevDef()
958   */
959  function updateSlice() {
960  /*update Slice slider
961   */
962      var slicesliderdiv;
963      var swidth = 145;
964      var activewidth = Math.max(Math.round((swidth * slidePointer /
             JSONnum) - 5), 0);
965      var inactivewidth = Math.max(Math.round((swidth - swidth *
             slidePointer / JSONnum) - 5), 0);
966      var sliderstr = '<table id="sliceslidertable"><tr>';
967      sliderstr += '<td width="' + spaddingX + 'px"></td>';
968      if (activewidth) {
969          sliderstr += '<td class="slider active" title="' + (
                 slidePointer) + "/" + (JSONnum) + '" width="' +
                 activewidth + 'px" height="10px">';
970          sliderstr += '</td>';
971      } else {
972          inactivewidth -= 5;
973      }
974      sliderstr += '<td class="slider knob tooltip" width="' + 10 + 'px
             " title="' + (slidePointer) + "/" + (JSONnum) + '"></td>';
975      if (inactivewidth) {
976          sliderstr += '<td class="slider inactive" title="' + (
                 slidePointer) + "/" + (JSONnum) + '" width="' +
                 inactivewidth + 'px" height="10px">';
977          sliderstr += '</td>';
978      }
979      sliderstr += '<td width="' + spaddingX + 'px"></td>';
980      sliderstr += '</tr></table>';
981
982      slicesliderdiv = document.getElementById('sliceslider');
983      slicesliderdiv.innerHTML = sliderstr;
984      slicesliderdiv.onmouseup = clickSlice;
985      slicesliderdiv.ondragstart = function () {
986          return false;
987      }
988
989  }
990
991  function clickSlice(event) {
992  /*move to position clicked on slice slider
993   */
994      if (event) {
995          xThumb = event.clientX;
996          yThumb = event.clientY;
997          var sliderLeft = document.getElementById("sliceslider").
                 getBoundingClientRect().left;
998          var sliderWidth = document.getElementById("sliceslider").
                 getBoundingClientRect().right - sliderLeft;
999          var perc = Math.min(Math.abs(xThumb - sliderLeft - spaddingX)
                  / (sliderWidth - 2 * spaddingX), 1);
1000         perc = Math.max(perc, 0);
1001         newslice = Math.round(JSONnum * perc);
1002         delta = newslice - slidePointer;
1003         if (delta > 0) {
```

```
            sliceNext(delta, false)
        } else if (delta < 0) {
            slicePrev(delta, false)
        }
    }
}

function sliceNext(delta, scaled) {
/*display next image in stack
 */
    // try {
    //  Tstart = performance.now();
    // } catch (err) {
    //  Tstart = new Date().getTime();
    // }

    try {
        clearTimeout(ActiveTask);
    } catch (err) {}
    if (scaled) {
        slidePointer += Math.floor(gTierCount * Math.abs(delta) / (
            zoom + xtrazoom)) + 1;
    } else {
        slidePointer += Math.floor(Math.abs(delta));
    }
    while (slidePointer >= JSONnum) {
        slidePointer -= JSONnum;
    }
    path = JSONout.slides[slidePointer].path;
    width = JSONout.slides[slidePointer].width;
    height = JSONout.slides[slidePointer].height;
    if (JSONout.slides[slidePointer].labelspath != undefined) {
        labelspath = JSONout.slides[slidePointer].labelspath;
        loadLabels();
    } else {
        labelspath = "";
    }
    updatePosition();
    init();
}

function slicePrev(delta, scaled) {
/*display previous image in stack
 */
    // try {
    //  Tstart = performance.now();
    // } catch (err) {
    //  Tstart = new Date().getTime();
    // }

    try {
        clearTimeout(ActiveTask);
    } catch (err) {}
    if (scaled) {
        slidePointer += -1 * Math.floor(gTierCount * Math.abs(delta)
            / (zoom + xtrazoom)) - 1;
    } else {
        slidePointer += -1 * Math.floor(Math.abs(delta));
    }
    while (slidePointer < 0) {
```

```
1062            slidePointer += JSONnum;
1063        }
1064        path = JSONout.slides[slidePointer].path;
1065        width = JSONout.slides[slidePointer].width;
1066        height = JSONout.slides[slidePointer].height;
1067        if (JSONout.slides[slidePointer].labelspath != undefined) {
1068            labelspath = JSONout.slides[slidePointer].labelspath;
1069            loadLabels();
1070        } else {
1071            labelspath = "";
1072        }
1073        updatePosition();
1074        init();
1075    }
1076
1077    function sliceNextDef(){
1078    /*default function for moving to next slice
1079     *used for click events as no argument required
1080     */
1081        sliceNext(1,true);
1082    }
1083
1084    function slicePrevDef(){
1085    /*default function for moving to previous slice
1086     * used for click events as no argument required
1087     */
1088        slicePrev(1,true);
1089    }
1090    /*** END SLICES ***/
1091
1092
1093    /*** OTHER UPDATES ***/
1094    function updatePosition() {
1095    /*updates position displayed
1096     */
1097        var innerDiv = document.getElementById("innerDiv");
1098        var clientX, clientY;
1099        var event = window.event;
1100        if (!event) {
1101            clientX = lasteventX;
1102            clientY = lasteventY;
1103        }else{
1104            clientX = event.clientX;
1105            clientY = event.clientY;
1106        }
1107        if (coords) {
1108            var errstr = "<b>Unable␣to␣resolve␣position.</b>"
1109                try {
1110                    document.getElementById('coords').innerHTML = "<b>&
                           nbsp;Position</b>␣(in␣" + resunits + ")<b>:</b><
                           br/>  " + Math.round((res * (-stripPx(
                           innerDiv.style.left) + clientX - 0) / (1 / (Math.
                           pow(2, gTierCount - 1 - zoom) / xtrazoom)) * 10))
                            / 10 + ",␣" + Math.round((res * (-stripPx(
                           innerDiv.style.top) + clientY - 16) / (1 / (Math.
                           pow(2, gTierCount - 1 - zoom) / xtrazoom)) * 10))
                            / 10 + ",␣" + Math.round(zres * slidePointer *
                           10) / 10;
1111                    removeError(errstr)
1112                } catch (err) {
```

```
1113                document.getElementById('coords').innerHTML = "<b>&
                       nbsp;Position</b> (in " + resunits + ")<b>:</b><
                       br/>  unknown";
1114                logError(errstr+" "+err)
1115            }
1116    }
1117 }
1118
1119 function updateInfo() {
1120 /*updates all sliders
1121  */
1122     updateZoom();
1123     updateThreshold();
1124     if (JSONnum) {
1125         updateSlice();
1126     }
1127     showThumb(); //should be in if statement
1128     updatePosition();
1129 }
1130
1131 function centreView() {
1132 /*center image with respect to the display
1133  */
1134     var innerDiv = document.getElementById("innerDiv");
1135     if ((getVar('vX').length) > 0) {
1136         innerDiv.style.left = -getVar('vX') * width / (Math.pow(2,
                gTierCount - 1 - zoom) / xtrazoom) + viewportWidth / 2 +
                "px";
1137     } else {
1138         innerDiv.style.left =  - (width / (Math.pow(2, gTierCount - 1
                - zoom) / xtrazoom)) / 2 + viewportWidth / 2 + 'px';
1139     }
1140     if ((getVar('vY').length) > 0) {
1141         innerDiv.style.top = -getVar('vY') * height / (Math.pow(2,
                gTierCount - 1 - zoom) / xtrazoom) + viewportHeight / 2 +
                 "px";
1142     } else {
1143         innerDiv.style.top =  - (height / (Math.pow(2, gTierCount - 1
                - zoom) / xtrazoom)) / 2 + viewportHeight / 2 + 'px';
1144     }
1145 }
1146
1147
1148 /*** THUMBNAIL ***
1149  * showThumb()
1150  * hideThumb()
1151  * moveThumb2()
1152  * clickThumb(event)
1153  * clickThumbC(event)
1154  ************************************************************
1155  * Note:
1156  * there are two thumbnails:
1157  *   normal one and
1158  *   crossectional one (denoted with 'C')
1159  * each thumbnail is a div (denoted by '0') consisting of
1160  *   an "image+label"-div (denoted by '') and
1161  *   a "position rectangle"-div (denoted by '2')
1162  ************************************************************/
1163 function showThumb() {
1164 /*display thumbnail
```

