CSS3 Transition

Contents

[Main objective 2](#_Toc307994776)

[How it was done in CSS3 3](#_Toc307994777)

[CSS3 Transition definition 3](#_Toc307994778)

[Making the transition works 3](#_Toc307994779)

[Background demo 3](#_Toc307994780)

[HTML 3](#_Toc307994781)

[CSS 4](#_Toc307994782)

[Javascript 5](#_Toc307994783)

[Shims used 8](#_Toc307994784)

[Introduction 8](#_Toc307994785)

[jQuery animate 8](#_Toc307994786)

[Things to shim 8](#_Toc307994787)

[Shims tested 8](#_Toc307994788)

[Improvements 9](#_Toc307994789)

[Final approach 10](#_Toc307994790)

[Animation process 10](#_Toc307994791)

[Compatibility and browser support 10](#_Toc307994792)

[Performance 10](#_Toc307994793)

[Image Loader 11](#_Toc307994794)

[Defining the url to Load 11](#_Toc307994795)

[PreLoader 12](#_Toc307994796)

# Main objective

The main objective of the project was a research of the CSS3 transition capabilities and how to use on and real project. CSS3 Transition is a new brand functionality included in the CSS3 W3C definition that allows developer to make transition for DOM properties over the time with an easing definition.

Since we have to test the transition property of the CSS3 we think of making some background animation using DIV. We think in making an opacity animation from 0 to 1 b between to DIV to make an image transition.

1 -> 0.5 -> 0

0 -> 0.5 -> 1

sWith this configuration of the animation process the two images blends in one moment and then the first one disappear and the second one appear.

The other usage for the CSS3 transition property is moving a DIV with a background image across the page. For this experience we use some character from the Plain Concepts game’s La Cura making then appear on the stage with an animation and the disappear.

# How it was done in CSS3

## CSS3 Transition definition

Before I start on how we implement this demo we have to talk about the usage of the CSS3 Transition property.

transition: all 1s ease-in-out;

This is the compact definition with include the property name, in this case all properties, the time value and the easing function.

Developers can define those properties individually like this:

transition-property: opacity;

transition-duration: 1s;

transition-timing-function: ease-in-out;

There is no different between the first definition and the second one.

## Making the transition works

To actually create an animation with the CSS3 Transition property there are several ways, using the CSS selector hover, blur, focus or by setting the opacity property to the applied object using Javascript or jQuery.

So if we apply the hover selector it’s should be like this:

.transitionDemo:hover

{

opacity: 0;

}

## Background demo

How the background demo was developed. We’re going to talk first about the html / css and then talk about the code and how we bind all the experience.

### HTML

<body>

    <div id="imageBackgroundNext" class="opacity0">

    </div>

    <div id="imageBackground">

    </div>

     <div class="clipping">

        <div id="character" class="characterBase">

        </div>

    </div>

    <div id="loadingProgress" class="loadingProgress transition">

    </div>

</body>

There are two DIV called imageBackground and imageBackgroundNext witch are used for the fade-in face-out opacity transition. There is also a DIV called ‘clipping’ used for clip the content of the children, which is the characterDIV, and the last DIV is the progress bar for the loading phrase of the demo.

### CSS

#imageBackground

{

background: url('../Images/backgrounds/cap01\_jef01\_lalengua.jpg');

background-size: auto;

background-repeat: no-repeat;

top: 0;

left: 0;

width: 760px;

height: 460px;

position: absolute;

}

.transition

{

-webkit-transition: opacity 1s ease-in-out;

-ms-transition: opacity 1s ease-in-out;

-moz-transition: opacity 1s ease-in-out;

transition-property: opacity;

transition-duration: 1s;

transition-timing-function: ease-in-out;

}

.clipping

{

clip: rect(0px, 760px, 460px, 0px);

top: 0;

left: 0;

right: 0;

bottom: 0;

width: 860px;

height: 560px;

position: absolute;

overflow: hidden;

}

.characterBase

{

background: url('../Images/characters/cientifica\_joven\_a.png');

background-repeat: no-repeat;

background-size: auto;

top: -50;

left: -800;

right: 0;

bottom: 0;

width: 860px;

height: 560px;

position: absolute;

