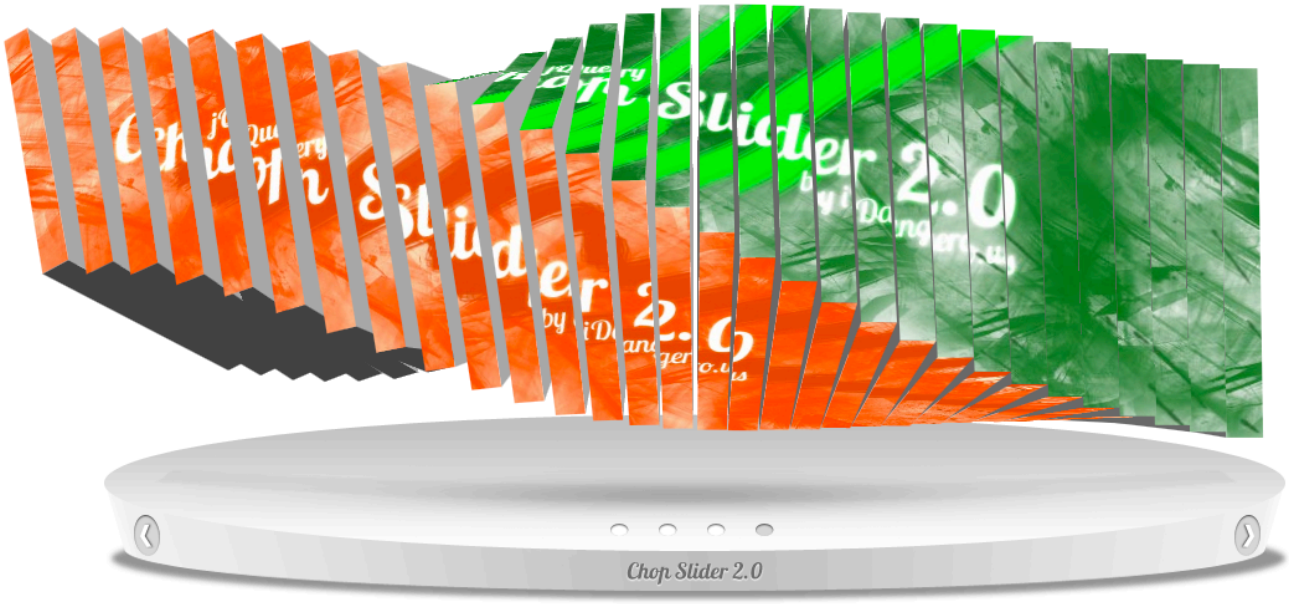
jQuery Chop Slider 2.0

by iDangero.us

The only limitation is your imagination!



Usage Documentation

For Version 2.2.0_Free

By Vladimir Kharlampidi
@nolimits4web
The iDangero.us
http://www.idangero.us/

Chop Slider v2
@chopslider
http://www.idangero.us/cs/

Table of Contents

1. Chop Slider

1.1. Slider Integration (HTML and CSS)

It is very easy to integrate Chop Slider into your page. So let's first of all look at the HTML code we need to use:

First of all you need to attach the jQuery library and Chop Slider scripts to your document. Add the following code to the HEAD section or in the end of the BODY section:

```
<script src="pathTojQuery/jquery-1.6.1.min.js"></script>
<script src="pathToChopSlider/jquery.id.chopslider-2.0.0.min.js"></script>
```

If you want to use Transition's library then we also need to attach it:

```
<script src="pathToLibrary/jquery.id.cstransitions-1.0.min.js"></script>
```

Then insert the following code inside of BODY:

```
<a id="slide-next" href="#"></a>
<a id="slide-prev" href="#"></a>
<div id="slider">
  <div class="slide cs-activeSlide"> <img src="1.jpg" /> </div>
  <!-- Slide with Link -->
  <div class="slide"> <a href="http://www.idangero.us"><img src="2.jpg" /></a> </
div>
  <div class="slide"> <img src="3.jpg" /> </div>
</div>
<!-- If you want to use Captions -->
<div class="slide-captions">
  <div class="sl-descr">
    <!-- You can use any HTML content inside of caption-->
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
  </div>
  <div class="sl-descr">
    <p>Caption text for second slide</p>
  </div>
  <!-- Empty caption-->
  <div class="sl-descr"></div>
</div>
<div class="caption"></div>
<!-- If you want to use Pagination -->
<div class="pagination">
  <span class="slider-pagination"></span>
  <span class="slider-pagination"></span>
  <span class="slider-pagination"></span>
</div>
```

At the example above we used slider with three images. As you can see the code is pretty simple.

Slider Navigation:

- Links with a "slide-next" and "slide-prev" ID are used here as a slider triggers to navigate between slides.

Slides:

- Every slide's image (IMG) must be wrapped with a DIV with a "slide" class, because Slider will operate with this divs
- If you want to use slide as a hyper-link all you need is to wrap the IMG with an <A> tag inside of "slide" DIV

All divs with a "slide" class wrapped with a one DIV with a "slider" id attribute. This is the main Slider container. We will use it for Slider initialization. You can stylize this element as you wish, wrap it with any additional elements you may need, but do not put anything inside except slides.

- First slide must have an additional "**cs-activeSlide**" class!

Captions:

- As you can see all captions for slides are in the "slide-captions" div. Caption's text for every slide is in the "sl-descr" div. These elements must be hidden. You can use any HTML content inside of each caption.
- Please note that the index number of caption must match to the same index number of slide
- If you want to use some slide without caption, just leave the "sl-descr" div empty
- All captions will be shown in the div with "caption" class. So to stylize captions you need to style only this element. And it is must be hidden by default.
- If you don't need want to use captions, just remove all appropriate elements used here for captions

Pagination:

- In the example above every caption is the empty SPAN element (but you can use any other element such a DIV, A, IMG, etc) with a "slider-pagination" class.
- All these pagination "buttons" wrapped with DIV with a "pagination" class
- The index number of pagination button must match to the same index number of slide. So the first pagination element will be for the first slide, last for the last slide, etc..
- If you do not want to use pagination just remove appropriate elements

In the example above you can use your own classes and id attributes, you will be able to set them on the Slider's initialization. Only two classes are reserved and required: "cs-activeSlide" and "cs-active-pagination" (if you are using pagination)

Now, let's look at the CSS code

```
#slide-prev {
      /* Any properties you need to stylize your Prev button */
}
#slide-next {
      /* Any properties you need to stylize your Next button */

}
#slider {
      /* Set the Slider's width, height and position to relative */
      width:900px;
      height:300px;
      position:relative;
      ........
}
/* Every slide must be hidden by default */
.slide { display:none; }
/*
   And only the slide with "cs-activeSlide" class must be visible!
   !Reserved Class name. Required class!
*/
.slide.activeSlide { display:block; }
/* Container with caption's text. Must be hidden */
.slide-descriptions {
```

```
      display:none;
}

/* Container with caption, must be hidden by default */
.caption {
    /* Style of container with Caption */
}

/* Container with Pagination buttons */
.pagination {
    /* Any properties you may need to stylize it */
}
.slider-pagination {
    /* Any properties you may need to stylize single pagination button */
}
/* This is a required class for the "Active" pagination button. ! Reserved Class
Name ! */
.cs-active-pagination {
      /* Different style for the Active pagination button */
}
```

That is all about required HTML and CSS code. As you can see it's very simple

1.2. Slider Initialization (Java Script)

We need to initialize the Chop Slider after the document is ready (is loaded). Use the following formatting inside of <script> tag (if you use it inside of your document file)

```
jQuery(function(){
      $("#slider").chopSlider(parameters)
})
```

As you can see it was pretty simple! Now let's look at the list of available parameters:

Parameter	Type of variable	Example	Required	Default	Description
Z-Index of the Slider on the time of animation					
loaderIndex	*number*	`18000`	optional	`20000`	z-index CSS property for the Slider Loader
CSS Class of the single "Slide"					
slide	*string*	`".slide"`	**required**	-	Selector of the single slide DIV
Controllers					
nextTrigger	*string*	`"a#slideNext"`	optional	-	Selector of the "Slide Next" button
prevTrigger	*string*	`"a#slidePrev"`	optional	-	Selector of the "Slide Prev" button

Parameter	Type of variable	Example	Required	Default	Description
hideTriggers	*boolean*	`true`	optional	`true`	If true, then the navigation buttons will be hidden in the beginning of transition and shown after transition is complete
sliderPagination	*String*	`".slider-pagination"`	optional	`-`	If true, then the navigation buttons will be hidden in the beginning of transition and shown after transition is complete
hidePagination	*boolean*	`true`	optional	`true`	If true, then the container with pagination buttons will be hidden in the beginning of transition and shown after transition is complete
Captions					
useCaptions	*boolean*	`true`	optional	`false`	should the slider to handle the slide's captions
everyCaptionIn	*string*	`".sl-descr"`	optional	`-`	selector of the container with every caption inside of it
showCaptionIn	*string*	`".caption"`	optional	`-`	selector of the container which will be used to show slide's captions

Parameter	Type of variable	Example	Required	Default	Description
hideCaptions	*boolean*	`false`	optional	`true`	If true, then the Caption's container will be hidden in the beginning of transition and shown after transition is complete
captionTransform	*string*	`"rotate(45deg)"`	optional	-	initial CSS3 Transform property of the caption before it will appear
Autoplay					
autoplay	*boolean*	`true`	optional	`false`	use auto play or not
autoplayDelay	*number*	`6000`	optional	`10000`	delay between slide transitions
Call Back Functions					
onStart	*function*	`function() {alert("Hello")}`	optional	-	custom function in the beginning of transition
onEnd	*function*	`function() {alert("Hello")}`	optional	-	custom function after the transition is complete
Debugging					
disableCSS3	*boolean*	`false`	optional	`false`	disable or enable CSS3 effects. If your transitions do not use rotation, translates and scale effects, it is better to disable it (set to "true")
Full 3D Mode					

Parameter	Type of variable	Example	Required	Default	Description
full3D	*string*	`"rotateY(40deg) rotateX(10deg)"`	optional	-	initial CSS3 3D Transform for the Slide's container
Specifying Transitions					
defaultParameters	*object*	`{ hPieces:2, vPieces:2,...}`	optional	See the default transition parameters in the next chapter	The default options for all following transitions. For example, if you will not specify some parameter in t2D transition, it will be get from the defaultParameters
t2D	*object*	`{ hPieces:2, vPieces:2,...}`	required	-	Parameters for the default 2D transition. This is the main an most common transition.
t3D	*object*	`{ hPieces:2, vPieces:2,...}`	optional	-	Parameters for the 3D Transition. It will be used if the browsers supports 3D Transforms
mobile	*object*	`{ hPieces:2, vPieces:2}`	optional	-	object with a parameters, that will be used for mobile devices
noCSS3	*object*	`{scaleX:1, scaleY:1}`	optional	-	object with a parameters, that will be used for old browsers without CSS3 transitions

Parameter	Type of variable	Example	Required	Default	Description
forceParameters	*object*	`{ hPieces:2, vPieces:2}`	optional	-	If you want to force some options for all types of transitions you need to specify theme here.

