

Script.aculo.us

The Wiki Docs

2007-06-13

| | | |
|-----------------|------------------------------------|-----------|
| Part I | Introduction | 4 |
| Part II | Usage | 4 |
| Part III | Effects | 5 |
| 1 | Core Effects | 5 |
| | Effect.Morph | 7 |
| | Effect.Move | 8 |
| | Effect.Opacity | 9 |
| | Effect.Parallel | 9 |
| | Effect.Scale | 10 |
| | Effect.Highlight | 10 |
| 2 | Combination Effects | 11 |
| | Effect.Appear | 12 |
| | Effect.BlindDown | 13 |
| | Effect.BlindUp | 14 |
| | Effect.DropOut | 14 |
| | Effect.Fade | 15 |
| | Effect.Fold | 15 |
| | Effect.Grow | 16 |
| | Effect.Puff | 17 |
| | Effect.Pulsate | 18 |
| | Effect.Shake | 18 |
| | Effect.Shrink | 18 |
| | Effect.SlideDown | 19 |
| | Effect.SlideUp | 20 |
| | Effect.Squish | 21 |
| | Effect.SwitchOff | 21 |
| | Effect.toggle | 21 |
| Part IV | Controls | 22 |
| 1 | Drag and Drop | 22 |
| | Draggables | 22 |
| | Draggable | 23 |
| | Droppables | 24 |
| | Droppables.add | 24 |
| | Droppables.remove | 25 |
| | Sortable | 25 |
| | Sortable.create | 26 |
| | Sortable.destroy | 28 |
| | Sortable.serialize | 28 |
| 2 | Autocompletion | 29 |
| | Autocompleter.Local | 29 |
| | Autocompleter.Base | 30 |
| | Ajax.Autocompleter | 31 |
| 3 | InPlace Editing | 33 |
| | Ajax.InPlaceEditor | 33 |
| | Ajax.InPlaceCollectionEditor | 36 |
| 4 | Slider | 37 |

Part V Tools 39

1 Builder 39
2 Sound 40

Part VI Appendix 41

1 Block Elements 41
2 Inline Elements 42
3 Giving Elements Layout 43

1 Introduction

script.aculo.us

Visual Effects Java Script library

About

This documentation is based on the text that is available in the wiki as of **12th June 2007**. Like in the [online version](#), there are some informations missing, like details about the new sound.js. Anyways, it's a good and near to complete offline version for everyone who needs to lookup specific details regarding Script.aculo.us.

The file you are looking at was compiled and marked up by me, [Kjell Bublitz](#). I completed some missing infos from top of the head here and there and fixed some typos aswell. I will take no further credit for the contents of this file. Like with the *prototype.chm* (which is available for download too), i am just the maintainer.

If you like this doc version, you should [check out my Prototype.chm](#) too.

Version

1.1 (added slider and updated some formatting)

Contact maintainer

You can find me at irc.freenode.net in #cakephp and #prototype as "m3nt0r". Or just go to my website i linked above and leave a comment or something.

2 Usage

Download

Go to the [script.aculo.us downloads page](#) to grab yourself the latest version in a convenient package. Follow the instructions there, then return here.

Unpack and Install

Put `prototype.js`, `scriptaculous.js`, `builder.js`, `effects.js`, `dragdrop.js`, `slider.js` and `controls.js` in a directory of your website, e.g. `/javascripts`.

Link to script.aculo.us in your HTML

Link to the scripts in the head of your document:

```
<script src="javascripts/prototype.js" type="text/javascript"></script>
<script src="javascripts/scriptaculous.js" type="text/javascript"></script>
```

By default, `scriptaculous.js` loads all of the other javascript files necessary for effects, drag-and-drop, sliders, and all of the other `script.aculo.us` features. If you don't need all of the features, you can limit the additional scripts that get loaded by specifying them in a comma-separated list, e.g.:

The scripts that can be specified are:

- builder
- effects
- dragdrop
- controls
- slider
- sound

```
<script src="scriptaculous.js?load=effects,dragdrop" type="text/javascript"></script>
```

Note: Some of the scripts require that others be loaded aswell in order to function properly.

