CSS3 Transitions and Transforms From Scratch

webdesign.tutsplus.com/articles/css3-transitions-and-transforms-from-scratch--webdesign-4975

There are some amazing examples of CSS transforms and transitions, and whilst you may be blown away by them, there's a good chance that you're also overwhelmed and a bit intimidated! This tutorial will take you back to the very basics. We're going to create some fundamental CSS3 transitional movements, step by step.

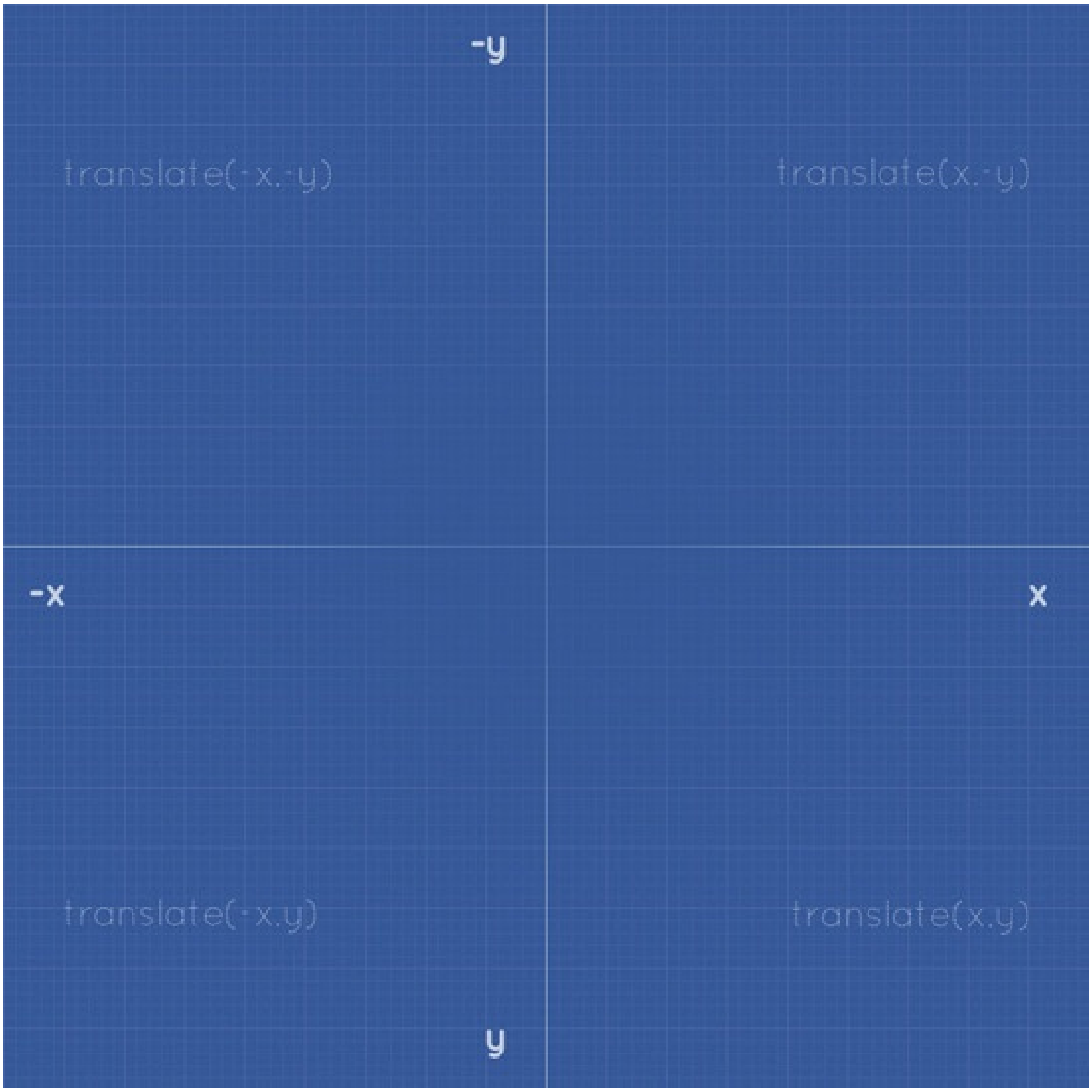
A Quick Note on Browser Support:

Support across browsers is already pretty reasonable. Firefox 3.5, Chrome, Opera 10.5, Safari 3.1, and Internet Explorer 9.0 have you covered where transforms are concerned. IE is a little less accommodating with regard to transitions, though support is planned for IE10.

The Axes and Grid

To help understand the movement easily we'll be working on an axis grid (which you'll probably recognize from basic math). We'll be using *x* and *y* coordinates to move our objects.

The only (crucial) difference is that on our axis the -y value is *above* the x axis, whilst it would ordinarily be below it. Why? Well, HTML and CSS (along with other web technologies like ActionScript) use an *inverted* Cartesian coordinate system because web pages start from top-left and read downwards. So now you know :)



Note: I'm going to assume that you're already familiar with HTML and CSS file structure. I'm going to skip explaining how to set up the CSS file, placing images and styling the Axis. Our focus will be on animating the objects. If you're not confident that your HTML + CSS skills are up to scratch, take a look at the new Tuts+ Premium HTML & CSS in 30 days (which is free) and you'll learn everything you need to know.

1. Horizontal Movement

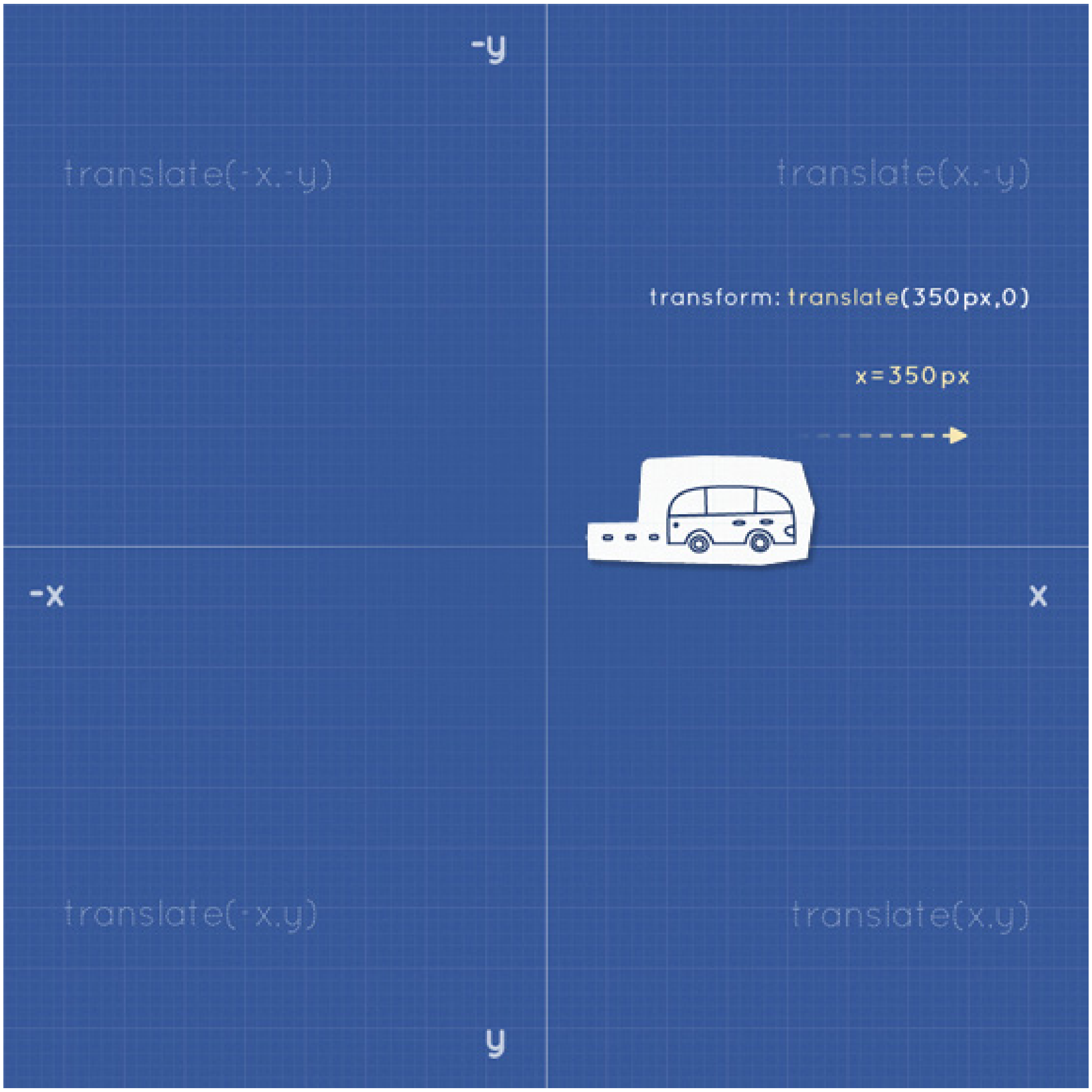
The first movement we'll demonstrate is "horizontal"; we'll animate the object to move to the right and to the left.