```
1165  */
1166
1167     //thumbnail of planar view
1168     var Thumb = document.getElementById('Thumb');
1169     var timg = document.getElementById('timg');
1170     timg.src = imgpath + '0-0-0' + imgtype;
1171     refreshUnderlay()
1172
1173     var Thumb0 = document.getElementById('Thumb0');
1174     Thumb0.style.height = gImageHeight / (Math.pow(2, gTierCount - 1)
            ) + 'px';
1175     Thumb0.style.width = gImageWidth / (Math.pow(2, gTierCount - 1))
            + 'px';
1176     Thumb0.style.display = "block";
1177     Thumb.style.display = "block";
1178
1179     var Thumb2 = document.getElementById('Thumb2');
1180     Thumb2.style.display = "block";
1181     Thumb2.onmouseup = clickThumb;
1182     Thumb.onmouseup = clickThumb;
1183     Thumb.ondragstart = function () {
1184         return false;
1185     }
1186
1187     //thumbnail of crossectional view
1188     var ThumbC = document.getElementById('ThumbC');
1189     ThumbC.innerHTML = '<div␣style="position:absolute;width:100%;text
            -align:center;"> cross-sectional␣view</div>';
1190     ThumbC.innerHTML += '<div><img␣src="' + rootpath + '/tc' +
            imgtype + '"></div>';
1191
1192     var Thumb0C = document.getElementById('Thumb0C');
1193     Thumb0C.style.height = JSONnum / (Math.pow(2, gTierCount - 1)) +
            1 + 'px';
1194     Thumb0C.style.width = gImageWidth / (Math.pow(2, gTierCount - 1))
             + 'px';
1195     Thumb0C.style.display = "block";
1196     ThumbC.style.display = "block";
1197
1198     var Thumb2C = document.getElementById('Thumb2C');
1199     Thumb2C.style.display = "block";
1200     Thumb2C.onmouseup = clickThumbC;
1201     ThumbC.onmouseup = clickThumbC;
1202     ThumbC.ondragstart = function () {
1203         return false;
1204     }
1205
1206 }
1207
1208 function hideThumb() {
1209 /*hide thumbnail
1210  */
1211     document.getElementById('Thumb0').style.display = "none";
1212     document.getElementById('Thumb0C').style.display = "none";
1213 }
1214
1215 function moveThumb2() {
1216 /*adjust thumbnail position indicator
1217  */
1218     //display rectangle of current view
```

```
1219      var innerDiv = document.getElementById("innerDiv");
1220
1221      //rectangle for cross-sectional view
1222      var Thumb2C = document.getElementById("Thumb2C");
1223      topT = stripPx(innerDiv.style.top);
1224      leftT = stripPx(innerDiv.style.left);
1225      Thumb2C.style.width = viewportWidth / (Math.pow(2, zoom) *
             xtrazoom) + 'px';
1226      Thumb2C.style.height = '0px';
1227      Thumb2C.style.left = -leftT / (Math.pow(2, zoom) * xtrazoom) + '
             px';
1228      Thumb2C.style.top = slidePointer / (Math.pow(2, gTierCount - 1))
             + 'px';
1229
1230      //rectangle for planar view
1231      var Thumb2 = document.getElementById("Thumb2");
1232      topT = stripPx(innerDiv.style.top);
1233      leftT = stripPx(innerDiv.style.left);
1234      Thumb2.style.width = viewportWidth / (Math.pow(2, zoom) *
             xtrazoom) + 'px';
1235      Thumb2.style.height = viewportHeight / (Math.pow(2, zoom) *
             xtrazoom) + 'px';
1236      Thumb2.style.left = -leftT / (Math.pow(2, zoom) * xtrazoom) + 'px
             ';
1237      Thumb2.style.top = -topT / (Math.pow(2, zoom) * xtrazoom) + 'px';
1238  }
1239
1240  function clickThumb(event) {
1241  /*move to position clicked in thumbnail
1242   */
1243      if (event) {
1244          xThumb = event.clientX; //X position of mouse on click with
                 respect to viewport
1245          yThumb = event.clientY;
1246          var innerDiv = document.getElementById("innerDiv");
1247
1248          //calculate new left position
1249          var ThumbWidth = stripPx(document.getElementById("Thumb0").
                 style.width); //width of thumbnail element
1250          var ThumbLeft = viewportWidth - stripPx(document.
                 getElementById("Thumb0").style.right) - stripPx(document.
                 getElementById("Thumb0").style.border) - ThumbWidth; //
                 left position of thumbnail element
1251          var xThumbRel = xThumb - ThumbLeft; //left position of click
                 with respect to thumbnail
1252          var ImageZoomedWidth = width / (Math.pow(2, gTierCount - zoom
                  - 1) / xtrazoom);
1253          var ImageZoomedLeft = ImageZoomedWidth * xThumbRel /
                 ThumbWidth;
1254          innerDiv.style.left = viewportWidth / 2 - ImageZoomedLeft + '
                 px';
1255
1256          //calculate new top position
1257          var ThumbHeight = stripPx(document.getElementById("Thumb0").
                 style.height); //height of thumbnail element
1258          var ThumbTop = viewportHeight - stripPx(document.
                 getElementById("Thumb0").style.bottom) - stripPx(document
                 .getElementById("Thumb0").style.border) - ThumbHeight; //
                 top position of thumbnail element
1259          var yThumbRel = yThumb - ThumbTop; //top position of click
```

```
            with respect to thumbnail
1260        var ImageZoomedHeight = height / (Math.pow(2, gTierCount -
                zoom - 1) / xtrazoom);
1261        var ImageZoomedTop = ImageZoomedHeight * yThumbRel /
                ThumbHeight;
1262        innerDiv.style.top = viewportHeight / 2 - ImageZoomedTop + '
                px';
1263
1264        //apply changes
1265        checkTiles(0);
1266        moveThumb2();
1267    }
1268 }
1269
1270 function clickThumbC(event) {
1271 /*move to position clicked in cross-section thumbnail
1272  */
1273    if (event) {
1274        xThumb = event.clientX;
1275        yThumb = event.clientY;
1276        var innerDiv = document.getElementById("innerDiv");
1277
1278        //calculate new left position
1279        var ThumbWidth = stripPx(document.getElementById("Thumb0C").
                style.width); //width of thumbnail element
1280        var ThumbLeft = viewportWidth - stripPx(document.
                getElementById("Thumb0C").style.right) - stripPx(document
                .getElementById("Thumb0C").style.border) - ThumbWidth; //
                left position of thumbnail element
1281        var xThumbRel = xThumb - ThumbLeft; //left position of click
                with respect to thumbnail
1282        var ImageZoomedWidth = width / (Math.pow(2, gTierCount - zoom
                 - 1) / xtrazoom);
1283        var ImageZoomedLeft = ImageZoomedWidth * xThumbRel /
                ThumbWidth;
1284        innerDiv.style.left = viewportWidth / 2 - ImageZoomedLeft + '
                px';
1285
1286        //calculate new slice number
1287        slidePointer = Math.round(JSONnum * (Math.abs(yThumb -
                viewportHeight + stripPx(document.getElementById("Thumb0C
                ").style.bottom) + stripPx(document.getElementById("
                Thumb0C").style.height)) + 2) / stripPx(document.
                getElementById("Thumb0C").style.height));
1288
1289        //apply changes
1290        sliceNext(1);
1291        moveThumb2();
1292    }
1293 }
1294 /*** END THUMBNAIL ***/
1295
1296 /*** CONTROLS-INFO ***
1297  * clickControls(event)
1298  * showControlsBtn()
1299  * hideControlsBtn()
1300  * showControls()
1301  * hideControls()
1302  */
1303 function clickControls(event) {
```

