Documentation of **TutorialGem**

Gemma Martini

January 10, 2018

Contents

1 Overview

TutorialGem is a `JS` library to create tutorials for web pages. It's very customizable and easy to use.
Once it's given the parameters it creates the whole tutorial, leaving the programmer free of worries. Nonetheless, it is possible to change the appearance of the tutorial by changing the colour of the tool tips, the arrows, their orientations and the background color of the elements.

2 Compatibility

TutorialGem works on web pages developed in HTML, Javascript and CSS. It's not guaranteed to work with particular JS libraries such as Angular and React. In particular, some dynamic mobile webpages don't allow this library to find out the necessary properties: it is recommended that you check before releasing the page with tutorial.
You will get the best results from the usage of **TutorialGem** if the elements of the tutorial have a quite small size, relatively to the size of the window. In order to use **TutorialGem** properly you should not use as tutorial elements those parts of the page that run scripts as they are loaded. Not following this good practice may lead to multiple executions of the script.
TutorialGem works on Chrome (from version 36), IE/Edge (from version 10), Firefox (from version 16), Safari (from version 9), Opera/Vivaldi (from version 23).

3 Usage

Before starting to discuss how to use the library some suggestion for a correct usage:

- Avoid using as tutorial elements nodes whose width or height is bigger than 9/10 of the window size. The layout of the tutorial will reduce the dimensions of that item, instead of increasing it;
- In order to give a better user experience you're suggested to use a variable in your `HTML` page to control the start of the tutorial. You may improve the user experience if you keep track of every single user's choice (for example using cookies to remember preferences);
- **TutorialGem** places arrows from the tool tip to the next element of the tutorial as shown in Figure 1. The ending position of the arrow may be changed by the user to `top, bottom, left` or `right`. If the choice isn't appropriate the arrow may look bad.

Let's now talk about how **TutorialGem** should be used. First of all you should decide the chain of the elements that need to be shown in the tutorial. Let's suppose you want to train the user to navigate a navBar made of 4 buttons, called `but1, but2, but3, but4` in this order. You have to create an array containing five objects with those fields:

`node` this element is the node you want to explain;

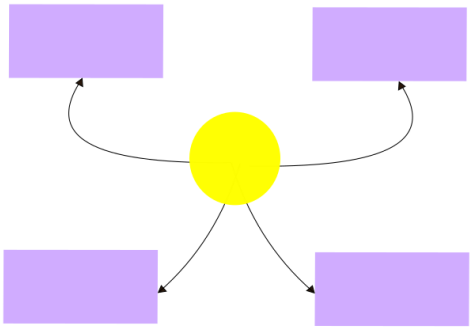

Figure 1: This is the standard orientation of arrows

`fromWhere` represents the position of the tip of the arrow on that node;

`backgroundColor` this field represents the colour of the background you would like to have behind the node;

`backgroundColor2` this is the background colour of the tooltip window;

`description` this is the text that should be written inside the tooltip window;

`myArrowColor` is the color of the arrow from the toolTip to the next element.

The colours should be written in `hex format`. Each of these descriptions should have from 30 to 240 characters, blank spaces included, so that the font size of the text doesn't become too small or too big.
The first of these five objects should have the first three fields unset, or set to **null**, so it will be considered as the fist introduction to this tutorial. On the other hand the last field shouldn't be specified in the last element, since the tool tip doesn't have anywhere to point at. Once the arroy is created, you can choose a function to be called when the tutorial is closed by the user. The function `tutGem` should be called with as first parameter the array and as second the function.

Here follows an example of usage:

```
//Step 1 create the data structure with the elements of the tutorial
//myNode, fromWhere, myBackgroundColor, myBackgroundColor2,
//myDescription, myArrowColor
var array = [];
//First toolTip
array.push(new genericNode(null, null, null, '#ffff66',
  "Hey mate, <br/> Let's say this is a travelling agency website.
  <br/> I'm your tool tip, here to show you around. <br/> See this
  arrow? Follow it and click on the highlighted element", null));
$(document).ready(function() {
  //First element of the tutorial
  array.push(new genericNode(document.getElementById("specialOffers"),
```

```
    null, '#ff0000', '#ffff66', "In this section you can find all our
    special offers! <br/> Never forget to have a look!", "#ffffcc"));
  //Second element of the tutorial
  array.push(new genericNode(document.getElementById("chooseDest"),
    null, '#9933ff', '#ffff66', "Do you have a place in particular on
    your mind? Here's where you can search your destination and find
    out how cheap it is to go there with us!", "#ffffcc"));
  //Third element of the tutorial
  array.push(new genericNode(document.getElementById("most"), "top",
    '#ffffff', '#ffff66', "Here are the most requested of our arranged
    travels. If you haven't decided yet where to go or what you want
    from your holidays don't miss this section. It will help you
    understanding what's popular! In the end, there must be a reason
    if something is very popular...", "#ffffcc"));
  //Fourth element of the tutorial
   array.push(new genericNode(document.getElementById("dest"), null,
    '#ffffff', '#ffff66', 'This is the destination column of the most
    requested travels. You can scroll it till you see a place that
    intrigues you.', "#ffffcc"));
  //Fifth element of the tutorial
  array.push(new genericNode(document.getElementById("descr"),
    "bottom", '#ffffff', '#ffff66', 'This description might help you
   in case you do not know much about that destination', "#ffffcc"));
  //Sixth element of the tutorial
  array.push(new genericNode(document.getElementById("contact"),
    null, '#88cc00', '#ffff66', 'Clicking here will lead you to all
    the directions to contact us. We hope you liked your first trip,
    although it is only around this page. We are sure that your next
    will be around the World. Click here to go to the main page<br/>
    <button type="button" class="btn btn-danger">Main page</button>',
    "#ffffcc"));
  setTimeout(function() {
    tutGem(array, function() {
      alert('Chiusura tutorial');
    });
  }, 10000);
});
```

4 Implementation

4.1 Overview

This library is made of some global variables (which represent a sort of "configuration file") and by the following functions:

`toolTipDimensions()` calculates the final dimensions of the toolTip based on the dimen-

sions of the window

`toolTipWidth()` **and** `toolTipHeight()` return respectively width and height of the tool tip, as calculated by `toolTipDimensions`;

`genericNode (myNode, fromWhere, myBackgroundColor, myBackgroundColor2, myDescription, myArrowColor)` This function creates a data structure which has those fields;

`scaleAndTranslate(node, mulFactor, transX, transY)` This function translates `node` of `(transX, transY)` first, then scales it of `mulFactor`

`createDiv (left, top, width, height, backgroundColor, borderRadius)` this function creates a div with the given parameters and returns it

`createBackground (elLeft, elTop, elWidth, elHeight, backgroundColor, borderRadius, withFrame)` This function creates a div in a position carefully decided considering the position of the element it should contain and returns it. The flag `withFrame` gives the chance to decide if the background should make a frame around the element or not;

`function shadePage()` This function creates a div that makes the whole page shaded

`drawArrow(startEl, finishEl, whereToPoint)` draws an arrow from `startEl` to `finishEl`. The vertexes names in the code can be seen in picture 2. There may be several different situations that are synthetized as follows: there are 4 different orientations for the arrow (4 different positions for `D, C, B, F, E`) and the heuristic decided to make the arrow look good in almost all cases is:

1. According to the angle that the segment between `A` and `B` makes an orientation of the arrow is decided. If the user decides otherwise that decision will change the position of the arrow;
2. That angle gets twice its previous dimension (but not more than $\pi/2$);
3. Around that line two very small angles are drawn and on that `CPA` and `CPG` are calculated.