}

.characterTransition

{

-webkit-transition: all 1s ease-in-out;

-ms-transition: all 1s ease-in-out;

-moz-transition: all 1s ease-in-out;

transition: all 1s ease-in-out;

-webkit-transform: all 1s scale(1.4);

-ms-transform: all 1s scale(1.4);

-moz-transform: all 1s scale(1.4);

transform: all 1s scale(1.4);

}

.loadingProgress

{

background-color: #FF0000;

border: thin solid #000000;

width: 100px;

height: 10px;

position: absolute;

}

There is nothing special in the css rather than the transition class. This class is used for apply the transition property to the DOM object. The good think about this is since is a class I can add or remove this class to an object to animate the transition or not, this enable scenarios where you want to set the opacity to 0 without animating it.

The other special consideration is that Transition is a new CSS3 property that is not defined in some browser, like IE9 for example. For this approach there are features in CSS that enable some vendor specific prefixes in order to enable non-standard CSS properties. For that reason we apply three vendors prefix, -webkit-, -ms-, -moz-.

### JavaScript

Now let’s talk about how the different parts of the demo were implemented. There are two main parts the background and the character.

#### Background

The code for the background is simple:

function changeNextBackground() {

    var background = $('#imageBackground');

    background.addClassCustom('transition');

    animateProperty(background, 'opacity', 0);

    var url = getNextUrl();

    currentUrl = url;

    var backgroundNext = $('#imageBackgroundNext');

    backgroundNext.css('background', 'url("' + url + '")');

    backgroundNext.addClassCustom('transition');

    backgroundNext.bind(getTransitionEndEvent(), onBackgroundTransitionCompleted);

    animateProperty(backgroundNext, 'opacity', 1, onBackgroundTransitionCompleted);

}

We add the ‘transition’ class to the two DIV involved, then we set the opacity to the background to 0, and for the backgroundNext to 1. When the transition is over we set the background with the same image url and set the background to 1 to reset the state to start the animation again.

function onBackgroundTransitionCompleted() {

    var background = $('#imageBackground');

    background.removeClassCustom('transition');

    background.css('background', 'url("' + currentUrl + '")');

    animateProperty(background, 'opacity', 1);

    var backgroundNext = $('#imageBackgroundNext');

    backgroundNext.removeClassCustom('transition');

    animateProperty(backgroundNext, 'opacity', 0);

    backgroundNext.addClassCustom('transition');

    $(this).unbind(getTransitionEndEvent(), onBackgroundTransitionCompleted);

    setTimeout(changeNextBackground, 1400);

}

A timeout function starts the process again in 1.4 seconds.

The function onBackgroundTransitionCompleted is called when the animation is over, to do that we need to subscribe to the event transitionEnd but since this CSS3 transition technology is still in draft we need to use the vendor specific tag. There is a function on the code to select the correct name for that.

function getTransitionEndEvent() {

    var vP = "";

    var transitionEnd = "transitionEnd";

    if ($.browser.webkit) {

        vP = "-webkit-";

        transitionEnd = "webkitTransitionEnd";

    } else if ($.browser.msie) {

        vP = "-ms-";

        transitionEnd = "MSTransitionEnd";

    } else if ($.browser.mozilla) {

        vP = "-moz-";

        transitionEnd = "transitionend";

    } else if ($.browser.opera) {

        vP = "-o-";

        transitionEnd = "oTransitionEnd";

    }

    return transitionEnd;

}

Having the transition property on a separate class enable scenario where you want to set the opacity to an element without animating it, so simply remove the class and the set the property with jQuery.

#### Characters

The character animation is the same implementation like the background. But in this case there are a two-step animation, the first one is animating the character from the left to the center of the viewport and then wait and then move the character to the right.

