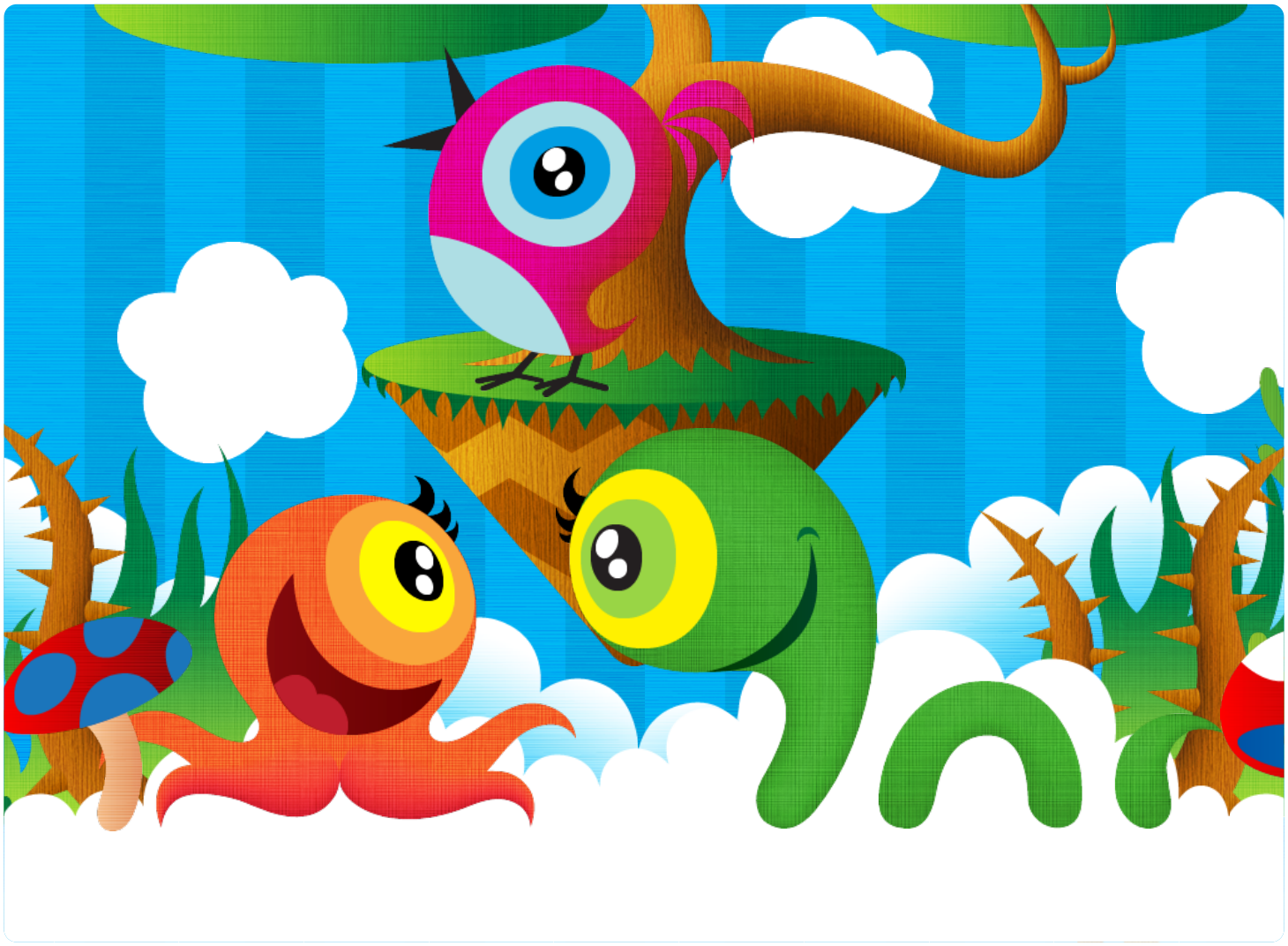
TUTORIAL

This tutorial is based on Robby Leonardi's design portfolio website and it will teach you the javascript animation trick behind that website. However the code will be simplified to make it easier to learn. I hope you will enjoy this tutorial and I hope it could be useful. Happy learning guys :)