`synchronizeCssStyles(src, destination, recursively)` this function copies the style (recursively if that flag has value `true`) of `src` into `destination`;

`focusItemCallback (toolTip, finalLeft, finalTop, scalingFactor, originalWidth, originalHeight)` this function is called inside `animatedZoom` when the animating element isn't the tool tip. In this function it is decided where to put the tool tip and than it is called the function that creates and animates the tool tip (`createToolTip`);

`createToolTipCallback (toolTip, finalLeft, finalTop, scalingFactor, originalWidth, originalHeight)` is called inside `animatedZoom` when the animating element is a tool tip. In this function, if it isn't the last step of the tutorial, the arrow between the tool tip and the next element is drawn and that element is kept light but not clickable;

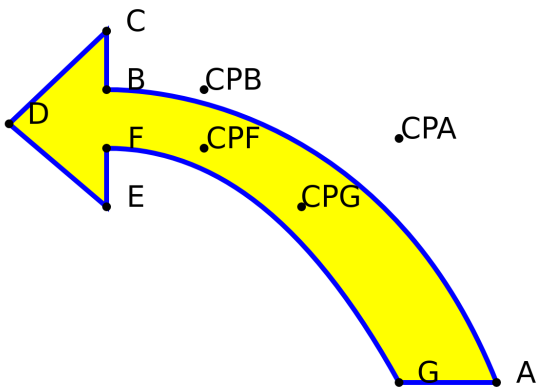

Figure 2: This is a sample of the arrow

`scrollIntoViewIfNeeded(el)` this function scrolls the page in order to keep `el` in the view only when it gets out of the view;

`LightenDarkenColor(col, amt)` this function makes `col` lighter or darker based on the value of `amt`;

`rgb2hex(rgb)` this function translates a color from `rgb` format to `hex`;

`drawTrapezoid(fromEl, toEl)` is used to draw the shadow of `toEl` from `fromEl`. There are four different directions for the trapezoids and they are chosen according to the relative position of the two rectangles `toEl` and `fromEl`. The colors of the shadows is chosen reducing the intensity of the color of the background of the bigger element. In the function the name of vertexes corresponds to the following sample (figure 3).

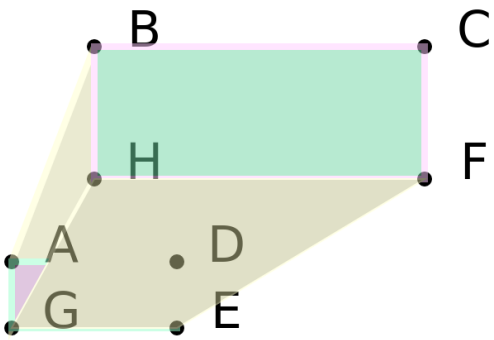

Figure 3: This is a sample of the shadow between two items

```
animatedZoom (currCallback, animatingEl, finalLeft, finalTop,
    scalingFactor, frameRate, numberOfTransformations, shrinkingFactor)
```

This function animates `animatingEl` moving it to its final position (`finalLeft, finalTop`) and scaling it of scalingFactor. The final position has been calculated before the scaling, since `scaleAndTranslate` translates it first. `frameRate` represents how often the animation should happen and `numberOfIterations` is the number of steps of the animation.
`shrinkingFactor` is used only when the animating element is a tool tip and it's 1 otherwise. At the end of the animation `currCallback` is called in order to draw some more things;

`focusItem(el)` creates a copy of the element, adds a background to it and animates it. Its final position is chosen according to the position of the next element if the tutorial, if it exists;

`createToolTip(x, y)` creates a tooltip centered in (`x,y`) and animates it;

`closeStep()` closes a step of the tutorial;

`tutStep(step)` produces the given step of the tutorial: it makes the whole page shaded and runs `createToolTip` or `focusItem`;

`tutGem(tutorialTable, callback)` sets all the parameters to call `tutStep` and chooses the behaviour in case of window resizing.

4.2 Code

4.2.1 toolTipDimensions

```
function toolTipDimensions() {
  var ratio = 1 / 5;
  var myWidth = window.innerWidth * ratio;
  if (myWidth > 300)
    myWidth = 300;
  if (myWidth < 100)
    myWidth = 100;

  var myHeight = window.innerHeight * ratio;
  if (myHeight > 300)
    myHeight = 300;
  if (myHeight < 100)
    myWidth = 100;
  // Both of them are in a range between 100 and 300
  if (myWidth > myHeight)
    myHeight = myWidth;
  else
    myWidth = myHeight;
  return [ myWidth, myHeight ];
}
```

4.2.2 toolTipWidth

```
function toolTipWidth() { return toolTipDimensions()[0]; }
```

4.2.3 toolTipHeight

```
function toolTipHeight() { return toolTipDimensions()[1]; }
```

4.2.4 genericNode

```
function genericNode(myNode, fromWhere, myBackgroundColor, myBackgroundColor2,
                     myDescription, myArrowColor) {
  this.myNode = myNode;
  this.fromWhere = fromWhere;
  this.myBackgroundColor = myBackgroundColor;
  this.myBackgroundColor2 = myBackgroundColor2;
  this.myDescription = myDescription;
  this.myArrowColor = myArrowColor;
}
```

4.2.5 scaleAndTranslate

```
function scaleAndTranslate(node, mulFactor, transX, transY) {
  var moveX = transX + 'px';
  var moveY = transY + 'px';
  node.style.transform = "translate(" + moveX + ", " + moveY + ") scale(" +
                         mulFactor + "," + mulFactor + ")";
}
```

4.2.6 createDiv

```
function createDiv(left, top, width, height, backgroundColor, borderRadius) {
  var currDiv = document.createElement("div");
  currDiv.style.backgroundColor = backgroundColor;
  currDiv.className = "highLayer";
  currDiv.style.position = "absolute";
  currDiv.style.top = top + "px";
  currDiv.style.left = left + "px";
  currDiv.style.width = width + "px";
  currDiv.style.height = height + "px";
  currDiv.style.borderRadius = borderRadius + "px";
  document.body.appendChild(currDiv);
  return currDiv;
}
```

4.2.7 createBackground

```
function createBackground(elLeft, elTop, elWidth, elHeight, backgroundColor,
                          borderRadius, withFrame) {
  var underEl;
  if (withFrame)
    underEl = createDiv(elLeft + borderRadius / 2, elTop + borderRadius / 2,
                        elWidth + borderRadius, elHeight + borderRadius,
                        backgroundColor, borderRadius);
  else
    underEl = createDiv(elLeft, elTop, elWidth, elHeight, backgroundColor,
                        borderRadius);
  return underEl;
}
```