```
1304 /*handles event when "Controls" button is clicked
1305  */
1306     if(isControl){
1307         hideControls();
1308     }else{
1309         showControls();
1310     }
1311 }
1312
1313 function showControlsBtn() {
1314 /*displays "Controls" button
1315  */
1316     document.getElementById("cntrlButton").style.display = "block";
1317 }
1318
1319 function hideControlsBtn() {
1320 /*hides "Controls" button
1321  */
1322     document.getElementById("cntrlButton").style.display = "none";
1323     if(isControl){
1324         hideControls();
1325     }
1326 }
1327
1328 function showControls() {
1329 /*displays div with image of controls
1330  */
1331     document.getElementById("controls").style.display = "block";
1332     $("#controls").animate({
1333         opacity : 0.75
1334     }, 500);
1335     isControl = true;
1336 }
1337
1338 function hideControls() {
1339 /*hides div with image of controls
1340  */
1341     $("#controls").animate({
1342         opacity : 0
1343     }, 500);
1344     setTimeout(function () {
1345         document.getElementById("controls").style.display = "none";
1346         isControl = false;
1347     }, 500);
1348 }
1349 /*** END OF CONTROLS-INFO ***/
1350
1351
1352 /*** AJAX/FILE LOADING ***
1353  * getHTTPObject()
1354  * getJSONAttribute(attribute, JSONdic, defval)
1355  * labelsHandler()
1356  * loadLabels()
1357  * JSONread()
1358  * loadJSON()
1359  */
1360 function getHTTPObject() {
1361 /*requests an object from the server and returns it
1362  */
1363     var xhr;
```

```
1364      try {
1365          xhr = new ActiveXObject("Msxml2.XMLHTTP");
1366      } catch (err) {
1367          try {
1368              xhr = new ActiveXObject("Microsoft.XMLHTTP");
1369          } catch (Err2) {
1370              xhr = false;
1371          }
1372      }
1373      if (!xhr && typeof XMLHttpRequest !== 'undefined') {
1374          xhr = new XMLHttpRequest();
1375      }
1376      return xhr;
1377  }
1378
1379  function getJSONAttribute(attribute, JSONdic, defval) {
1380  /*attempts to get and return requested attribute from a JSON
        dictionary,
1381   *returns defval if not found
1382   */
1383      if (attribute in JSONdic) {
1384          var ans = JSONdic[attribute];
1385          return ans;
1386      }
1387      return defval;
1388  }
1389
1390  labels = getHTTPObject();
1391
1392  function labelsHandler() {
1393  /*places labels in the labels div
1394   */
1395      if (labels.readyState == 4) {
1396          var labels2 = eval('(' + labels.responseText + ')');
1397          var lab = labels2.labels.length;
1398
1399          for (var $i = 0; $i < lab; $i++) {
1400              var label = labels2.labels[$i].label;
1401              var name = label;
1402              var nX = labels2.labels[$i].x;
1403              var nY = labels2.labels[$i].y;
1404
1405              if (labels2.labels[$i].name != undefined) {
1406                  name = labels2.labels[$i].name;
1407              }
1408
1409              if (labels2.labels[$i].url != undefined) {
1410                  label = '<a href="' + labels2.labels[$i].url + '"
                        title="' + name + '" target="_blank">' + label +
                        '</a>';
1411              }
1412
1413              pinImage = document.createElement("div");
1414              pinImage.style.position = "absolute";
1415              pinImage.style.left = (nX * gImageWidth / (Math.pow(2,
                    gTierCount - 1 - zoom) / xtrazoom)) + "px";
1416              pinImage.style.top = (nY * gImageHeight / (Math.pow(2,
                    gTierCount - 1 - zoom) / xtrazoom)) + "px";
1417              pinImage.style.width = 8 * label.length + "px";
1418              pinImage.style.height = "2px";
```

```
1419             pinImage.style.zIndex = 1;
1420             pinImage.setAttribute("id", "L" + $i);
1421             pinImage.innerHTML = label;
1422             document.getElementById("imageLabels").appendChild(
                    pinImage);
1423         }
1424     }
1425 }
1426
1427 function loadLabels() {
1428 /*requests labels to be loaded
1429  */
1430     var urlLabels = labelspath;
1431     var pinImage = document.getElementById("L0");
1432     if (pinImage) {
1433         imageLabels = document.getElementById("imageLabels");
1434         var divs = imageLabels.getElementsByTagName("div");
1435         while (divs.length > 0)
1436             imageLabels.removeChild(divs[0]);
1437     } else {
1438         labels.open("GET", urlLabels, true);
1439         labels.onreadystatechange = labelsHandler;
1440         labels.send(null);
1441     }
1442 }
1443
1444 function JSONread() {
1445 /*reads in the information about the stack of images and
1446  *sets global variables according to its content
1447  */
1448     var isError = false;
1449     if (JSONrequest.readyState == 4 && JSONrequest.status != 404) {
1450         removeError("<b>JSON␣not␣ready.</b>");
1451         removeError("<b>Couldn't␣load␣JSON.</b>␣Status:");
1452         isError = false;
1453         try {
1454             JSONout = eval('(' + JSONrequest.responseText + ')');
1455             removeError("Failed␣to␣read␣JSON:␣");
1456         } catch (err) {
1457             isError = true;
1458             document.getElementById('debug').innerHTML = "Failed␣to␣
                    read␣JSON:␣" + err;
1459             logError("Failed␣to␣read␣JSON:␣" + err);
1460         }
1461         if (!isError) {
1462             JSONnum = JSONout.slides.length;
1463             /***useful DICOM tags
1464              * (0018,0050) Slice Thickness              ------- slices
                     can have overlap so don't use this!
1465              * (0018,0088) Spacing Between Slices      <------ This
                     is the z-Resolution
1466              * (0018,1050) Spatial Resolution
1467              * (0018,1164) Imager Pixel Spacing
1468              * (0018,6048) Pixel Component Range Start
1469              * (0018,604A) Pixel Component Range Stop
1470              * (0018,604C) Pixel Component Physical Units
1471              * (0018,1240) Upper/Lower Pixel Values
1472              * (0018,6024) Physical Units X Direction
1473              * (0018,6026) Physical Units Y Direction
1474              * (0018,9322) Reconstruction Pixel Spacing
```

