Making Cool Stuff with Advanced CSS Animator

Last time, Nancy introduced you to the DMX Zone Advanced CSS Animator and showed you the basics for using it, along with some very basic examples. In this instalment, Nancy will show you practical applications and how you can make use of this new tool in your everyday workflow.

One thing that I want to put out there is that animation, CSS or otherwise, should not be used just for animation's sake. A little animation when it serves a purpose, such as to call attention to the state of a button, for example, can be very helpful and add value to the web page. On the other hand, zooming and popping all over the place takes away value from any project and can be most annoying, so we want to be sure we avoid that pitfall.

When planning your animation, ask yourself why you want a particular object to move. Is the movement something that helps distinguish a clicked button from the others? Or does a sliding panel allow the user to view content in a more orderly fashion instead of scrolling or flipping through a number of pages? These are good uses of animation and CSS animation does much of this easily. Let's take a look at a few examples.

Sliding Content

Anyone here remember the web of the mid 90's when we had pages that went on and on .. and on? Then we went through the circa-2000 web where the goal was to keep the web page content in browser height? What did that give us? More, albeit shorter pages. Where once we had 15 scroll-till-your-fingers-fall-off dissertations, now we had in its place 65 short click-till-you-fell-over pages that took 10 clicks to get what you wanted.

Enter Web 2.0 and ways to compartmentalize content. The Accordion allowed you to click a bar to reveal sections hidden till you needed them. Tabs were introduced to allow you to view what was behind Door #3. And then there is the slider. Click a button and more content comes careening invariably from .. somewhere. I'm going to show you how to do the latter today as it's quite useful. I was tempted to put a vroooom sound byte on the paragraph that does the sliding, but my kids won't let me do stuff like that any more.

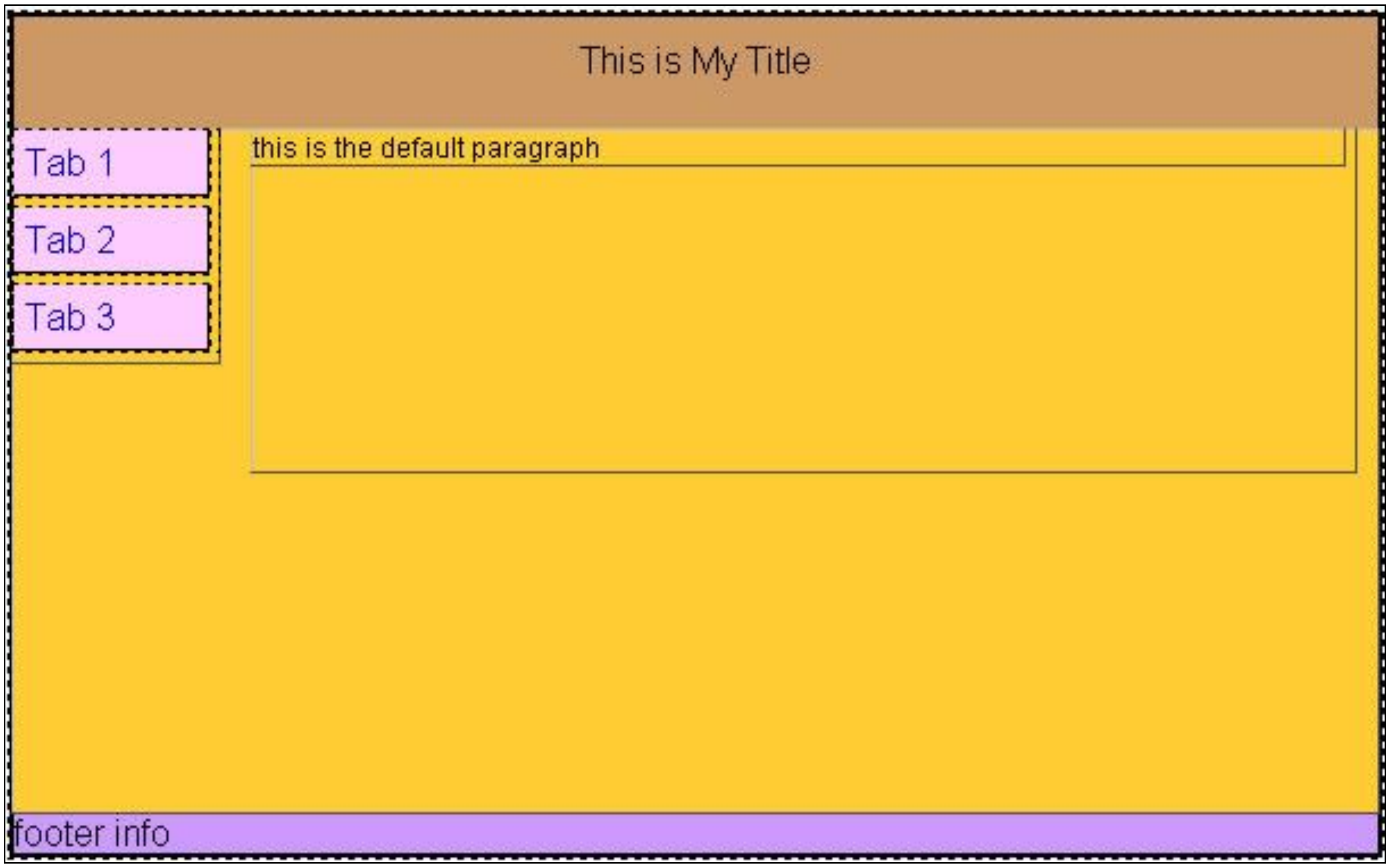
So here we go The page for this is going to be very basic and I have colored the various sections in different colors. I know most of us wouldn't design a web page that looks like this, but I wanted to make sure you would see them easily.