1. Parallax Effect on Mouse Move



First, all of images and their data will be stored in an array called **objectArray**.

```
var objectArray = new Array();

fillObjectArray();

function fillObjectArray()
{
     var birdDiv = document.getElementById("bird");
     var birdX = 312;
     var birdY = 33;
     var birdFactor = 0.05;
```

```
//object array after fillObjectArray() function is executed

objectArray[0][0] = birdDiv;
objectArray[0][1] = 312;
objectArray[0][2] = 33;
objectArray[0][3] = 0.05;

objectArray[1][0] = bush1Div;
objectArray[1][1] = -28;
objectArray[1][2] = 352;
```

```
        var birdArray = new Array();
        birdArray.push(birdDiv, birdX, birdY, birdFactor);
        objectArray.push(birdArray);

        var bush1Div = document.getElementById("bush1");
        var bush1X = -28;
        var bush1Y = 352;
        var bush1Factor = 0.06;
        var bush1Array = new Array();
        bush1Array.push(bush1Div, bush1X, bush1Y,
        bush1Factor);
        objectArray.push(bush1Array);

        ...
}
```

```
objectArray[1][3] = 0.06;

...
```

Next code is the engine of this parallax effect. **tempX** is the mouse x position. **windowWidth** is an inner width of user's browser window. **objectArray[i][3]** is the parallax factor number. The bigger this number, the more parallax effect on mouse movement. Foreground objects such as snake and squid have bigger **objectArray[i][3]** values compare to background objects such as flying island and bird. **objectArray[i][1]** is the image's original x position before it is shifted. On mouse movement, **objectArray[i][1]** value will be reduced or added by **objectArray[i][3] * (0.5 * windowWidth - tempX)**. This **objectArray[i][3] * (0.5 * windowWidth - tempX)** means the parallax factor number is multiplied by the x mouse distance from browser window's center. Then this value will be stored in **yourDivPositionX**, and **yourDivPositionX** will be used to update the images' x position. Since this function is applied to every images inside **objectArray**, all of those images will be shifted every time user moves the mouse.

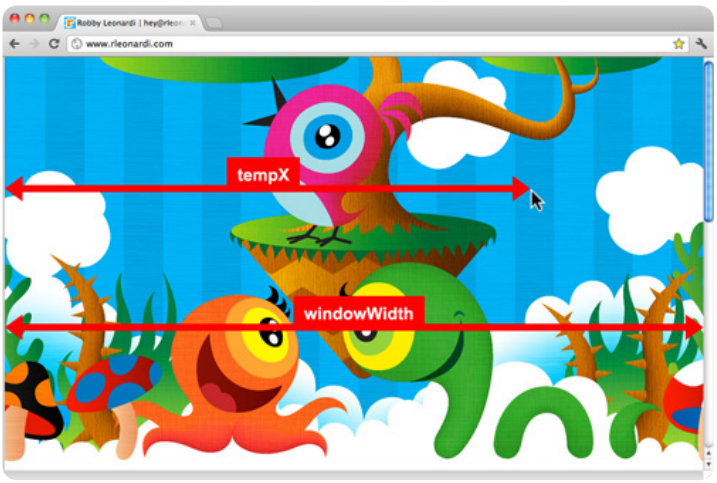


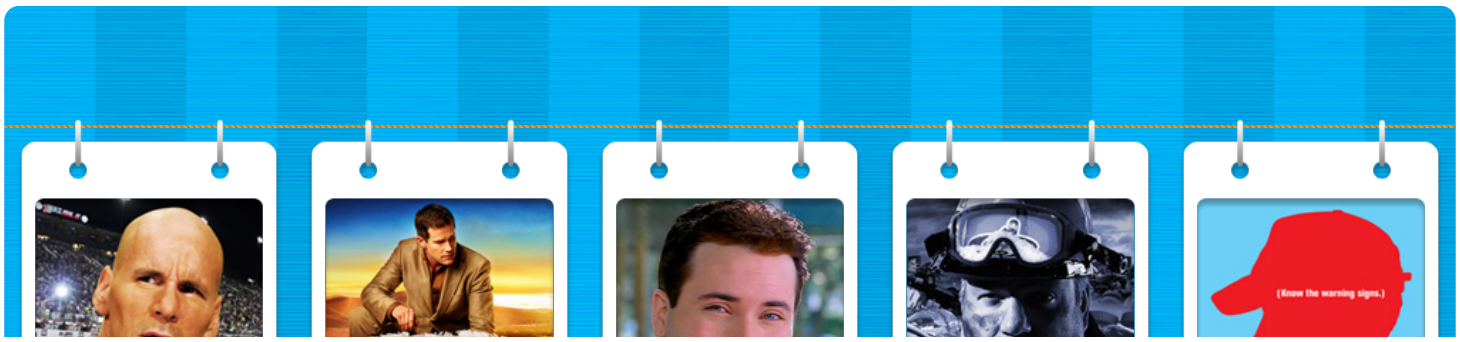
image 1.1.