Moving to the Right

To move an object from its initial position we use: `transform: translate(x,y);`, where the x value should be positive and the y value should be 0 to move the object to the right.

HTML

Open your favorite Text Editor and enter the following html markup, then save the file.



```
<div id="axis" class="one">
<img class="object van move-right" src="images/van-to-
right.png"/>
</div>
```

We've assigned three classes to the image:

- **object:** We use this class to set general rules for all the objects we will use.
- **van:** We're going to use different objects to demonstrate each animation, so we'll apply different classes to them as well. This way we can position each object separately.
- **move-right:** We'll move the object using this class, each movement will have different class.

CSS

Firstly, we'll position the object (our van image) to the center of the grid.

```
.object {
 position: absolute;
}
.van {
 top: 45%;
 left: 44%;
}
```

In this example we are going to move the object 350px to the right. The syntax is `transform: translate(350px,0);` and the object will move when the Axis is hovered over. We therefore declare it with `#axis:hover .move-right`.

```
#axis:hover .move-right{
 transform: translate(350px,0);
 -webkit-transform: translate(350px,0); /** Chrome & Safari
**/
 -o-transform: translate(350px,0); /** Opera **/
 -moz-transform: translate(350px,0); /** Firefox **/
}
```

The CSS transform property will only move the object from one point to another, it will not animate between the two states. To do this we need to add a transition property in the .object class.

```
.object {
 position: absolute;
 transition: all 2s ease-in-out;
 -webkit-transition: all 2s ease-in-out; /** Chrome & Safari
**/
 -moz-transition: all 2s ease-in-out; /** Firefox **/
 -o-transition: all 2s ease-in-out; /** Opera **/
}
```

This transition rule will tell the browser to animate `all` properties attached to the object for `2 seconds` using an `ease-in-out` timing function (tween), without delay.

We can use 6 types of transition-timing-functions:

- **linear:** the transition will have constant speed from start to end.
- **ease:** the transition will start slowly, then get faster, before ending slowly.
- **ease-in:** the transition will start slowly.
- **ease-out:** the transition will end slowly.
- **ease-in-out:** the transition starts and ends slowly.
- **cubic-bezier:** define specific values for your own transition.

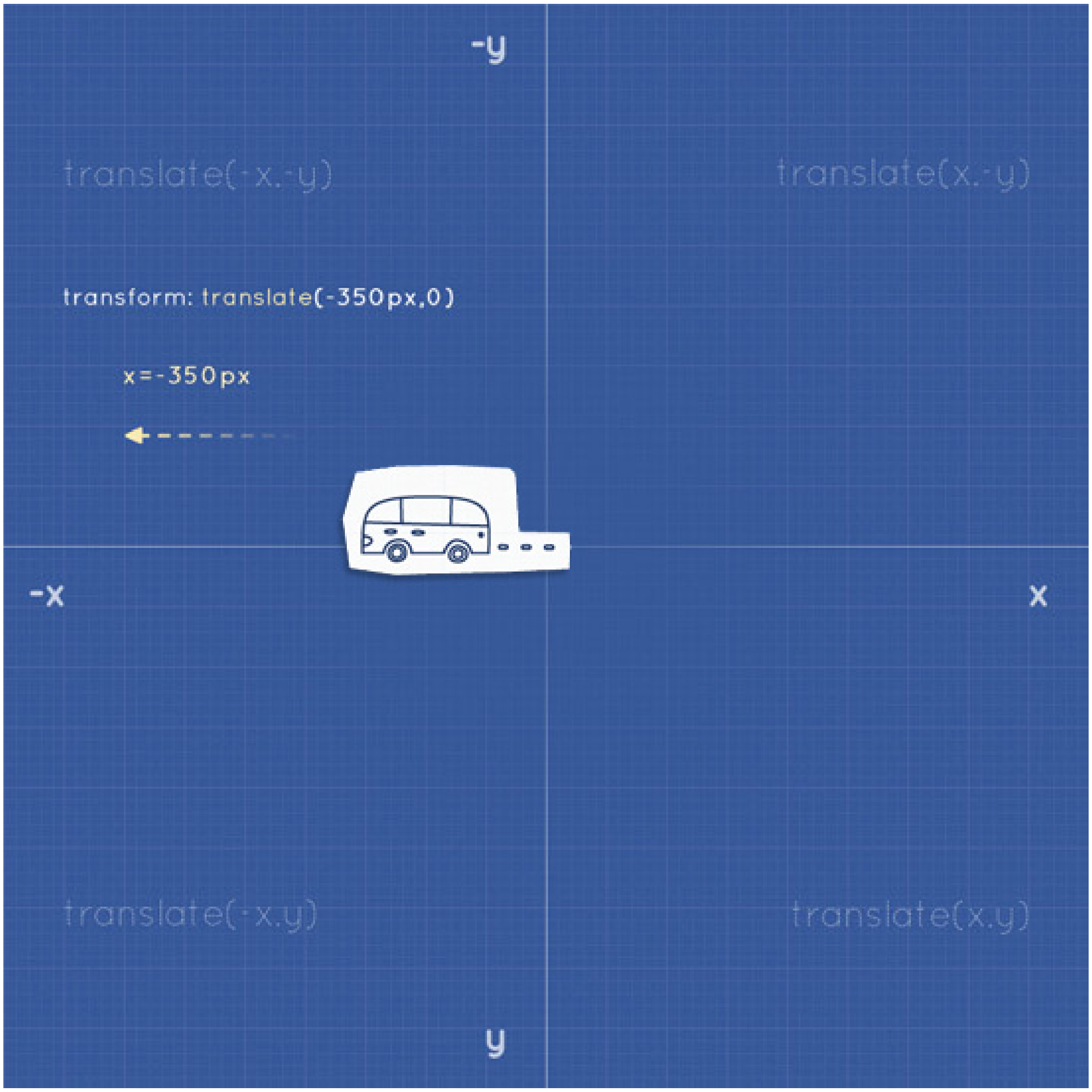
View Demo

Moving to the Left

To move an object to the left we simply need to enter a negative value in the **x** coordinate, while the **y** coordinate should remain **0**. In this example we will move the object `-350px` to the left.