```
1475              * (0028,0030) Pixel Spacing                <------ This
                     is x and y resolution as a string seperated by a "\"
1476              * (0028,0034) Pixel Aspect Ratio           ------- for
                     CT should always be = 1
1477              * (0028,0108) Smallest Pixel Value in Series
1478              * (0028,0109) Largest Pixel Value in Series
1479              * (0054,1001) Units                        <------ Does
                     not seem to be commonly defined, but standart is mm/
                     voxel
1480              */
1481
1482             /*get extra image info from JSON file if available*/
1483             height = getJSONAttribute("height", JSONout, height);
1484             width = getJSONAttribute("width", JSONout, width);
1485
1486             resunits = getJSONAttribute("Units", JSONout, resunits);
1487             resunits = getJSONAttribute("resunits", JSONout, resunits
                    );
1488
1489             res = getJSONAttribute("PixelSpacing", JSONout, res);
1490             res = getJSONAttribute("0028,0030", JSONout, res); //need
                     to check format
1491             res = getJSONAttribute("res", JSONout, res);
1492
1493             zres = getJSONAttribute("SpacingBetweenSlices", JSONout,
                    zres);
1494             zres = getJSONAttribute("0018,0088", JSONout, zres); //
                    need to check format
1495             zres = getJSONAttribute("zres", JSONout, zres);
1496
1497             /*try{ densM = JSONout.RescaleSlope; } catch(err) {} //
                    need to convert
1498             try{ densM = JSONout["0028,1052"]; } catch(err) {} //need
                     to convert
1499             try{ densB = JSONout.RescaleIntercept; } catch(err) {} //
                    need to convert
1500             try{ densB = JSONout["0028,1053"]; } catch(err) {} //need
                     to convert
1501              */
1502             densmin = getJSONAttribute("densmin", JSONout, densmin);
1503             densmax = getJSONAttribute("densmax", JSONout, densmax);
1504             thresLower = Math.max(densmin, thresLower);
1505             thresUpper = Math.min(densmax, thresUpper);
1506
1507             imgtype = getJSONAttribute("filetype", JSONout, ".jpg");
1508
1509             start = getVar('start');
1510             if (start.length > 0) {
1511                 slidePointer = parseInt(start, 10);
1512             } else {
1513                 slidePointer = parseInt(JSONnum / 3, 10);
1514             }
1515
1516             path = JSONout.slides[slidePointer].path; //path to
                    specific slice images
1517             height = getJSONAttribute("height", JSONout.slides[
                    slidePointer], height); //height of specific slice
1518             width = getJSONAttribute("width", JSONout.slides[
                    slidePointer], width); //width of specific slice
1519
```

```
1520            if (JSONout.slides[slidePointer].labelspath != undefined)
                    {
1521                labelspath = JSONout.slides[slidePointer].labelspath;
1522                loadLabels();
1523            }
1524
1525            loadedJSON = true;
1526
1527            startup();
1528
1529        }
1530
1531    } else {
1532        if (JSONrequest.readyState != 4) {
1533            logError("<b>JSON not ready.</b>");
1534        } else {
1535            logError("<b>Couldn't load JSON.</b> Status:" +
                       JSONrequest.status);
1536        }
1537    }
1538 }
1539
1540 function loadJSON() {
1541 /*requests content of file describing the image stack
1542  */
1543     JSONrequest = getHTTPObject();
1544     JSONrequest.onreadystatechange = JSONread;
1545     JSONrequest.open("GET", JSON, true);
1546     JSONrequest.send(null);
1547 }
1548 /*** END AJAX/FILE LOADING ***/
1549
1550
1551
1552 /*** MOUSE WHEEL HANDLES ***
1553  *Mouse wheel settings (radio buttons):
1554  * wheelMode1()
1555  * wheelMode2()
1556  *Generic handlers:
1557  * handle(delta)
1558  * wheel(event)
1559  */
1560 function wheelMode1() {
1561 /*Mouse wheel used for zoom
1562  */
1563     document.getElementById('chkMW1').checked = true;
1564     document.getElementById('chkMW1fa').className = "fa fa-square";
1565     document.getElementById('chkMW2fa').className = "fa fa-square-o";
1566     wheelmode = 0;
1567 }
1568
1569 function wheelMode2() {
1570 /*Mouse wheel used for slices
1571  */
1572     document.getElementById('chkMW2').checked = true;
1573     document.getElementById('chkMW1fa').className = "fa fa-square-o";
1574     document.getElementById('chkMW2fa').className = "fa fa-square";
1575     wheelmode = 1;
1576 }
1577
```

```
1578 function handle(delta) {
1579 /*handles mouse-wheel rotation
1580  */
1581     //calls the right function with the rotation argument from the
         mouse-wheel.
1582     try {
1583         clearTimeout(ActiveTask);
1584     } catch (err) {}
1585     wheelobs += delta;
1586     if (wheelobs <= -2) {
1587         wheelobs = 0;
1588         if (wheelmode == 0) {
1589             zoomIn();
1590         } else {
1591             slicePrev(delta, true);
1592         }
1593     } else if (wheelobs >= 2) {
1594         wheelobs = 0;
1595         if (wheelmode == 0) {
1596             zoomOut();
1597         } else {
1598             sliceNext(delta, true);
1599         }
1600     }
1601 }
1602
1603 function wheel(event) {
1604 /*called after mouse wheel is moved
1605  */
1606     //called after mouse wheel is moved
1607     var delta = 0;
1608     if (!event) {
1609         event = window.event;
1610     }
1611     if (event.wheelDelta) {
1612         delta = event.wheelDelta / 120;
1613     } else if (event.detail) {
1614         delta = -event.detail / 3;
1615     }
1616     if (delta) {
1617         //alert("wheel");
1618         handle(delta);
1619     }
1620     if (event.preventDefault) {
1621         event.preventDefault();
1622     }
1623     event.returnValue = false;
1624 }
1625
1626 function addWheelEvent() {
1627 /*adds event listener for mousewheel
1628  */
1629     if (window.addEventListener) {
1630         window.addEventListener('DOMMouseScroll', wheel, false); //FF
1631         window.addEventListener('mousewheel', wheel, false); // Opera
             , Chrome,Safari
1632         //window.addEventListener('wheel', wheel, false); //IE9+
1633     }else{
1634         try{
1635             if(elem.attachEvent) {
```

```
1636                elem.attachEvent ("onmousewheel", wheel); // IE8-?
                       IE7 emulator doesn't know this
1637            }
1638        }catch(err){}
1639    }
1640    window.onmousewheel = document.onmousewheel = wheel;
1641 }
1642 /*** END MOUSE WHEEL HANDLES ***/
1643
1644 /*** KEYBOARD HANDLES ***
1645  * capturekey(e)
1646  */
1647 function capturekey(e) {
1648 /*calls appropriate functions in case of key being pressed
1649  */
1650     var k = (typeof event != 'undefined') ? window.event.keyCode : e.
          keyCode;
1651     try {
1652         clearTimeout(ActiveTask);
1653     } catch (err) {}
1654     if (k == 187 || k == 61 || k == 107) {
1655         /* =/+ (in FF 61, else 187) or Numpad+ (107) -> zoom in */
1656         zoomIn();
1657     } else if (k == 189 || k == 173 || k == 109) {
1658         /* -/_ (in FF 173, else 189) or Numpad- (109) -> zoom out */
1659         zoomOut();
1660     } else if (k == 39 || k == 40 || k == 34) {
1661         /* RightArrow (39), DownArrow (40) or PageDown (34) -> next
               slice */
1662         sliceNext(1);
1663     } else if (k == 37 || k == 38 || k == 33) {
1664         /* LeftArrow (37), UpArrow (38) or PageUp (33) -> previous
               slice */
1665         slicePrev(-1);
1666     } else if (k == 82) {
1667         /* r (82) -> increase minimum threshold */
1668         thresLower = Math.min(1 * thresLower + 1,thresUpper);
1669         updateThreshold();
1670         checkTiles(1);
1671     } else if (k == 70) {
1672         /* f (70) -> dencrease minimum threshold */
1673         thresLower = Math.max(1 * thresLower - 1,densmin);
1674         updateThreshold();
1675         checkTiles(1);
1676     } else if (k == 84) {
1677         /* t (84) -> increase maximum threshold */
1678         thresUpper = Math.min(1 * thresUpper + 1,densmax);
1679         checkTiles(1);
1680     } else if (k == 71) {
1681         /* g (71) -> decrease maximum threshold */
1682         thresUpper = Math.max(1 * thresUpper - 1,thresLower);
1683         checkTiles(1);
1684     } else if (k == 81) {
1685         /* q (81) -> zoom in */
1686         zoomIn();
1687     } else if (k == 69) {
1688         /* e (69) -> zoom out */
1689         zoomOut();
1690 //  } else if (k == 87) {
1691 //      /* w (87) -> pan */
```