First step:

function animateCharacter() {

    completeType = 1;

    var character = $('#character');

    character.removeClassCustom('characterTransition');

    animateProperty(character, 'left', -800);

    var url = getCharacterNextUrl();

    character.css('background', 'url("' + url + '")');

    character.addClassCustom("characterTransition");

    animateProperty(character, 'left', -100, onCharacterTransitionCompleted);

}

When the transition is over the function onCharacterTransitionCompleted is executed, in this function we check the state of the animation:

function onCharacterTransitionCompleted(e) {

    if (completeType == 1) {

        setTimeout(continueAnimation, 1000 \* 2);

    }

    else if (completeType == 2) {

        setTimeout(animateCharacter, 1000 \* 3);

    }

}

If the animation is in the first phrase we set a timeout of 2 seconds to the function continueAnimation:

function continueAnimation() {

    var character = $('#character');

    animateProperty(character, 'left', 800, onCharacterTransitionCompleted);

    completeType = 2;

}

When this function finish we start the process again.

# Shims used

## Introduction

Shims are some pieces of code that are used to supply some missing part in the development process. So here we’re going to talk how we can do CSS3 Transition on a browser that doesn’t support CSS3 transitions.

## jQuery animate

For the animation support we’re going to use the animate function of the jQuery plug-in, this is not the 100% of the support of the CSS3 Transition but at least we have support to simple animations.

## Things to shim

There are two things on the CSS3 Transition implementation that we need to provide. The most obvious is the animation support itself, but the other part is the ‘transitionEnd’ event. So we need to find a way to be notified back when a jQuery animation is over.

## Shims tested

We test two shims, jQuery.Transions[[1]](#footnote-1) and transitions[[2]](#footnote-2) both of them have the same approach to enable Transition support in IE9.

They read the CSS Transition property separately and then bind events with the hover, blur and focus events, which means that those shims only works if you use the hover css selector. Of course those shims don’t have the ‘transitionEnd’ event and you can’t listen of the completion of the transition. Basically they read the CSS transition events and create jQuery animation for animating the properties.

Both shims cover different parts of the implementation but they are not a real option to use on a real project. My final approach was not use those shims and create a helpers functions to enable the CSS transition in modern browsers and jQuery animation in IE9.

Let’s see an example of this.

$.prototype.addClassCustom = function (value) {

    if (Modernizr.csstransitions) {

        this.addClass(value);

    }

    else {

        this.data('animation', true);

    }

};

$.prototype.removeClassCustom = function (value) {

    if (Modernizr.csstransitions) {

        this.removeClass(value);

    }

    else {

        this.data('animation', null);

    }

};

Those are the addClass and removeClass function from the jQuery plugin. If the browser support CSS we add the class, if not then add this animation value to true, to know that we want to animate properties. We didn’t add the CSS transition properties in IE9 to prevent a race condition on the browser.

function animateProperty(element, propertyName, value, completed) {

    if (!Modernizr.csstransitions) {

        if ($(element).data('animation') == true) {

            var properties = '{"' + propertyName + '":' + value + '}';

            properties = jQuery.parseJSON(properties);

            $(element).animate(properties, 1000, 'linear', completed);

        }

        else {

            $(element).css(propertyName, value);

        }

    }

    else {

        $(element).css(propertyName, value);

    }

}

The animation function is query if the browser supports CSS Transition if it does then simply apply the css property and the browser will handle the animation process. If the browser doesn’t support CSS Transtion then we need to check if the animation property is set to true on the object and then animate the property.

## Improvements

This implementation doesn’t read the time value from the CSS transition so it not a real shims, it’s just a function to help creating the transitions, at the end those shims read the CSS transition properties and create the animation based on the hover, blur and focus selector so you can do the same without the shim. In this demo we use Javascript so much, but as I said before CSS Transition is great for simple transition but not for creating complex interaction with DOM objects and Javascript is there to rescue you.

# Final approach

## Animation process

Using the CSS3 transition is a simple process, just simply specify the property you want to animate the time and there you go, animation is now ready to go. The bad thing is that is super hard for designers to create a consistent interaction for the user with this simple tool. Interaction seem to be more complex that a simple opacity animation, and there is no visual editor like Microsoft Expression Blend for HTML5.

Transitions works in the same value as animation is created on Silverlight, when you are animating a property it’s doesn’t matter from witch value are you coming from, just starting the animation from this value to the destination value. But for example in Silverlight we have something called Storyboards. Storyboards are grouped animation of properties that have the same start time. So is super easy to start creating complex animation from the same start point in time with multiple elements.