HTML

Create another HTML file and enter the following markup.



```
<div id="axis" class="two">
 <img class="object van move-left" src="images/van-to-left.png"/>
</div>
```

This time we use the `move-left` class to set the css rule for moving the object to the left.

CSS

Then, we enter -350px as the x coordinate. `transform: translate(-350px,0);` to move the object to the left. Quite easy, right?

```
#axis:hover .move-left {
 transform: translate(-350px,0);
 -webkit-transform: translate(-350px,0); /** Safari & Chrome
**/
 -o-transform: translate(-350px,0); /** Opera **/
 -moz-transform: translate(-350px,0); /** Firefox **/
}
```

Since we have previously set the transition rule in our .object class, we don't need to set it again.

View Demo

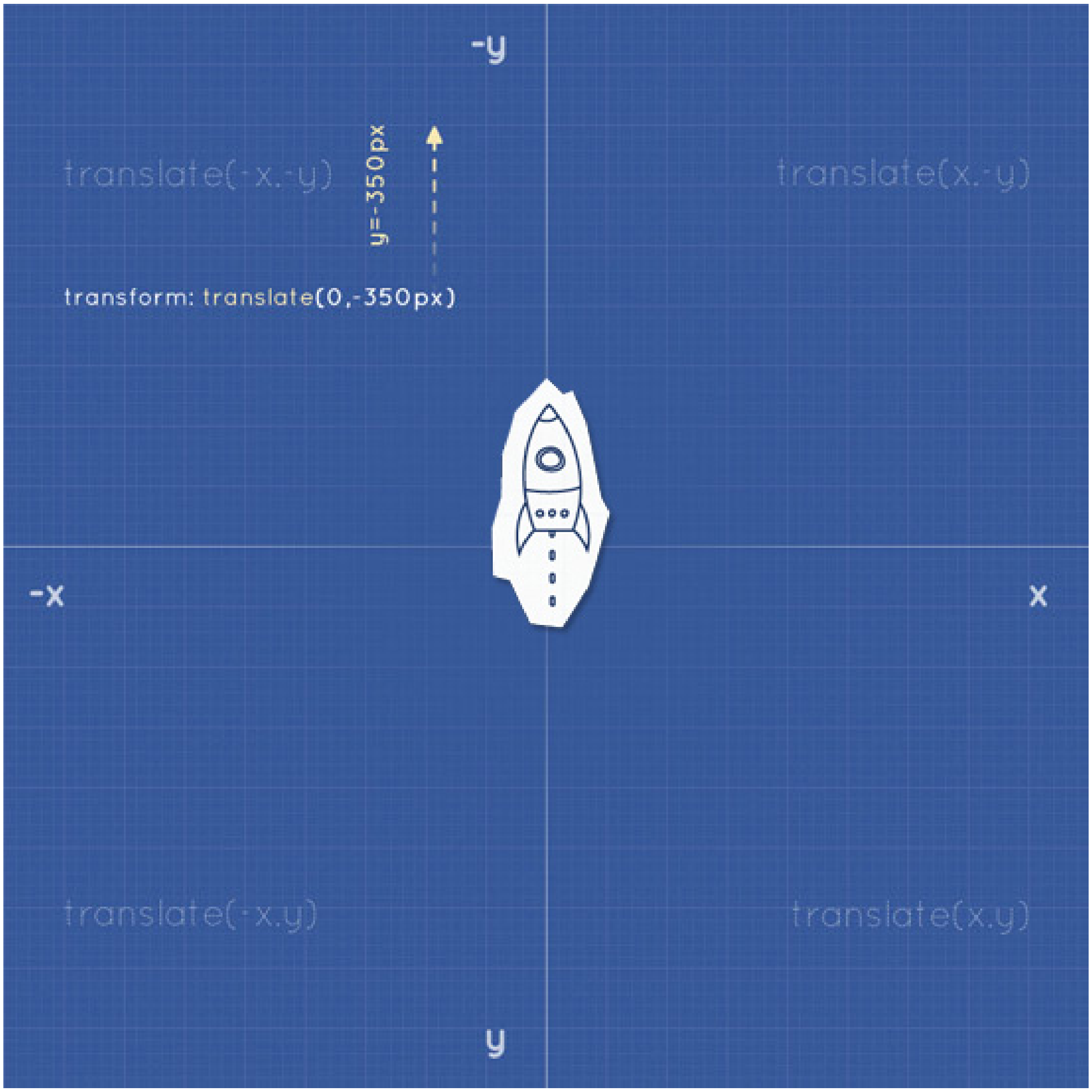
2. Vertical Movement

Moving an object vertically is pretty easy, almost identical to moving horizontally. The only difference is that we will use the -y value to move the object upward and the y value to move downward.

Moving Upwards

HTML

The HTML markup is identical to the previous two examples. However, we're replacing the object with a rocket (for illustrative purposes), and the class that we use for our upwards movement is **move-up**.



```
<div id="axis">
 <img class="object rocket move-up"
src="images/rocket.png"/>
</div>
```

CSS

As with our van, we'll position the rocket in the center of the grid:

```
.rocket {
 top: 43%;
 left:
44%;
}
```

As we've mentioned before, the y coordinate should be negative, in order to move the rocket upwards. in this case we move it 350px up.

```
#axis:hover .move-up {
 transform: translate(0,-350px);
 -webkit-transform: translate(0,-
350px);
 -o-transform: translate(0,-350px);
 -moz-transform: translate(0,-350px);
}
```