Here's the mark up for the basic page. Copy it into a new Dreamweaver page and save it as slider.html and we'll get started.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>My Slider</title>
<style type="text/css">
#container {
      width: 600px;
      margin: 0px auto;
      border: solid #000 2px;
      min-height: 350px;
      font-family: Arial, Helvetica, sans-serif;
}
#container #header {
      background-color: #C96;
      height: 45px;
      text-align: center;
      padding-top: 5px;
}
#container #content {
      background-color: #FC3;
      min-height: 300px;
}
#container #content #nav {
      float: left;
      width: 80px;
      margin: 0;
}
#navlist {
      margin: 0;
      display: block;
      padding: 0;
}
#container #content #nav li {
      display: block;
      padding: 5px 0px 5px 5px;
      background-color: #FCF;
      list-style-type: none;
      margin-bottom: 5px;
      border-bottom: solid #000 1px;
      border-right: solid #000 1px;
}
#nav a {
      text-decoration: none;
}
#container #content #mainContent {
      float: left;
      font-size: 12px;
      margin: 10px;
}
#container #footer {
      background-color: #C9F;
}

/* ~~ miscellaneous float/clear classes ~~ */
.clearfloat {
      clear:both;
      height:0;
      font-size: 1px;
      line-height: 0px;
}
</style>
</head>

<body>
<div id="container">
<div id="header">
My Title
</div>
```

```
<div id="content">
<div id="nav">
<ul id="navlist">
<li><a href="#">Tab 1</a></li>
<li><a href="#">Tab 2</a></li>
<li><a href="#">Tab 3</a></li>
</ul>
</div>
<div id="mainContent">
  <p>This is the default paragraph</p>
</div>
<br class="clearfloat" />
</div>

<div id="footer">
footer info
<br class="clearfloat" />
</div>

</div>
</body>
</html>
```

The structure itself is simple. We have a page with an 800 wide container that is centered in the browser. We have a header, a left hand navigation bar and a right side content page that will have additional content fly in from the right merely for effect. You will want to note a few CSS items here and I have purposefully added them later in the process so we can point them out as we go.



The first animation is the buttons. Rather than put the animation on the link (a) tag, I have put it on the LI tag so the entire box will move out when the mouse runs over it rather than just the link, which is just text, making for a better effect. I will select the first li tag in the list of tags under the Document window and open the Behaviors Panel on the Tag Inspector panel. I click on DMXzone/Advanced CSS Animator and the dialogue opens. I want to adjust the width of the li box so I click on width and the right arrow to move width to Used Properties. I already know that the li box is 80px wide because that is what was specified in the CSS for the page. So my starting width will be 80px and my finishing width will be 90px, which will move it out to the right just a little bit. I'll accept those values and set the event trigger to be onMouseEnter which gives me a little smoother transition than onMouseOver does. I need to have a way to return the nav buttons to their original position, so I'll repeat the procedure, but this time from 90 px to 80 px and use the onMouseLeave event which does a little better job then onMouseOut does. I'll test this in Live View and it does what I want so I'll repeat the procedure for the other two buttons. Take a look at the code for the finished buttons:

```
<div id="nav">
<ul id="navlist">
<li onclick="dmxAnimate('#paradefault, #para2, #para3', {},
{opacity:'hide'});dmxAnimate('#para1', {}, {top:'0px',left:'0px',opacity:'show'})"
onmouseenter="dmxAnimate(this, {width:'80px'}, {width:'90px'})"
onmouseleave="dmxAnimate(this, {width:'90px'}, {width:'80px'})"><a href="#">Tab
1</a></li>

<li onclick="dmxAnimate('#paradefault, #para1, #para3', {},
{opacity:'hide'});dmxAnimate('#para2', {}, {top:'0px',left:'0px',opacity:'show'})"
onmouseenter="dmxAnimate(this, {width:'80px'}, {width:'90px'})"
onmouseleave="dmxAnimate(this, {width:'90px'}, {width:'80px'})"><a href="#">Tab
2</a></li>
```

```
<li onclick="dmxAnimate('#paradefault, #para1, #para2', {},
{opacity:'hide'});dmxAnimate('#para3', {}, {top:'0px',left:'0px',opacity:'show'})"
onmouseenter="dmxAnimate(this, {width:'80px'}, {width:'90px'})"
onmouseleave="dmxAnimate(this, {width:'90px'}, {width:'80px'})"><a href="#">Tab
3</a></li>
</ul>
</div>
```

Now for the text that will change. We don't have these in our page yet, so let's create them with this code. Within the mainContent div, add this code:

```
<div id="mainContent">
<p id="paradefault"> this is the default paragraph</p>
  <p id="para1">This is the first paragraph.  Benchmarking against industry leaders, an
essential process, should be a top priority at all times an important ingredient of
business process reengineering building flexibility through spreading knowledge and
self-organization. </p>
   <p id="para2">This is the second paragraph.  The balanced scorecard, like the
executive dashboard, is an essential tool organizations capable of double-loop
learning, by moving executive focus from lag financial indicators to more actionable
lead indicators. While those at the coal face don't have sufficient view of the overall
goals.</p>
    <p id="para3">This is the third paragraph.  Ut labore et dolore magna aliqua.
Excepteur sint occaecat velit esse cillum dolore duis aute irure dolor. Quis nostrud
exercitation sed do eiusmod tempor incididunt ut enim ad minim veniam. Ullamco laboris
nisi lorem ipsum dolor sit amet, qui officia deserunt. Sunt in culpa sed do eiusmod
tempor incididunt excepteur sint occaecat. Eu fugiat nulla pariatur. </p>
</div>
```

In this case, it's just an announcement so you will know the paragraph actually changed, but you could have anything that you wanted to change within each paragraph. The important thing here is that each paragraph needs to have an id so we can manipulate it with the Advanced CSS Animator. There are also some CSS changes to make in order to get the paragraphs to function correctly. I don't want them to fly in lower and lower so I'm going to make the content div relatively positioned so I can make the navigation, mainContent and fly in paragraphs positioned absolutely within them. By doing this, the paragraphs will fly in at the same position (at the top) of the content div. This is the CSS for the paragraphs:

The content div CSS should now look like this:

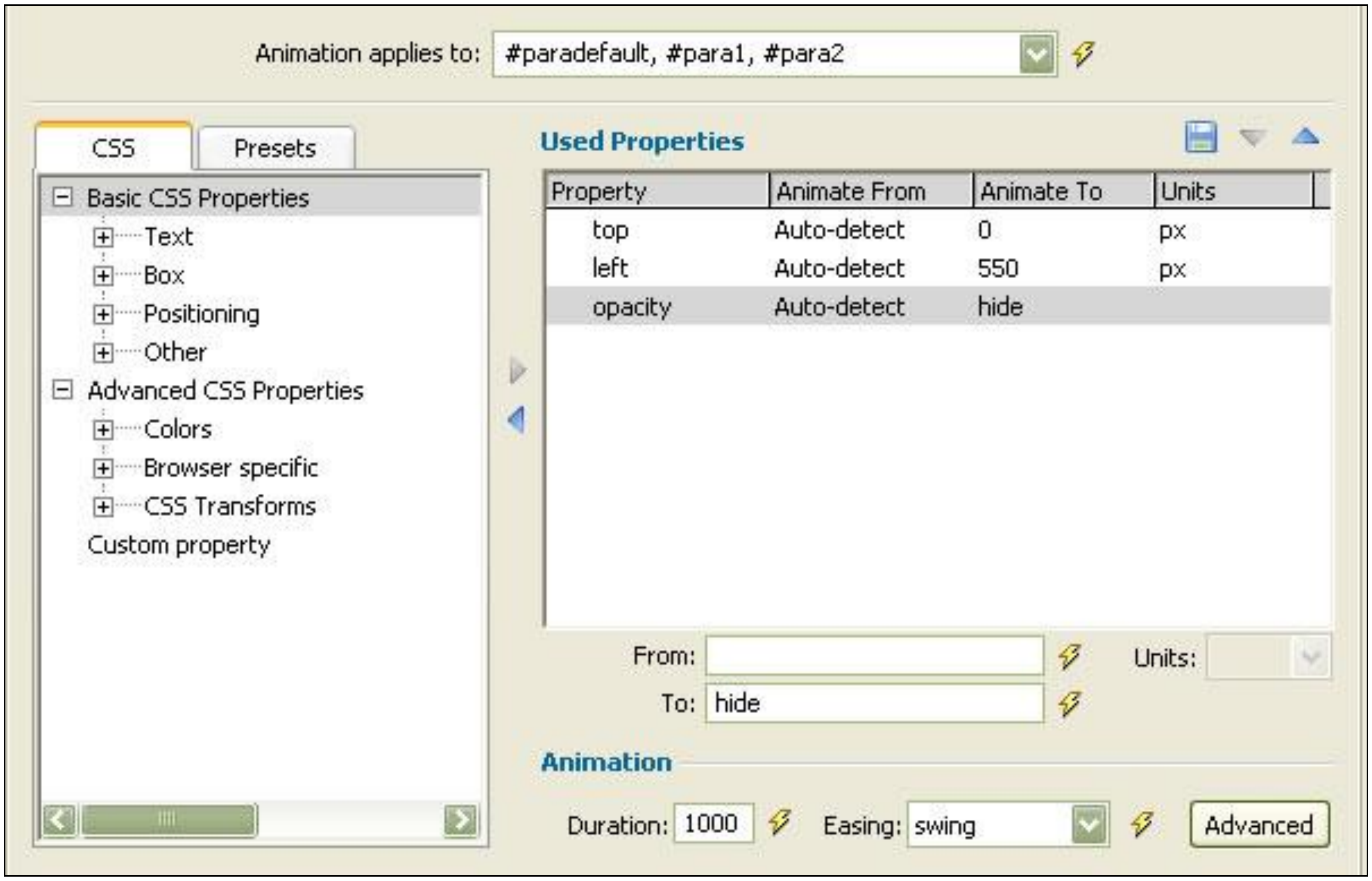
```
#container #content {
      background-color: #FC3;
      min-height: 300px;
      position: relative;
}
```

The nav will now be absolutely positioned and so will mainContent which will ensure those divs start at the top of the content div. And the paragraphs are absolutely positioned relative to the content div at 0 for the top and 0 for the left margin. The paragraphs that will fly in have 550 for the left position or off the right side of the page.