1.3. 2D Transitions Parameters

Parameter	Type of variable	Example	Required	Default	Description
type	*string*	`"multi"`	optional	`"vertical"`	type of transition, can be "**vertical**", "**horizontal**", **"slide"** or "**multi**"
hPieces	*number*	`5`	optional	`10`	amount of horizontal pieces. Used in "horizontal" and "multi" transitions
vPieces	*number*	`5`	optional	`10`	amount of vertical pieces. Used in "vertical" and "multi" transitions
xOffset	*number or string (for "random")*	`50` or `"random"`	optional	`0`	horizontal offset (in px) of slider's pieces, or how strong it will be "stretched" horizontally

Parameter	Type of variable	Example	Required	Default	Description
yOffset	*number or string (for "random")*	`-50` or `"random"`	optional	`0`	vertical offset (in px) of slider's pieces, or how strong it will be "stretched" vertically
rotate	*number or string (for "random")*	`50` or `"random"`	optional	`0`	rotate angle of every piece (in degrees)
rotateSymmetric	*boolean*	`true`	optional	`true`	if true, then all pieces will be rotated to the same angle specified in "rotate" parameter. If false, then all pieces will be rotated from -rotate/2 to +rotate/2
scaleX	*number or string (for "random")*	`1.1` or `"random"`	optional	`1`	horizontal scaling for every piece
scaleY	*number or string (for "random")*	`-0.3` or `"random"`	optional	`1`	vertical scaling for every piece
translateX	*number or string (for "random")*	`100` or `"random"`	optional	`0`	horizontal translate (offset) in px for each piece
translateY	*number or string (for "random")*	`-50` or `"random"`	optional	`0`	vertical translate (offset) in px for each piece
ease1	*string*	`"ease-in"`	optional	`"ease"`	Easing function for the First Part of animation

Parameter	Type of variable	Example	Required	Default	Description
ease2	*string*	`"ease-out"`	optional	`"ease"`	Easing function for the Third Part of animation
origin	*string*	`"left top"`	optional	`"50% 50%"`	Transform origin CSS3 property for each piece
dur1	*number*	`1200`	optional	`1000`	duration in ms while the each piece will reach its new position(will get its new transformation) **First part of**
dur2	*number*	`300`	optional	`600`	after the first part, each piece will fade to the next(or prev) slide's same piece for this duration (in ms) **Second part**
dur3	*number*	`900`	optional	`1000`	and the last part - duration (in ms) while the new piece with the image of new slide will get into initial position. In most cases must be the same as **dur1** **Third part**
pieceDelay	*number*	`80`	optional	`50`	delay (in ms) between pieces animation

Parameter	Type of variable	Example	Required	Default	Description
xFadeDelay	*number*	`600`	optional	`300`	delay for each piece between First and Second parts of animation. This value allows you to hold (or to stop for a while) transition after the First part.
startFrom	*number or string (for "random")*	`13` or `"random"`	optional	`0`	Number of first slice in transition queue. Transition will start from this piece. For example, if you have 100 pieces and you will set this option to 100, then transition will start from the last piece. You will get "reverse" transition

Parameter	Type of variable	Example	Required	Default	Description
multiDelay	*string*	`"linear"` or `"random"`	optional	`"progressive"`	Type of delay for "Multi" transition types. Can be "linear", "progressive", "horizontal" or "vertical". With "progressive" type slices will be animated in diagonally way, in "linear" - slice-by-slice. In "horizontal" - column by column. In "vertical" - row by row
borderRadius	*number*	`100`	optional	`0`	Border radius for the slices with rounded corners. "Sexy" transitions are made with this option
halfTransition	*boolean*	`true`	optional	`false`	Set this option to **true** if you want to make "Half" Transition
halfTransitionOpacity	*number*	`0.3`	optional	`1`	Slice opacity in the end of Half transition. The slice's opacity will be animated from 1 to this value. Must be from 0 to 1

Parameter	Type of variable	Example	Required	Default	Description
prevTransition	*object*	`{rotate:-10}`	optional	-	object with a parameters, that will be used for Prev transition (when you click on the Prev button). If not specified, then both Next and Prev transitions will be the same

Important Notes:

- please note that you can use **negative values** for the following parameters : xOffset, yOffset, rotate, scaleX, scaleY, translateX, translateY, origin
- to learn more **rotate, scale, translate** values and how do they work please visit this site: http://www.w3.org/TR/css3-2d-transforms/#transform-functions
- to learn how the **origin** works look here: http://www.w3.org/TR/css3-2d-transforms/#transform-origin-property
- to learn how the **ease** works and what values it can accept look here: http://dev.w3.org/csswg/css3-transitions/#transition-timing-function_tag
- it is not recommended to use **more than 100 pieces** in one transition, it could work very slow
- if you are using transition with 0 value of scaleX or scaleY, than you probably do not need the Second Part of animation, in this case set the **dur2** property to "1" ms

Example with parameters

Ok, let's look at the Slider initialization example again but with parameters:

```
jQuery(function(){
      $("#slider").chopSlider({
            /* Slide Element */
            slide : ".slide",
            /* Controlers */
            nextTrigger : "a#slide-next",
            prevTrigger : "a#slide-prev",
            hideTriggers : true,
            sliderPagination : ".slider-pagination",
            /* Captions */
            useCaptions : true,
            everyCaptionIn : ".sl-descr",
            showCaptionIn : ".caption",
            captionTransform : "scale(0) translate(-600px,0px) rotate(45deg)",
            /* Autoplay */
            autoplay : true,
            autoplayDelay : 6000,
```

```
            /* Default Parameters */
            defaultParameters : {
                  type: "vertical",
                  xOffset: 20,
                  yOffset: 20,
                  hPieces : 10,
                  vPieces: 20,
                  rotate : 10 ,
                  rotateSymmetric: false,
                  scaleX:0.5,
                  scaleY:-0.5,
                  translateX:10,
                  translateY:10,
                  ease1:"ease",
                  ease2:"ease",
                  origin:"center center",
                  dur1: 1000,
                  dur2 :600,
                  dur3: 1000,
                  pieceDelay : 50,
                  xFadeDelay :0,
                  prevTransition :  {
                        rotate:-10,
                        xOffset:10,
                        startFrom:10
                  }
            },
            /* For Mobile Devices */
            mobile: {
                  disableCSS3:true,
                  dur1:1200,
                  dur2:1200,
                  dur3:1200,
                  hPieces:4,
                  vPieces:4,
                  pieceDelay:120,
                  rotate:0,
                  yOffset:0,
                  scaleX:1,
                  scaleY:1
            },
            /* For Old and IE Browsers */
            noCSS3:{
                  dur1:1200,
                  dur2:1200,
                  dur3:1200,
                  hPieces:4,
                  vPieces:4,
                  pieceDelay:120,
                  xFadeDelay :200
            },
            onStart: function(){ /* Do Something*/ },
            onEnd: function(){ /* Do Something*/ }
      })
})
```

That was example with almost all possible parameters. Of course there is no sense to specify parameters that you do not need. For example, if you do not use scaling no need to set scaleX and scaleY parameters.

In the example above you can see different parameters for "previous transition", "mobile" and for browsers without CSS3 transitions. And again do not need to list all parameters here, just specify only those you want to change!

1.4. Understanding of 2D transition and its parameters

Now let's see on screenshots how the main parameters works in different type of transitions

Here is our initial slide:

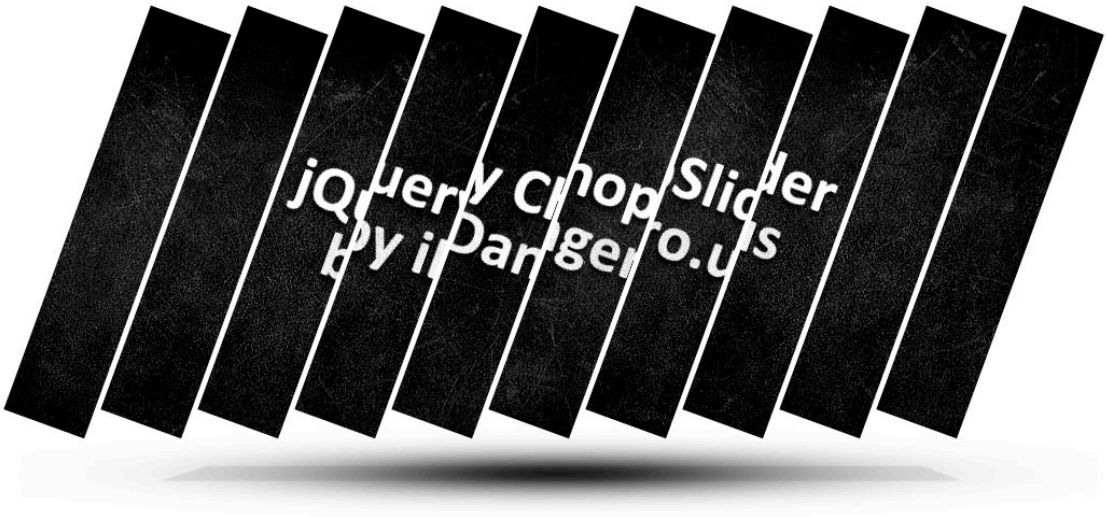
"Vertical" Type

Ok, let's set **xOffset** parameter to 50:

Now we see like the image is "stretched" horizontally. What is **50 px** that we set in **xOffset**? That is mean that now slide's width increased to 100px (on **50 px** from each side!) - the left border of first piece is on 50px to left of its initial position and the right border of the last slide is on 50px to the right from its initial right border.

