

## PRODUCT SPECIFICATIONS

Motion Graphics using jQuery and CSS3

by

**Armagan Tekdoner**

Date: March 2015

Barrie, Ontario, Canada

Web Page: [javascript.grifare.info/BigSwitch/big-switch.html](http://javascript.grifare.info/BigSwitch/big-switch.html)

Official Website: [grifare.info](http://grifare.info)

Introduction	2
Style	3
Functionality	5
List of Files and Directory Structure	11
Pitfalls to Avoid	12

## Introduction

This single-page hand-coded website is a Motion Graphics study that utilises CSS3 animations hand-in-hand with jQuery, on HTML5 platform. May be beneficial for intermediate level web coders, front-end designers, CSS lovers, and client-side technology enthusiasts.

Perhaps this study will not change the axis of rotation of Earth but there are solutions to issues any web developer may come across, during this or that stage of his/her life.

Wheel-reinventing has been avoided while trying to do something unique. This may not be an advanced study for the seasoned professionals but it certainly is not elementary, either.

Although CSS3 transformations are still highly experimental and heavily vendor-dependent in the year of 2015 when this exercise has been made live online, I believe some of them will become standards while many finding their ways in the global recycle bin.

The question is, what the hell some dancing geometric shapes or some rotated text are for, in the first place? Well, we are not obliged to apply everything available to everything. If we keep the number of elements and speed of moving parts at very low levels, subtle transitions may improve the look and feel of websites. It is good to have the technology available but let us use it responsibly.

The entire idea and the logic flow is my own work. No copy-paste job here. Apart from the images used, which are generic images with no copyright restrictions, I have not borrowed any assets. It is intended to be a fun exercise.

To analyse the codes more easily, in addition to checking the source code, you can find all files here: <http://javascript.grifare.info/BigSwitch>

All codes are well-commented. Remember, this is open source, anyone may use anything here.

No conditions apply!

Hope you enjoyed it.

## Style

There are advanced CSS3 tricks.

big-switch.css file I have prepared, begins with importing these 2 third-party files:

```
/* imports a stylesheet for some buttons */  
@import 'shiny-css-buttons.css';  
/* imports a stylesheet for the analogue switch */  
@import 'analogue-switch.css';
```

(Remember to use @import rules before your own CSS code begins!)

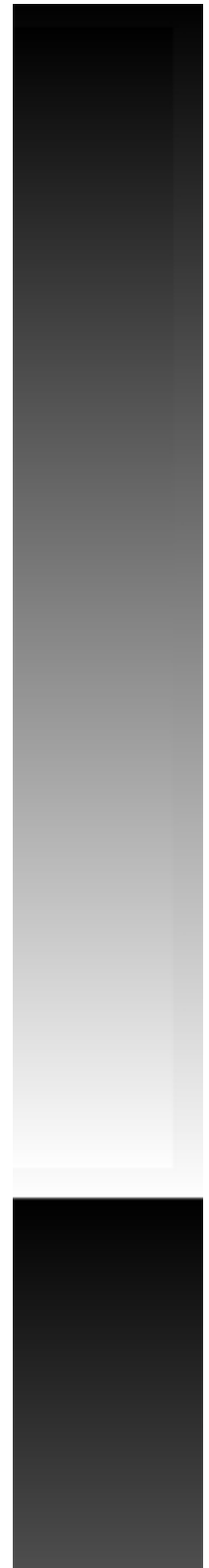
big-switch.css sets all design rules and values that are displayed before any button is clicked. I relied on DOM selectors and the classes or IDs used, belong to third-party files that are imported.

The stage-looking background shown on the right is generated by using Microsoft's service hyperlinked at the bottom of the page.

And is transitioned, when the special-effects.css file is in action.

Many variations are possible not only thanks to the background-color codes, but thanks to other elements styling values, such as positioning, as well. It is worthwhile to note that MS uses "background" not "background-color". If your target screen size will be large, try fixed footers that stick to bottom, to experience more variations.

The special-effects.css file I created, is inserted into the header when the analogue switch is turned on and that contributes to the show significantly. It reveals many hidden elements and contains all animation rules. It overrides the rules as it comes after the main css file. You can find 5 keyframes, a few transitions, and transformations there.



<http://ie.microsoft.com/Testdrive/Graphics/CSSGradientBackgroundMaker/Default.html>

This switch on the right, believe it or not, is NO image: it is a purely CSS creature. It is also a third party plug-in.

The author releases the CSS file and basic HTML codes for use, but with no functionality attached.

Author's URL, where there are other switches:

<http://codepen.io/maturo>

(Caina Maturo, SAO PAULO, BRAZIL)



Including iron background and sketches, there are 11 images in total and they add up to only 232kb.