```
#mainContent #paradefault {
      display: block;
      width: 480px;
      top: 0px;
      left: 0px;
      position: absolute;
}
#mainContent #para1 {
      display: none;
      width: 480px;
      position: absolute;
      top: 0px;
      left: 550px;
}
#mainContent #para2 {
      display: none;
      width: 480px;
      position: absolute;
      top: 0px;
      left: 550px;
}
#mainContent #para3 {
      display: none;
      width: 480px;
      position: absolute;
      top: 0px;
      left: 550px;
}
```

You will also notice that the default paragraph has its display property set to block, which the paragraphs which are not yet seen are set to display:none so that they are invisible to begin with.   Now we're ready to apply the Advanced CSS Animator behaviors. Select the first link again and add on to the behaviors for the links. Open the Advanced CSS Animator and select opacity and add it to the Used Properties box. We're first going to hide whatever we have in the page at the current time. Keep in mind that you don't always know what the user currently has in front of them, but you do know which link you are working with. We're working with the first tab and it governs the first paragraph, so we want to hide all the other paragraphs.

By default, the Advanced CSS Animator only works with one div at a time, but you can set up the opacity, select the default paragraph from the dropdown list and then click in the dropdown list and add in the others yourself. This will save you from individually having to apply the same behaviour three times. Add opacity to the Used Properties with a result of hide and then set the top to be 0 and the left to be 550px to move the paragraphs not being used out to the right of the page. Set the event trigger to be the onClick event in the Tag Inspector.


Animation applies to: #paradefault, #para1, #para2

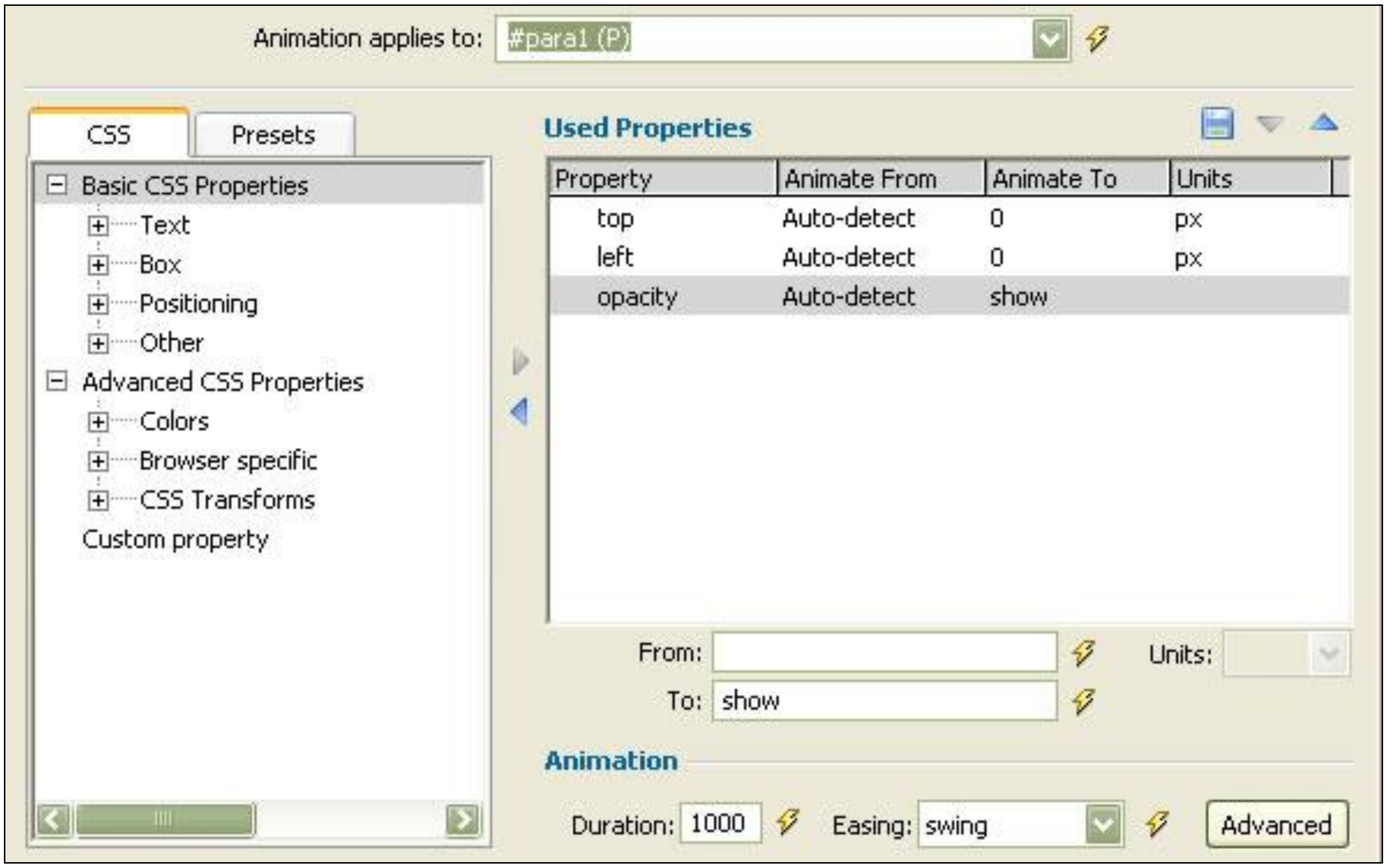
Property	Animate From	Animate To	Units
top	Auto-detect	0	px
left	Auto-detect	550	px