4.2.8 shadePage

```
function shadePage() {
  var shadeDiv = document.getElementById("shadeDiv");
  if (shadeDiv == null) {
    shadeDiv = document.createElement("div");
    shadeDiv.id = "shadeDiv";
    var currWidth, currHeight;
    currWidth = document.body.getBoundingClientRect().width;
    currHeight = document.body.getBoundingClientRect().height;
    if (currWidth < window.innerWidth)
      currWidth = window.innerWidth;
    if (currHeight < window.innerHeight)
      currHeight = window.innerHeight;
    shadeDiv.style.width = currWidth + "px";
    shadeDiv.style.height = currHeight + "px";
    document.body.appendChild(shadeDiv);
  }
  shadeDiv.className = "shade superBottomLayer";
}
```

4.2.9 drawArrow

```
function drawArrow(startEl, finishEl, whereToPoint) {
  // The application point changes its position due to the destination
  // position relatively to the origin Using getBoundingClientRect to be sure
  // to have the right position and dimension
  rect = startEl.getBoundingClientRect();
  var top0 = rect.top + window.pageYOffset;
  var left0 = rect.left + window.pageXOffset;
  var width0 = rect.width;
  var height0 = rect.height;
```

```
var rect = finishEl.getBoundingClientRect();
var leftD = rect.left + window.pageXOffset;
var topD = rect.top + window.pageYOffset;
var widthD = rect.width;
var heightD = rect.height;
var originX = leftO + widthO / 2;
var originY = topO + heightO / 2;
var destinationX, destinationY;
var CE = heightD / 2;
if (CE > 1 / 5 * window.innerHeight)
  CE = 1 / 5 * window.innerHeight;
if (CE < 1 / 20 * window.innerHeight)
  CE = 1 / 20 * window.innerHeight;
var xB, yB, xC, yC, xE, yE, xF, yF, xCPF, yCPF, xCPB, yCPB, xCPA, yCPA, xCPG,
    yCPG;
// There are 8 possibilities in order to draw a nice arrow
// According to the picture in the documentation:
// D=(destinationX, destinationY)
// B=(destinationX +3/2 widthD/2, destinationY-CE/6)
// F=(xB, destinationY + CE/6)
// C=(xB, destinationY - CE/2)
// E=(xB, destinationY + CE/2)
// G=(originX, originY)
// A=(originX, originY)
// For the control points there is an heuristic which takes into consideration
// the position of the arrow, its orientation and the position of the origin
var xA = originX;
var yA = originY;
var xG = originX;
var yG = originY;
var xMiddleCE, yMiddleCE; // the mid point of CE
if (whereToPoint == null) {
  if (topD + heightD / 2 < (topO + (heightO / 2)))
    whereToPoint = "bottom";
  if (topD + heightD / 2 >= (topO + (heightO / 2)) &&
      (leftD + widthD / 2 < (leftO + (widthO / 2))))
    whereToPoint = "right";
  if (topD + heightD / 2 >= (topO + (heightO / 2)) &&
      (leftD + widthD / 2 >= (leftO + (widthO / 2))))
    whereToPoint = "left";
}
var CPBDistance = 3 * CE;
var mulAngle = 3;
if (whereToPoint === "bottom") {
  destinationX = leftD + widthD / 2;
  destinationY = topD + heightD;
```

```
    yB = destinationY + CE;
    xB = destinationX - CE / 6;
    yC = yB;
    xC = destinationX - CE / 2;
    yE = yB;
    xE = destinationX + 1 / 2 * CE;
    yF = yB;
    xF = destinationX + 1 / 6 * CE;
    xCPB = xB;
    yCPB = yB + CPBDistance;
    xCPF = xF;
    yCPF = yF + CPBDistance;
  } else if (whereToPoint === "top") {
    destinationX = leftD + widthD / 2;
    destinationY = topD;
    yB = destinationY - CE;
    xB = destinationX + CE / 6;
    yC = yB;
    xC = destinationX + CE / 2;
    yE = yB;
    xE = destinationX - 1 / 2 * CE;
    yF = yB;
    xF = destinationX - 1 / 6 * CE;
    xCPB = xB;
    yCPB = yB - CPBDistance;
    xCPF = xF;
    yCPF = yF - CPBDistance;
  } else if (whereToPoint === "right") {
    destinationX = leftD + widthD;
    destinationY = topD + heightD / 2;
    xB = destinationX + CE;
    yB = destinationY + CE / 6;
    xC = xB;
    yC = destinationY + CE / 2;
    xE = xB;
    yE = destinationY - 1 / 2 * CE;
    xF = xB;
    yF = destinationY - 1 / 6 * CE;
    xCPB = xB + CPBDistance;
    yCPB = yB;
    xCPF = xF + CPBDistance;
    yCPF = yF;
  } else if (whereToPoint === "left") {
    destinationX = leftD;
    destinationY = topD + heightD / 2;
    xB = destinationX - CE;
```

```
  yB = destinationY - CE / 6;
  xC = xB;
  yC = destinationY - CE / 2;
  xE = xB;
  yE = destinationY + 1 / 2 * CE;
  xF = xB;
  yF = destinationY + 1 / 6 * CE;
  xCPB = xB - CPBDistance;
  yCPB = yB;
  xCPF = xF - CPBDistance;
  yCPF = yF;
}

xMiddleCE = (xC + xE) / 2;
yMiddleCE = (yC + yE) / 2;
// Changing the orientation of y axis
var angle = Math.atan2(yA - yMiddleCE, xMiddleCE - xA);
var controlAngle;
// Quadrant 1
if ((0 <= angle) && (angle <= Math.PI / 2)) {
  if ((whereToPoint === "left") || (whereToPoint === "top")) {
    controlAngle = mulAngle * angle;
    if (controlAngle > Math.PI / 2)
      controlAngle = Math.PI / 2;
  }
  if ((whereToPoint === "right") || (whereToPoint === "bottom")) {
    controlAngle = (Math.PI - angle) * mulAngle;
    if (controlAngle > Math.PI / 2)
      controlAngle = Math.PI / 2;
    controlAngle = Math.PI / 2 - controlAngle;
  }
}
// Quadrant 2
if (angle > Math.PI / 2) {
  if ((whereToPoint === "left") || (whereToPoint === "bottom")) {
    controlAngle = mulAngle * (angle - Math.PI / 2);
    if (controlAngle > Math.PI / 2)
      controlAngle = Math.PI / 2;
    controlAngle = Math.PI / 2 + controlAngle;
  }

  if ((whereToPoint === "right") || (whereToPoint === "top")) {
    controlAngle = (Math.PI - angle) * mulAngle;
    if (controlAngle > Math.PI / 2)
      controlAngle = Math.PI / 2;
    controlAngle = Math.PI - controlAngle;
```

```
  }
}
// Quadrant 3
if (angle <= -Math.PI / 2) {
  if ((whereToPoint === "left") || (whereToPoint === "top")) {
    controlAngle = (angle + Math.PI / 2) * mulAngle;
    if (controlAngle < -Math.PI / 2)
      controlAngle = -Math.PI / 2;
    controlAngle = controlAngle - Math.PI / 2;
  }
  if ((whereToPoint === "right") || (whereToPoint === "bottom")) {
    controlAngle = (Math.PI + angle) * mulAngle;
    if (controlAngle > Math.PI / 2)
      controlAngle = Math.PI / 2;
    controlAngle = controlAngle - Math.PI;
  }
}
// Quadrant 4
if ((angle > -Math.PI / 2) && (angle < 0)) {
  if ((whereToPoint === "left") || (whereToPoint === "bottom")) {
    controlAngle = angle * mulAngle;
    if (controlAngle < -Math.PI / 2)
      controlAngle = -Math.PI / 2;
  }
  if ((whereToPoint === "right") || (whereToPoint === "top")) {
    controlAngle = (Math.PI / 2 + angle) * mulAngle;
    if (controlAngle > Math.PI / 2)
      controlAngle = Math.PI / 2;
    controlAngle = controlAngle - Math.PI / 2;
  }
}
xCPA = xA + Math.cos(controlAngle + constAngle) * CPBDistance;
yCPA = -Math.sin(controlAngle + constAngle) * CPBDistance + yA;
xCPG = xA + Math.cos(controlAngle - constAngle) * CPBDistance;
yCPG = yA - Math.sin(controlAngle - constAngle) * CPBDistance;
var xD = destinationX;
var yD = destinationY;
var svg = document.createElementNS("http://www.w3.org/2000/svg", "svg");
var svgNS = svg.namespaceURI;
svg.setAttribute('height',
                 document.body.getBoundingClientRect().height + "px");
svg.setAttribute('width', document.body.getBoundingClientRect().width + "px");
svg.setAttribute('style', 'position:absolute;top:0;left:0');
svg.setAttribute('class', "highLayer");
svg.id = "svg";
var path = document.createElementNS(svgNS, 'path');
```

```
path.setAttribute('d', 'M ' + xA + ' ' + yA + ' C ' + xCPA + ' ' + yCPA +
                           ', ' + xCPB + ' ' + yCPB + ', ' + xB + ' ' + yB +
                           ' L ' + xC + ' ' + yC + ' L ' + xD + ' ' + yD +
                           ' L ' + xE + ' ' + yE + ' L ' + xF + ' ' + yF +
                           ' C ' + xCPF + ' ' + yCPF + ', ' + xCPG + ' ' +
                           yCPG + ', ' + xG + ' ' + yG + ' Z');
if (tutTable[curr + 1].myArrowColor == null)
  arrCol = "yellow";
else
  arrCol = tutTable[curr + 1].myArrowColor;
path.setAttribute('fill', arrCol);
path.id = "path";
svg.appendChild(path);
/*// debugging points
//<!-- Mark relevant points -->
svg.setAttribute('class', "superHighLayer");
var g = document.createElementNS(svgNS, 'g');
g.setAttribute('stroke', 'black');
g.setAttribute('stoke-width', "4");
g.setAttribute('fill', "black");
svg.appendChild(g);
var pointA = document.createElementNS(svgNS, 'circle');
pointA.setAttribute('cx', xA);
pointA.setAttribute('cy', yA);
pointA.setAttribute('r', "3");
g.appendChild(pointA);
var pointB = document.createElementNS(svgNS, 'circle');
pointB.setAttribute('cx', xB);
pointB.setAttribute('cy', yB);
pointB.setAttribute('r', "3");
g.appendChild(pointB);
var pointC = document.createElementNS(svgNS, 'circle');
pointC.setAttribute('cx', xC);
pointC.setAttribute('cy', yC);
pointC.setAttribute('r', "3");
g.appendChild(pointC);
var pointD = document.createElementNS(svgNS, 'circle');
pointD.setAttribute('cx', xD);
pointD.setAttribute('cy', yD);
pointD.setAttribute('r', "3");
g.appendChild(pointD);
var pointE = document.createElementNS(svgNS, 'circle');
pointE.setAttribute('cx', xE);
pointE.setAttribute('cy', yE);
pointE.setAttribute('r', "3");
g.appendChild(pointE);
```

```
  var pointF = document.createElementNS(svgNS, 'circle');
  pointF.setAttribute('cx', xF);
  pointF.setAttribute('cy', yF);
  pointF.setAttribute('r', "3");
  g.appendChild(pointF);
  var pointG = document.createElementNS(svgNS, 'circle');
  pointG.setAttribute('cx', xG);
  pointG.setAttribute('cy', yG);
  pointG.setAttribute('r', "3");
  g.appendChild(pointG);
  var pointCPB = document.createElementNS(svgNS, 'circle');
  pointCPB.setAttribute('cx', xCPB);
  pointCPB.setAttribute('cy', yCPB);
  pointCPB.setAttribute('r', "3");
  g.appendChild(pointCPB);
  var pointCPF = document.createElementNS(svgNS, 'circle');
  pointCPF.setAttribute('cx', xCPF);
  pointCPF.setAttribute('cy', yCPF);
  pointCPF.setAttribute('r', "3");
  g.appendChild(pointCPF);
  var pointCPA = document.createElementNS(svgNS, 'circle');
  pointCPA.setAttribute('cx', xCPA);
  pointCPA.setAttribute('cy', yCPA);
  pointCPA.setAttribute('r', "3");
  g.appendChild(pointCPA);
  var pointCPG = document.createElementNS(svgNS, 'circle');
  pointCPG.setAttribute('cx', xCPG);
  pointCPG.setAttribute('cy', yCPG);
  pointCPG.setAttribute('r', "3");
  g.appendChild(pointCPG);*/
  document.body.appendChild(svg);
}
```