Use in your HTML

To call upon the functions, use HTML script tags. The best way is to define them like this:

```
<script type="text/javascript" language="javascript">
  // 
    <a href="#">Effect.Appear</a>('element id');
  // ]]&gt;
&lt;/script&gt;</pre></div><div data-bbox="163 352 873 369" data-label="Text"><p>This way, you won't have to worry about using characters like &lt; and &gt; in your Java Script code.</p></div><div data-bbox="163 379 518 394" data-label="Text"><p>You can also use effects inside event handlers:</p></div><div data-bbox="179 405 523 437" data-label="Text"><pre>&lt;div onclick="new <a href="#">Effect.SwitchOff</a>(this)"&gt;
  Click here if you've seen enough.
&lt;/div&gt;</pre></div><div data-bbox="163 449 896 477" data-label="Text"><p>If you want to get tricky with it, you can pass extra options to the effect like 'duration', 'fps' (frames per second), and 'delay'.</p></div><div data-bbox="179 488 634 520" data-label="Text"><pre>&lt;div onclick="new <a href="#">Effect.BlindUp</a>(this, {duration: 16})"&gt;
  Click here if you want this to go sloooooow.
&lt;/div&gt;</pre></div><div data-bbox="163 532 288 552" data-label="Section-Header"><h2>Next steps</h2></div><div data-bbox="163 563 688 579" data-label="Text"><p>Have a look at the <a href="#">Demos</a> to catch a glimpse of what you can achieve.</p></div><div data-bbox="94 609 258 630" data-label="Section-Header"><h2>3 Effects</h2></div><div data-bbox="91 642 307 660" data-label="Section-Header"><h3>3.1 Core Effects</h3></div><div data-bbox="163 668 885 697" data-label="Text"><p>The six core effects <a href="#">Effect.Opacity</a>, <a href="#">Effect.Scale</a>, <a href="#">Effect.Morph</a>, <a href="#">Effect.Move</a>, <a href="#">Effect.Highlight</a> and <a href="#">Effect.Parallel</a> are the foundation of the script.aculo.us Visual Effects Java Script library.</p></div><div data-bbox="163 708 245 729" data-label="Section-Header"><h2>Syntax</h2></div><div data-bbox="163 740 440 756" data-label="Text"><p>The basic syntax to start an effect is:</p></div><div data-bbox="184 767 680 780" data-label="Text"><pre>new Effect.<b>EffectName</b>( element, required-params, [options] );</pre></div><div data-bbox="184 790 901 819" data-label="Text"><p><b>element:</b> Can be either a string containing the id of the element, or a Java Script DOM element object.</p></div><div data-bbox="184 817 899 833" data-label="Text"><p><b>required-params:</b> Depends on the effect being called and may not be needed. Most effects do</p></div>
```

not have required parameters. See the documentation for the core effects to learn if the effect has required parameters or if this parameter should be omitted.

options: Used to give any additional customization parameters to the effect.

There are general and effect-specific options.

Example

```
new Effect.Opacity('my element',
  { duration: 2.0,
    transition: Effect.Transitions.linear,
    from: 1.0, to: 0.5 });
```

Common parameters

All core effects support following settings in their options parameter:

| Option | Since | Description |
|------------|---------|---|
| duration | V1.0 | Duration of the effect in seconds, given as a float. Defaults to 1.0. |
| fps | V1.0 | Target this many frames per second. Default to 25. Can't be higher than 100. |
| transition | V1.0 | Sets a function that modifies the current point of the animation, which is between 0 and 1. Following transitions are supplied: Effect.Transitions.sinoidal (default), Effect.Transitions.linear, Effect.Transitions.reverse, Effect.Transitions.wobble and Effect.Transitions.flicker. |
| from | V1.0 | Sets the starting point of the transition, a float between 0.0 and 1.0. Defaults to 0.0. |
| to | V1.0 | Sets the end point of the transition, a float between 0.0 and 1.0. Defaults to 1.0. |
| sync | V1.0 | Sets whether the effect should render new frames automatically (which it does by default). If true, you can render frames manually by calling the render() instance method of an effect. This is used by Effect.Parallel(). |
| queue | V1.5 | Sets queuing options. When used with a string, can be 'front' or 'end' to queue the effect in the global effects queue at the beginning or end, or a queue parameter object that can have {position:'front/end', scope:'scope', limit:1}. For more info on this, see Effect Queues |
| delay | V1.5 | Sets the number of seconds to wait before the effect actually starts. Defaults to 0.0. |
| direction | unknown | Sets the direction of the transition. Values can be either 'top-left', 'top-right', 'bottom-left', 'bottom-right' or 'center' (Default). Applicable only on Grow and Shrink effects. |

Additionally, the options parameter also can be supplied with callback methods, so you can have Java Script executed at various events while the effect is running. The callbacks are supplied with a reference to the effect object as a parameter. Here is an example of getting the element id passed by reference into a callback function:

```
function callback(obj){
  for(var i in obj.effects){
    alert(obj.effects[i]['element'].id);
  }
}
```

| Callback | Since | Description |
|--------------|-------|---|
| beforeStart | V1.0 | Called before the main effects rendering loop is started. |
| beforeUpdate | V1.0 | Called on each iteration of the effects rendering loop, before the redraw takes places. |
| afterUpdate | V1.0 | Called on each iteration of the effects rendering loop, after the redraw takes places. |
| afterFinish | V1.0 | Called after the last redraw of the effect was made. |