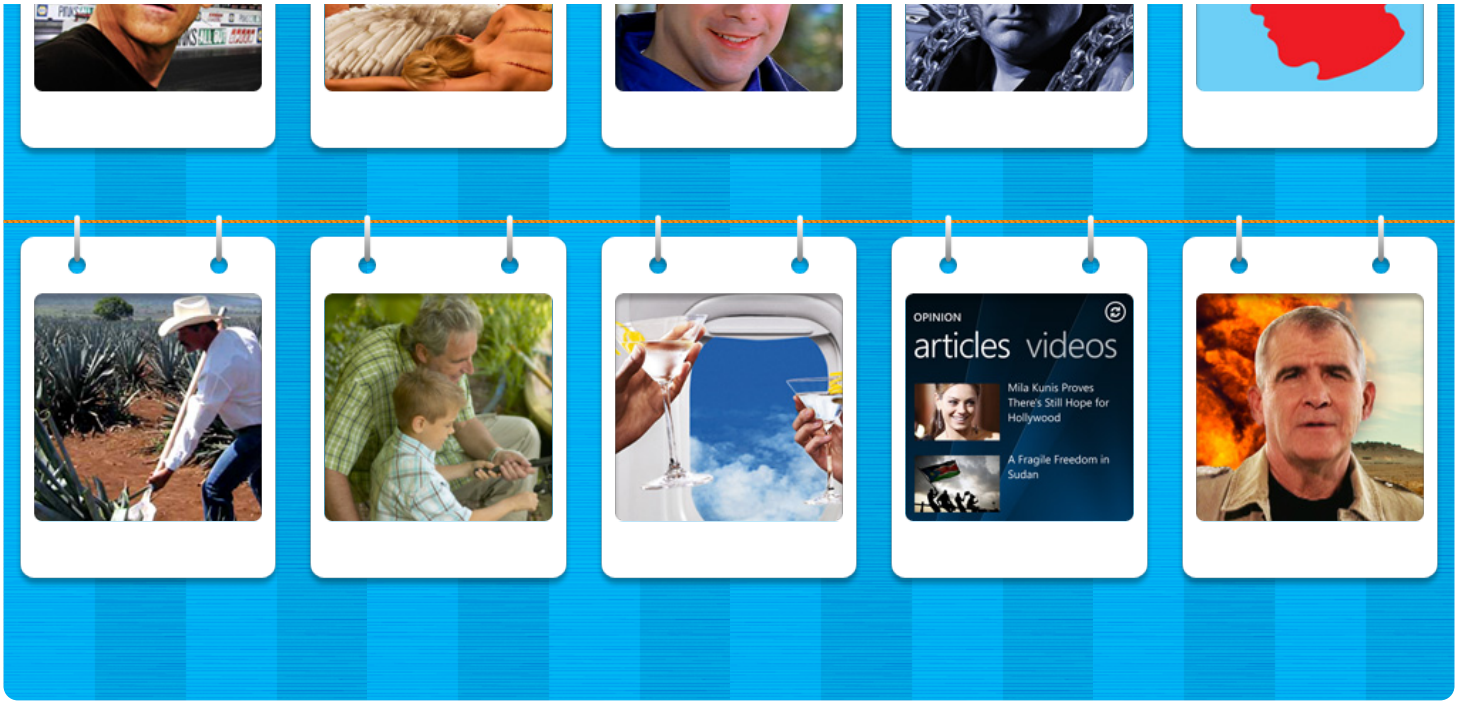
```
function moveDiv(tempX)
{
      for (var i=0;i<objectArray.length;i++)
      {
            var yourDivPositionX = objectArray[i][3] * (0.5
            * windowWidth - tempX) + objectArray[i][1];
            objectArray[i][0].style.left = yourDivPositionX
            + "px";
      }
}
```