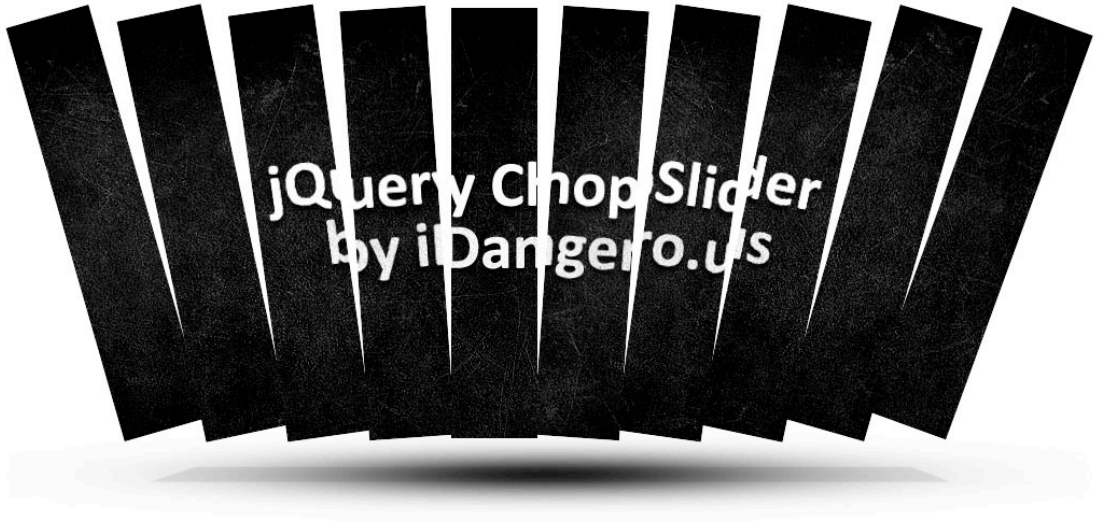
yOffset for verical type work like simple vertical offset. If you set it to 50px, then the slider will just moved down to 50px. If you set it to -50px, then the slider will just moved up to 50px.

Let's set **rotate** parameter to 20 (degrees), **rotateSymmetric** is equal to **true:**



Now you see that every piece has been rotate to 10 degrees.

Let's set **rotateSymmetric** to **false**:



In this case every piece will be rotated from -10 degrees to 10 degrees with a step equal to (rotate*2)/vPieces

Let's set **rotate** to **-10** degrees:

Let's set **xOffset** to **-100** px:

We see some kind of shrunk effect. If you will set the xOffset equal to slide's width but negative, then all pieces will be swapped from (last piece will be the first, etc..)

Not let's set the **scaleX** parameter to **0.5**. In this case width of every piece will reduced by half:

"Horizontal" Type

Let's set **yOffset** to **50** px:

It works in the same way like with **xOffset** for vertical type, but vertically.

Now let's set **rotate** to **10** degrees, **rotateSymmetric** is equal to **true:**

The same but with **rotateSymmetric** equal to **false:**

Like with vertical type, every piece will be rotated from -10 degrees to 10 degrees with a step equal to (**rotate***2)/**hPieces**

Ok, now let's see at the **xOffset**. With horizontal type it works in the same way as with vertical type, but it looks totally different. It will look like skew effect

xOffset is equal to **100** px:

The left border of **last** piece is on 50px to left of its initial position and the right border of the first slide is on 50px to the right from its initial right border. If you will set xOffset to -100px, then you'll have inverted skew effect.

"Slide" Type

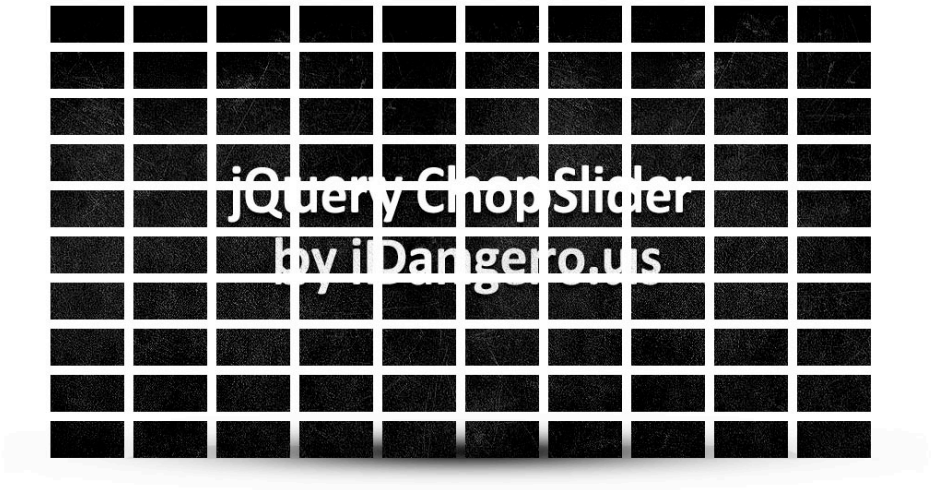
Slide is the most simple type of transition. It works in horizontal way. From parametric options it supports: **pieceDelay**, **vPieces** and **dur1.** Script will automatically detect direction of new slide and depending on this will move slides to the left or to the right. Here is screenshot:

Image will be sliced on the **vPieces** number of slices and animated (slide) for the **dur1** duration.

"Multi" Type

Multi type of transition is like combination of horizontal and vertical type, but looks much more awesome.

Let's set **xOffset** and **yOffset** to 50 px:



rotate to **10** degrees:

rotateSymmetric to **false**:

As you can see it's very easy to use slider's main parameters.

Scaling

If you need to use scaling, use scaleX and scaleY parameters.

E.g. If you will set **scaleX** to -1 and **scaleY** to 1, then you get something like a "horizontal flip" effect.

Translate

Use **translateX** and **translateY** values carefully! For example if you are not use scaling and rotation, then translateX will just move pieces horizontally, and translateY - vertically. But if you will use rotation and some of translate, then **piece will move along the circle!**

Just note that the scale, rotate and translate values are depend on each other.

Transition Parts

As mentioned above every Slider's transition consists of three parts. Let's try to understand every part of this transition. But let's imagine that we have only one piece, not container with a lot of pieces like in real. And we have "initial slide" (current active slide) and "new slide" (it is hidden at the moment). We will have transition from "initial slide" to "new slide":

First Part.

In this part the piece will be animated to its new position (xOffset, yOffset) and its new transformation (rotate, scaleX, scaleY, translateX, translateY) for the time equal to the **dur1** parameter. For piece's transformation (only!, not for position) will be used easing function specified

in the **ease1** parameter. Then it will hold its new position and transformation for the time equal to the **xFadeDelay** parameter. Then second part...

Second Part.

Now the appropriate piece of "**new slide**" will fade in for the time equal to the **dur2** parameter. This "new piece" will already have the same "new position and new transformation". And after that, the "old piece" with image of "**initial slide**" will be removed. Then third part...

Third Part.

Now the "new piece" will be animated to the initial position (with zero offsets) and to the initial transformation (with zero rotate, translate and scales equal to 1) for the time equal to the **dur3** parameter. For new piece transformation (only!, not for position) will be used easing function specified in the **ease2** parameter.

Now about real transition. It works the same with every piece but with a delay specified in **pieceDelay** parameter.

1.5. 3D Transitions Parameters

3D Transition supports all the same options as 2D, and couple of new. Let's look at them:

Parameter	Type of variable	Example	Required	Default	Description
The following parameters will do nothing in 3D Transition: dur2, origin, halfTransition, halfTransitionOpacity, borderRadius					
3D Transitions do not support "random" parameters					
type	**string**	`"vertical3D"`	**required**	–	type of 3D transition, can be "**vertical3D**", "**horizontal3D**", "**multiFlipV**", "**multiFlipH**"
Rotate Angles Rotate angles for 3D transitions work a little bit different than for 2D transitions, so please read carefully. For example "rotate" parameter is not some specific X, Y or Z angle. For every type it is different. Such kind of difference was made to make effects similar to 2D with same rotate option.					
rotate	**number**	`30`	**optional**	`0`	rotate angle of every piece. But not the same as for 2D Transitions. For the "vertical3D" it is Y angle, for the "horizontal3D" it is X angle, for the "multiFlipV" and "multiFlipH" it is Z angle

Parameter	Type of variable	Example	Required	Default	Description
rotateMid	**number**	`90`	**optional**	`0`	rotate angle of the main 3D angle - "middle" angle. For "vertical3D" and "horizontal3D" the value of this option is "-180" in the end of transition. For "multiFlipV" and "multiFlipH" it is "90" in the end of transition.
rotateZ	**number**	`45`	**optional**	`0`	Another new angle value. For the "vertical3D" and "horizontal3D" it is Z angle of slice. For the "multiFlipV" it is Y angle, and for the "multiFlipH" it is X angle. **For the "horizontal3D" and "vertical3D" this angle will be always not symmetric for each piece!**
rotateSymmetric	**boolean**	`true`	**optional**	`true`	if true, then all pieces will be rotated to the same angle specified in "rotate" parameter. If false, then all pieces will be rotated from -rotate/2 to +rotate/2. **For the "multiFlipV" and "multiFlipH" types this parameter also depends on rotateZ option**
translateZ	**number**	`50`	**optional**	`0`	Z translate (offset) in px for each piece
scaleZ	**number**	`1.1`	**optional**	`1`	Z scaling for every piece
showFaces	**boolean**	`true`	**optional**	`true`	If true, then Chop Slider will add 3D back faces to make the 3D effect more realistic

Parameter	Type of variable	Example	Required	Default	Description
faces	**object**	`{ top: "#555", right: "#222", bottom :"#ff0", left: "#333" }`	**optional**	`{top: "#999", right: "#999", bottom :"#333", left: "#555"}`	Background property for back faces. You can even use here images
thickness	**number**	`5`	**optional**	`10`	Thickness of 3d slices in px. Works in **MultiFlip** and in **Sphere** 3D transitions
animateContainer	**string**	`"rotateX(30deg)"`	**optional**	–	initial CSS3 Transform property of the container with animated slides. Container will be animated for the **animateContainerDuration** time and after that will be return back for the **animateContainerDuration2** time. **Time of container animation do not depend on pieceDelay value!**
animateContainerDuration	**number**	`1200`	**optional**	equal to **dur1**	
animateContainerDuration2	**number**	`1200`	**optional**	equal to **dur3**	
containerDelay	**number**	`1000`	**optional**	`0`	If you need to hold animated container for some time, you need to specify it here

Important Notes:

- please note that you can use **negative values** for the following parameters : scaleZ, translateZ, rotateMid, rotate and rotateZ
- to learn more **rotate, scale, translate** values and how do they work please visit this site: http://www.w3.org/TR/css3-2d-transforms/#transform-functions
- to learn how the **origin** works look here: http://www.w3.org/TR/css3-2d-transforms/#transform-origin-property
- to learn how the **ease** works and what values it can accept look here: http://dev.w3.org/csswg/css3-transitions/#transition-timing-function_tag
- it is not recommended to use **more than 100 pieces** in one transition, it could work very slow

Example with parameters

Ok, let's look at the Slider initialization example again but with parameters, but before you should remember on important thing:

When you specify 3D effect, don't forget to specify 2D transition in "**defaultParameters**" or "**t2D**" parameter!!!