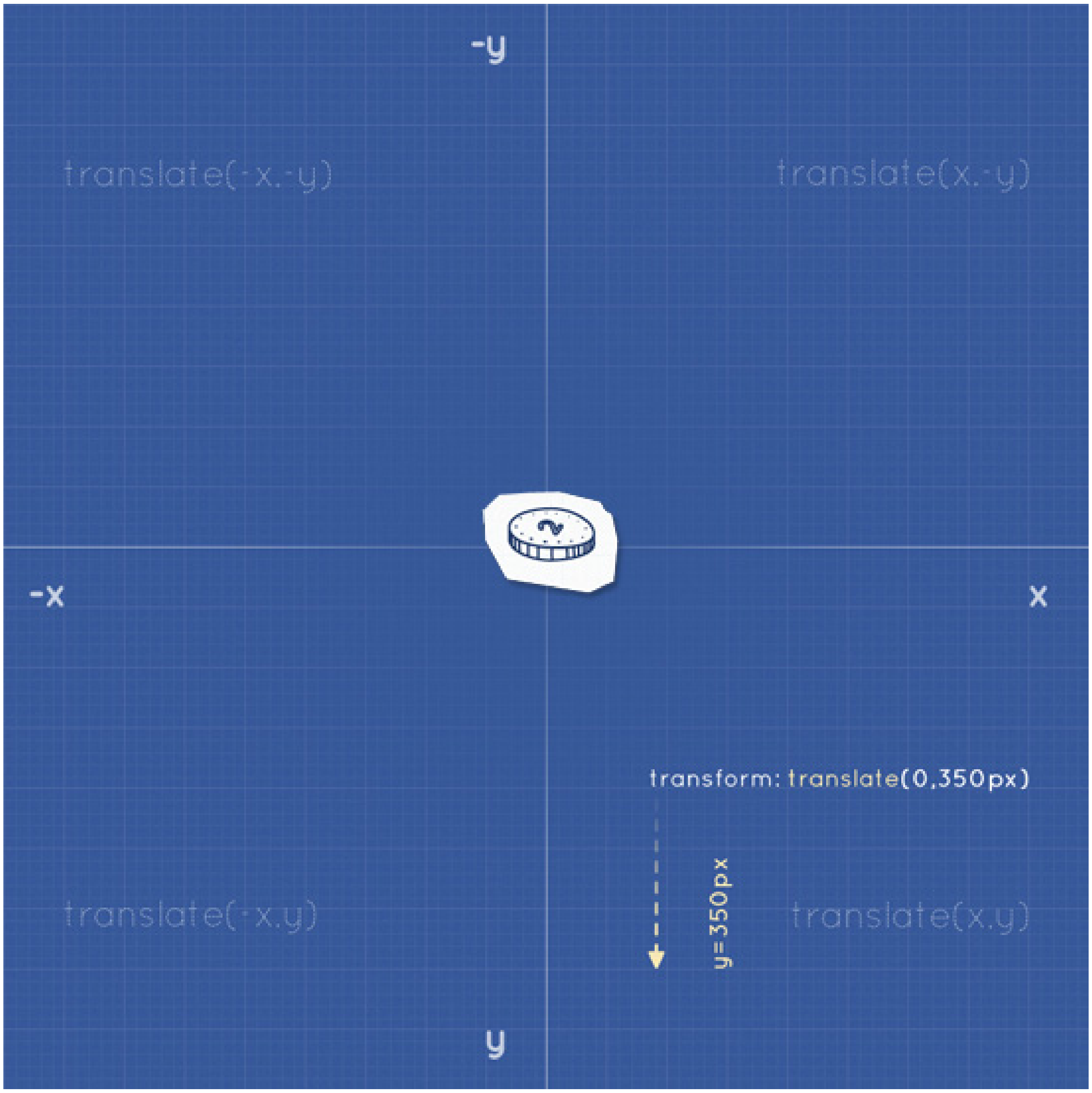
View Demo

Moving Downwards

The method for moving an object downwards is (surprise, surprise) the opposite of moving upwards; the y coordinate value should be positive and the x coordinate should remain 0. The syntax is `transform: translate(0,y);`

HTML

In this example, we'll demonstrate downwards movement using a coin. Genius!



```
<div id="axis" class="four">
<img class="object coin move-down"
src="images/coin.png"/>
</div>
```

CSS

```
#axis:hover .move-down {
 transform: translate(0,350px);
 -webkit-transform:
translate(0,350px);
 -o-transform: translate(0,350px);
 -moz-transform: translate(0,350px);
}
```

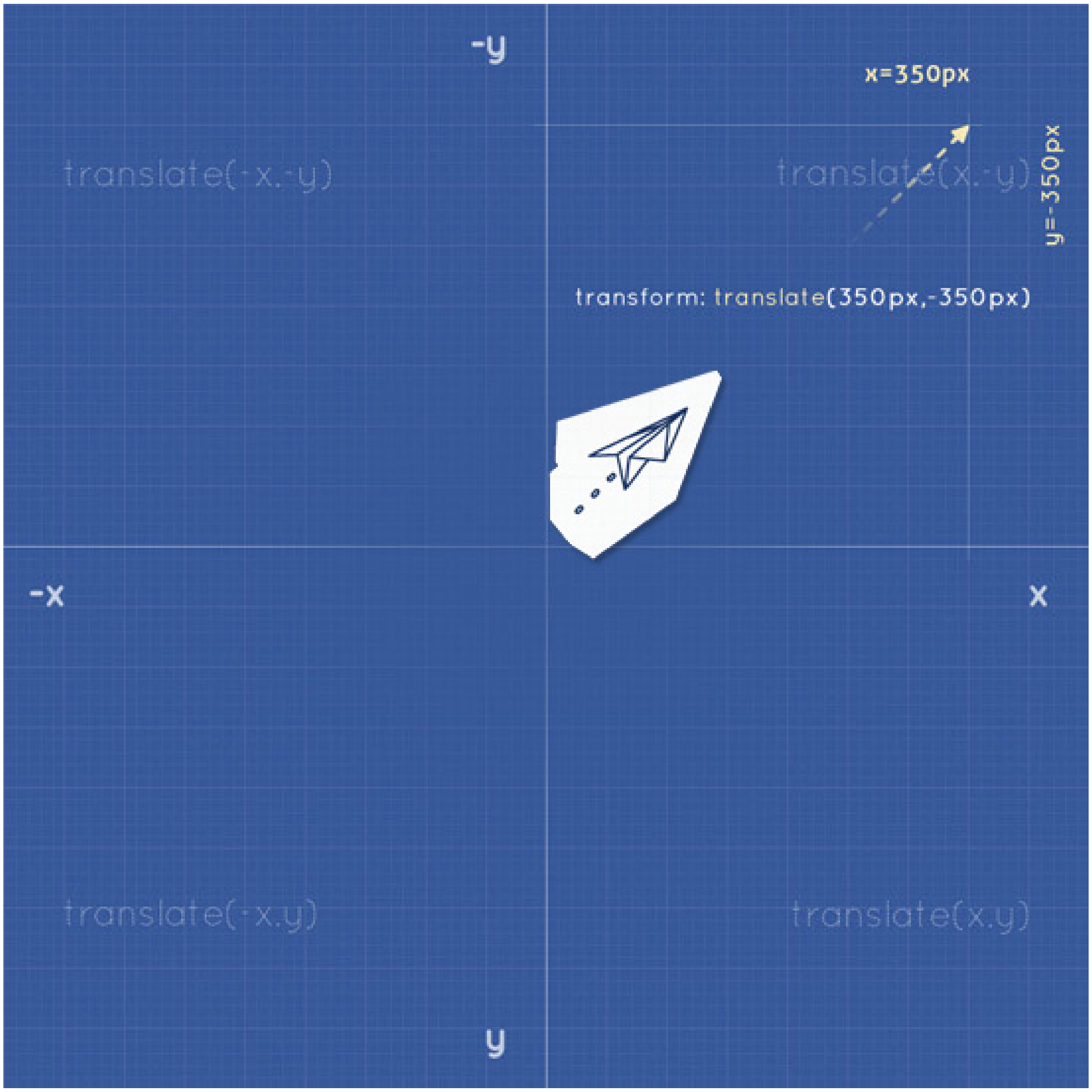
View Demo

3. Diagonal Movement

To create a diagonal transition, we'll combine values of both coordinates **x** and **y**. The syntax should be `transform: translate(x,y)`. Depending on the direction, the value of **x** and **y** could be negative or positive.

HTML

And to demonstrate our diagonal movement, we'll use a paper plane.



```
<div id="axis" class="five">
<img class="object plane move-ne" src="images/paper-
plane.png"/>
</div>
```

CSS

We'll direct movement toward the north east. For the **x** coordinate value we enter a positive value (350px) and for the **y** coordinates we enter a negative value (-350px). The syntax will therefore look like this:

```
transform: translate(350px,-350px);
```

```
#axis:hover .move-ne {
 transform: translate(350px,-350px);
 -webkit-transform: translate(350px,-
350px);
 -o-transform: translate(350px,-350px);
 -moz-transform: translate(350px,-350px);

}
```

Feel free to experiment and direct the movement of objects along the other diagonal axes.