Within the effect object, there are several useful variables you can access:

| Variable | Since | Description |
|------------------|-------|--|
| effect.element | V1.0 | The element the effect is applied to. |
| effect.options | V1.0 | Holds the options you gave to the effect. |
| effect.frame | V1.0 | The number of the last frame rendered. |
| effect.startOn | V1.0 | The times (in ms) when the effect was started, and when it will be finished. |
| effect.finishOn | V1.0 | The times (in ms) when the effect was started, and when it will be finished. |
| effect.effects[] | V1.0 | On an Effect.Parallel effect, there's an effects[] array containing the individual effects the parallel effect is composed of. |

Example usage of Callback functions

```
function myCallBackOnFinish(obj){
  alert("the Element's id the effect was applied to is :"+ obj.element.id);
}
function myCallBackOnStart(obj){
  alert("the Element object the effect will be applied to is :"+ obj.element);
}
new Effect.Highlight(myObject,
  { startcolor:'#ffffff',
    endcolor:'#ffffcc',
    duration: 0.5,
    afterFinish: myCallBackOnFinish,
    BeforeStart: myCallBackOnStart
  });
```

3.1.1 Effect.Morph

This effect changes the CSS properties of an element.

Availability

script.aculo.us V1.7 and later.

Syntax

```
$('#morph_example').morph('background:#080;color:#fff');
```

Examples

```
new Effect.Morph('error message',{
  style:'background:#f00; color:#fff; border: 20px solid #f88; font-size:2em',
```

```
duration:0.8
});
```

Style as a hash (keys should be javascript names, rather than CSS ones i.e. 'backgroundColor' rather than 'background-color'):

```
new Effect.Morph('example',{
  style:{
    width:'200px'
  }
});
```

You can also use `$('#element_id').morph({width:'200px'})`, which is a bit shorter.

Effect-specific paramters

| Option | Description |
|--------|--|
| style | the target style of your element, writing with the standard CSS syntax, or as a hash |

Details

Effect.Morph takes original styles given by CSS style rules or inline style attributes into consideration when calculating the transforms. It works with all length and color based CSS properties, including margins, paddings, borders, opacity and text/background colors.

Notes

The original style for an element **must** be in its style attribute, not in an external stylesheet, for Scriptactulous to morph it.

3.1.2 Effect.Move

This effect moves an element. Effect.MoveBy is older name.

Availability

script.aculo.us V1.0 and later.

Syntax

```
new Effect.Move('id of element', y, x, [options]);
new Effect.Move(element, y, x, [options]);
```

Examples

This will move object to corner of the window (x=0; y=0):

```
new Effect.Move (obj,{ x: 0, y: 0, mode: 'absolute'});
```

This will move object 30px up and 20px to the right (the default mode is 'relative'):

```
new Effect.Move (obj,{ x: 20, y: -30, mode: 'relative'});
```

Options

For more informations about options see [Core Effects](#) (part about Common parameters)

Notes

Make sure all the elements you are moving is either absolute or relative. Leaving out the 'position'-tag will break IE compatibility (it will work in Firefox though)

3.1.3 **Effect.Opacity**

This effect changes an element's opacity (transparency).

Availability

script.aculo.us V1.0 and later.

Syntax

```
new Effect.Opacity('id of element', [options]);
new Effect.Opacity(element, [options]);
```

Examples

```
new Effect.Opacity('id_of_element', {duration:0.5, from:1.0, to:0.7});
```

Will fade the element from 100% to 70% over the space of 1/2 second.

Notes

Microsoft Internet Explorer can only set opacity on elements that have a 'layout' (see Giving Elements Layout).

3.1.4 **Effect.Parallel**

This is a special effect to allow to combine more than one core effect into a parallel effect. It's the only effect that doesn't take an element as first parameter, but an array of subeffects.

Availability

script.aculo.us V1.0 and later.

Syntax

```
new Effect.Parallel([array of subeffects], [options]);
```

Example

```
new Effect.Parallel(
  [ new Effect.MoveBy(element, 100, 0, { sync: true }),
    new Effect.Opacity(element, { sync: true, to: 0.0, from: 1.0 } ) ],
  { duration: 0.5,
    afterFinish: function(effect) { Element.hide(effect.effects[0].this.parentNode); }
  }
);
```

3.1.5 **Effect.Scale**

This effect changes an elements width and height dimensions and the base for em units. This allows for smooth, automatic relative scaling of elements contained within the scaled element.