View complete:

Javascript CSS

2. Thumbnail Shifting Animation





This code shifts the thumbnail group:

```
thumbnailPaperContainerArray[i].style.left = thumbnailPaperContainerLeftEarlyPositionX + thumbnailPaperContainerShiftAmount + "px";
```

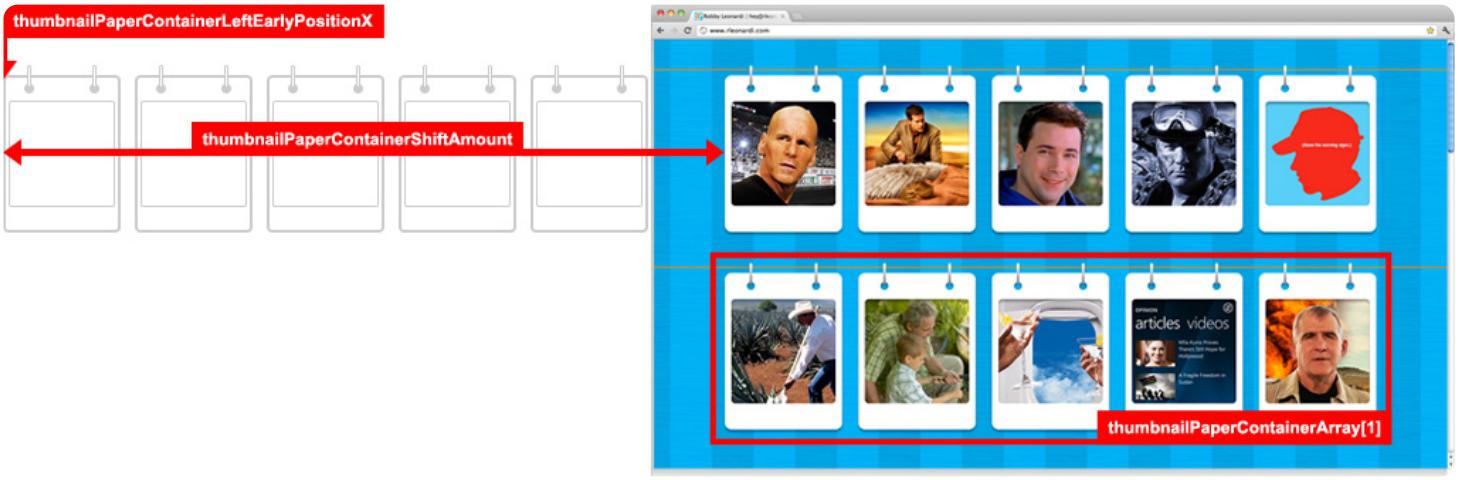
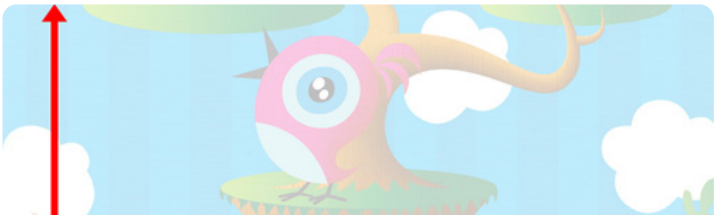


image 2.1.

thumbnailPaperContainerLeftEarlyPositionX is the thumbnail group's x position when it is still hidden. When the mouse is scrolled down, the webpage's y position is calculated and transferred to **thumbnailPaperContainerShiftAmount**, and this makes **thumbnailPaperContainerArray[i]** shifts horizontally. The higher the vertical position of **thumbnailPaperContainerArray[i]**, the bigger the value of **thumbnailPaperContainerShiftAmount**.

The equation of **thumbnailPaperContainerShiftAmount**:



```
var thumbnailPaperContainerShiftAmount = (verticalPosition + windowHeight - contentThumbnailPaperDiv.offsetTop - thumbnailPaperContainerArray[i].offsetTop) * thumbnailPaperContainerShiftSpeed;
```

Again, this code just calculates and transfers the webpage's vertical

image 2.2.

position to **thumbnailPaperContainerShiftAmount**. **verticalPosition** is the webpage's y position. **windowHeight** is an inner height of the browser window. **contentThumbnailPaperDiv.offsetTop** is the y position of thumbnail group container. **thumbnailPaperContainerArray[i].offsetTop** is the y position of each thumbnail group. And finally **thumbnailPaperContainerShiftSpeed** is a factor number. The bigger this number, the bigger is **thumbnailPaperContainerShiftAmount**. In this example, **thumbnailPaperContainerShiftSpeed** is set to 2.