Typography:

`font-family: "Lucida Console", Monaco, monospace;`

I think Lucida is great in rendering pin-sharp text. No other font was used.

Colour Scheme:

Shades of RED, GREEN, and GREY, as this study is about switches. When the D-day show begins, the violet background shows up to imitate lightening strike. Images have been selected accordingly as well.

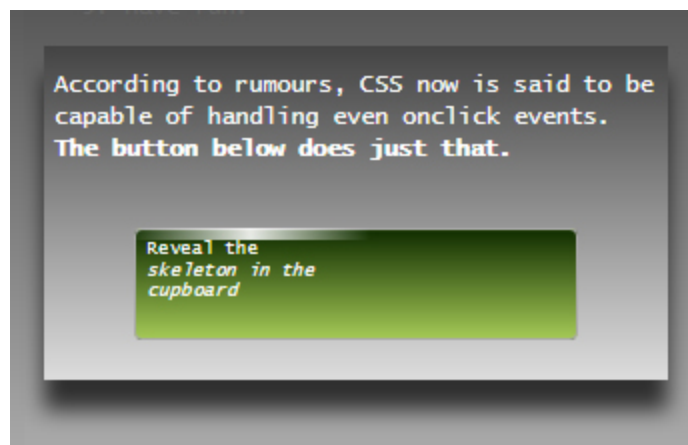
## Functionality

In total, there are 3 buttons a user can turn on and off and there is nothing else to do apart from leaving the page. This is not a user-interaction study.

The URL never changes and anything can be reset by page refresh at anytime, as this study is just a client-side exercise.

Many elements have been styled by using DOM selectors and establishing many classes and IDs thus avoided.

### TRICK 1: Green button: CSS-driven event handling



The GREEN button displayed above (on the left on the screen) is intended as an exercise to demonstrate how CSS3-driven event handling happens.

The button itself (which is a third-party study) and the text effects that come with it, is made up of purely CSS and its source code is available at this URL:

[http://javascript.grifare.info/BigSwitch/css/shiny\\_css\\_buttons.css](http://javascript.grifare.info/BigSwitch/css/shiny_css_buttons.css)

borrowed and adapted from:

<http://www.paulund.co.uk/>

However, the functioning is my job. Here is how it works in plain English.

There is an invisible div beneath it. When that green button is clicked, that div becomes visible, which incorporates the RED button that will hide itself along with whatever that div has, on mouse click.

The keyword to keep in mind here is TARGET.

What TARGET does is:


```
#myElement:target { display: block; }
```

works like

```
<button onclick="#myElement">Click me</button>
```

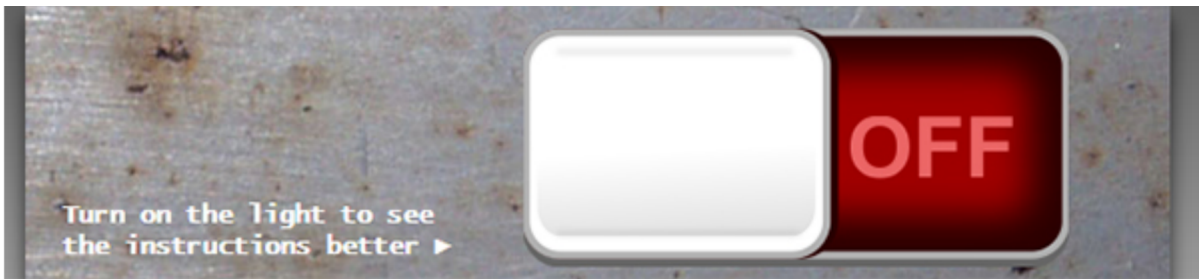
or

```
<a href="#myElement">Go to my element</a>
```