We will use example from the previous chapter but now with 3D transition.

```
jQuery(function(){
      $("#slider").chopSlider({
            /* Slide Element */
            slide : ".slide",
            /* Controlers */
            nextTrigger : "a#slide-next",
            prevTrigger : "a#slide-prev",
            hideTriggers : true,
            sliderPagination : ".slider-pagination",
            /* Captions */
            useCaptions : true,
            everyCaptionIn : ".sl-descr",
            showCaptionIn : ".caption",
            captionTransform : "scale(0) translate(-600px,0px) rotate(45deg)",
            /* Autoplay */
            autoplay : true,
            autoplayDelay : 6000,
            /* Default Parameters */
            defaultParameters : {
                  type: "vertical",
                  xOffset: 20,
                  yOffset: 20,
                  hPieces : 10,
                  vPieces: 20,
                  rotate : 10 ,
                  rotateSymmetric: false,
                  scaleX:0.5,
                  scaleY:-0.5,
                  translateX:10,
                  translateY:10,
                  ease1:"ease",
                  ease2:"ease",
                  origin:"center center",
                  dur1: 1000,
                  dur2 :600,
                  dur3: 1000,
                  pieceDelay : 50,
                  xFadeDelay :0,
                  prevTransition :  {
                        rotate:-10,
                        xOffset:10,
                        startFrom:10
                  }
            },
            /* For Browsers that support 3D Transforms */
            t3D: {
                  type: "vertical3D",
                  animateContainer:false,
                  containerDelay:0,
                  xOffset: 0,
```

```
        yOffset: 0,
        vPieces: 30,
        rotate : 0,
        rotateMid : -90,
        rotateZ : 0,
        rotateSymmetric: false,
        scaleX:1,
        scaleY:1,
        scaleZ:1,
        translateX:0,
        translateY:0,
        translateZ:0,
        dur1: 900,
        dur3: 0,
        xFadeDelay: 0 ,
        pieceDelay : 60,
        multiDelay : "linear",
        startFrom:0,
        showFaces:true, //not required. It is true by default
        /* If you want to change faces' background:
        faces : {
              top: "#ff0",
              right: "yellow",
              bottom: "url(images/face-image.png) repeat",
              left: "#000"
        },
        */
        ease1:"ease",
        ease2:"ease",
        halfTransition : false,
        prevTransition : {startFrom:"end"}
        origin:"50% 50%",
        ease1:"ease",
        ease2:"ease",
        halfTransition : false,
        prevTransition : {startFrom:"end"}
},
/* For Mobile Devices */
mobile: {
        disableCSS3:true,
        dur1:1200,
        dur2:1200,
        dur3:1200,
        hPieces:4,
        vPieces:4,
        pieceDelay:120,
        rotate:0,
        yOffset:0,
        scaleX:1,
        scaleY:1
},
/* For Old and IE Browsers */
noCSS3:{
        dur1:1200,
        dur2:1200,
        dur3:1200,
        hPieces:4,
        vPieces:4,
        pieceDelay:120,
        xFadeDelay :200
```

```
            },
            onStart: function(){ /* Do Something*/ },
            onEnd: function(){ /* Do Something*/ }
        })
})
```

That was example with almost all possible parameters. Of course there is no sense to specify parameters that you do not need. For example, if you do not use scaling no need to set scaleX and scaleY parameters.

In the example above you can see different parameters for "previous transition", "mobile" and for browsers without CSS3 transitions. And again do not need to list all parameters here, just specify only those you want to change!

1.6. Understanding of 3D transition and its parameters

3D Transition works in the way similar to 2D Transition. First difference is the Transition Parts. Let's look at them

3D Transition's Parts

Every 3D Transition consists of 2 parts! (There were 3 parts in 2D Transitions). And again let's imagine that we have only one piece, not container with a lot of pieces like in real. And we have "initial slide" (current active slide) and "new slide" (it is hidden at the moment). We will have transition from "initial slide" to "new slide":

First Part.

In this part the piece will be animated to its new position (xOffset, yOffset) and its new transformation (rotate, rotateZ, rotateMid, scaleX, scaleY, scaleZ, translateX, translateY and translateZ) for the time equal to the **dur1** parameter. For piece's transformation and position will be used easing function specified in the **ease1** parameter. Then it will hold its new position and transformation for the time equal to the **xFadeDelay** parameter. Then second part...

Second Part.

Now the appropriate piece of "**new slide**" will fade in for the time equal to the **dur2** parameter. This "new piece" will already have the same "new position and new transformation". And after that, the "old piece" with image of "**initial slide**" will be removed. Then third part...

Third Part.

Now the peice will be animated to the initial position (with zero offsets) and to the initial transformation (with zero rotate, rotateZ, translates and scales equal to 1) for the time equal to the **dur3** parameter. For piece's transformation and position will be used easing function specified in the **ease2** parameter. "rotateMid" parameter will be set to -90 degrees (for vertical3D and horizontal3D) and to 180 degrees for multiFlipV and multiFlipH

Now about real transition. It works the same with every piece but with a delay specified in **pieceDelay** parameter.

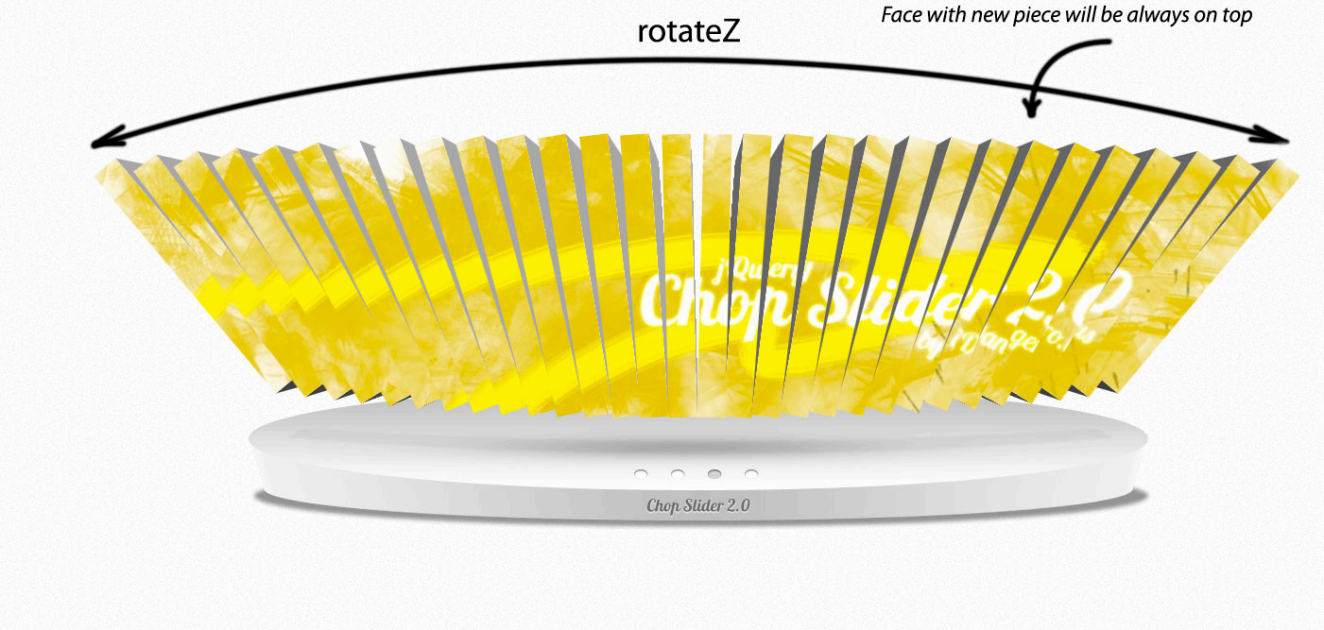
As you can see it is simple. Now let's look at different 3D Transition types and their difference.

The key for the successfully understanding of these 3D transitions is to understand how the rotation angles work!

vertical3D - 3D Blocks transition

Ok, let's look at the transformed vertical3D slide:

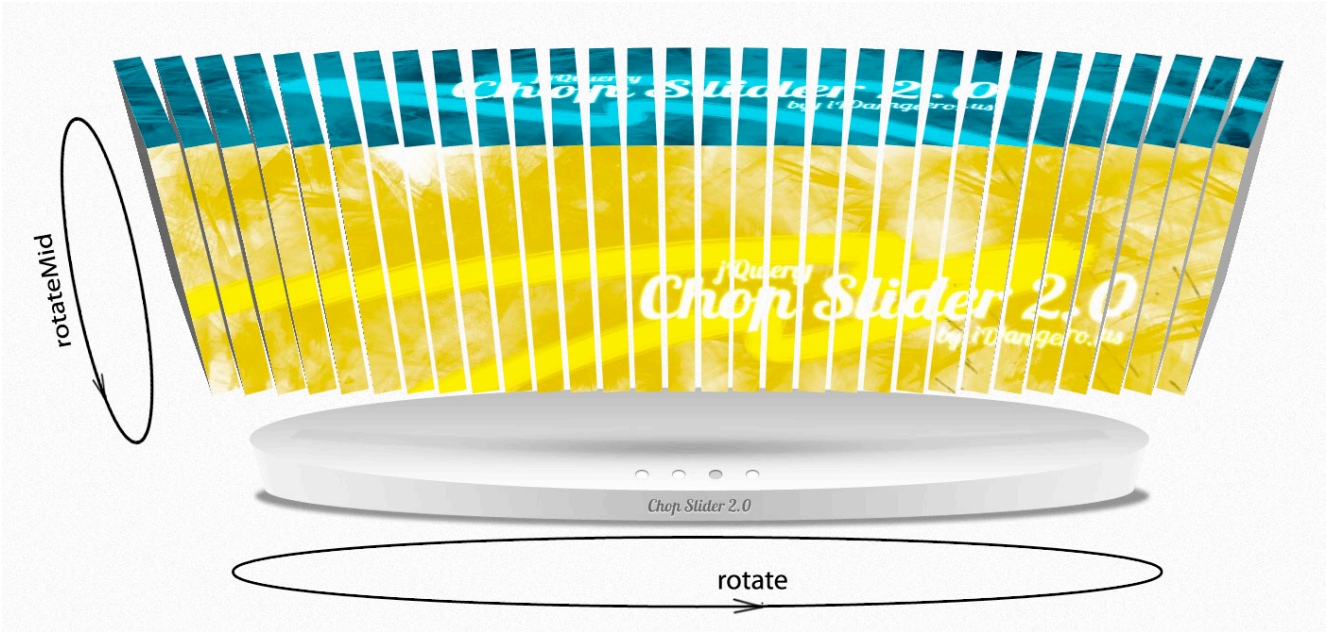
rotateZ = 40, xOffset:100:



rotateMid= -30, xOffset:100:

rotateMid= -30, rotate= -10, rotateZ=0, rotateSymmetric:true, xOffset:100:

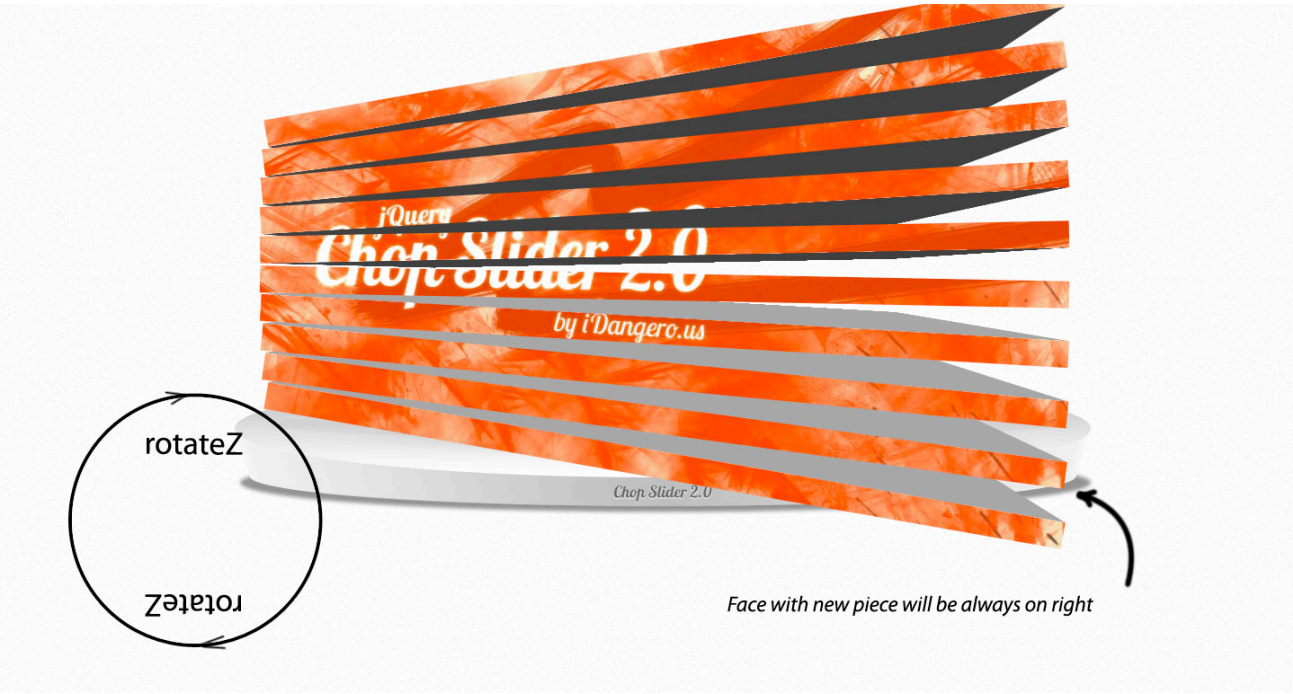
rotateMid= -30, rotate= -10, rotateZ=0, rotateSymmetric:false, xOffset:100:



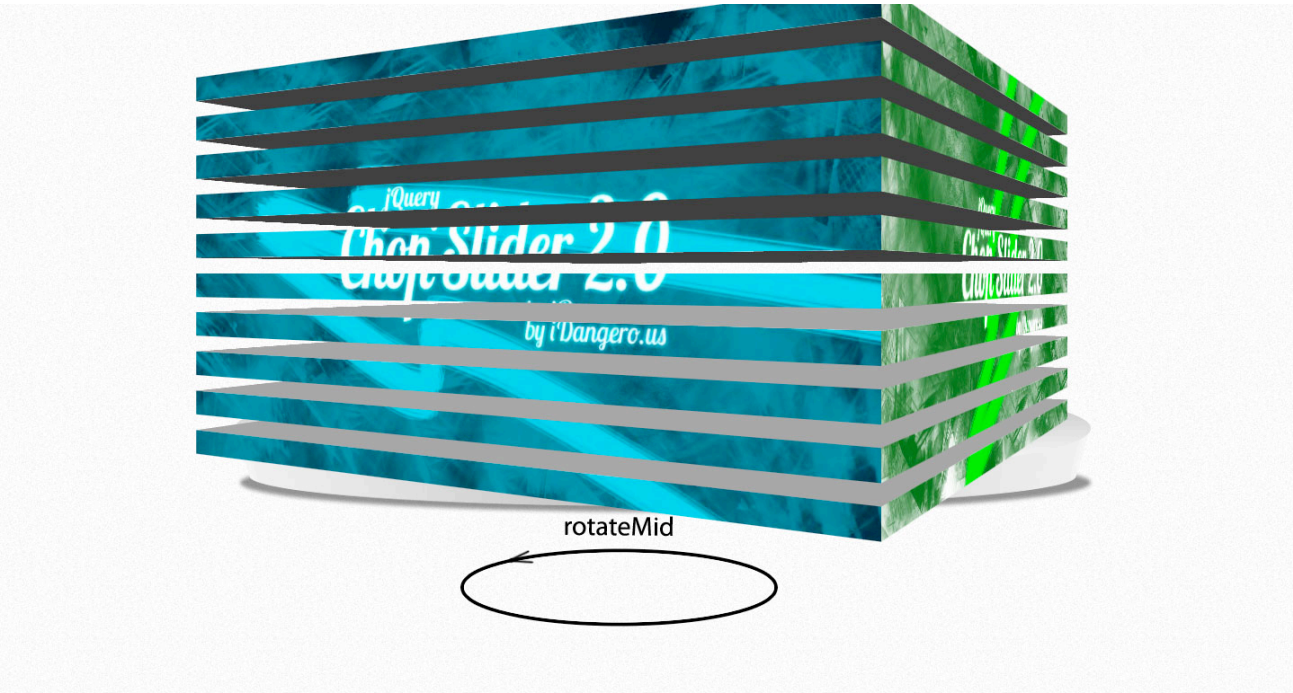
Now let's look on the next type of 3D Transition:

horiozontal3D - 3D Blocks transition

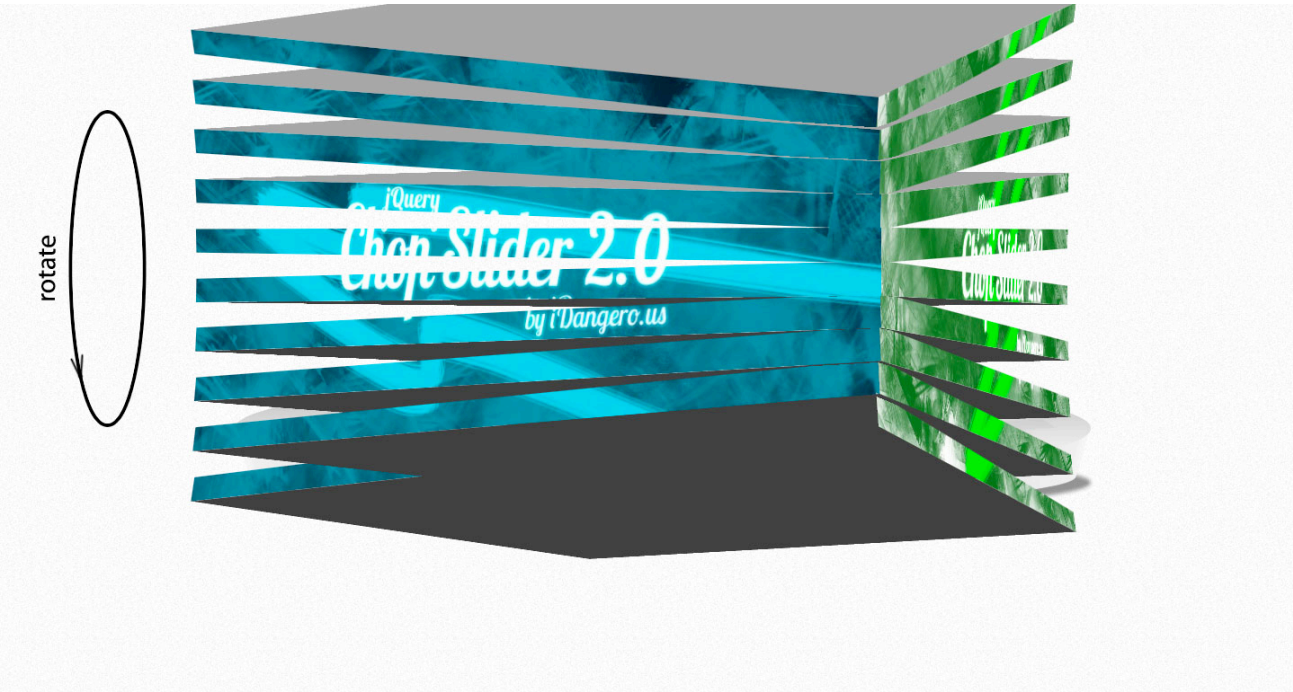
rotateZ = 10:



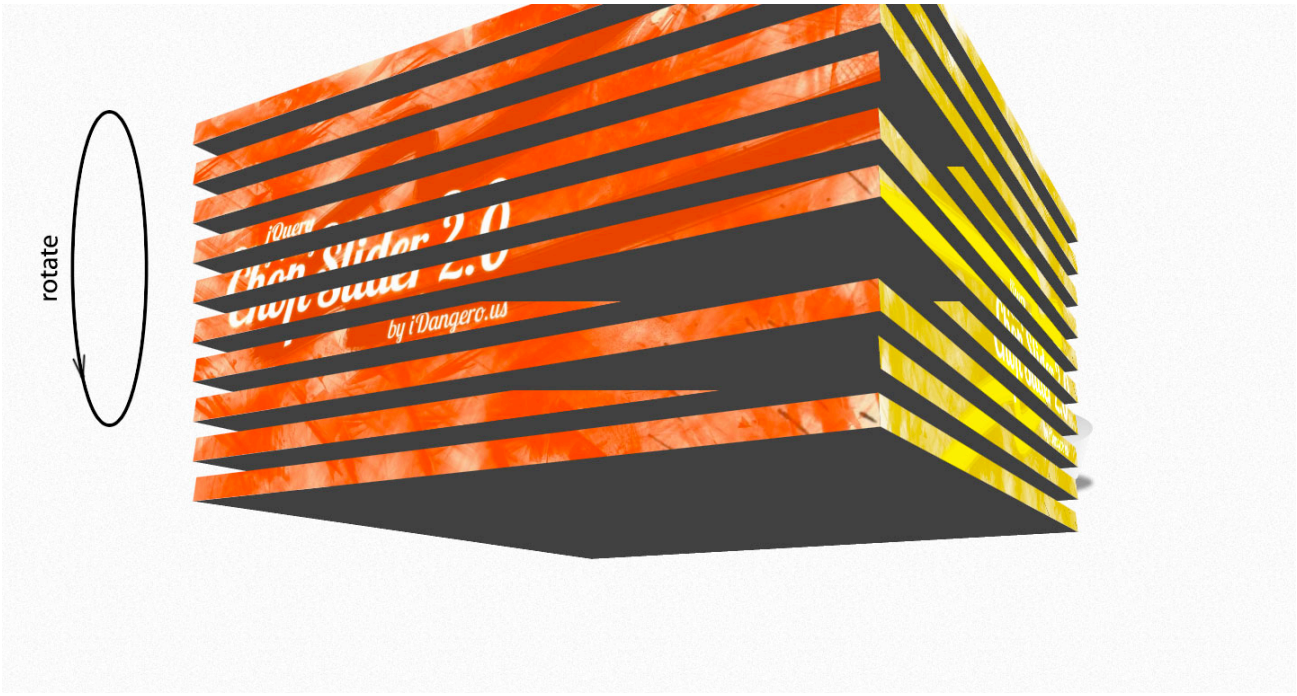
rotateZ = 0, rotateMid= -30:



rotateZ = 0, rotateMid= -30, rotate=20, rotateSymmetric = false:



rotateZ = 0, rotateMid= -30, rotate=20, rotateSymmetric = true:



multiFlipV - 3D Flips Transition

The only difference between multiFlipV and multiFlipH types is that they use different rotate angles to rotate the piece to the face with "new" pice. multiFlipV - will make the vertical 180 degrees "flip" and multiFlipH - horizontal flip. This types of transitions similar to "multi" 2D type of transition.

3D Flips is the most powerful type of 3D Transition. It allows you to replicate 2D analogs but in 3D:

rotate = 20, rotateSymmetric = true:

rotate = 20, rotateSymmetric = false:

rotate = 0, rotateZ = 20, rotateSymmetric = false:

rotate = 0, rotateZ = 20, rotateSymmetric = true:

rotate = 0, rotateZ = 0, rotateMid=-45:

multiFlipH - 3D Flips Transition

The only difference between multiFlipV and multiFlipH types is that they use different rotate angles to rotate the piece to the face with "new" pice. multiFlipV - will make the vertical 180 degrees "flip" and multiFlipH - horizontal flip. This types of transitions similar to "multi" 2D type of transition.

3D Flips is the most powerful type of 3D Transition. It allows you to replicate 2D analogs but in 3D:

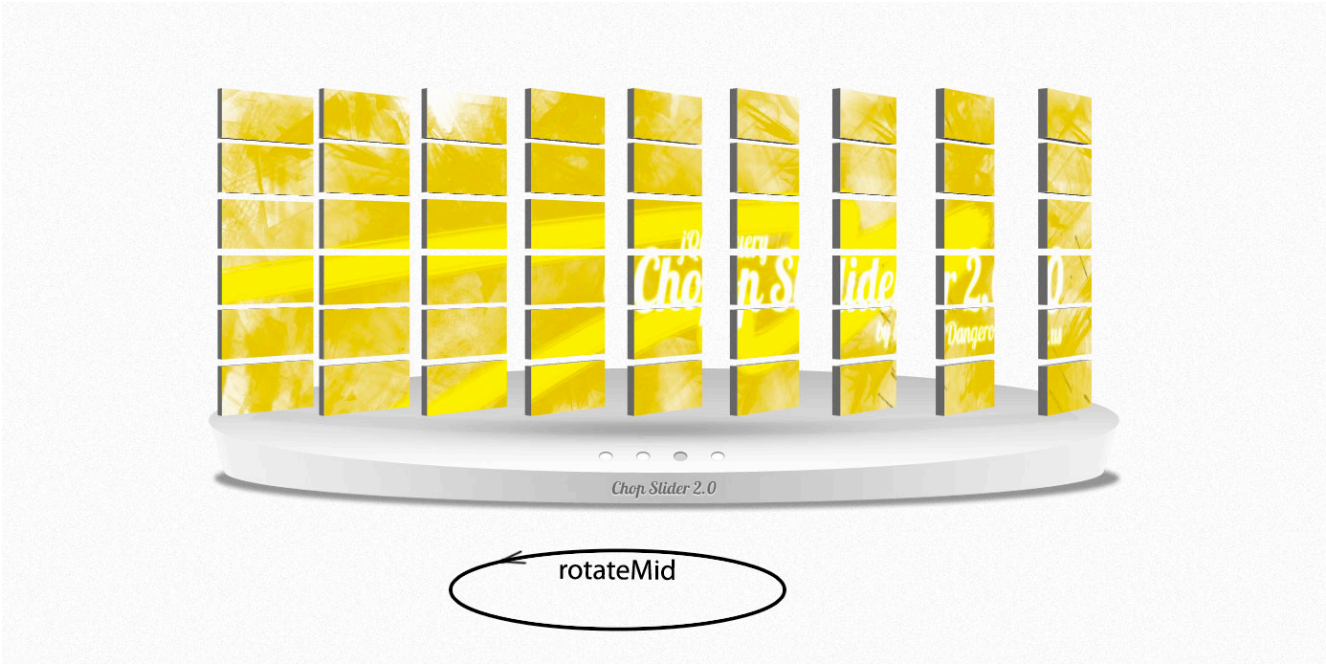
rotate = 20, rotateSymmetric = true:

rotate = 20, rotateSymmetric = false:

rotate = 0, rotateZ = 50, rotateSymmetric = false:

rotate = 0, rotateZ = 50, rotateSymmetric = true:

rotate = 0, rotateZ = 0, rotateMid=-45:



1.7. Animation of Slider's 3D Container

Chop Slider allows to animate Container with pieces in 3D. For this case it you will need to use animateContainer property:

```
....
t3D : {
      ....
      animateContainer : "rotateX(45deg)",
      ...
},
....
```

In this case container will be animated to the specified position for the dur1 time and right after that will return to its initial position for the dur3 time.

This animation do not depend on pieceDelay property, so if you need to hold animated container for a while you will need to use additional containerDelay property:

```
....
t3D : {
      ....
      animateContainer : "rotateX(45deg)",
      containerDelay : 1200, //container delay in ms
      ...
},
....
```

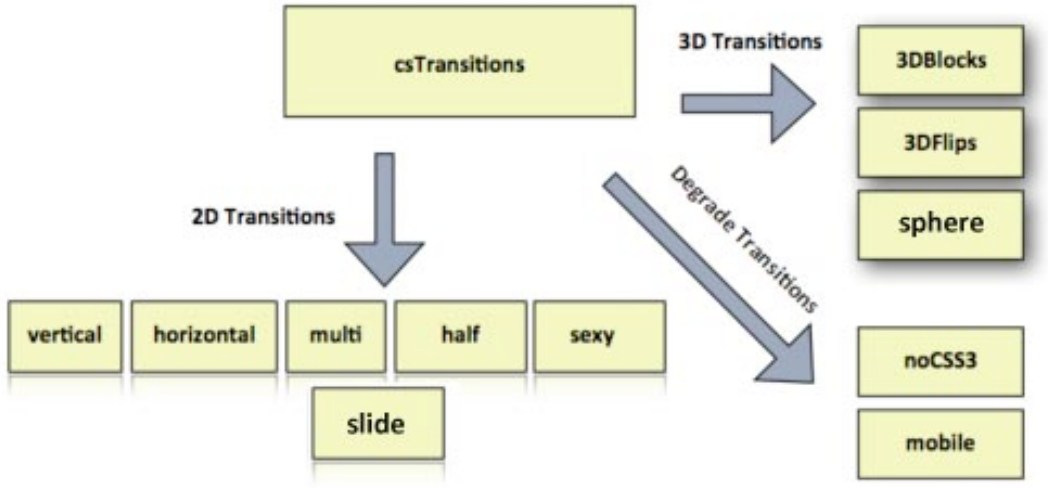
With this option the last example from the previous chapter will look like that:

2. Transitions Library

One of the biggest Chop Slider 2 innovations is the Transition Library. Transitions Library is the additional script file that contains a lot of predefined transition effects:

- 122 2D Transitions - the most common case. These 105 effects will work in all major browsers, except IE. In IE they will look a little bit different but still awesome
- 60 3D Transitions - for the modern browsers. Currently work in: Chrome (Windows Vista+, Mac OSX), Safari (Windows XP+, Mac OSX, iOS)
- 5 "No CSS3" Transitions - for the browsers that do not support CSS3 Transforms like IE6-9
- 5 "Mobile" Transitions - for the mobile browsers

It's very easy to use Transitions Library. So the Transitions Library comes with the following Transitions Groups:



Transitions Library 1.2 contains the following amount of transitions:

- vertical : 23 transitions (from 0 to 22)
- horizontal : 21 transitions (from 0 to 20)
- multi : 30 transitions (from 0 to 29)
- half : 31 transitions (from 0 to 30)
- sexy : 10 transitions (from 0 to 9)
- 3DBlocks : 16 transitions (from 0 to 15)
- 3DFlips : 26 transitions (from 0 to 25)
- sphere: 18 transitions (from 0 to 17)
- noCSS3 : 5 transitions (from 0 to 4)
- mobile : 5 transitions (from 0 to 4)

The biggest advantage of the Transitions Library is that you do not need to write all these parameters for every type of transition.