When **thumbnailPaperContainerArray[i]** is in the middle of browser window, **thumbnailPaperContainerArray[i]** needs to be locked so it will not keep shifting horizontally.

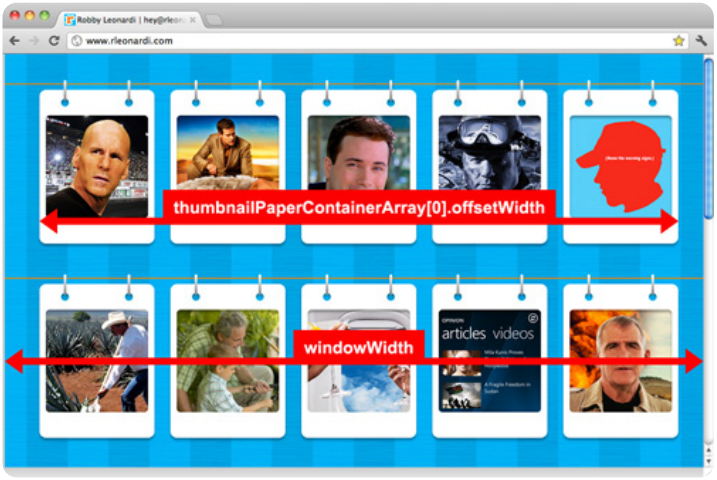


image 2.3.

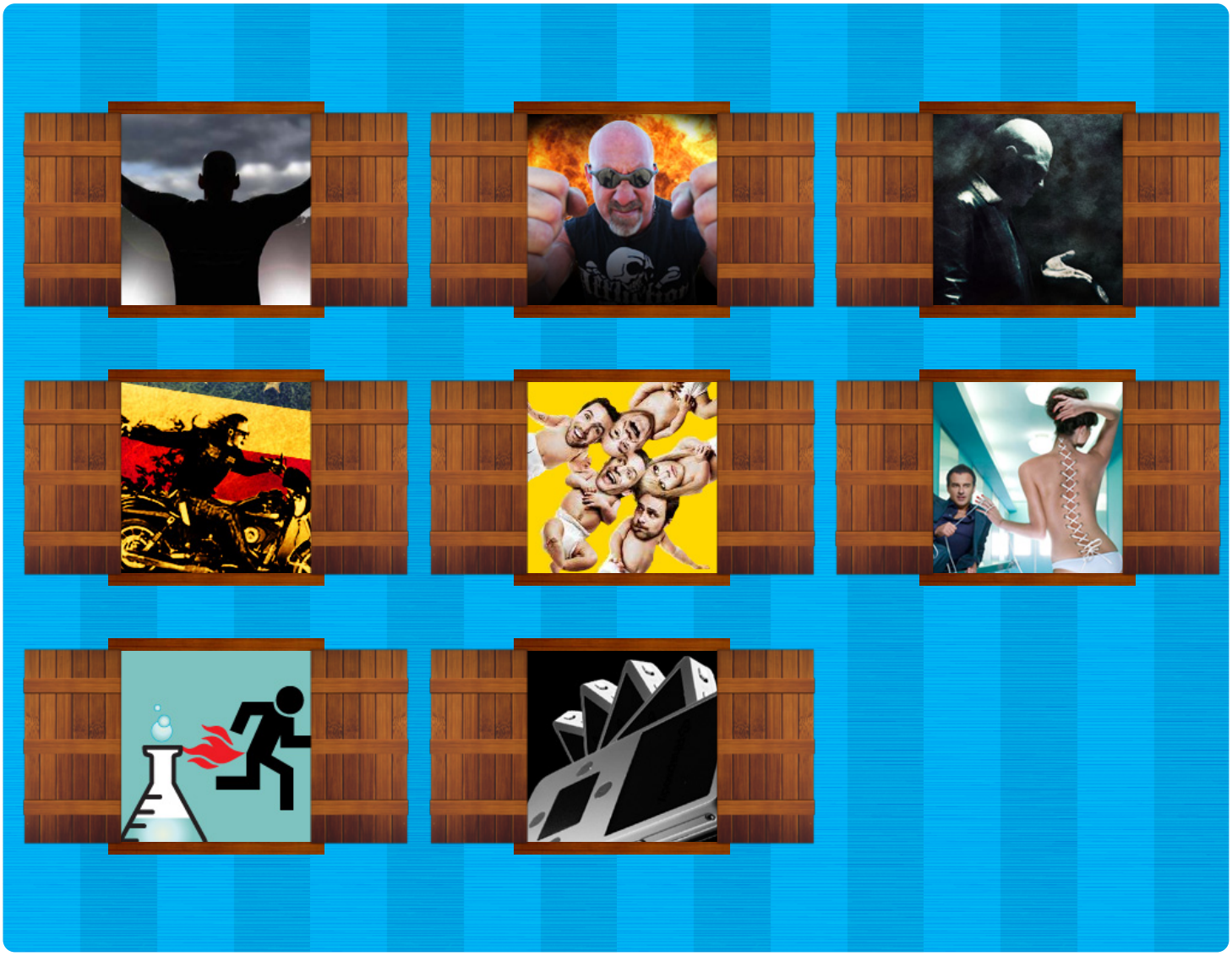
```
if (thumbnailPaperContainerLeftEarlyPositionX +
thumbnailPaperContainerShiftAmount > 0.5 * (windowWidth -
thumbnailPaperContainerArray[0].offsetWidth))
{
     thumbnailPaperContainerArray[i].style.left = 0.5 *
     (windowWidth -
     thumbnailPaperContainerArray[i].offsetWidth) + "px";
}
```

