API: Create your own effect

Rhino. lider gives you the possibility to write your own cool effects and use them like any other effect. We will explain, what every effect needs, what functions to call and how to use the options properly.

Arguments

Every effect-function is called with three arguments: `$slider, params, callback` . Those arguments need to be in every effect in this particular order.

$slider	This is the actual slideshow-element (like #slideshow) and is needed to get the different vars like `items` or `active`
params	Params contains the settings and settings.direction for shorthand use. With params.settings you can access every option from defaults or your call. The full list of options and arguments can be found here
callback	The callback function resets the styles of the active- and the next-element after the effect has finished.

Where do I put the effect?

Effects in rhinoslider are build as a jQuery extension. They can be found in `$.fn.rhinoslider.effects` . Just add your effect (e.g. slide) to the object and you're good to go:

```
$.fn.rhinoslider.effects = {
                slide: function($slider, params, callback)
.
```

Preparations

Sometime you need to change some built-in settings in order to make your effect work. This is where the preparations come in. Like effects, preparations are build as an jQuery extension:

```
$.fn.rhinoslider.preparations = {
        slide: function($slider, settings, vars){}
}
```

Like effects, preparations have three arguments which need to be in every preparation-function: `$slider, settings, vars` .

For the slide-effect we need the slider and the items to have `overflow: hidden` (which could/should be done by css).

```
slide: function ($slider, settings, vars) {
        vars.items.css('overflow', 'hidden');
        $slider.css('overflow', 'hidden');
}
```

The effect

At the beginning of the effect-function we need to get some variables from the slider:

```
var vars = $slider.data('slider:vars');
var settings = params.settings, direction = params.direction;
var values = [];
```

Of course you could save yourself the `settings = params.settings` and the `direction = params.direction` but lets leave that to personal style.

`vars` now contains all elements, just like buttons, items, prefix and some other stuff. A complete list can be found here.

Options

Slide needs a few options: `direction, animateActive` and `easing` :

direction	In order to determine the animationparameters,we need to know in which direction the items will slide.
animateActive	This determines, if the active-element should be animated and slide out or if it should stay and leave the animation to the next-element.
easing	The standard values here are `linear` and `swing` but since jQuery easing can really enhance every effect, we like to encourage you to use it. jQuery easing can be found here.

Have a look at the direction:

```
switch (direction) {
        case 'toTop':
                values.top = -vars.container.height();
                values.left = 0;
                values.nextTop = -values.top;
                values.nextLeft = 0;
                break;
        case 'toBottom':
                values.top = vars.container.height();
                values.left = 0;
                values.nextTop = -values.top;
                values.nextLeft = 0;
                break;
        case 'toRight':
                values.top = 0;
                values.left = vars.container.width();
                values.nextTop = 0;
                values.nextLeft = -values.left;
                break;
        case 'toLeft':
                values.top = 0;
                values.left = -vars.container.width();
                values.nextTop = 0;
                values.nextLeft = -values.left;
                break;
}
```

As you can see, depending on the direction the parameters are set. Before the animation, we have to make sure that our next-

element won't be behind the active-element. This might not be important in this effect but just be sure to check on it.

```
//put the "next"-element on top of the others and show/hide it,
//depending on the effect
vars.next.css({
        zIndex: 2,
        opacity: 1
});
```

Animation

Moving on to the animation. Since our active-element might not be animated due to user-settings we need to check first:

```
if (settings.animateActive) {
        vars.active.css({
                top: 0,
                left: 0
        }).animate({
                top: values.top,
                left: values.left,
                opacity: 1
        }, settings.effectTime, settings.easing);
}
```

So we have the active-element checked of our list. Now to the next-element. This needs to be positioned outside the slideshow first:

```
vars.next.css({
        top: values.nextTop,
        left: values.nextLeft
});
```

After every effect you need to reset all elements to their initial position and set the classes for active- and next-element so the next effect can run as expected. This is done by the callback-function. Start it after your next-element has been animated:

```
vars.next
        .css({
                top: values.nextTop,
                left: values.nextLeft
        }).animate({
                top: 0,
                left: 0,
                opacity: 1
        }, settings.effectTime, values.nextEasing, function () {
                //reset element-positions
                callback($slider, settings);
        });
```