```
1692 //      pan(-1,0);
1693 //  } else if (k == 83) {
1694 //      /* s (83) -> pan */
1695 //      pan(1,0);
1696 //  } else if (k == 65) {
1697 //      /* a (65) -> pan */
1698 //      pan(0,-1);
1699 //  } else if (k == 68) {
1700 //      /* d (68) -> pan */
1701 //      pan(0,1);
1702     } else if (k == 88) {
1703         /* x (88) -> previous slice */
1704         slicePrev(-1);
1705     } else if (k == 67) {
1706         /* c (67) -> next slice */
1707         sliceNext(1);
1708     }
1709 }
1710
1711 if (navigator.appName != "Mozilla") {
1712     document.onkeyup = capturekey;
1713 } else {
1714     document.addEventListener("keypress", capturekey, true);
1715 }
1716 /*** END KEYBOARD HANDLES ***/
1717
1718
1719 /*** MOUSE HANDLES ***
1720  *Generic:
1721  * is_touch_device()
1722  * adjustRuler()
1723  *Mouse Click Settings (Radio Buttons):
1724  * clickMode1()
1725  * clickMode2()
1726  *Mouse Drag Event:
1727  * startMove(event)
1728  * processMove(event)
1729  * stopMove()
1730  *Apple Specific Events:
1731  * appleStartTouch(event)
1732  * appleMoveEnd(event)
1733  * appleMove(event)
1734  * appleMoving(event)
1735  */
1736 /**Generic**/
1737 function is_touch_device() {
1738 /*checks if client device supports touch commands
1739  */
1740     try {
1741         document.createEvent("TouchEvent");
1742         return true;
1743     } catch (err) {
1744         return false;
1745     }
1746 }
1747
1748 function adjustRuler(){
1749 /*calculates and sets style of the ruler div
1750  */
1751     var innerDiv = document.getElementById("innerDiv");
```

```
1752      var clientX, clientY;
1753      var event = window.event;
1754      if (!event) {
1755          clientX = lasteventX;
1756          clientY = lasteventY;
1757      }else{
1758          clientX = event.clientX;
1759          clientY = event.clientY;
1760      }
1761      Mtop = res * (clientY - dragStartTop) / (1 / (Math.pow(2,
             gTierCount - 1 - zoom) / xtrazoom));
1762      Mleft = res * (clientX - dragStartLeft) / (1 / (Math.pow(2,
             gTierCount - 1 - zoom) / xtrazoom));
1763      Mdist = Math.round(Math.sqrt(Math.pow(Mtop, 2) + Math.pow(Mleft,
             2)) * 10) / 10;
1764
1765      rulerdiv = document.getElementById('ruler');
1766      rulerdiv.alt = Mdist + resunits;
1767      rulerwidth = Math.sqrt(Math.pow((clientY - dragStartTop), 2) +
             Math.pow((clientX - dragStartLeft), 2));
1768      rulerangle = Math.atan2(Mtop, Mleft) * 180 / Math.PI;
1769      rulerdiv.style.top = ((dragStartTop + clientY) / 2) + 'px';
1770      rulerdiv.style.left = ((dragStartLeft + clientX - rulerwidth) /
             2) + 'px';
1771      rulerdiv.style.height = '0px';
1772      rulerdiv.style.width = Math.round(rulerwidth) + 'px';
1773
1774      if (rulerangle < -90 || rulerangle > 90) {
1775          rulerangle = rulerangle + 180;
1776      }
1777      //rotation might be an issue on Safari and on IE9-
1778      rulerdiv.style.transform = "rotate(" + rulerangle + "deg)"; //;-
             ms-transform: rotate(30deg);-webkit-transform: rotate(30deg)
1779      rulerdiv.style.webkitTransform = "rotate(" + rulerangle + "deg)";
1780      rulerdiv.innerHTML = engUnit(Mdist,resunits);
1781      rulerdiv.style.zIndex = "2";
1782      rulerdiv.style.display = "block";
1783  }
1784
1785  /**Mouse Click Settings (Radio Buttons)**/
1786  function clickMode1() {
1787  /*on Mouse Click/Drag: pan/drag the image
1788   */
1789      var ruler = document.getElementById('ruler');
1790      ruler.style.display = "none";
1791      if (is_touch_device()) {
1792          document.getElementById("cmlbl").innerHTML = 'Touch␣Mode:';
1793      }
1794      document.getElementById('chkMC1').checked = true;
1795      document.getElementById('chkMC1fa').className = "fa␣fa-square";
1796      document.getElementById('chkMC2fa').className = "fa␣fa-square-o";
1797      document.getElementById('sleepRuler').style.display = "block";
1798      document.getElementById('outerDiv').style.cursor = "move";
1799      clickmode = 0;
1800  }
1801
1802  function clickMode2() {
1803  /*on Mouse Click/Drag: measure distance
1804   */
1805      if (is_touch_device()) {
```

```
1806          document.getElementById("cmlbl").innerHTML = 'Touch␣Mode:';
1807      }
1808      document.getElementById('chkMC2').checked = true;
1809      document.getElementById('chkMC1fa').className = "fa␣fa-square-o";
1810      document.getElementById('chkMC2fa').className = "fa␣fa-square";
1811      document.getElementById('sleepRuler').style.display = "none";
1812      document.getElementById('outerDiv').style.cursor = "default";
1813      clickmode = 1;
1814  }
1815
1816  /**Mouse Drag Event**/
1817  function startMove(event) {
1818  /*called on mousedown -
1819   *saves initial position and defines mousemove event
1820   */
1821      innerDiv = document.getElementById("innerDiv");
1822      if (!event) {
1823          event = window.event;
1824      }
1825      dragStartLeft = event.clientX;
1826      dragStartTop = event.clientY;
1827      mTop = stripPx(innerDiv.style.top);
1828      mLeft = stripPx(innerDiv.style.left);
1829      if (clickmode == 0) {
1830          dragging = true;
1831      } else {
1832          measuring = true;
1833      }
1834
1835      return false;
1836  }
1837
1838  function processMove(event) {
1839  /*updates the coordinates displayed and
1840   *executes the appropriate function in case of panning or measuring
         active
1841   */
1842      var Mtop, Mleft, Mdist;
1843      var ruler;
1844      var rulerwidth, rulerangle;
1845      innerDiv = document.getElementById("innerDiv");
1846
1847      if (!event) {
1848          event = window.event;
1849      }
1850      if (event) {
1851          lasteventX = event.clientX
1852          lasteventY = event.clientY
1853      }
1854
1855      updatePosition();
1856
1857      if (dragging) {
1858          innerDiv.style.top = mTop + (lasteventY - dragStartTop) + 'px
               ';
1859          innerDiv.style.left = mLeft + (lasteventX- dragStartLeft) + '
               px';
1860      } else if (measuring) {
1861          adjustRuler();
1862      }
```