First of all look at the demo-transitions-library.html file from the downloaded archive with Chop Slider 2, choose the transition you want to use. For example you like "vertical 6th: transition. Here is our Slider's code:

```
jQuery(function(){
      $("#slider").chopSlider({
            /* Slide Element */
            slide : ".slide",
            /* Controlers */
            nextTrigger : "a#slide-next",
            prevTrigger : "a#slide-prev",
            hideTriggers : true,
            sliderPagination : ".slider-pagination",
            /* Captions */
            useCaptions : true,
            everyCaptionIn : ".sl-descr",
            showCaptionIn : ".caption",
            captionTransform : "scale(0) translate(-600px,0px) rotate(45deg)",
            /* Autoplay */
            autoplay : true,
            autoplayDelay : 6000,

            /* 2D Transition (required) */
            t2D : csTransitions['vertical'][6],

            /* Pick the Transition for 3D (Optional) */
            t3D: csTransitions['3DBlocks'][4],

            /* For Mobile Devices (Optional) */
            mobile: csTransitions['mobile'][2],

            /* For Old and IE Browsers (Optional) */
            noCSS3: csTransitions['noCSS3'][3],

            onStart: function(){ /* Do Something*/ },
            onEnd: function(){ /* Do Something*/ }
      })
})
```

Now let's see on some awesome tricks that you can do with Transition Library.

1.You want to use few transitions

For example you like no only "vertical 6" but also "multi 21" transitions:

```
jQuery(function(){
	$("#slider").chopSlider({
		.....

		/* 2D Transitions (required) */
		t2D : [
			csTransitions['vertical'][6],
			csTransitions['multi'][21]
		],

		/* Pick the Transitions for 3D (Optional) */
		t3D: [
			csTransitions['3DBlocks'][4],
			csTransitions['3DBlocks'][12],
			csTransitions['3DFlips'][6]
		],

		/* For Mobile Devices (Optional) */
		mobile: csTransitions['mobile'][2],

		/* For Old and IE Browsers (Optional) */
		noCSS3: csTransitions['noCSS3'][3],

		......
	})
})
```

As you can see in this case we should use Array formatting and put any number of transitions you want to use inside of it.

In this case when you specify few transitions they will be picked up randomly every time you launch the transition

2.You want to use all transitions from the Group

If you want to use for example all "Sexy" transition use such kind of formatting:

```
jQuery(function(){
	$("#slider").chopSlider({
		.....

		/* 2D Transitions (required) */
		t2D : csTransitions['sexy']['random'],

		/* Pick the Transitions for 3D (Optional) */
		t3D: csTransitions['3DFlips']['random'],

		/* For Mobile Devices (Optional) */
		mobile: csTransitions['mobile'][2],

		/* For Old and IE Browsers (Optional) */
		noCSS3: csTransitions['noCSS3'][3],
```

```
            ......
      })
})
```

In this case 2D Transition will be picked up randomly from all transitions fromthe "Sexy" group and 3D Transition will be picked up randomly from all transitions from the "3DFlips" group

You can use such kind of formatting: all from group and some from other groups:

```
jQuery(function(){
      $("#slider").chopSlider({
            .....

            /* 2D Transitions (required) */
            t2D : [
                  csTransitions['sexy']['random'],
                  csTransitions['half'][15],
                  csTransitions['vertical'][16]
            ],

            /* Pick the Transitions for 3D (Optional) */
            t3D: [
                  csTransitions['3DFlips']['random'],
                  csTransitions['3DBlocks'][10]
            ],

            /* For Mobile Devices (Optional) */
            mobile: csTransitions['mobile'][2],

            /* For Old and IE Browsers (Optional) */
            noCSS3: csTransitions['noCSS3'][3],

            ......
      })
})
```

3.You want to use all available transitions

```
jQuery(function(){
      $("#slider").chopSlider({
            .....

            /* 2D Transitions (required) */
            t2D : [
                  csTransitions['sexy']['random'],
                  csTransitions['half']['random'],
                  csTransitions['horizontal']['random'],
                  csTransitions['multi']['random'],
                  csTransitions['vertical']['random']
            ],

            /* Pick the Transitions for 3D (Optional) */
            t3D: [
                  csTransitions['3DFlips']['random'],
                  csTransitions['3DBlocks']['random']
            ],

            /* For Mobile Devices (Optional) */
            mobile: csTransitions['mobile']['random'],

            /* For Old and IE Browsers (Optional) */
            noCSS3: csTransitions['noCSS3'][3],
```

```
            ......
      })
})
```

4.You want to force some parameters for all specified transitions

In this case you need to use "forceParameters" parameter:

```
jQuery(function(){
      $("#slider").chopSlider({

            /* 2D Transitions (required) */
            t2D : [
                  csTransitions['sexy']['random'],
                  csTransitions['half']['random']
            ],

            /* Pick the Transitions for 3D (Optional) */
            t3D: csTransitions['3DBlocks']['random'],

            /* For Mobile Devices (Optional) */
            mobile: csTransitions['mobile']['random'],

            /* For Old and IE Browsers (Optional) */
            noCSS3: csTransitions['noCSS3'][3],

            /* Force parameters that you need to change */
            forceParameters: {
                  hPieces: 10,
                  vPieces: 10
            },
            ......
      })
})
```

In the example above number of horizontal and vertical pieces will be always 10 for any transition from library!

5. You want to change some specific transition parameters

For example you like transition "vertical 6" but you want to change number of vPieces and rotate angle of predefined transition. You need to use this syntax:

```
jQuery(function(){
      $("#slider").chopSlider({
            ...
            /* 2D Transitions (required) */
            t2D : $.extend(
                        csTransitions['vertical'][6],
                        {
                              /* Parameters you want to change */
                              vPieces:20,
                              rotate:90
                        }
            ),
            /* Pick the Transitions for 3D (Optional) */
            t3D: csTransitions['3DBlocks']['random'],

            /* For Mobile Devices (Optional) */
            mobile: csTransitions['mobile']['random'],
```

```
            /* For Old and IE Browsers (Optional) */
            noCSS3: csTransitions['noCSS3'][3],

            ...
      })
})
```

But of course you can write it compact:

```
...
t2D : $.extend( csTransitions['vertical'][6 ], {vPieces:20, rotate:90} ),
...
```

Please note, that with $.extend() function you can extend some specific transitions, it will not work with "random" from library.

If you want to use few extended transitions:

```
...
t2D : [
      $.extend( csTransitions['vertical'][6], {vPieces:20, rotate:90} ),
      $.extend( csTransitions['multi'][5], {hPieces:20, rotate:90} )
],
...
```

6. How to make 3D transition from 2D

For example you like the "vertical 1" transition but you can not find 3D transition which will be similar to this ones. In this case you can create 3D transition from this 2D transition using the following trick:

```
jQuery(function(){

      /*
      First of all you need to create empty object before the Slider's
      initialization
      */

      var myNew3DTransition = {}

      $("#slider").chopSlider({
            .....
            /* 2D Transitions (required) */
            t2D : csTransitions['vertical'][1],

            /* Extend the previous transition */
            t3D: $.extend(
                  myNew3DTransition ,
                  csTransitions['vertical'][1],
                  {
                        type: "multiFlipV",
                        hPieces:1,
                        multiDelay:"linear"
                  }
            ),
            .....
      })
```

```
})
```

As you can see it was pretty easy and now your 2D transition will be the same but with awesome 3D effect in browsers that support 3D transforms

When you make 3D Transition from 2D remember the following:

- Use only "multiFlipV" or "multiFlipH" 3D type

- When you make 3D from "vertical 2D" you need to set **hPieces** to 1 and **multiDelay** to "linear"

- When you make 3D from "horizontal 2D" you need to set **vPieces** to 1 and **multiDelay** to "linear"

4. Using Multiple Chop Sliders

4.1 Using Different Sliders on one page

Since 2.0.5 update Chop Slider 2 supports for multiple sliders on one page. It is very easy to use it, just a few things you should remember:

- for every new slider you need to use absolutely different CSS classes (or ID attributes)

- when using few sliders on one page, Full 3D mode could work not properly

So let's see on two new variables that we need to use for multiple sliders:

Parameter	Type of variable	Example	Required	Default	Description
activeSlideClass	*string*	`"activeSlide"`	optional	`"cs-activeSlide"`	name of class that will be used by Slider to determine CSS class of the "active" slide
activePaginationClass	*string*	`"active-pagination"`	optional	`"cs-active-pagination"`	name of class that will be used by Slider to determine CSS class of the "active" pagination button

Let's look on example from the included **demo-miltiple.html** file.

HTML code:

```
<!--
In this demo will use two sliders with similar styling, so we will use similar
classes for the styles and will add additional different classes that will be used
by the script
```

```
-->
<!-- First Slider -->
<div class="wrapper">
  <div class="s-shadow-b"></div>
  <a id="slide-next" class="snext-1" href="#"></a>
  <a id="slide-prev" class="sprev-1" href="#"></a>
  <div id="slider-1">
    <div class="slide slide1 cs-activeSlide"> <img src="images/slider/1.jpg"
width="900" height="300" alt="slide1" /> </div>
    <div class="slide slide1"> <img src="images/slider/2.jpg" width="900"
height="300" alt="slide2" /> </div>
    <div class="slide slide1"> <img src="images/slider/3.jpg" width="900"
height="300" alt="slide3" /> </div>
    <div class="slide slide1"> <img src="images/slider/4.jpg" width="900"
height="300" alt="slide4" /> </div>
  </div>
  <div class="pagination">
    <span class="slider-pagination sp-1"></span>
    <span class="slider-pagination sp-1"></span>
    <span class="slider-pagination sp-1"></span>
    <span class="slider-pagination sp-1"></span>
  </div>
</div>
<!-- Second Slider -->
<div class="wrapper slider-2">
  <div class="s-shadow-b"></div>
  <a id="slide-next" class="snext-2" href="#"></a>
  <a id="slide-prev" class="sprev-2" href="#"></a>
  <div id="slider-2">
    <div class="slide slide2 cs-activeSlide-2"> <img src="images/slider/3.jpg"
width="900" height="300" alt="slide1" /> </div>
    <div class="slide slide2"> <img src="images/slider/4.jpg" width="900"
height="300" alt="slide2" /> </div>
    <div class="slide slide2"> <img src="images/slider/1.jpg" width="900"
height="300" alt="slide3" /> </div>
    <div class="slide slide2"> <img src="images/slider/2.jpg" width="900"
height="300" alt="slide4" /> </div>
  </div>
  <div class="pagination">
    <span class="slider-pagination sp-2"></span>
    <span class="slider-pagination sp-2"></span>
    <span class="slider-pagination sp-2"></span>
    <span class="slider-pagination sp-2"></span>
  </div>
</div>
```