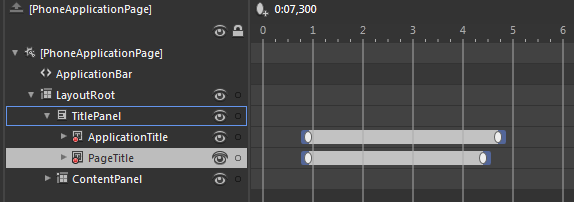


Ilustración 1 Expression Blend with Storyboard view

## Compatibility and browser support

CSS3 transition is part of the standard but still in draft

| IE6.0 | IE7.0 | IE8.0 | IE9.0 | IE10 W8 | Chrome13 | Chrome14 | Safari4.x | Safari5.x | Firefox3.6 | Firefox6.x | Opera11 | WP7WP 7 | WP7.5WP 7.5 | AndroidAnd-roid | Safari iOSiOS 3.1.3 | Safari iOSiOS 4.1.2 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | No | No | No[[3]](#footnote-3) | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | Yes | No | Yes |

## Performance

Performance is important today, even for Web Applications. As my personal experience I found that Internet Explorer is the best on drawing and updating DOM objects, for example IE10 at Windows 8 have the best performance of all browsers using the transition property. The animation are super smooth all the time, but for example with Google Chrome or Firefox sometimes there are some frames that are dropped. I’m talking about simple opacity and movement animation not complex animation with transforms, just moving object around.

# Image Loader

The image loader is a three steps process, the first process is generating the url images to load, the second one is to actually load the images and notify a script to count and process the images, the final step is warp-it all in a loading script and start the animation process.

## Defining the url to Load

There are three groups of images to load, backgrounds, characters and items. To define those urls, there are three functions that fill an array and the call the ‘loadImageArray’.

var imagesValues = new Array();

var characterValues = new Array();

var rainValues = new Array();

var myLoader = new preloader();

$(document).ready(startAnimationEngine);

function startAnimationEngine() {

    var functionsIndex = 0;

    var functions = [];

    functions.push(loadCharacters);

    functions.push(loadItems);

    functions.push(loadMainBackgrounds);

    myLoader.progress = function () {

        var w = $('#loadingProgress');

        w.width(w.width() + 1);

    };

    myLoader.completed = function () {

        functionsIndex++;

        if (functionsIndex < functions.length) {

            functions[functionsIndex]();

        }

        else {

            $('#loadingProgress').remove();

            setTimeout(changeNextBackground, 1000);

            setTimeout(startAnimateCharacter, 1000);

        }

    };

    functions[functionsIndex]();

}

This is the first function executed on the app, when the DOM elements are ready. The function subscribes to the progress and completed event of the preloader object, making the progress bar increase the width in one pixel at time, and when the loader completed the whole process the next function is called. When all the functions are executed we removed the ‘loadingProgress’ object from the DOM and the set to timeout two function, changeNextBackground and startAnimateCharacter.

## PreLoader

The preloader object is responsible for loading content from the server, the first implementation only loads images but can be customized to support more file types.

var preloader = (function (global) {

    var loadedcount = 0;

    var max;

    return function preloader() {

        var instance = this;

        this.loadImageArray = function (array) {

            if (array !== undefined) {

                loadedcount = 0;

                max = array.length;

                for (var i = 0; i < array.length; i++) {

                    var item = array[i];

                    instance.loadImage(item);

                }

            }

        };

        this.loadImage = function (url) {

            var img = new Image();

            img.src = url;

            img.onload = function () { complete(false, this); };

            img.onerror = function (e) { complete(true, this); };

        };

        function complete(error, element) {

            if (error) {

                console.log('Error downloading image ' + element.src);

            }

            loadedcount++;

            instance.progress(element);

            if (loadedcount == max) {

                instance.completed(loadedcount);

            }

        }

    };

})(this);

All the magic occur at the loadImage function. There I create an instance of the Image object and then subscribe to the onload and onerror function to notify me back when the loading is complete, once is completed I check if this item is the last one and it is then fire the completed event.

1. <https://github.com/louisremi/jquery.transition.js> [↑](#footnote-ref-1)
2. <http://browserexperiments.com/shims/Transitions/transitions.js> [↑](#footnote-ref-2)
3. In IE9 there are shims to use [↑](#footnote-ref-3)