The final result

That's it. Your effect should now look like this:

```
slide: function ($slider, params, callback) {
        var vars = $slider.data('slider:vars');
        var settings = params.settings;
```

```
        var direction = params.direction;
        var values = [];

        //check, in which direction the content will be moved
        switch (direction) {
                case 'toTop':
                        values.top = -vars.container.height();
                        values.left = 0;
                        values.nextTop = -values.top;
                        values.nextLeft = 0;
                        break;
                case 'toBottom':
                        values.top = vars.container.height();
                        values.left = 0;
                        values.nextTop = -values.top;
                        values.nextLeft = 0;
                        break;
                case 'toRight':
                        values.top = 0;
                        values.left = vars.container.width();
                        values.nextTop = 0;
                        values.nextLeft = -values.left;
                        break;
                case 'toLeft':
                        values.top = 0;
                        values.left = -vars.container.width();
                        values.nextTop = 0;
                        values.nextLeft = -values.left;
                        break;
        }

        //put the "next"-element on top of the others and show/hide it,
        //depending on the effect
        vars.next.css({
                zIndex: 2,
                opacity: 1
        });

        //if animateActive is false, the active-element will not move
        if (settings.animateActive) {
                vars.active.css({
                        top: 0,
                        left: 0
                }).animate({
                        top: values.top,
                        left: values.left,
                        opacity: 1
                }, settings.effectTime, settings.easing);
        }
        vars.next
        //position "next"-element depending on the direction
        .css({
                top: values.nextTop,
                left: values.nextLeft
        }).animate({
                top: 0,
                left: 0,
                opacity: 1
        }, settings.effectTime, settings.easing, function () {
                //reset element-positions
                callback($slider, settings);
        });
}
```

Our effect has some tiny changes to that but this should work nonetheless. If this would be the only effect in your version of the slideshow, the effect-extension should like this:

```
$.fn.rhinoslider.effects = {
        slide: function ($slider, params, callback) {
                var vars = $slider.data('slider:vars');
                var settings = params.settings;
                var direction = params.direction;
                var values = [];

                //if showtime is 0, content is sliding permanently
                //so linear is the way to go
                values.easing = settings.showTime === 0 ? 'linear' : settings.easing;
                values.nextEasing = settings.showTime === 0 ? 'linear' : settings.easing;
                $slider.css('overflow', 'hidden');

                //check, in which direction the content will be moved
                switch (direction) {
                        case 'toTop':
                                values.top = -vars.container.height();
                                values.left = 0;
                                values.nextTop = -values.top;
                                values.nextLeft = 0;
                                break;
                        case 'toBottom':
                                values.top = vars.container.height();
                                values.left = 0;
                                values.nextTop = -values.top;
                                values.nextLeft = 0;
                                break;
                        case 'toRight':
                                values.top = 0;
                                values.left = vars.container.width();
                                values.nextTop = 0;
                                values.nextLeft = -values.left;
                                break;
                        case 'toLeft':
                                values.top = 0;
                                values.left = -vars.container.width();
                                values.nextTop = 0;
                                values.nextLeft = -values.left;
                                break;
                }

                //put the "next"-element on top of the others and show/hide it,
                //depending on the effect
                vars.next.css({
                        zIndex: 2,
                        opacity: 1
                });

                //if animateActive is false, the active-element will not move
                if (settings.animateActive) {
                        vars.active.css({
                                top: 0,
                                left: 0
                        }).animate({
                                top: values.top,
                                left: values.left,
                                opacity: 1
                        }, settings.effectTime, values.easing);
                }
                vars.next
                //position "next"-element depending on the direction
```

```
                .css({
                        top: values.nextTop,
                        left: values.nextLeft
                }).animate({
                        top: 0,
                        left: 0,
                        opacity: 1
                }, settings.effectTime, values.nextEasing, function () {
                        //reset element-positions
                        callback($slider, settings);
                });
        }
}
```

API: Variables

VARIABLE	TYPE	DESCRIPTION
$. lider	jQuery-object	The element (like . l) the slider is called upon
effects	$.fn.rhinoslider.effects	Contains all effects
preparations	$.fn.rhinoslider.preparations	Contains all preparations
settings	object	Contains all defaults and/or user-defined options
vars.active	jQuery-object	Contains the current active item
vars.bullets	array (jQuery-object)	Contains the bullets
vars.buttons	array (jQuery-object)	Contains buttons: play, next, prev
vars.container	jQuery-object	The container wrapped around $slider
vars.intervalAutoPlay	bool / interval	Is false, when autoplay is canceled, contains the interval if autoplay is used
vars.isPlaying	bool	Is set whenever autoplay is toggled
vars.items	array (jQuery-objects)	Contains the items
vars.navigation	jQuery-object	Container around the bullets
vars.next	jQuery-object	Is set by next() or prev(), contains the next item
vars.original	HTMLObject	Contains the original HTMLObject the slider was called upon
vars.playedArray	array (bool)	Contains bool values for each item
vars.playedCounter	int	Counts how many items have been displayed
vars.prefix	string	The prefix for all classnames