4.2.10 synchronizeCssStyles

```
function synchronizeCssStyles(src, destination, recursively) {

  // if recursively = true, then we assume the src dom structure and destination
  // dom structure are identical (ie: cloneNode was used)

  // window.getComputedStyle vs document.defaultView.getComputedStyle
  // @TBD: also check for compatibility on IE/Edge
  var cssObj = window.getComputedStyle(src, null);
  var style = ""
  var cssObjProp;
  for (i = 0; i < cssObj.length; i++) {
```

```
    cssObjProp = cssObj.item(i);
    style += cssObjProp + ":" + cssObj.getPropertyValue(cssObjProp) + "; ";
  }
  destination.style.cssText = style;
  // destination.style.cssText = document.defaultView.getComputedStyle(src,
  // "").cssText;

  if (recursively) {
    var vSrcElements = src.getElementsByTagName("*");
    var vDstElements = destination.getElementsByTagName("*");

    for (var idx = vSrcElements.length; idx--;) {
      var vSrcElement = vSrcElements[idx];
      var vDstElement = vDstElements[idx];
      style = "";
      cssObj = window.getComputedStyle(vSrcElement, null);
      for (i = 0; i < cssObj.length; i++) {
        cssObjProp = cssObj.item(i);
        style += cssObjProp + ":" + cssObj.getPropertyValue(cssObjProp) + "; ";
      }
      vDstElement.style.cssText = style;

      // vDstElement.style.cssText =
      // document.defaultView.getComputedStyle(vSrcElement, "").cssText;
    }
  }
}
```