View Demo

4. Rotation

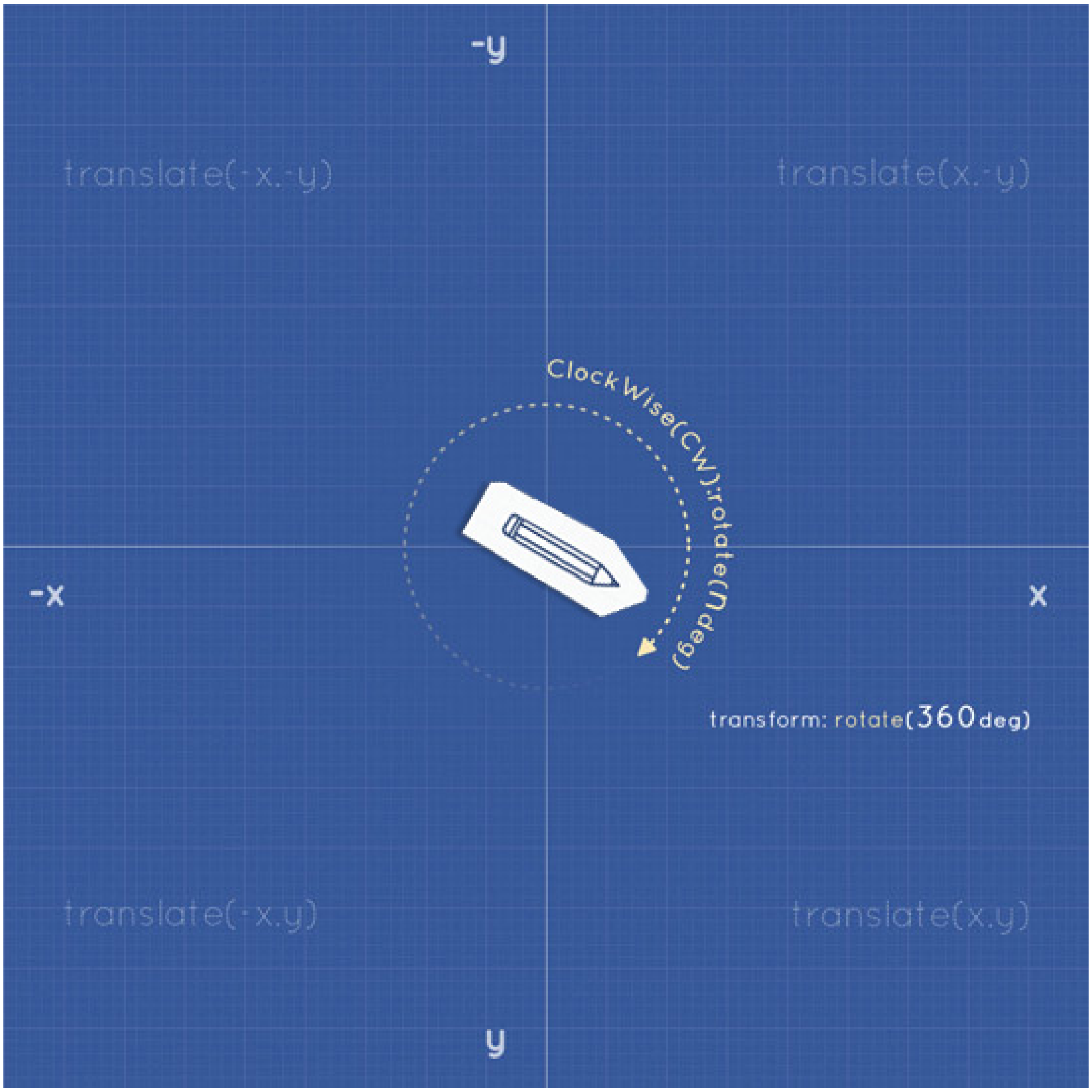
Rotational movement in CSS3 is regulated using a radial coordinate from 0° to 360°. To rotate an object simply apply the following css property: `transform: rotate(ndeg);` where **n** is the degree of rotation.

360° Clockwise

To rotate an object clockwise, we enter a positive value for the `rotate(ndeg)` property.

HTML

For this example we'll use a pencil to demonstrate the movement.



```
<div id="axis" class="six">
<img class="object pencil rotate360cw"
src="images/pencil.png"/>
</div>
```

CSS

And we'll rotate the object 360 degrees clockwise.

```
#axis:hover .rotate360cw {
 transform: rotate(360deg);
 -webkit-transform:
rotate(360deg);
 -o-transform: rotate(360deg);
 -moz-transform: rotate(360deg);
}
```

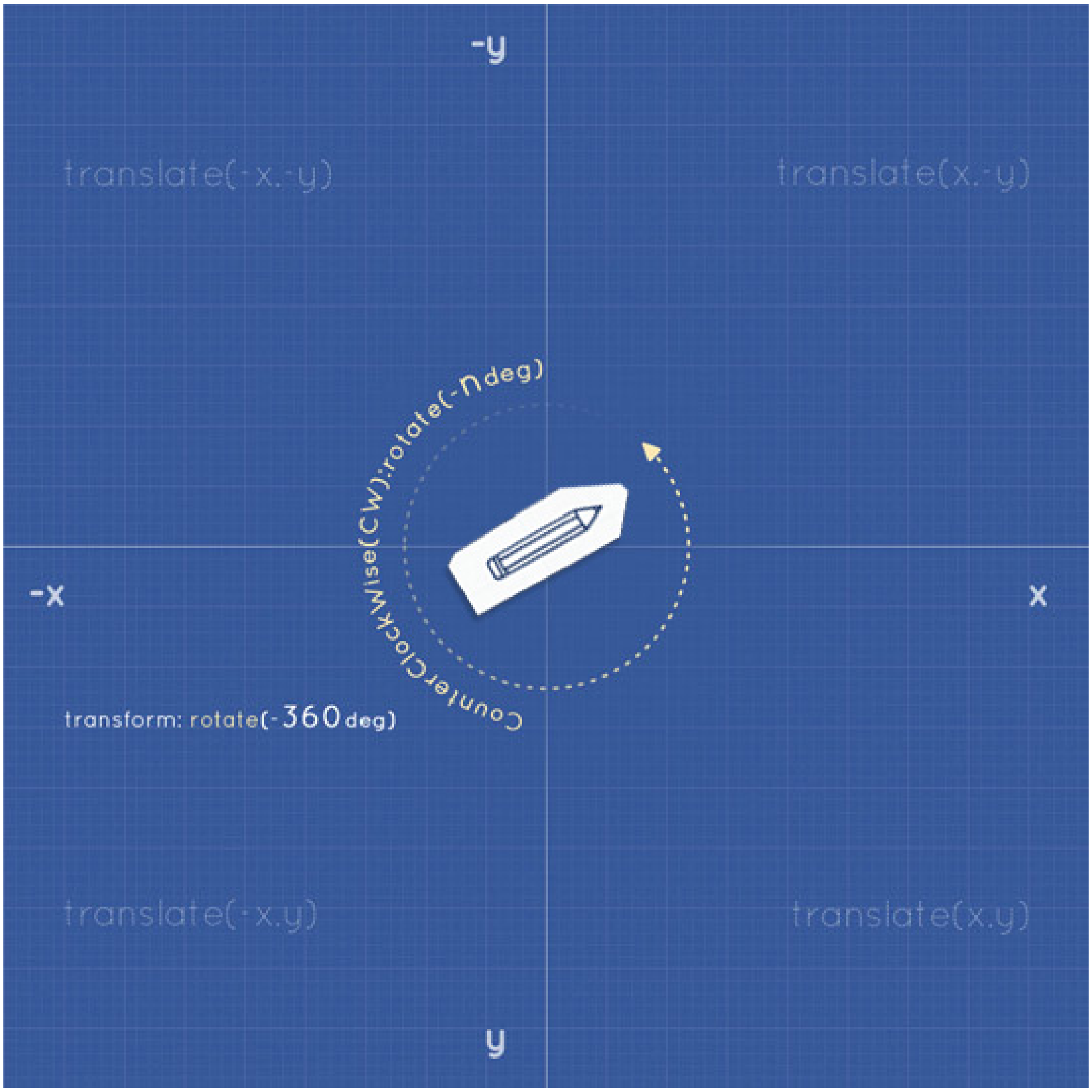
View Demo

360° Counter Clockwise

To perform counter-clockwise rotation we enter (you guessed it) a negative value.