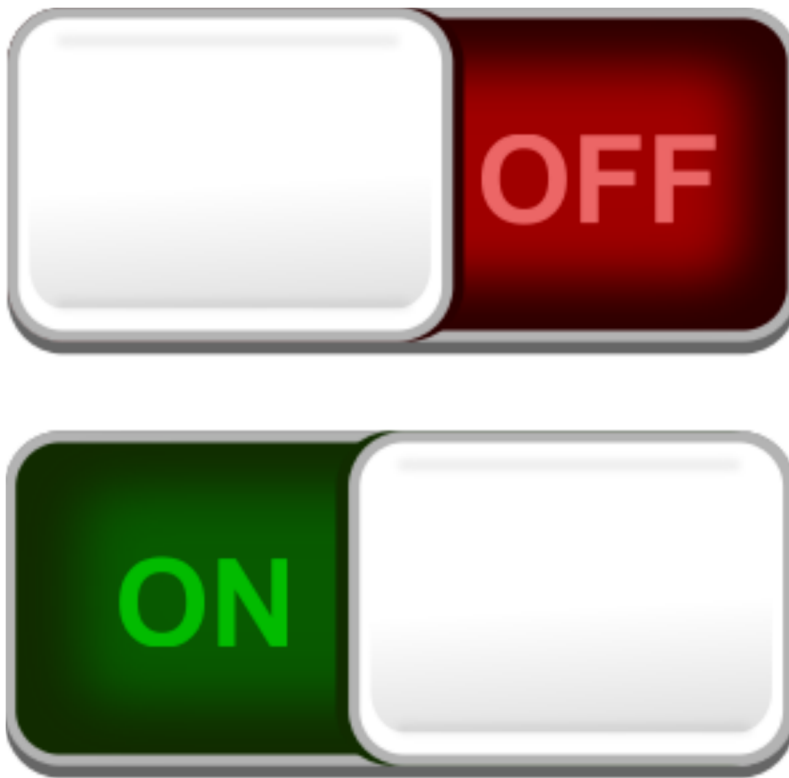
<p>According to rumours, CSS now is said to be capable of handling even onclick events. The button below does just that.</p> <div data-bbox="272 766 686 867"><p>Reveal the skeleton in the cupboard</p></div>  <div data-bbox="272 1255 686 1356"><p>Hide the skeleton in the cupboard</p></div>	<p>The magical codes here are:</p> <pre>/* CSS-driven onclick event trick */ #hidden-secret-layer1{     display: none; }  #hidden-secret-layer1:target{     display: block; }  #hidden-secret-layer2:target{     display: none; }</pre>
---	---

For further analysis, please check both HTML and CSS files.

## TRICK 2: jQuery-driven Switch



This switch above, is made up of the 2 images below and the simple image swap is in action.



When the switch is turned on, 3 actions happen.

- The light bulb image is also swapped with its alit version
- With 130miliseconds of delay, the text colour of instructions changes to pale yellow from grey
- The label pointing at the switch now reads another copy

When the switch is turned off:

- The light bulb image is swapped with its turned-off version
- The text colour of instructions changes back to grey immediately
- The label pointing at the switch switches back to its original copy




Here is what you would find in the JavaScript file:

```
function turnOnTheLight() {
    // displays green label of the switch when the bulb is turned on
    $( "section div:nth-of-type(1)" ).html("<img src='images/button-green.png'
alt='switch' title='light turned on' />");
    //selects the element, swaps the image
    $( "header div" ).html("<img src='images/lightbulb2.png' alt='light bulb
alit' title='light turned on' />");
    //selects the element, changes the font colour with delay
    $('ol')
        .delay(130)
        .queue( function(next){
            $(this).css('color', '#f8f19d');
            next();
        });
    // displays a different message when the bulb is turned on
    $( "section h3:nth-of-type(1)" ).html("Turn off the light to save <br />the
planet &#9658;");
}
```

Make it fun.

## INSTRUCTIONS

1. Build a simple HTML page that uses a few elements that could be used to build a switch.
2. Style the switch using CSS so that it looks convincing.
3. Incorporate some sort of CSS transform, transition effect or a simple CSS animation.
4. Employ one or more jQuery effects to some aspect of the user interface.
5. Have fun.



### TRICK 3: Analogue Switch, which does the majority of the show, using both jQuery and CSS3



My D-day show incorporates many jQuery and CSS tricks, which are my own ideas and works.

Here is the drive-thru synopsis of the approximately 50-second show, along with explanations about how it happens. The events start once the switch is turned on.

What	How	When
Background changes	Jquery appends another file called special-effects.css to header, and the current background definition is overridden by the new CSS file.	Immediately
Label that begins with "Turn on..." changes.	jQuery changes the text using .text method.	Immediately
Function call	doomsday function is called	Immediately
Lightening1 shown: iron background is replaced with the lightening photo.	jQuery changes the iron background using .css method into the centre.	After 800 ms
Lightening1 removed: lightening photo background is replaced with the background colour.	jQuery removes the background image using .css method.	After 1100 ms
Lightening2 shown: same lightening photo is displayed in broader background.	jQuery inserts the same lightening photo using .css method, in repeat mode, horizontally.	After 1900 ms
Space image shows up where lightening 1 stroke.	jQuery inserts space photo using .css method, into the element where the iron background was first displayed.	After 2200 ms
Lightening2 removed: lightening photo background is replaced with the background colour.	jQuery removes the background image using .css method.	After 2300 ms