4.2.11 focusItemCallback

```
function focusItemCallback(toolTip, finalLeft, finalTop, scalingFactor,
                           originalWidth, originalHeight) {
  // Called when it's the last iteration
  var animatedItemRect =
      document.getElementById("backgroundDiv").getBoundingClientRect();
  var finalWidth = animatedItemRect.width;
  var finalHeight = animatedItemRect.height;
  // Finding real allignment from left and top
  var finalLeft = animatedItemRect.left + window.pageXOffset;
  var finalTop = animatedItemRect.top + window.pageYOffset;
  // Finding center for toolTip
  if (curr == tutTable.length - 1) {
    // Last step of tutorial
    var x = finalLeft + finalWidth - (toolTipWidth() / 2);
    var y = finalTop - (toolTipHeight() / 2);
  } else {
```

```
    // If node isn't the last element of the tutorial
    holeEl = tutTable[curr + 1].myNode;
    var nextRect = holeEl.getBoundingClientRect();
    var nRl = nextRect.left + window.pageXOffset;
    var nRt = nextRect.left + window.pageYOffset;
    // Classifying relative position using quadrants around center of
    // zoomedEl
    // Quadrant 2 & quadrant 3
    if ((nRl + nextRect.width / 2 <= (finalLeft + finalWidth / 2))) {
      x = finalLeft + (toolTipWidth() / 2);
      y = finalTop + finalHeight + (toolTipHeight() / 2);
    }
    // Quadrant 1 & quadrant 4
    if ((nRl + nextRect.width / 2 > (finalLeft + finalWidth / 2))) {
      x = finalLeft + finalWidth - (toolTipWidth() / 2);
      y = finalTop + finalHeight + (toolTipHeight() / 2);
    }
    var value = toolTipWidth() / 2 + itemFrameRatio * window.innerWidth;

    while (x - value < window.pageXOffset) {
      x++;
    }
  }
  //  (x - toolTipWidth() / 2 - itemFrameRatio * window.offsetWidth));
  createToolTip(x, y);
}
```

4.2.12 createToolTipCallback

```
function createToolTipCallback(toolTip, finalLeft, finalTop, scalingFactor,
                               originalWidth, originalHeight) {
  // Called when it's the last iteration
  var finalWidth = originalWidth * scalingFactor;
  var finalHeight = originalHeight * scalingFactor;
  if (curr != tutTable.length - 1) {
    // If it's not the last step of tutorial
    holeEl = tutTable[curr + 1].myNode;
    drawArrow(toolTip, holeEl, tutTable[curr + 1].fromWhere);
    // Keeping light holeEl
    var holeRect = holeEl.getBoundingClientRect();
    var hRl = holeRect.left + window.pageXOffset;
    var hRt = holeRect.top + window.pageYOffset;
    var hRw = holeRect.width;
    var hRh = holeRect.height;
    var newHole = holeEl.cloneNode(true);
    synchronizeCssStyles(holeEl, newHole, true);
```

```
    newHole.id = "newHole";
    var holeBorderRadius = window.getComputedStyle(holeEl, null)
                                 .getPropertyValue("border-top-left-radius");
    var holeBackground = createBackground(hRl, hRt, hRw, hRh,
                                          tutTable[curr + 1].myBackgroundColor,
                                          parseInt(holeBorderRadius), false);
    holeBackground.appendChild(newHole);
    holeBackground.className = "hole";
    holeBackground.id = "holeBackground";

    // Setting functionality to that part of the window
    var clickDiv = createDiv(hRl, hRt, hRw, hRh, "transparent", 0);
    clickDiv.id = "clickDiv";
    clickDiv.className = "superHighLayer";
    clickDiv.onclick = function() {
      closeStep();
      curr++;
      contItem = 1;
      contToolTip = 1;
      tutStep();
    };
  }
}
```

4.2.13 scrollIntoViewIfNeeded

```
function scrollIntoViewIfNeeded(el) {
  var myRect = el.getBoundingClientRect();
  if ((myRect.top < 0) || (myRect.left < 0) ||
      (myRect.top + myRect.height > window.innerHeight) ||
      (myRect.left + myRect.width > window.innerWidth))
    el.scrollIntoView();
}
```

4.2.14 LightenDarkenColor

```
function LightenDarkenColor(col, amt) {
  var usePound = false;
  if (col[0] == "#") {
    col = col.slice(1);
    usePound = true;
  }
  var num = parseInt(col, 16);
  var r = (num >> 16) + amt;
  if (r > 255)
    r = 255;
  else if (r < 0)
```

```
    r = 0;
  var b = ((num >> 8) & 0x00FF) + amt;
  if (b > 255)
    b = 255;
  else if (b < 0)
    b = 0;
  var g = (num & 0x0000FF) + amt;
  if (g > 255)
    g = 255;
  else if (g < 0)
    g = 0;
  return (usePound ? "#" : "") + (g | (b << 8) | (r << 16)).toString(16);
}
```

4.2.15 rgb2hex

```
function rgb2hex(rgb) {
  if (/^#[0-9A-F]{6}$/i.test(rgb))
    return rgb;

  rgb = rgb.match(/^rgb\((\d+),\s*(\d+),\s*(\d+)\)$/);
  function hex(x) { return ("0" + parseInt(x).toString(16)).slice(-2); }
  return "#" + hex(rgb[1]) + hex(rgb[2]) + hex(rgb[3]);
}
```