HTML

We're still using the pencil as our object, but we'll change its class to .rotate360ccw.



```
<div id="axis" class="seven">
<img class="object pencil rotate360ccw"
src="images/pencil.png"/>
</div>
```

CSS

```
#axis:hover .rotate360ccw {
 transform: rotate(-360deg);
 -webkit-transform: rotate(-
360deg);
 -o-transform: rotate(-360deg);
 -moz-transform: rotate(-360deg);
}
```

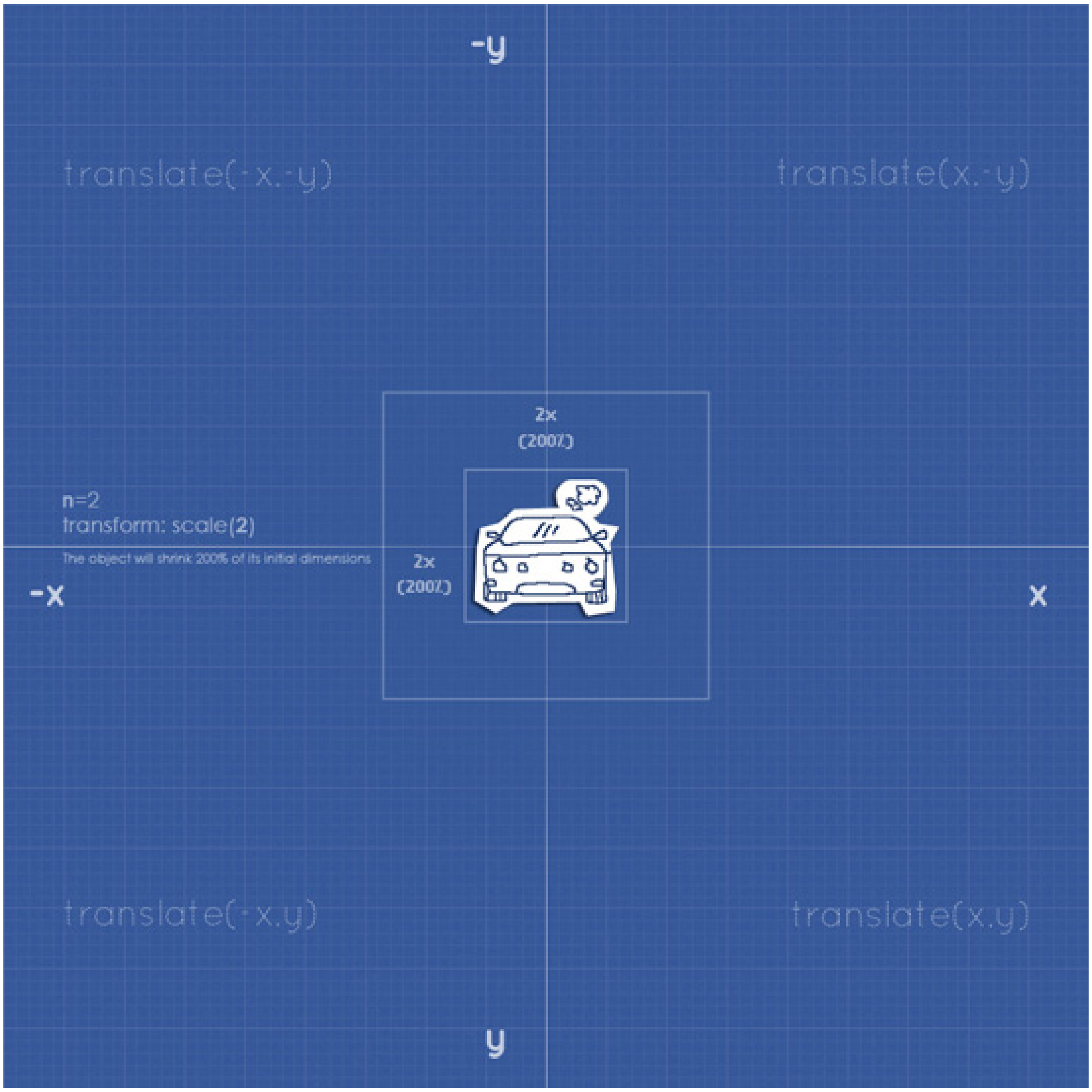
View Demo

5. Scaling

Scale is an interesting feature in CSS3. By using the scale(n) or scale(x,y) property we can enlarge or shrink an object within our HTML. The object will be scaled according to n/x,y value, where the x-axis is for the width and the y-axis represents the height. For example, if we enter scale(2), the object will be scaled twice (200% larger) along both axes, from its initial dimension.

Let’s take a look at the example below.

In this example we'll increase the size of the car by 200%, giving the illusion that the car is moving closer (hopefully).



HTML

Again, the HTML markup has barely changed, but this time we use a car as the object.

```
<div id="axis" class="eight">
<img class="object car larger"
src="images/car.png"/>
</div>
```

CSS

```
#axis:hover .larger{
 transform: scale(2);
 -webkit-transform:
scale(2);
 -o-transform: scale(2);
 -moz-transform: scale(2);
}
```