opacity	Auto-detect	hide	

From:
To: hide
Units:

Animation

Duration: 1000 Easing: swing

Now apply the Advanced CSS Animator again but this time select #para1 from the dropdown list and set the top to be 0 and the left to be 0 and the opacity to be show as the result. This will move the paragraph in to the page and absolutely position it within the mainContent div. Apply the same behaviors to the other two tabs and #para 2 and #para3 and test the result. This is a basic example but I think you get the idea. With the Advanced CSS Animator, you can add all kinds of multiple effects and you can apply them to multiple regions all at the same time! Pretty incredible, eh?


Animation applies to: #para1 (P)

Property	Animate From	Animate To	Units
top	Auto-detect	0	px
left	Auto-detect	0	px
opacity	Auto-detect	show	

From:
To: show
Units:

Animation
Duration: 1000 Easing: swing

A Simple Image Gallery

In this example, we are going to create an image gallery using HTML, CSS and the Advanced CSS Animator, which will add supporting javascripts for browsers that don't support CSS Animation as yet. This is one of the best things about the Advanced CSS Animator, because you don't have to worry about what browsers can see your work and which ones are going to bomb out. It will just work everywhere and there is no Flash required.

Start with a new blank HTML page and **save** it as mygallery.html. To give some structure to the page, create a container div to center the gallery and an image container div within that container. We'll also create a third div below the image container to hold the controls. Here is the HTML for the basic page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Untitled Document</title>
</head>

<body>
<div id="container">
<div id="imagecontainer">
<!--end imagecontainer --></div>
<div id="controls">
<ul>
```

```
<li>1</li>
<li>2</li>
<li>3</li>
<li>4</li>
</ul>
<!--end controls --></div>
<!--end container --></div>
</body>
</html>
```