4.2.16 drawTrapezoid

```
function drawTrapezoid(fromEl, toEl) {
  var previousTrapezoid = document.getElementById("trapezoids");
  if (previousTrapezoid != null)
    previousTrapezoid.remove();
  var xA, yA, xB, yB, xC, yC, xD, yD, xE, yE, xF, yF;
  var fromRect = fromEl.getBoundingClientRect();
  var toRect = toEl.getBoundingClientRect();
  yA = fromRect.top + window.pageYOffset;
  xA = fromRect.left + window.pageXOffset;
  xD = xA + fromRect.width;
  yD = yA;
  xB = toRect.left + window.pageXOffset;
  yB = toRect.top + window.pageYOffset;
  xC = xB + toRect.width;
  yC = yB;
  xF = xC;
  yF = yC + toRect.height;
  var xH = xB;
  var yH = yF;
  xE = xD;
```

```
yE = yD + fromRect.height;
var xG = xA;
var yG = yE;
var radiusTo = parseInt(window.getComputedStyle(toEl, null)
                              .getPropertyValue("border-top-left-radius"));
if (radiusTo != null) {
  var scalingF =
      window.getComputedStyle(toEl, null).getPropertyValue("transform");
  var delta = (1 - Math.sqrt(2) / 2) * radiusTo * toEl.currScaling;
  xB = xB + delta;
  yB = yB + delta;
  xC = xB + toRect.width - 2 * delta;
  yC = yB;
  xF = xC;
  yF = yC + toRect.height - 2 * delta;
  xH = xB;
  yH = yF;
}
var radiusFrom = parseInt(window.getComputedStyle(fromEl, null)
                                .getPropertyValue("border-top-left-radius"));
if (radiusFrom != null) {
  var delta2 = (1 - Math.sqrt(2) / 2) * radiusFrom;
  xA = xA + delta2;
  yA = yA + delta2;
  xD = xA + fromRect.width - 2 * delta2;
  yD = yA;
  xE = xD;
  yE = yD + fromRect.height - 2 * delta2;
  xG = xA;
  yG = yE;
}
var svg = document.createElementNS("http://www.w3.org/2000/svg", "svg");
var svgNS = svg.namespaceURI;
svg.setAttribute('height',
                 document.body.getBoundingClientRect().height + "px");
svg.setAttribute('width', document.body.getBoundingClientRect().width + "px");
svg.setAttribute('style', 'position:absolute;top:0;left:0');
svg.setAttribute('class', "middleLayer");
svg.setAttribute('opacity', '0.5');
svg.id = "trapezoids";
var path1 = document.createElementNS(svgNS, 'path');
var path2 = document.createElementNS(svgNS, 'path');
var fromX, fromY, toX, toY;
fromX = fromRect.left + window.pageXOffset + fromRect.width / 2;
fromY = fromRect.top + window.pageYOffset + fromRect.height / 2;
toX = toRect.left + window.pageXOffset + toRect.width / 2;
```

```
toY = toRect.top + window.pageYOffset + toRect.height / 2;
// Q1
if (toX >= fromX && toY <= fromY) {
  path1.setAttribute('d', 'M ' + xA + ' ' + yA + ' L ' + xB + ' ' + yB +
                          ' L ' + xH + ' ' + yH + ' L ' + xG + ' ' + yG +
                          'Z');
  path2.setAttribute('d', 'M ' + xG + ' ' + yG + ' L ' + xH + ' ' + yH +
                          ' L ' + xF + ' ' + yF + ' L ' + xE + ' ' + yE +
                          'Z');
}
// Q2
if (toX < fromX && toY < fromY) {
  path1.setAttribute('d', 'M ' + xG + ' ' + yG + ' L ' + xH + ' ' + yH +
                          ' L ' + xF + ' ' + yF + ' L ' + xE + ' ' + yE +
                          'Z');
  path2.setAttribute('d', 'M ' + xE + ' ' + yE + ' L ' + xF + ' ' + yF +
                          ' L ' + xC + ' ' + yC + ' L ' + xD + ' ' + yD +
                          'Z');
}
// Q3
if (toX <= fromX && toY >= fromY) {
  path1.setAttribute('d', 'M ' + xB + ' ' + yB + ' L ' + xA + ' ' + yA +
                          ' L ' + xD + ' ' + yD + ' L ' + xC + ' ' + yC +
                          'Z');
  path2.setAttribute('d', 'M ' + xC + ' ' + yC + ' L ' + xD + ' ' + yD +
                          ' L ' + xE + ' ' + yE + ' L ' + xF + ' ' + yF +
                          'Z');
}

// Q4
if (toX > fromX && toY > fromY) {
  path1.setAttribute('d', 'M ' + xG + ' ' + yG + ' L ' + xH + ' ' + yH +
                          ' L ' + xB + ' ' + yB + ' L ' + xA + ' ' + yA +
                          'Z');
  path2.setAttribute('d', 'M ' + xA + ' ' + yA + ' L ' + xB + ' ' + yB +
                          ' L ' + xC + ' ' + yC + ' L ' + xD + ' ' + yD +
                          'Z');
}
var color, colorWeaker1, colorWeaker2;
var style = (window.getComputedStyle(fromEl, null)
                   .getPropertyValue("background-color"));
if (style != 'rgba(0, 0, 0, 0)') {
  color = rgb2hex(style);
} else
  color = tutTable[curr].myBackgroundColor;
colorWeaker1 = LightenDarkenColor(color, 40);
```

```
colorWeaker2 = LightenDarkenColor(color, 60);
if (color === "#ffffff")
  color = "#f2f2f2";
path1.setAttribute('fill', colorWeaker1);
path1.setAttribute('stroke', color);
path1.setAttribute('stroke-width', "2");
path1.id = "path1";
svg.appendChild(path1);
path2.setAttribute('fill', colorWeaker2);
path2.setAttribute('stroke', color);
path2.setAttribute('stroke-width', "1");
path2.id = "path2";
svg.appendChild(path2);
/* // debugging points
 //<!-- Mark relevant points -->
 svg.setAttribute('class', "superHighLayer");
 var g = document.createElementNS(svgNS, 'g');
 g.setAttribute('stroke', 'black');
 g.setAttribute('stoke-width', "4");
 g.setAttribute('fill', "black");
 svg.appendChild(g);
 var pointA = document.createElementNS(svgNS, 'circle');
 pointA.setAttribute('cx', xA);
 pointA.setAttribute('cy', yA);
 pointA.setAttribute('r', "3");
 g.appendChild(pointA);
 var pointB = document.createElementNS(svgNS, 'circle');
 pointB.setAttribute('cx', xB);
 pointB.setAttribute('cy', yB);
 pointB.setAttribute('r', "3");
 g.appendChild(pointB);
 var pointC = document.createElementNS(svgNS, 'circle');
 pointC.setAttribute('cx', xC);
 pointC.setAttribute('cy', yC);
 pointC.setAttribute('r', "3");
 g.appendChild(pointC);
 var pointD = document.createElementNS(svgNS, 'circle');
 pointD.setAttribute('cx', xD);
 pointD.setAttribute('cy', yD);
 pointD.setAttribute('r', "3");
 g.appendChild(pointD);
 var pointE = document.createElementNS(svgNS, 'circle');
 pointE.setAttribute('cx', xE);
 pointE.setAttribute('cy', yE);
 pointE.setAttribute('r', "3");
 g.appendChild(pointE);
```

```
    var pointF = document.createElementNS(svgNS, 'circle');
    pointF.setAttribute('cx', xF);
    pointF.setAttribute('cy', yF);
    pointF.setAttribute('r', "3");
    g.appendChild(pointF);
    var pointG = document.createElementNS(svgNS, 'circle');
    pointG.setAttribute('cx', xG);
    pointG.setAttribute('cy', yG);
    pointG.setAttribute('r', "3");
    g.appendChild(pointG);
    var pointH = document.createElementNS(svgNS, 'circle');
    pointH.setAttribute('cx', xH);
    pointH.setAttribute('cy', yH);
    pointH.setAttribute('r', "3");
    g.appendChild(pointH);
   */
  document.body.appendChild(svg);
}
```