thumbnailPaperContainerLeftEarlyPositionX + thumbnailPaperContainerShiftAmount is x position of **thumbnailPaperContainerArray[i]** when the user scrolls the mouse. **0.5 * (windowWidth - thumbnailPaperContainerArray[0].offsetWidth)** is x position when **thumbnailPaperContainerArray[i]** exactly at the middle of user's browser window. **windowWidth** is an inner width of the browser window, and **thumbnailPaperContainerArray[0].offsetWidth** is the first thumbnail group width. This code basically says if any thumbnail group moves beyond the middle of browser window, put it back to the middle again.

View complete:

Javascript CSS

3. Sequential Window Opening Animation



The opening window animation is created by using this sequential window image. It is one long image that is shifted horizontally to create an opening window illusion. There are 7 opening window slides, and each of them has the same width (**slide_width**). So when it is time to shift to next window slide, the horizontal position of this sequential window image will be added or reduced by **slide_width**. Time to shift the next window slide is triggered by scrolling the mouse.

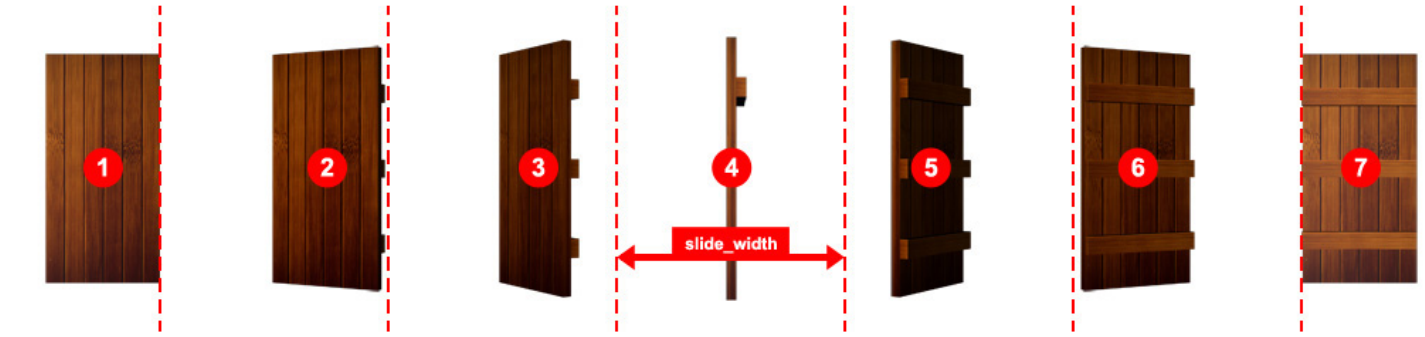


image 3.1.