```
1863 }
1864
1865 function stopMove() {
1866 /*called on mouseup -
1867  *resets the mousemove events and updates tiles in case image was
         panned
1868  */
1869     if (dragging) {
1870         dragging = false;
1871         //only load new tiles once moving has stopped
1872         checkTiles(0);
1873     }
1874     measuring = false;
1875 }
1876
1877 /**Apple Device Event Handlers Block**/
1878 function appleStartTouch(event) {
1879 /*Touch event started
1880  */
1881     innerDiv = document.getElementById("innerDiv");
1882     if (event.touches.length == 1) {
1883         touchIdentifier = event.touches[0].identifier;
1884         dragStartLeft = event.touches[0].clientX;
1885         dragStartTop = event.touches[0].clientY;
1886         mTop = stripPx(innerDiv.style.top);
1887         mLeft = stripPx(innerDiv.style.left);
1888
1889         if (clickmode == 0) {
1890             dragging = true;
1891         } else {
1892             measuring = true;
1893         }
1894         return true;
1895     }
1896 }
1897
1898 function appleMoveEnd(event) {
1899 /*Touch event ended
1900  */
1901     dragging = false;
1902     measuring = false;
1903     appleMove(event);
1904 }
1905
1906 function appleMove(event) {
1907 /*Touch event ongoing
1908  */
1909     var Mtop, Mleft, Mdist;
1910     var ruler;
1911     var rulerwidth, rulerangle;
1912     innerDiv = document.getElementById("innerDiv");
1913     if (event) {
1914         lasteventX = event.changedTouches[0].clientX
1915         lasteventY = event.changedTouches[0].clientY
1916     }
1917     updatePosition();
1918     if ((event.changedTouches.length == 1) && (dragging == true) && (
           touchIdentifier == event.changedTouches[0].identifier)) {
1919         innerDiv.style.top = mTop + (lasteventY - dragStartTop) + 'px
             ';
```

```
1920            innerDiv.style.left = mLeft + (lasteventX - dragStartLeft) +
                    'px';
1921        } else if ((event.changedTouches.length == 1) && (measuring ==
                true) && (touchIdentifier == event.changedTouches[0].
                identifier)) {
1922            adjustRuler();
1923        }
1924        event.preventDefault();
1925        checkTiles(0);
1926    }
1927
1928    function appleMoving(event) {
1929    /*handles apple touch moving event
1930    */
1931        event.preventDefault();
1932        appleMove(event);
1933    }
1934    /*** END MOUSE HANDLES ***/
1935
1936    /*** INITIATION ***
1937     * startup()
1938     * init()
1939     * initOnclicks()
1940     * winsize()
1941     * $(document).ready
1942     */
1943    function startup() {
1944    /*initialises all the variables, loads JSON, etc.
1945    */
1946
1947        // console.timeStamp("startup")
1948
1949        //Threshold sliders
1950        thresLower = Math.max(densmin, -1000);
1951        thresUpper = Math.min(densmax, 1000);
1952
1953        if (typeof jQuery === 'undefined') {
1954            // no jQuery
1955            logError("Please␣make␣sure␣your␣browser␣supports␣jQuery.");
1956        }
1957
1958        winsize();
1959        document.getElementById('error').innerHTML = ""; //error div
                displays overlay at top of screen
1960        //one can use logError and removeError to access it
1961        var divs = document.getElementById("imageLabels").
                getElementsByTagName("div");
1962        while (divs.length > 0){
1963            imageLabels.removeChild(divs[0]);
1964        }
1965        var width2 = getVar('width');
1966        var height2 = getVar('height');
1967        var coords2 = getVar('coords');
1968        if (width2.length > 0) width = width2;
1969        if (height2.length > 0) height = height2;
1970        if (coords2.length > 0) coords = coords2;
1971
1972        var start = getVar('start');
1973        if (start.length > 0) {
1974            slidePointer = parseInt(start, 10);
```

```
1975      }
1976
1977      //try to load JSON. the JSON file can either be specified
              directly, or a root directory can be given.
1978      var rootpath2 = getVar('root');
1979      if (rootpath2.length > 1) rootpath = rootpath2;
1980      var JSON2 = getVar('JSON');
1981      if (JSON2.length > 1) { //if specified directly, then load JSON
              file
1982          JSON = JSON2;
1983      } else {
1984          JSON = rootpath + "/infoJSON.txt"; //otherwise load the file
                with name "infoJSON.txt" from rootpath specified
1985      }
1986      if (JSON.length > 1 && !loadedJSON) { //for 3D images display
              navigation elements for moving through the slices
1987          getJSON = true;
1988          document.getElementById("slices").style.display = "block";
1989          loadJSON();
1990          document.getElementById('wheelMode').style.display = "block";
1991          return;
1992      }
1993
1994      /* resolution (in plane) in px/resunits */
1995      var res2 = getVar('res');
1996      if (res2.length > 1) {
1997          try {
1998              res = parseFloat(res2);
1999          } catch (err) {}
2000      }
2001      if (isNumeric(res)) {
2002          try {
2003              res = parseFloat(res);
2004          } catch (err) {
2005              res = 1.0;
2006          }
2007      } else {
2008          res = 1.0;
2009      }
2010
2011      /* z-resolution (between slides) in px/resunits */
2012      zres2 = getVar('zres');
2013      if (zres2.length > 1) {
2014          try {
2015              zres = parseFloat(zres2);
2016          } catch (err) {}
2017      }
2018      if (isNumeric(zres)) {
2019          try {
2020              zres = parseFloat(zres);
2021          } catch (err) {
2022              zres = 1.0;
2023          }
2024      } else {
2025          zres = 1.0;
2026      }
2027
2028      /* unit for the resolution */
2029      resunits2 = getVar('resunits');
2030      if (resunits2.length > 0)
```

```
2031        resunits = resunits2;
2032    if (resunits.length <= 0) {
2033        resunits = "px";
2034    }
2035
2036    imgpath = path;
2037
2038    //calculate number of zoom levels
2039    gImageWidth = width;
2040    gImageHeight = height;
2041    tempWidth = gImageWidth;
2042    tempHeight = gImageHeight;
2043    divider = 2;
2044    gTierCount = 1;
2045    while (tempWidth > tileSize || tempHeight > tileSize) {
2046        tempWidth = Math.floor(gImageWidth / divider)
2047            tempHeight = Math.floor(gImageHeight / divider);
2048        divider *= 2;
2049        if (tempWidth % 2)
2050            tempWidth++;
2051        if (tempHeight % 2)
2052            tempHeight++;
2053        gTierCount++;
2054    }
2055    vTier2 = getVar('vT');
2056    if (vTier2.length > 0) {
2057        zoom = Math.max(0,vTier2);
2058        if (zoom > gTierCount - 1){
2059            xtrazoom = Math.min(zoom - gTierCount + 1,xtrazoomMax);
2060            zoom = gTierCount - 1;
2061        }
2062    } else {
2063        zoom = gTierCount - 1;
2064    }
2065
2066    //position innerDiv containing the image relative to viewport
2067    centreView();
2068
2069    /*attach functions to outerDiv which contains the innerDiv
2070     * this is important as the user may click in an area outside the
2071     * actual image and this way can still interact with the image,
2072     * e.g. if the image is out of the viewing area
2073     */
2074    var outerDiv = document.getElementById("outerDiv");
2075    outerDiv.onmousedown = startMove;
2076    outerDiv.onmousemove = processMove;
2077    outerDiv.onmouseup = stopMove;
2078    outerDiv.ondragstart = function () {
2079        return false;
2080    }
2081
2082    /*Capture Mobile Device Events*/
2083    outerDiv.ontouchstart = appleStartTouch;
2084    outerDiv.ontouchend = appleMoveEnd;
2085    outerDiv.ontouchmove = appleMoving;
2086    outerDiv.ongesturestart = function (event) {
2087        event.preventDefault();
2088        gestureScale = event.scale;
2089        parent.document.ontouchmove = function (event) {
2090            event.preventDefault();
```