4.2.17 animatedZoom

```
function animatedZoom(currCallback, animatingEl, finalLeft, finalTop,
                      scalingFactor, frameRate, numberOfTransformations,
                      shrinkingFactor) {
  var node = animatingEl;
  rect = node.getBoundingClientRect();
  var originalTop = rect.top + window.pageYOffset;
  var originalLeft = rect.left + window.pageXOffset;
  var originalWidth = rect.width;
  var originalHeight = rect.height;
  var scaling =
      (Math.pow(scalingFactor * shrinkingFactor, 1 / numberOfTransformations));
  var translatingX = (finalLeft - originalLeft) / numberOfTransformations;
  var translatingY = (finalTop - originalTop) / numberOfTransformations;
  id = setInterval(frame, frameRate);
  function frame() {
    if (shrinkingFactor == 1) {
      nowScaling = Math.pow(scaling, contItem) / shrinkingFactor;
      nowTranslatingX = translatingX * contItem;
      nowTranslatingY = translatingY * contItem;

      scaleAndTranslate(node, nowScaling, nowTranslatingX, nowTranslatingY);
      scrollIntoViewIfNeeded(node);
      if ((tutTable[curr].myNode != null) &&
          ((contItem % 2 == 0) || (contItem == numberOfTransformations))) {
        document.getElementById("backgroundDiv").currScaling =
```

```
            Math.pow(scaling, contItem);
        drawTrapezoid(document.getElementById("currHoleBackground"),
                      document.getElementById("backgroundDiv"));
      }
      if (contItem == numberOfTransformations) {
        if (shrinkingFactor == 1)
          currZoomedItem = animatingEl;
        clearInterval(id);
        currCallback(animatingEl, finalLeft, finalTop, scalingFactor,
                     originalWidth, originalHeight);
      } else
        contItem++;
    } else {
      nowScaling = Math.pow(scaling, contToolTip) / shrinkingFactor;
      nowTranslatingX = translatingX * contToolTip;
      nowTranslatingY = translatingY * contToolTip;
      scaleAndTranslate(node, nowScaling, nowTranslatingX, nowTranslatingY);
      if (contToolTip == numberOfTransformations) {
        if (shrinkingFactor == 1)
          currZoomedItem = animatingEl;
        clearInterval(id);
        currCallback(animatingEl, finalLeft, finalTop, scalingFactor,
                     originalWidth, originalHeight);
      } else
        contToolTip++;
    }
  }
  frame();
}
```

4.2.18 focusItem

```
function focusItem(el) {
  // Scrolling to center vertically
  scrollIntoViewIfNeeded(el);
  var copyEl = el.cloneNode(true);
  synchronizeCssStyles(el, copyEl, true);
  copyEl.id = "copyEl";
  var rect = el.getBoundingClientRect();
  // leftEl is the position of the element and his copy from left
  var leftEl = rect.left + window.pageXOffset;
  // topEl is the position of the element and his copy from top
  var topEl = rect.top + window.pageYOffset;
  var widthEl = el.offsetWidth;
  var heightEl = el.offsetHeight;
  // Keeping the colour of element inserting background div
```

```
var backColor = tutTable[curr].myBackgroundColor;
var backgroundDiv = createBackground(leftEl, topEl, widthEl, heightEl,
                                     backColor, 2 * backgroundFrame, true);
backgroundDiv.id = "backgroundDiv";
backgroundDiv.className = "veryHighLayer";
var lighterColor = LightenDarkenColor(backColor, 30);
dontClickDiv = createBackground(backgroundFrame, backgroundFrame, widthEl,
                                heightEl, "white", 0, false);
dontClickDiv.setAttribute('style',
                          dontClickDiv.getAttribute('style') + " opacity:0");
dontClickDiv.className = "superHighLayer";
backgroundDiv.appendChild(copyEl);
backgroundDiv.appendChild(dontClickDiv);
backgroundDiv.style.padding = backgroundFrame + "px";
var divRect = backgroundDiv.getBoundingClientRect();
var x = window.innerWidth / 2 + window.pageXOffset;
var y = window.innerHeight / 2 + window.pageYOffset;
//(x,y) is the center of the window
if (curr != tutTable.length - 1) {
  // In order not to cover the next element of the tutorial there are many
  // cases to take into consideration
  var nextEl = tutTable[curr + 1].myNode;
  var nextElRect = nextEl.getBoundingClientRect();
  var finalTop, finalLeft, ratioX, ratioY;
  if (((nextElRect.left + nextElRect.width / 2) <
       window.pageXOffset + window.innerWidth) &&
      ((nextElRect.top + nextElRect.height / 2) <
       window.pageYOffset + window.innerHeight)) {
    // NextEl is inside current window -> 4 cases (relatively to currentEl the
    // one whose attributes are in divRect) The element has a frame around
    // itself
    ratioX = ((1 - itemFrameRatio) * window.innerWidth / 2) / (divRect.width);
    ratioY =
        ((1 - itemFrameRatio) * window.innerHeight / 2) / (divRect.height);
    if (ratioX <= ratioY)
      mulFactor = ratioX;
    else
      mulFactor = ratioY;
    if (mulFactor > maxMulFactor)
      mulFactor = maxMulFactor;
    var currentX = divRect.left + divRect.width / 2;
    var currentY = divRect.top + divRect.height / 2;
    // Quadrant 1 -> go bottom-left
    if (((nextElRect.left + nextElRect.width / 2) >= currentX) &&
        ((nextElRect.top + nextElRect.height / 2) <= currentY)) {
      finalLeft = window.pageXOffset + itemFrameRatio * window.innerWidth +
```

```
                divRect.width / 2 * mulFactor - divRect.width / 2;
    finalTop = (1 - itemFrameRatio) * window.innerHeight +
               window.pageYOffset - toolTipHeight() -
               (divRect.height * mulFactor / 2) - divRect.height / 2;
  }
  // Quadrant 2 -> go bottom-right
  if ((nextElRect.left + nextElRect.width / 2 < currentX) &&
      ((nextElRect.top + nextElRect.height / 2) < currentY)) {
    finalLeft = (1 - itemFrameRatio) * window.innerWidth +
                window.pageXOffset - (divRect.width * mulFactor / 2) -
                divRect.width / 2;
    finalTop = (1 - itemFrameRatio) * window.innerHeight +
               window.pageYOffset - toolTipHeight() -
               (divRect.height * mulFactor / 2) - divRect.height / 2;
  }
  // Quadrant 3 -> go top-right
  if ((nextElRect.left + nextElRect.width / 2 <= currentX) &&
      ((nextElRect.top + nextElRect.height / 2) >= currentY)) {
    finalLeft = (1 - itemFrameRatio) * window.innerWidth +
                window.pageXOffset - (divRect.width * mulFactor / 2) -
                divRect.width / 2;
    finalTop = window.pageYOffset +
               itemFrameRatio * window.innerHeight / 2 +
               divRect.height * mulFactor / 2 - divRect.height / 2;
  } // Quadrant 4 -> go top-left
  if ((nextElRect.left + nextElRect.width / 2 > currentX) &&
      ((nextElRect.top + nextElRect.height / 2) > currentY)) {
    // Since scaleAndTranslate translates first it's necessary to find out
    // the left allingment that doesn't become negative when the element is
    // scaled
    finalLeft = window.pageXOffset + itemFrameRatio * window.innerWidth +
                divRect.width / 2 * mulFactor - divRect.width / 2;
    finalTop = window.pageYOffset +
               itemFrameRatio * window.innerHeight / 2 +
               divRect.height * mulFactor / 2 - divRect.height / 2;
  }
}
// NextEl is outside from current window
else {
  // The element has a frame around itself
  ratioX = ((1 - itemFrameRatio) * window.innerWidth) / (divRect.width);
  ratioY = ((1 - itemFrameRatio) * window.innerHeight) / (divRect.height);
  if (ratioX <= ratioY)
    mulFactor = ratioX;
  else
    mulFactor = ratioY;
```

```
    if (mulFactor > maxMulFactor)
      mulFactor = maxMulFactor;
    finalTop = y - (divRect.height / 2);
    finalLeft = x - divRect.width / 2;
    if ((finalTop + divRect.height / 2 + (divRect.height / 2) * mulFactor +
         toolTipHeight()) >
        (window.innerHeight * (1 - itemFrameRatio) + window.pageYOffset))
      // If there's not enough room for the tool tip
      finalTop = window.innerHeight * (1 - itemFrameRatio) +
                 window.pageYOffset - toolTipHeight() -
                 (divRect.height * maxMulFactor / 2) - divRect.height / 2;
  }
} else {
  // The element is the last element
  // The element has a frame around itself
  ratioX = ((1 - itemFrameRatio) * window.innerWidth) / (divRect.width);
  ratioY = ((1 - itemFrameRatio) * window.innerHeight) / (divRect.height);
  if (ratioX <= ratioY)
    mulFactor = ratioX;
  else
    mulFactor = ratioY;
  if (mulFactor > maxMulFactor)
    mulFactor = maxMulFactor;
  finalTop = y - (divRect.height / 2);
  finalLeft = x - divRect.width / 2;
  if (finalTop + divRect.height / 2 - (divRect.height / 2) * mulFactor -
          toolTipHeight() <
      window.pageYOffset + itemFrameRatio * window.innerHeight)
    // If there's not enough room for the tool tip (which is on top this time)
    finalTop = window.pageYOffset + itemFrameRatio * window.innerHeight +
               toolTipHeight() + (divRect.height * maxMulFactor / 2) -
               divRect.height / 2;
}
// Keeping visible the current element of the tutorial, if it exists
if (curr != 0) {
  var currHoleRect = tutTable[curr].myNode.getBoundingClientRect();
  var newCurrHole = tutTable[curr].myNode.cloneNode(true);
  synchronizeCssStyles(tutTable[curr].myNode, newCurrHole, true);
  newCurrHole.id = "newCurrHole";
  var holeBorderRadius = window.getComputedStyle(tutTable[curr].myNode, null)
                              .getPropertyValue("border-top-left-radius");
  var currHoleBackground = createBackground(
      currHoleRect.left + window.pageXOffset,
      currHoleRect.top + window.pageYOffset, currHoleRect.width,
      currHoleRect.height, tutTable[curr].myBackgroundColor,
      parseInt(holeBorderRadius), false);
```

```
    currHoleBackground.appendChild(newCurrHole);
    currHoleBackground.className = "hole";
    currHoleBackground.id = "currHoleBackground";
    currHoleBackground.className = "bottomLayer";
  }

  animatedZoom(focusItemCallback, backgroundDiv, finalLeft, finalTop, mulFactor,
               intervalOfTimer, numberOfIterations, 1);
}
```