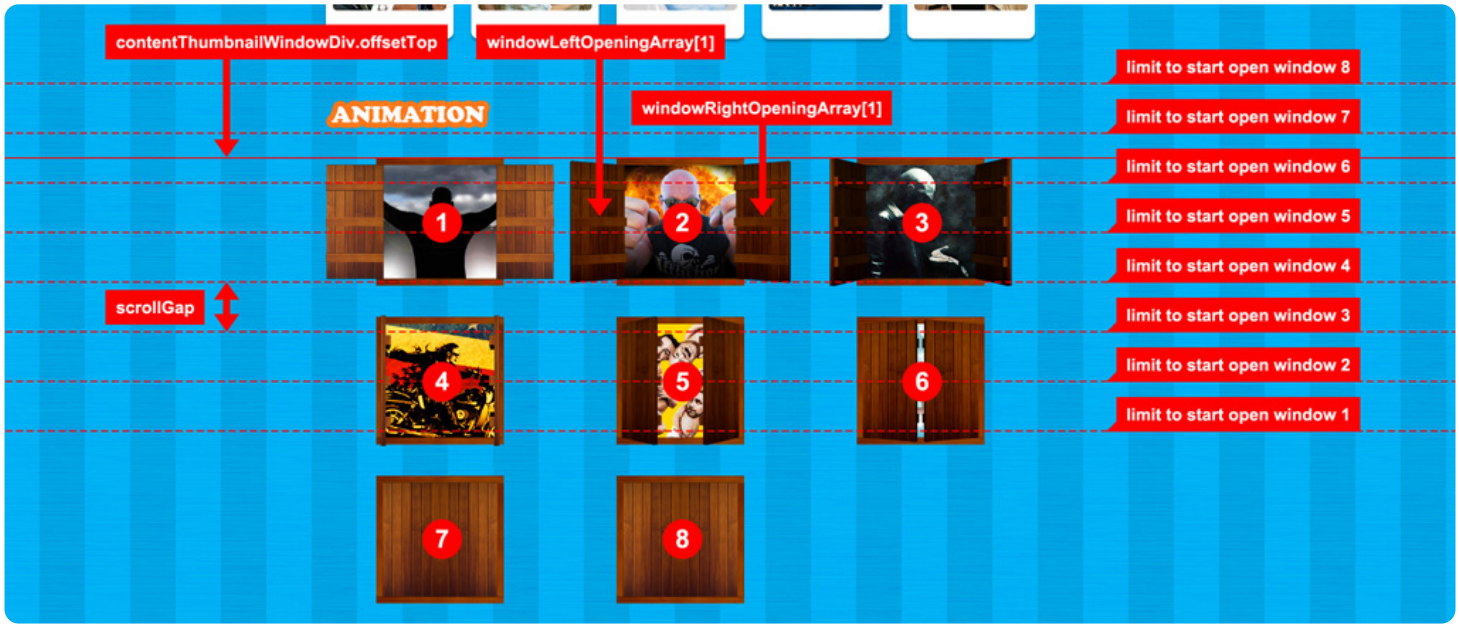


image 3.2.

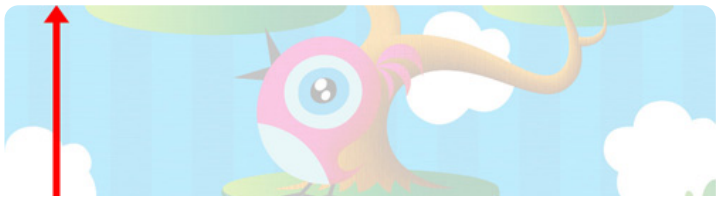
Image 3.2. will explain the concept of this window opening animation. First, pay attention at **contentThumbnailWindowDiv.offsetTop**. Its y position is the same with the first window thumbnail, and it is drawn with continuous red line. When the user scroll the page, this line will move to different areas marked between dotted lines.

This **contentThumbnailWindowDiv.offsetTop** line is above "limit to start open window 6". It means the sixth window thumbnail will shift its sequential window image (image 3.1.) to slide number 2, which shows only small window opening.