CSS:

```
/* Active Elements For the Second Slider. Rules are the same as for the First
Slider */
.cs-activeSlide-2 {
      display:block
}
.cs-active-pagination-2 {
      background:url(../images/navi.png) no-repeat left top;
}
```

JavaScript:

```
jQuery(function(){
	$("#slider-1").chopSlider({
		/* Slide Element */
		slide : ".slide1",
		activeSlideClass : "cs-activeSlide",
		/* Controlers */
		nextTrigger : "a.snext-1",
		prevTrigger : "a.sprev-1",
		hideTriggers : true,
		sliderPagination : ".sp-1",
		activePaginationClass : "cs-active-pagination",
		/* For first slider we will use all Multi Transitions */
		t2D : csTransitions['multi']['random']

	})
	$("#slider-2").chopSlider({
		/* Slide Element */
		slide : ".slide2",
		activeSlideClass : "cs-activeSlide-2",
		/* Controlers */
		nextTrigger : "a.snext-2",
		prevTrigger : "a.sprev-2",
		hideTriggers : true,
		sliderPagination : ".sp-2",
		activePaginationClass : "cs-active-pagination-2",
		/* For second slider we will use all Multi Transitions */
		t2D : csTransitions['vertical']['random']

	})
})
```

As you can see it is pretty simple. In the example above we do not use Captions. But if you need to use different captions for these Sliders you need to use different Caption parameters **showCaptionIn** and **everyCaptionIn** for every Slider.

4.2 One Multi-Slider From Two

Now let's see how to make one slider from two. In other words we need to use control navigation for both Sliders.

Let's change example from previous chapter, so the control arrows and pagination of first Slider will also run animation on second Slider:

```
jQuery(function(){
	/*
	First Slider - always must be the "main" slider!
	*/
	$("#slider-1").chopSlider({
		/* Slide Element */
		slide : ".slide1",
		activeSlideClass : "cs-activeSlide",
		/* Controlers */
		nextTrigger : "a.snext-1",
		prevTrigger : "a.sprev-1",
		hideTriggers : true,
		sliderPagination : ".sp-1",
		activePaginationClass : "cs-active-pagination",
```

```
            /* For first slider we will use all Multi Transitions */
            t2D : csTransitions['multi']['random'],
            onStart: function() {
                  // For pagination
                  $.chopSlider.slide({
                        slideTo : $(".cs-active-pagination").index()
                  })
            }
      })
      $("#slider-2").chopSlider({
            /* Slide Element */
            slide : ".slide2",
            activeSlideClass : "cs-activeSlide-2",
            /* Controlers */
            // We need to use triggers of First Slider
            nextTrigger : "a.snext-1",
            prevTrigger : "a.sprev-1",
            hideTriggers : true,
            /* For second slider we will use all Multi Transitions */
            t2D : csTransitions['vertical']['random']

      })
})
```

As you can see for such kind of case we need to use triggers from the First Slider for the Second Slider.

But it is not so simple with pagination, we cannot use same **sliderPagination** property for both sliders, so we need to use some kind of hack using API function $.chopSlider.slide() and onStart callback function for the first Slider.

5. Chop Slider's External API

Important note!

If you are using multiple Sliders, then all following API $.chopSlider functions will be related to the last Slider on the page!

5.1 $.chopSlider.redefine()

This function allows you to redefine default slider's parameters that were set on slider's initialization. It accepts the same parameters specified in table above. **But do not need to use all parameters, use only those you want to redefine!**

Here are some examples of its usage:

```
      //Enable auto play
      $("#play-button").click(function(){
            $.chopSlider.redefine({
                  autoplay:true,
                  autoplayDelay:6000
            })
      });

      //Disable auto play
      $("#stop-button").click(function(){
            $.chopSlider.redefine({
                  autoplay:false
            })
```

```
});

//Disable CSS3 effects
$("#no-css3").click(function(){
	$.chopSlider.redefine({
		disableCSS3:true
	})
})

//Change the 2D Transition parameters.
$("#change-transition").click(function(){
	$.chopSlider.redefine({
		t2D: {
			scaleX:2,
			scaleY:0.5,
			dur1:1000,
			dur2:100,
			dur3:1000
		}
	})
})
```

5.2 $.chopSlider.slide()

This function allows you to run transition with parameters different from default ones. And again, you can use only parameters you want to make different. This function support one more additional parameter - "**direction**", which you can use for transition to the next slide, or to the previous slide.

Parameter	Type of variable	Example	Required	Default	Description
direction	*string*	`"prev"`	**optional**	`"next"`	Direction of transition. Can be "**next**" or "**previous**"
slideTo	*number*	`3`	**optional**	–	Index number of the "new" slide. You can use this option for your custom pagination

Important point, usage of this function **will not affect** to the default parameters (only autoplay can be changed), so after such kind of custom transition, slider parameters will return to default ones, specified on initialization.

Here are couple of usage examples:

```
//Custom transition to the next slide with custom parameters
$("#custom-transition1").click(function(){
	$.chopSlider.slide({
		direction:"next", /*Optional, default value is "next"*/
		scaleX:1,
		scaleY:1,
		rotate:360,
```

```
                dur1:1000,
                dur2:600,
                dur3:1000,
                pieceDelay : 50
        })
})

//Custom transition to the previous slide
$("#custom-transition2").click(function(){
        $.chopSlider.slide({
                direction:"previous", /* Optional, default value is "next" */
                t2D : {
                        scaleX:0,
                        scaleY:0,
                        rotate:360,
                        dur1:1000,
                        dur2:600,
                        dur3:1000,
                        hPieces:20,
                        vPieces:20,
                        pieceDelay : 30
                }
        })
})

//Custom transition to the next slide which will enable Auto-play
$("#custom-transition3").click(function(){
        $.chopSlider.slide({
                autoplay:true,
                autoplayDelay:10000,
                t2D : {
                        scaleX:0,
                        scaleY:0,
                        rotate:360,
                        dur1:1000,
                        dur2:600,
                        dur3:1000,
                        hPieces:20,
                        vPieces:20,
                        pieceDelay : 30
                }
        })
})

//Or you can run the transition with default parameters to the next slide
like that
$("#custom-transition4").click(function(){
        $.chopSlider.slide()
})

//Or to the previous slide like that
$("#custom-transition4").click(function(){
        $.chopSlider.slide({direction:"prev"})
})
```

5.3 $.chopSlider.isAnimating()

This function will return "1" if the transition is running and will return "0" after the transition is complete.

```
$("#runSuperEffect").click(function(){
	if($.chopSlider.isAnimating() == 1) {
		alert ("Sorry but wait until the transition will end");
	}
	else {
		//do something
	}
})
```

5.4 $.chopSlider.hasCSS3()

This function is very useful to determine is the browser supports CSS3 transitions or not.

```
if( $.chopSlider.hasCSS3() ) {
	// Do something
	alert ("Hola! Your browser supports CSS3 transitions!");
}
else {
	// Do something
	alert ("Oops! Your browser do not support CSS3 transitions!");
}
```

5.5 $.chopSlider.has3D()

This function is very useful to determine is the browser supports 3D transitions or not.

```
if( $.chopSlider.has3D() ) {
	// Do something
	alert ("Hola! Your browser supports 3D transitions!");
}
else {
	// Do something
	alert ("Oops! Your browser do not support 3D transitions!");
}
```

6. $(element).csTransform() additional jQuery method

This is additional cross-browser (not IE) jQuery method allows to apply CSS3 transforms (or CSS3 animation) to any element.

csTransform() function accepts two different sets of parameters.

6.1. csTransform(transform, duration, callback)

Variables	Type of variable	Example	Required	Default	Description
transform	*string*	`"rotate(90deg)"`	**required**	-	CSS3 transform property
duration	*number*	`600`	optional	`0`	duration of animation
callback	*function*	`function() {alert("transition completed") }`	optional	-	Selector of the single slide DIV

Here are couple of examples:

```
//Element will be transformed for 600ms
$("#somelement").click(function(){
	$(this).csTransform("rotate(45deg) scale(0.5) translateX(200px)",600)
})

//Element will be transformed imidieatly
$("#somelement").click(function(){
	$(this).csTransform("rotate(45deg) scale(0.5) translateX(200px)")
})

//Rotate again the same element after its transformation
$("#somelement").click(function(){
	$(this).csTransform(
		"rotate(45deg) scale(0.5)",
		900,
		function(){
			$(this).csTransform("rotate(15deg) scale(0.5)", 1900)
		}
	)
})
```

6.2. csTransform(parameters, callback)

Parameter	Type of variable	Example	Required	Default	Description
transform	*string*	`"rotate(90deg)"`	**required**	-	CSS3 transform property
time	*number*	`600`	optional	`0`	duration of animation
ease	*string*	`"ease-in"`	optional	`"ease"`	Transition easing function

Parameter	Type of variable	Example	Required	Default	Description
origin	*string*		optional	`"50% 50%"`	Transform origin property
delay	*number*	`300`	optional	`0`	Delay before transition (animation) will start
animateOrigin	*boolean*	`true`	optional	`false`	If true, the transform origin property will be animated too. Useful only if you will apply .csTransform() method to the same element more that one time

Here is example:

```
//Element will be transformed for 600ms after the 200ms delay
//with custom origin and ease
//and then it will be transformed again with other origin
$("#somelement").click(function(){
      $(this).csTransform(
            {
                  transform:"rotate(45deg) scale(0.5) translateX(200px)",
                  delay:200,
                  time:600,
                  ease:"ease-in-out",
                  origin:"0px 0px"
            },
            function(){
                  $(this).csTransform(
                        {
                              transform:"rotate(90deg) scale(1) translateX(500px)",
                              delay:100,
                              time:1600,
                              ease:"ease-in-out",
                              origin:"120px 50px",
                              animateOrigin:true
                        }
                  )
            }
      )
})
```

Of course first variation with (transform, duration, callback) much more easy to use, but when you need to get full control over your transition use second variation!

Please note! If you will set "**disableCSS3"** property on Slider's initialization to "**true**", then the csTransform() function **will not work**!