```
2091            };
2092        }
2093        outerDiv.ongestureend = function (event) {
2094            event.preventDefault();
2095            if (event.scale > gestureScale) {
2096                zoomIn();
2097            } else {
2098                zoomOut();
2099            }
2100            parent.document.ontouchmove = null;
2101        };
2102
2103        /*now call original initialisation function for the rest.*/
2104        winsize();
2105        init();
2106
2107        /*Capture touch device events*/
2108        if (is_touch_device()) {
2109            /*for touch devices no mouse wheel support, but touching =
                   zooming*/
2110            document.getElementById("wheelMode").style.display = 'none';
2111            document.getElementById("cmlbl").innerHTML = 'Touch␣Mode:';
2112        } else {
2113            /*for normal devices mousewheel support includes different
                   modes*/
2114            document.getElementById("wheelMode").style.display = 'block';
2115        }
2116
2117
2118        /*entry animation
2119         * moves in toolbox from the side, once ready
2120         * this is not onlz a nice effect, but prevents
2121         * users from trying to access functions before
2122         * they have been loaded. Unfortunately on older
2123         * browsers animations aren't supported so the
2124         * panel will be off screen and never appear
2125         */
2126        setTimeout(function () {
2127            $("#overlay").animate({
2128                left : "0px"
2129            }, 1500, "swing");
2130        }, 1500);
2131
2132        // console.timeStamp("startup end");
2133
2134        initOnclicks();
2135        addWheelEvent();
2136    }
2137
2138    function init() {
2139    /*initiate, but also called for new slides...
2140    */
2141        var imageLabels = document.getElementById("imageLabels");
2142        var divs = imageLabels.getElementsByTagName("div");
2143
2144        // try {
2145        //  Tstart = performance.now();
2146        // } catch (err) {
2147        //  Tstart = new Date().getTime();
2148        // }
```

```
2149
2150        while (divs.length > 0) {
2151            imageLabels.removeChild(divs[0]);
2152        }
2153        imgpath = path;
2154        ActiveTask = setTimeout(function () {
2155            checkTiles(1);
2156        }, 0);
2157        updateInfo(); //check if all necessary
2158    } //*** End of Init()
2159
2160    function initOnclicks() {
2161    /*initiates html elements
2162    */
2163        /*hide elements*/
2164        document.getElementById("controls").style.display = "none";
2165        /*create onclick events*/
2166        document.getElementById('chkMW1').onclick = wheelMode1;
2167        document.getElementById('chkMW2').onclick = wheelMode2;
2168        document.getElementById('chkMW1div').onmouseup = wheelMode1;
2169        document.getElementById('chkMW2div').onmouseup = wheelMode2;
2170        document.getElementById('chkMC2').onclick = clickMode2;
2171        document.getElementById('chkMC1').onclick = clickMode1;
2172        document.getElementById('chkMC2div').onmouseup = clickMode2;
2173        document.getElementById('chkMC1div').onmouseup = clickMode1;
2174        document.getElementById("cntrlButton").onmouseup = clickControls;
2175        document.getElementById("closeControls").onmouseup = hideControls
               ;
2176        document.getElementById("zoomouticon").onmouseup = zoomOut;
2177        document.getElementById("zoominicon").onmouseup = zoomIn;
2178        document.getElementById("sliceprevicon").onmouseup = slicePrevDef
               ;
2179        document.getElementById("slicenexticon").onmouseup = sliceNextDef
               ;
2180        document.getElementById("zoomouticonfa").onmouseup = zoomOut;
2181        document.getElementById("zoominiconfa").onmouseup = zoomIn;
2182        document.getElementById("slicepreviconfa").onmouseup =
               slicePrevDef;
2183        document.getElementById("slicenexticonfa").onmouseup =
               sliceNextDef;
2184    }
2185
2186    function winsize() {
2187    /*repositions the image and deals with variable changes in case of
           the window being rescaled
2188     *this always happens on page load!
2189     */
2190        var contentbox, overlayTop;
2191        var errmsg = "<b>Failed␣on␣resize:</b>␣";
2192        tviewportWidth = viewportWidth;
2193        tviewportHeight = viewportHeight;
2194
2195        if (typeof(window.innerWidth) == 'number') {
2196            viewportWidth = window.innerWidth;
2197            viewportHeight = window.innerHeight;
2198        } else if (document.documentElement && (document.documentElement.
               clientWidth || document.documentElement.clientHeight)) {
2199            viewportWidth = document.documentElement.clientWidth;
2200            viewportHeight = document.documentElement.clientHeight;
2201        } else if (document.body && (document.body.clientWidth ||
```

```
        document.body.clientHeight)) {
2202        viewportWidth = document.body.clientWidth;
2203        viewportHeight = document.body.clientHeight;
2204    }
2205
2206    try {
2207        contentbox = document.getElementById('contentbox');
2208        viewportWidth = contentbox.clientWidth;
2209        viewportHeight = contentbox.clientHeight;
2210    } catch (err) {
2211        //setTimeout("winsize();", 5000);
2212    }
2213
2214
2215    if (viewportHeight != tviewportHeight || viewportWidth !=
        tviewportWidth) {
2216        if (viewportHeight != tviewportHeight){
2217            try {
2218                overlayTop = Math.max(Math.min((viewportHeight - 500)
                        , 145), 0);
2219                document.getElementById("overlay").style.top =
                        overlayTop + 'px';
2220                document.getElementById("overlay").style.height = (
                        viewportHeight - overlayTop) + 'px';
2221            } catch (err) {
2222                logError(errmsg + err);
2223                //Fails on load as element not yet created
2224            }
2225        }
2226        try {
2227            if (viewportWidth >= 700) {
2228                if(loadedJSON){
2229                  showThumb();
2230                }
2231                showControlsBtn();
2232            } else if (viewportWidth <= 600) {
2233                hideThumb();
2234                hideControlsBtn();
2235            }
2236            removeError(errmsg);
2237            //updateInfo();
2238        } catch (err) {
2239            logError(errmsg + err);
2240            //Fails on load as element not yet created
2241        }
2242    }
2243
2244 }
2245
2246 window.onresize = winsize;
2247
2248 /*execute once page is loaded*/
2249 $(document).ready(
2250     function(){
2251         winsize();
2252         startup();
2253     }
2254 );
2255 /*** END INITIATION ***/
```

CSS Stylesheet

Listing 5: code for MCTV.css

```
@charset 'UTF-8';
body{
    height: 100%;
    margin: 0pt;
    background-color: #000;
    color: white;
}

a{
    border: 0px;
    border-color: #000;
    color: #ffffff;
    text-decoration: none;
}
a:hover{
    border: 0px;
    border-color: #000;
    color: #ffffff;
    text-decoration: none;
    font-size: 110%;
}
a:visited{
    border: 0px;
    border-color: #000;
    color: #ffffff;
    text-decoration: none;
}

div#outerDiv{
    position: relative;
    overflow: hidden;
    height: 100%;
    width: 100%;
    border: 1px solid black;
    cursor: move;
}

div#outerDiv0 {
    position: absolute;
    overflow: hidden;
    top: 0px;
    left: 0px;
    height: 100%;
    width: 100%;
}