For the first window thumbnail, pay attention at **contentThumbnailWindowDiv.offsetTop** and "limit to start open window 1". There are 6 dotted lines below **contentThumbnailWindowDiv.offsetTop** until we reach "limit to start open window 1". Since "limit to start open window" means the sequential window image (image 3.1.) shows slide number 2, 6 dotted lines below **contentThumbnailWindowDiv.offsetTop** means the sequential window image (image 3.1.) shows slide number 7, which is a fully opened window.

For the second window thumbnail, there are 5 dotted lines below **contentThumbnailWindowDiv.offsetTop** until we reach "limit to start open window 2". It means the sequential window image (image 3.1.) shows slide number 6. This will be applied to all of the next window thumbnails so every one of them will show its own sequential window image slide number.

Now let's go to the process of this window animation. Code will detect the webpage's y position, and its value will be stored in **verticalPosition**. This vertical position will be added and subtracted with some variables (**windowHeight** and **limitToStartOpenWindowFromBottom**), and its value will be compared to the position of window thumbnails container or **contentThumbnailWindowDiv.offsetTop** with some adjustment. There are 7 conditionals since there are 7 slides of sequential window image (image 3.1.), so each conditional will be represented by one slide of image.



This is the first of seven conditionals. If **verticalPosition + windowHeight - limitToStartOpenWindowFromBottom** falls under this category, which is less than **(i + 1) * scrollGap + contentThumbnailWindowDiv.offsetTop)**, then the window should be closed. For the left window panel (**windowLeftOpeningArray[i]**), this is done with **windowLeftOpeningArray[i].style.left = "0px"**, since leftmost

sequential window image (image 3.1.) shows a closed window.

```
//opening window image show slide 1
if (verticalPosition + windowHeight -
limitToStartOpenWindowFromBottom < (i + 1) * scrollGap +
contentThumbnailWindowDiv.offsetTop)
{
     windowLeftOpeningArray[i].style.left = "0px";
     windowRightOpeningArray[i].style.left =
     earlyWindowRightPositionX + "px";
}
```

The second until sixth conditionals have a very similar code structure, so they will be combined together with the loop **for (var j=1; j< 6; j++)**. For the left window panel (**windowLeftOpeningArray[i]**), the second conditional will show an almost closed window, and the sixth conditional will show an almost fully opened window (image 3.1.). Each of **verticalPosition + windowHeight - limitToStartOpenWindowFromBottom** will be tested with 2 values. The top limit is **(i + j + 1) * scrollGap + contentThumbnailWindowDiv.offsetTop)**, and the bottom limit is **(i + j) * scrollGap + contentThumbnailWindowDiv.offsetTop**. Then the sequential window image (image 3.1.) will be shifted to its position with **windowLeftOpeningArray[i].style.left = (-j * slide_width) + "px"**. This will make the window changes its looks from almost closed until fully opened (slide number 2-6 in image 3.1.).

```
//opening window image show slide 2 - 6
for (var j=1; j< 6; j++)
{
     if ((verticalPosition + windowHeight -
     limitToStartOpenWindowFromBottom >= (i + j) *
     scrollGap + contentThumbnailWindowDiv.offsetTop) &&
     (verticalPosition + windowHeight -
     limitToStartOpenWindowFromBottom < (i + j + 1) *
     scrollGap + contentThumbnailWindowDiv.offsetTop))
     {
          windowLeftOpeningArray[i].style.left = (-j *
          slide_width) + "px";
          windowRightOpeningArray[i].style.left =
          earlyWindowRightPositionX + (j * slide_width) +
          "px";
     }
}
```

image 3.3.

This will be the last conditional, if **verticalPosition + windowHeight - limitToStartOpenWindowFromBottom** is more than **(i + 6) * scrollGap + contentThumbnailWindowDiv.offsetTop)**, then the sequential window image (image 3.1.) will be shifted to slide number 7. This slide shows a fully opened window image. For the left window panel (**windowLeftOpeningArray[i]**), this is done with **windowLeftOpeningArray[i].style.left = (-6 * slide_width) + "px"**.

```
//opening window image show slide 7
if (verticalPosition + windowHeight - limitToStartOpenWindowFromBottom >= (i + 6) * scrollGap +
contentThumbnailWindowDiv.offsetTop)
{
     windowLeftOpeningArray[i].style.left = (-6 * slide_width) + "px";
     windowRightOpeningArray[i].style.left = earlyWindowRightPositionX + (6 * slide_width) + "px";
}
```

View complete:

Javascript CSS