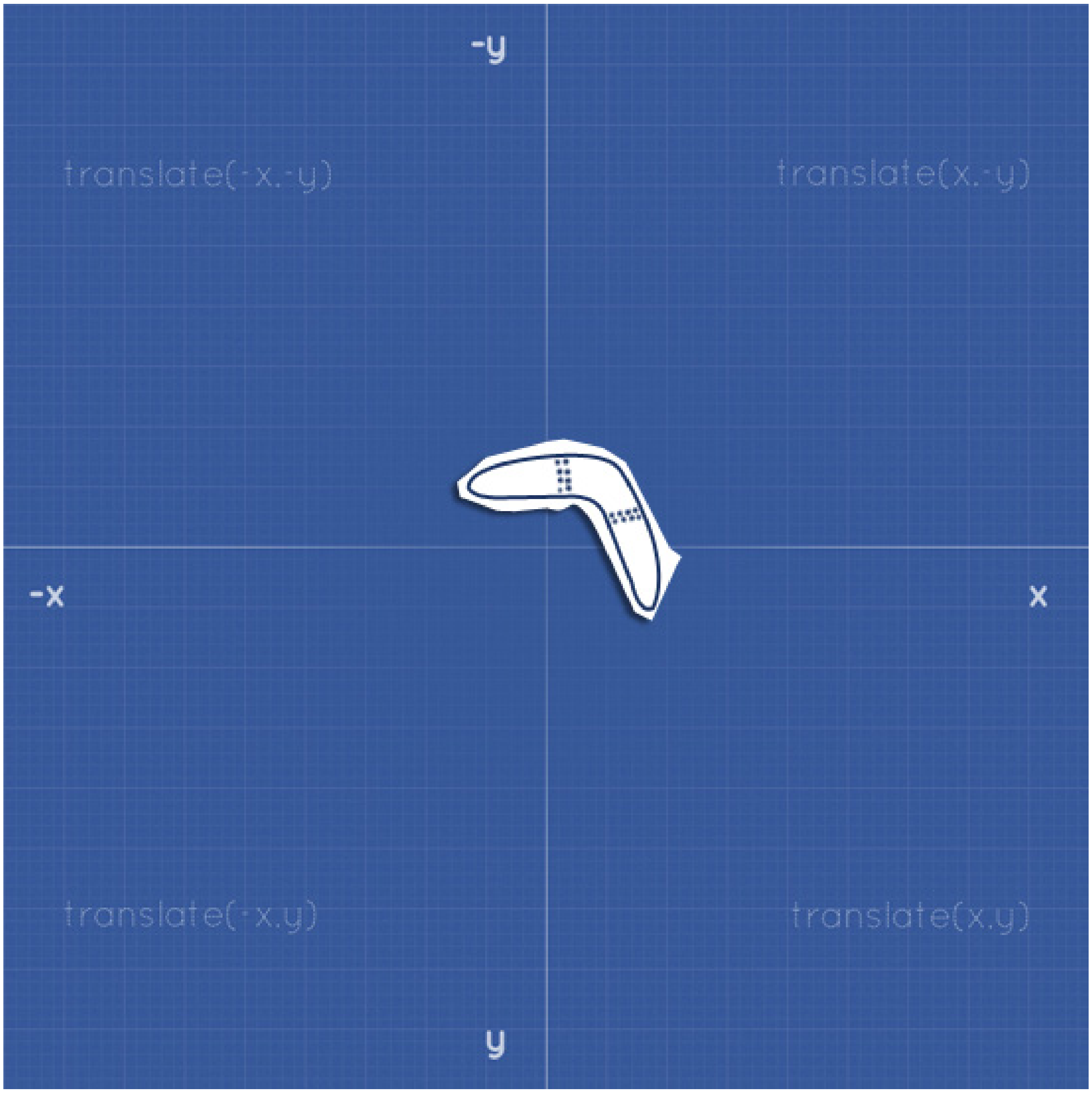
View Demo

6. Multiple Movements

After having played with basic movements and transforms, we'll now try to combine some of them. Let's take a look.

HTML

This time, we'll be using a boomerang to demonstrate the animation.



```
<div id="axis" class="ten">
<img class="object boomerang multiple-move"
src="images/pencil.png"/>
</div>
```

CSS

The plan is to move the boomerang diagonally, while performing rotations at the same time. In order to do this, we simply have to list the transformations, separated by spaces.

```
#axis:hover .multiple {
 transform: translate(350px, -350px) rotate(360deg);
 -webkit-transform: translate(350px, -350px)
rotate(360deg);
 -o-transform: translate(350px, -350px) rotate(360deg);
 -moz-transform: translate(350px, -350px) rotate(360deg);
}
```

View Demo

Conclusion

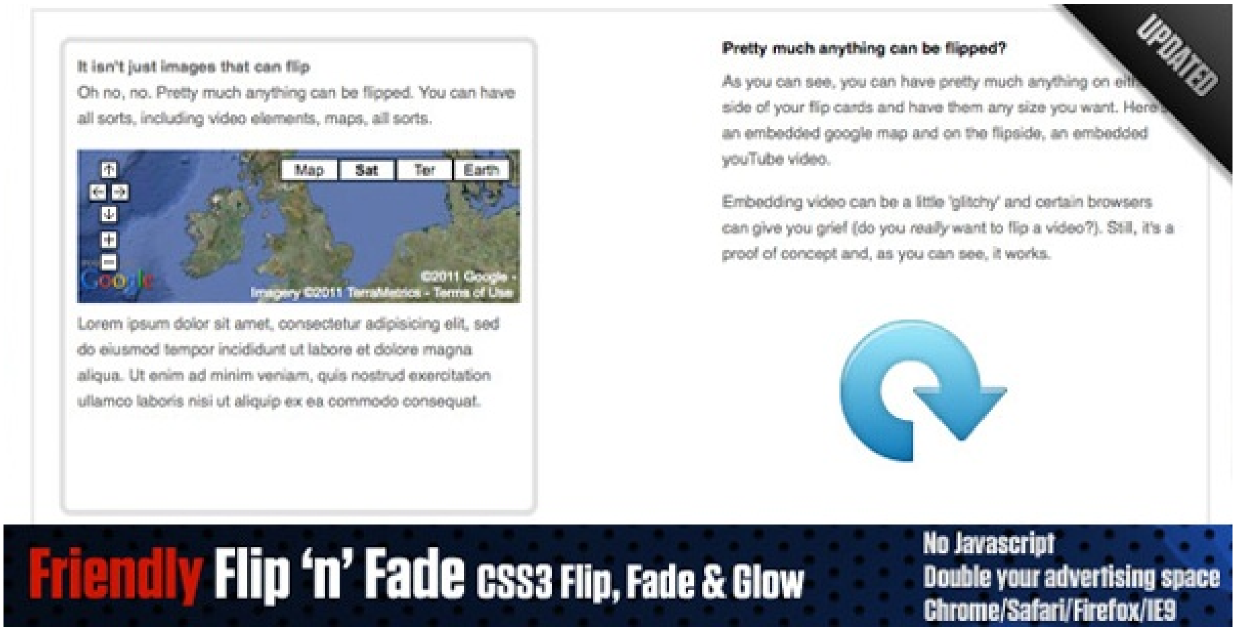
These are basic examples, and there's huge scope for developing them further! Remember: take browser support into account when using CSS3 properties, and don't go crazy with these effects - they should enhance your design, not drown it!

More CSS3 Transition Examples

For more inspiration, take a look at some of the cool CSS3 animations and effects on Envato Market. Here are a few examples:

1. Friendly Flip 'n' Fade

These gorgeous, subtle and useful CSS3 flip, fade and glow transitions help you extend a website’s real estate and utilise your space more effectively. Any content at all, including videos, maps, images, etc., can all be flipped (or faded).



It isn't just images that can flip

Oh no, no. Pretty much anything can be flipped. You can have all sorts, including video elements, maps, all sorts.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Pretty much anything can be flipped?

As you can see, you can have pretty much anything on eit side of your flip cards and have them any size you want. Here an embedded google map and on the flipside, an embedded youTube video.