What	How	When
Skeletons enter, travelling from upper left to lower right.	jQuery inserts skeletons sketch using .css method. The positioning and timing makes them move from left to right.	After 2500 ms
Red text appears that says "you opened a hole".	jQuery displays the previously hidden span using .css method, converting it to display:block	After 3500 ms
Skeletons removed.	jQuery removes skeletons sketch using .css method.	After 4500 ms
Switches removed and their labels changed.	jQuery removes 2 elements using .css and .empty methods, changes messages using .text method.	After 9000 ms
Light bulb is alit again, instructions become "you are in trouble" warnings, stage background colour goes black.	jQuery removes h2 and h3 elements using .css method, changes messages and swaps image using .text and .html methods, changes background to black and font-color to pale yellow using .css method.	After 13
Header and footer removed, light bulb removed, everywhere goes black, "you are in trouble" warnings repeated.	jQuery removes elements using .css method, a for loop posts the warning 300 times using .append method.	After 18 s
Everything is removed, rotating sky begins.	jQuery removes elements using .css and .remove methods, another div is made visible changing its CSS property from display:none to display:block from which sky image is seen. It rotates using CSS animations.	After 23 s
Rotating sky fades out, scary skeleton fades in.	Using jQuery methods of .fadeIn .fadeOut and .attr image swap is realised.	After 30 s
The window through which the sky is seen, widens.	jQuery changes width and height properties using .css method.	After 38 s
Scary skeleton becomes background	jQuery removes everything using .empty method and inserts the skeleton using .css method.	After 45 s
Show ends	jQuery adds a "Thank you" text 10 times using .append method.	After 48 s
Back to square one	Page refresh happens: location.reload	After 50 s

Turning off the switch does not pause: it reloads the page.

## List of Files and Directory Structure

File Name	Directory	Source
big-switch.html	root	Armagan Tekdoner
big-switch.css	css	Armagan Tekdoner
special-effects.css	css	Armagan Tekdoner
analogue-switch.css	css	Caina Maturo
shiny-css-buttons.css	css	Paul Underwood
big-switch.js	js	Armagan Tekdoner
jquery-1.11.2.min.js	js	jquery.com
button-green.png	images	Georgian College
button-red.png	images	Georgian College
lightbulb1.png	images	<a href="https://lh4.googleusercontent.com/-katLGTSCm2Q/UJC0_N7XCrI/AAAAAAAAABq0/6GxNfNW-Ra4/s300">https://lh4.googleusercontent.com/-katLGTSCm2Q/UJC0_N7XCrI/AAAAAAAAABq0/6GxNfNW-Ra4/s300</a>
lightbulb2.png	images	
lightening-strike.jpg	images	<a href="http://gallsources.com">http://gallsources.com</a>
metal-background.jpg	images	<a href="http://textures8.com">http://textures8.com</a>
scary-skeleton.jpg	images	<a href="http://freewallpapersbackgrounds.com">http://freewallpapersbackgrounds.com</a>
skeleton-in-closet.jpg	images	<a href="https://melanieyeyocarter.wordpress.com">https://melanieyeyocarter.wordpress.com</a>
skeletons-dancing.png	images	<a href="http://www.wpclipart.com">http://www.wpclipart.com</a>
space.jpg	images	<a href="http://www.azdeepskies.com">http://www.azdeepskies.com</a>
space-hole.jpg	images	<a href="http://www.azdeepskies.com/">http://www.azdeepskies.com/</a>

## Pitfalls to Avoid

A small list of troubles I identified that might be helpful for those who are not industry professionals.

### CSS

- `@import` rules cannot be used @ our heart's content: It should be placed before our CSS codes begin.
- CSS animations are very unreliable, versions, vendors... and wrong timings.

### jQuery and JavaScript

- Dynamically inserted elements by jQuery, cannot be selected using DOM selectors, as they do not exist in the DOM structure.

Let us say we insert a div as the second div of the page:

```
$( 'body' ).html( '<div>blah blah</div>' );
```

But then, this will NOT work:

```
$( 'div:nth-of-type(2)' ).css( 'color','red' );
```

- Dynamically inserted images by jQuery, should have the relative paths to the file into which they are being inserted, NOT to that of the .js file in use.

Consider this directory structure:

index.html

myImage.jpg

scripts/script.js

The jQuery code inside the script.js file, as if it is inside index.html, should be something like this:

```
$(' body ').html( '' );
```

and NOT

```
$(' body ').html( '' );
```

- Do not forget to place your code **INSIDE** this function:

```
$(document).ready(function() {  
    $some code  
}); // close doc dot ready
```

- Click counters. If you declare a counter inside a function that needs to use the number of clicks, the counter will reset itself at each click and it will never count!

Here is my suggestion:

```
var clickCounter = 0;
$(document).ready(function() {
    $( "section" ).click(function() {
        // check if counter is even
        if ( clickCounter % 2 == 0 ){
            function1();
        }
        // check if counter is odd
        if (clickCounter % 2 == 1 ){
            function2();
        }
        // increment the counter
        clickCounter++;
    }); // close click function
}); // close doc dot ready
```