.nofa{
    /*set this to "block" if not using font-awesome*/
    display: none;
}

div#outerDiv{
    position: relative;
    overflow: hidden;
    height: 100%;
```

```
57      width: 100%;
58      border: 1px solid black;
59  }
60
61  div#outerDiv0 {
62      position: absolute;
63      overflow: hidden;
64      top: 0px;
65      left: 0px;
66      height: 100%;
67      width: 100%;
68  }
69
70  /*toolbox*/
71  div#overlay {
72      position: absolute;
73      z-index: 100;
74      height: 100%;
75      padding-top: 0;
76      background-color: #ffffff;
77      background-color: rgba(255, 255, 255, 0.5);
78      border-width: 0px;
79      border-color: #266c8e;
80      color: #000;
81      font-size: 16px;
82      font-family: sans-serif;
83      text-shadow: none;
84      text-align: left;
85  }
86
87  /*slider*/
88  /*slider colour*/
89  .slider.active {
90      color: #369aca;
91      background-color: #369aca;
92  }
93  .slider.active2 {
94      color: #266c8e;
95      background-color: #266c8e;
96  }
97  .slider.inactive {
98      color: #d4d4d4;
99      background-color: #d4d4d4;
100 }
101 .slider.knob {
102     color: #000000;
103     background-color: #000000;
104 }
105 .slider.active:hover {
106     opacity: 0.6;
107     cursor: pointer;
108 }
109 .slider.active2:hover {
110     opacity: 0.6;
111     cursor: pointer;
112 }
113 .slider.inactive:hover {
114     opacity: 0.6;
115     cursor: pointer;
116 }
```

```
117 /*slider icons*/
118 div.icon{
119     position: absolute;
120     z-index: 102;
121 }
122 div.icon img{
123     width: 20px;
124     border-width: 0px;
125 }
126 div.icon img:hover{
127     width: 22px;
128     border-width: 0px;
129     line-height: 20px;
130     cursor: pointer;
131 }
132 div.icon span{
133     width: 20px;
134     border-width: 0px;
135     color: #266c8e;
136 }
137 div.icon span:hover{
138     border-width: 0px;
139     font-size: 110%;
140     line-height: 20px;
141     cursor: pointer;
142 }
143 div.icon.left{
144     left: 10px;
145 }
146 div.icon.right{
147     left: 190px;
148 }
149 div.icon.zoom{
150     top: 15px;
151 }
152 div.icon.slices{
153     top: 45px;
154 }
155 /*slider positioning*/
156 div.slider{
157     position: absolute;
158     z-index: 101;
159     left: 15px;
160     height: 10px;
161     width: 190px;
162 }
163 div.slider.zoom{
164     top: 20px;
165 }
166 div.slider.slices{
167     top: 50px;
168 }
169 /*slider tables*/
170 #thresslidertable{
171     table-layout: fixed;
172     height: 2px;
173     min-height: 10px;
174     border: 0px;
175     border-collapse: collapse;
176     margin: 0;
```

```
177     padding: 0;
178 }
179 #zoomslidertable{
180     table-layout: fixed;
181     height: 2px;
182     min-height: 10px;
183     min-width: 0px;
184     border: 0px;
185     border-collapse: collapse;
186     margin: 0;
187     padding: 0;
188 }
189 #sliceslidertable{
190     table-layout: fixed;
191     height: 2px;
192     min-height: 10px;
193     min-width: 0px;
194     border: 0px;
195     border-collapse: collapse;
196     margin: 0;
197     padding: 0;
198 }
199
200 /*radio buttons*/
201 .optionbtnlbl input{
202     position: relative;
203     left: 0px;
204     height: 15px;
205     width: 15px;
206     line-height: 20px;
207 }
208 .optionbtnlbl{
209     position: relative;
210     left: 30px;
211     height: 20px;
212     width: 150px;
213     background-color: #266c8e;
214     margin-bottom: 2px;
215     color: #ffffff;
216     text-align: left;
217     line-height: 20px;
218 }
219 .optionbtnlbl:hover{
220     font-size: 110%;
221     line-height: 20px;
222     cursor: pointer;
223 }
224
225
226 /*ruler*/
227 .ruler {
228     display: block;
229     z-index: 200;
230     height: 0px;
231     border: 1px solid #369aca;
232 }
233 .rulerlbl{
234     color: white;
235     text-align: center;
236     text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px
```

```
        black;
}
#sleepRuler{
    position: absolute;
    left: 30px;
    width: 150px;
}
hr.scale {
    width: 50px;
}


/*thumbnail*/
div.thumbnail {
    display: none;
    position: absolute;
    overflow: hidden;
    z-index: 102;
    width: 256px;
    border: 5px solid #ffffff;
    border: 5px solid rgba(255, 255, 255, 0.5);
    color: white;
    text-align: center;
    text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px
        black;
}
/*thumbnail position indicator*/
div.thumbpos {
    display: none;
    position: absolute;
    overflow: hidden;
    z-index: 101;
    background-color: rgba(54, 154, 202, 0.03);
    border: 1pt solid #369aca;
    cursor: pointer;
}
/*thumbnail image*/
div.thumbimg {
    display: none;
    position: absolute;
    overflow: hidden;
    z-index: 100;
    top: 0px;
    right: 0px;
    cursor: pointer;
}


/*key controls info*/
div.controls{
    position: absolute;
    overflow: hidden;
    z-index: 500;
    top: 25%;
    left: 25%;
    height: 50%;
    width: 50%;
    background-color: #ffffff;
    background-color: rgba(255, 255, 255, 0.75);
    opacity: 0;
```

```
}
div.controls img{
    display: block;
    position: relative;
    top: 5%;
    max-height: 90%;
    max-width: 90%;
    margin: auto;
}
div.close{
    display: block;
    position: absolute;
    top: 0px;
    right: 0px;
    height: 20px;
    width: 20px;
    background-color: rgba(255, 255, 255, 0);
    color: #266c8e;
    font-family: Arial, Verdana, sans-serif;
}
div.close:hover{
    font-size: 110%;
    cursor: pointer;
}
#cntrlButton{
    position: absolute;
    left: 30px;
    height: 20px;
    width: 150px;
    background-color: #266c8e;
    color: #ffffff;
    text-align: left;
    line-height: 20px;
}
#cntrlButton:hover{
    font-size: 110%;
    line-height: 20px;
    cursor: pointer;
}


/*error bar and debug*/
div.error {
    position: absolute;
    overflow: hidden;
    z-index: 900;
    top: 0px;
    width: 100%;
    max-height: 20%;
    background-color: #ff0000;
    background-color: rgba(255, 0, 0, 0.5);
    border-width: 0px;
    padding-top: 0;
    color: #ffffff;
    font-size: 16px;
    font-family: sans-serif;
    text-align: center;
    text-shadow: none;
}
.debug {
```

```
355     display: none;
356 }
357 
358 /*info*/
359 /*reference bar*/
360 div#copy a {
361     color: gray;
362     font-family: sans-serif;
363     text-align: center;
364     text-decoration: none;
365     text-shadow: none;
366 }
367 /*Southampton logo*/
368 .whitelogo{
369     border: 0px;
370     -webkit-filter: grayscale(100%) brightness(100);
371     -moz-filter: grayscale(100%) brightness(100);
372     -ms-filter: grayscale(100%) brightness(100);
373     -o-filter: grayscale(100%) brightness(100);
374     filter: grayscale(100%) brightness(100);
375     /*-ms-filter:progid:DXImageTransform.Microsoft.BasicImage(
            grayscale=1);*/
376 }
```