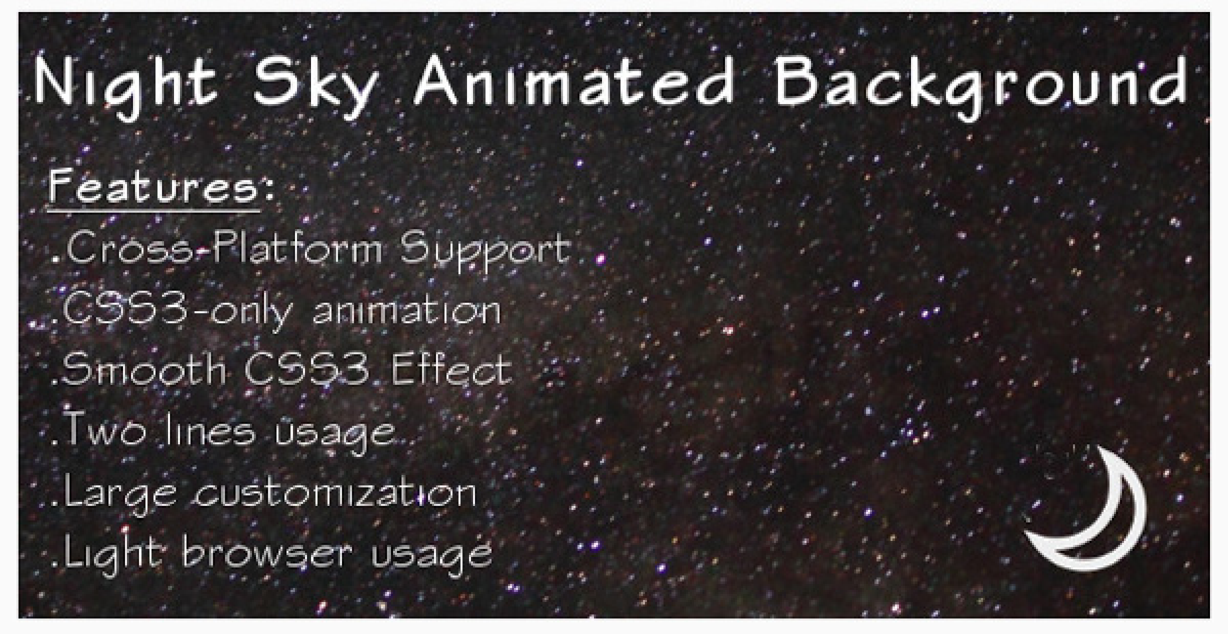
Embedding video can be a little 'glitchy' and certain browsers can give you grief (do you *really* want to flip a video?). Still, it's a proof of concept and, as you can see, it works.

UPDATED
Map
Sat
Ter
Earth
Google
©2011 Google - Imagery ©2011 TerraMetrics - Terms of Use
Friendly Flip ‘n’ Fade CSS3 Flip, Fade & Glow
No Javascript
Double your advertising space
Chrome/Safari/Firefox/IE9

Friendly Flip 'n' Fade on Envato Market

2. Night Sky Animated Background

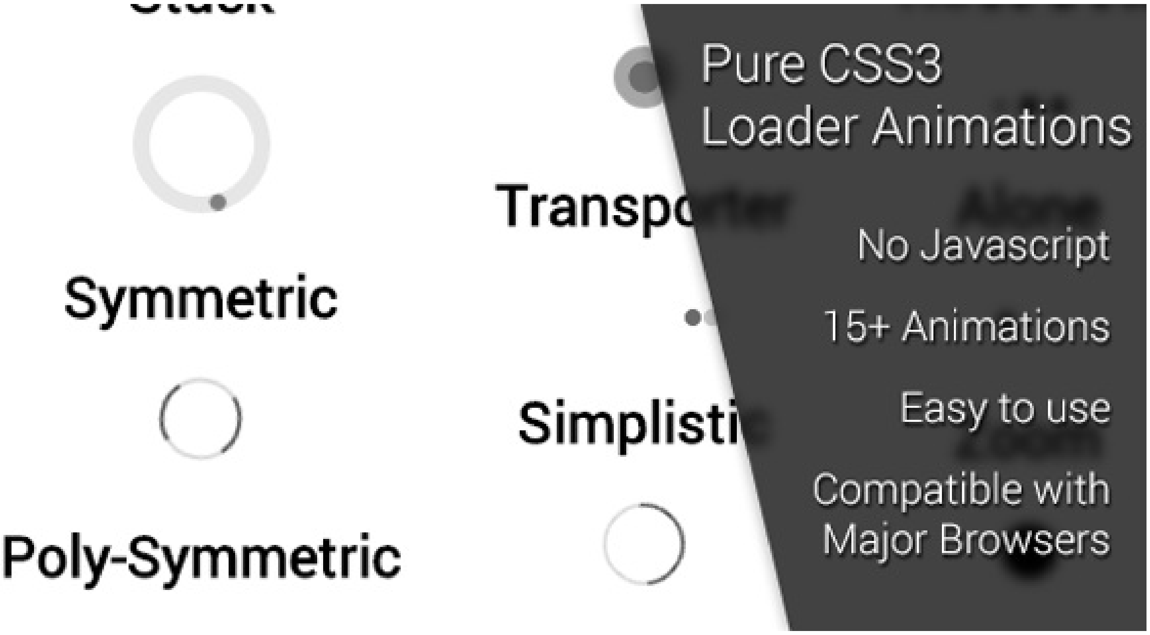
This is a CSS / jQuery based utility that will let you create an unusual animated night sky background effect for your website.



Night Sky Animated Background on Envato Market

3. Pure CSS3 Loader Animations

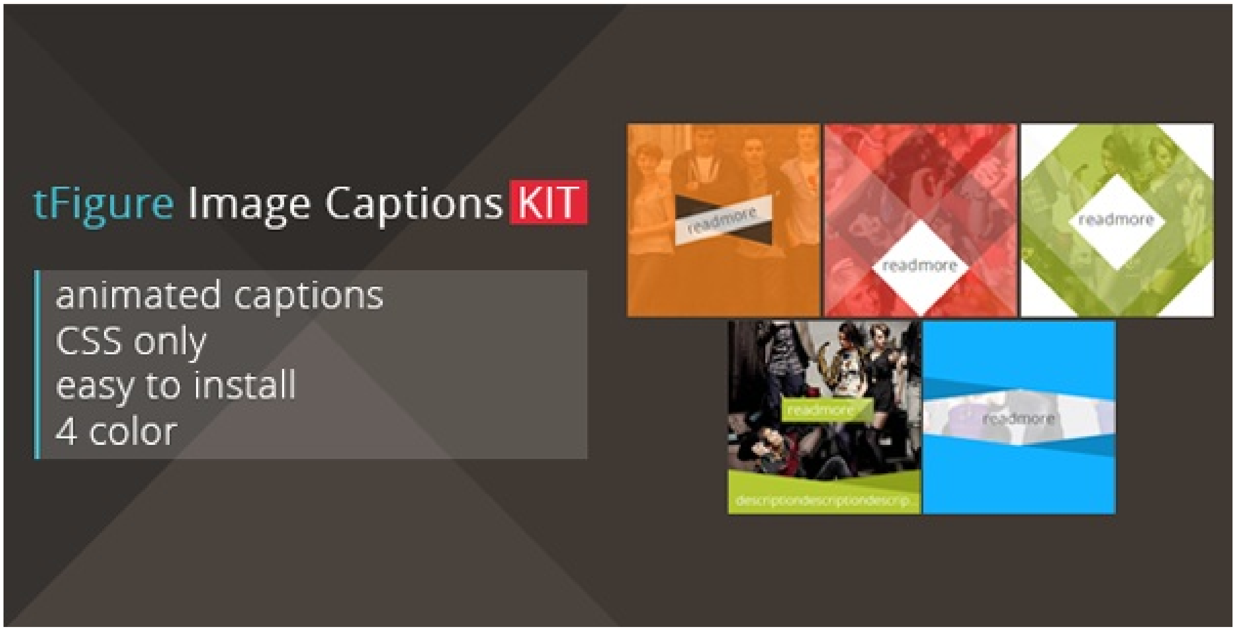
This is a kit of Pure CSS3 Loader Animations. These 15+ CSS3 animation effects are compatible with all major browsers.



Pure CSS3 Loader Animations on Envato Market

4. tFigure - Image Captions

This is a collection of five CSS-only animated Image Captions. This kit offers some great effects and will save you a lot of time when creating websites.



tFigure - Image Captions on Envato Market

5. Social Mosaic Wall

This item lets you build your own mosaic grid with your social media for staff and more. It includes 28 social icons and a range of different grid configurations.

Social Mosaic Wall on Envato Market