That doesn't look like much if you take a look in Live View or a browser. So let's add some basic CSS to tidy it up. We want the container to center our gallery in the browser, so we'll give the container a width and set the margins to automatically center no matter the size of the browser. Here is the container CSS:

```
#container {
      width: 800px;
      margin: 0px auto;
}
```

You can use whatever width you want for your container, but keep in mind that without a width, the autocentering doesn't work.

Next we're going to define the size of the image container. I have already created my images to be 400 px by 300 px so those are the dimensions of my image container as well. I want to center it within the container as well so I have specified that in the CSS for the image container. I have also set the overflow to hidden so there won't be anything leaking out the sides anywhere.

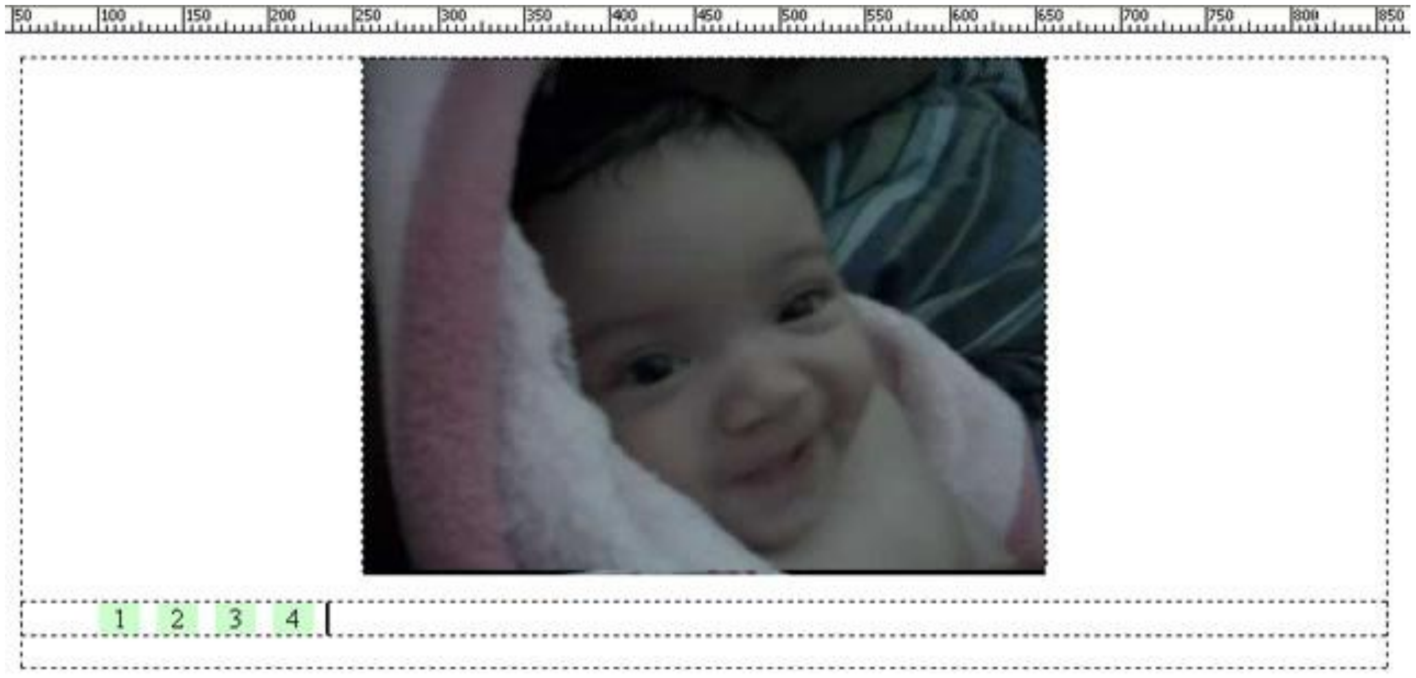
```
#imagecontainer {
      height: 300px;
      width: 400px;
      margin: 0px auto;
      overflow: hidden;
}
```

You may have noticed in the HTML that I used an unordered list for my controls, which are numbers 1-4 since I have only 4 images in my gallery. Obviously the number changes to match the number of images if you are making a larger (or smaller) gallery. In the CSS for the controls, I will specify display to be inline so the list items are side by side in one row. The CSS for the controls follows:

```
#controls ul li {
      display: inline;
      margin: 0px 3px 0px 3px;
      background-color: #CFC;
      padding: 0px 8px;
}
```

I have also put a light yellow background to give more of a block appearance to the controls so they aren't just a number out there and I added a little horizontal padding to spread them out a little. The last thing I have done is to place the images into the image container on top of each other with the first image on top. Because I have hidden the overflow, this in essence stacks the pictures and now I can use the Advanced CSS Animator to change what is in the image container depending on the control selected. There are many ways one could do this and we'll look at few.

One of the easiest, in my opinion is to stack the images and hide or show the image depending on the control selected. This is similar to the old JavaScript Show/Hide behaviour, but this Is done using CSS. The Advanced CSS Animator makes it point and click easy.



The simple html and css interface showing the first image in our gallery

Applying the Animator

The first thing to do is to select the trigger to start the chain of events we are about to create. I am going to use the onClick event to act as my trigger so I need to make each number into a link and because I am using the CSS Advanced Animator to define what happens when the link is clicked, I'll just put **javascript:;** into the link box in Dreamweaver's Property Inspector and that will create the **a** tag that I need to define my link in the Behaviors Panel. Next, I select the a tag in the Document Inspector at the base of Dreamweaver's document window and select the Tag Inspector Panel and the Behaviors tab on that panel and from there, the Advanced CSS Animator. As in the previous example, when I click on the first link, I am going to set the opacity of the first picture to show and the other pictures to hide. Likewise, when I select the second link, I want to set the second picture to show and hide the others and so forth. This is similar to what we did before, but this time, we are showing one picture that is stacked and hiding the others with a fading transition.

That's it for this time. We'll be cooking up some more advanced uses of the Advanced CSS Animator in the future, so stay tuned!

Sliding Panels Extension

Just as a side note – you can also use the Sliding Panels DMXzone extension to create amazing sliding effects out of the box. See for more info http://www.dmxzone.com/go?17564