4.2.19 createToolTip

```
function createToolTip(x, y) {
  // Creating new div in (x,y)
  var color2 = tutTable[curr].myBackgroundColor2;
  var text = tutTable[curr].myDescription;
  var backgroundDiv =
      createDiv(x - toolTipWidth() / 2, y - toolTipHeight() / 2, toolTipWidth(),
                toolTipHeight(), color2, toolTipWidth());
  backgroundDiv.className = "tTip";
  var textDiv = document.createElement("div");
  textDiv.id = "toolTipTextDiv";
  backgroundDiv.id = "toolTipDiv";
  backgroundDiv.className = "superHighLayer";
  textDiv.innerHTML += text;
  textDiv.className += " textTip";
  var closeTip = document.createElement("a");
  closeTip.className = "closeTip";
  closeTip.innerHTML += '×';
  closeTip.id = "closeTip";
  closeTip.onclick = function() {
    closeStep();
    callback();
  };
  backgroundDiv.appendChild(textDiv);
  backgroundDiv.appendChild(closeTip);
  textDiv.style.fontSize = "30px";
  closeTip.style.fontSize = "35px";
  closeTip.style.position = "fixed";
  closeTip.style.left =
      5 / 8 * backgroundDiv.getBoundingClientRect().width + "px";
  closeTip.style.top =
      1 / 15 * backgroundDiv.getBoundingClientRect().height + "px";
  while (textDiv.getBoundingClientRect().height +
             closeTip.getBoundingClientRect().height >
         backgroundDiv.getBoundingClientRect().height) {
```

```
    textDiv.style.fontSize = parseInt(textDiv.style.fontSize) - 1 + "px";
    closeTip.style.fontSize = parseInt(closeTip.style.fontSize) - 1 + "px";
  }
  // The div is shrank in order to animate scaling later
  var shrinkingFactor = 50;
  // The backgroundDiv should be positioned centered in (x,y)
  var finalLeft = x - (toolTipWidth() / 2);
  var finalTop = y - (toolTipHeight() / 2);
  // Begin animation that will zoom backgroundDiv
  animatedZoom(createToolTipCallback, backgroundDiv, finalLeft, finalTop, 1,
               intervalOfTimer, numberOfIterations, shrinkingFactor);
}
```

4.2.20 closeStep

```
function closeStep() {
  clearInterval(id);
  for (var i = 0; i < ids.length; i++) {
    var el = document.getElementById(ids[i]);
    if (el)
      el.remove();
  }
}
```

4.2.21 tutStep

```
function tutStep() {
  // Shading the page
  shadePage();
  var el = tutTable[curr].myNode;
  var x, y;
  if (curr == 0) {
    // First step of tutorial -> only show initial tool tip
    x = window.innerWidth / 2 + window.pageXOffset;
    y = window.innerHeight / 2 + window.pageYOffset;
    //(x,y) is the center of the window
    createToolTip(x, y);
  } else {
    focusItem(el);
  }
  // The other elements are created inside animatedZoom
}
```

4.2.22 tutGem

```
function tutGem(tutorialTable, doneCallback) {
  window.onresize = function(event) {
```

```
    closeStep();
    tutStep();
  };
  tutTable = tutorialTable;
  callback = doneCallback;
  curr = 0;
  contItem = 1;
  contToolTip = 1;
  tutStep();
}
```

5 Conclusion

As explained before, the library works on several web pages and may be used on most modern web browser. Its aim is not to overtake the other libraries, but to offer a more animated option to those who are looking for better aestethic performances.