Availability

script.aculo.us V1.0 and later.

Syntax

```
new Effect.Scale('id of element', percent, [options]);  
new Effect.Scale(element, percent, [options]);
```

Effect-specific parameters

| Option | Description |
|-----------------|---|
| scaleX | Sets whether the element should be scaled horizontally, defaults to true. |
| scaleY | Sets whether the element should be scaled vertically, defaults to true. |
| scaleContent | Sets whether content scaling should be enabled, defaults to true. |
| scaleFromCenter | If true, scale the element in a way that the center of the element stays on the same position on the screen, defaults to false. |
| scaleMode | Either 'box' (default, scales the visible area of the element) or 'contents' (scales the complete element, that is parts normally only visible byscrolling are taken into account). You can also precisely control the size the element will become by assigning the originalHeight and originalWidth variables to scaleMode as follows: <pre>scaleMode: { originalHeight: 400, originalWidth: 200 }</pre> |
| scaleFrom | Sets the starting percentage for scaling, defaults to 100.0. |

3.1.6 **Effect.Highlight**

This effect Flashes a color as the background of an element. It is mostly used to draw attention to a part of the page that has been updated via javascript or AJAX, when the update would not otherwise be obvious.

Availability

script.aculo.us V1.0 and later.

Syntax

```
new Effect.Highlight('id of element', [options]);  
new Effect.Highlight(element, [options]);
```

Effect-specific parameters

| Option | Description |
|--------------|---|
| duration | Duration of the effect in seconds, given as a float. Defaults to 1.0. |
| startcolor | Sets the color of first frame of the highlight the highlight. Defaults to "#ffff99" (light yellow) |
| endcolor | Sets the color of the last frame of the highlight. This is best set to the background color of the highlighted element. Defaults to "#ffffff" (white) |
| restorecolor | Sets the background color of the element after the highlight has finished. Defaults to the current background-color of the highlighted element (see Note) |

Examples

```
new Effect.Highlight('my_field', {startcolor:'#ff99ff', endcolor:'#999999'})
```

Notes

If the restorecolor option is not given, Effect.Highlight tries to find out the current background color of the element, which will only work reliably across browsers if the color is given with a CSS rgb triplet, like `rgb(0, 255, 0)`.

Be aware of the syntax: this effect strictly requires a "new" in front, otherwise you will get a javascript error.

3.2 Combination Effects

All the combination effects are based on the six Core Effects, and are thought of as examples to allow you to write your own effects.

- [Effect.Appear](#), [Effect.Fade](#)
- [Effect.Puff](#)
- [Effect.DropOut](#)
- [Effect.Shake](#)
- [Effect.SwitchOff](#)
- [Effect.BlindDown](#), [Effect.BlindUp](#)
- [Effect.SlideDown](#), [Effect.SlideUp](#)
- [Effect.Pulsate](#)
- [Effect.Squish](#)
- [Effect.Fold](#)
- [Effect.Grow](#)
- [Effect.Shrink](#)

Additionally, there's the `Effect.toggle` utility method for elements you want to show temporarily with a Appear/Fade, Slide or Blind animation.

[Effect.toggle](#) uses any of the following pairs:

| Toggle Parameter | Description |
|------------------|---|
| appear | Effect.Appear , Effect.Fade |
| slide | Effect.SlideDown , Effect.SlideUp |

blind [Effect.BlindDown](#), [Effect.BlindUp](#)

Have a look at the [Combination Effects Demo online](#).

3.2.1 **Effect.Appear**

Make an element appear. If the element was previously set to `display:none`; inside the `style` attribute of the element, the effect will automatically show the element. This means that it must be placed under the `style` attribute of an object, and not in the CSS in the head of the document or a linked file.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.Appear('id_of_element');
```

Options

- duration: 1.0 (in seconds)
- from: 0.0 - 1.0 (percent of opacity to start)
- to: 0.0 - 1.0 (percent of opacity to end)

Example

```
Effect.Appear('id_of_element', { duration: 3.0 });
```

Notes

Can take an options parameter, to control the underlying [Effect.Opacity](#) effect.

Works safely with most HTML elements, except table rows, table bodies and table heads.

There is a problem with floating Elements in Safari.

If you need the div to be floated you can do it like this:

```
<a href="#" onclick="new Effect.Appear('appear-div');">Click to appear</a>
<div style="float: right">
  <div id="appear-div" style="display: none;">
    Only this div has to appear!
  </div>
</div>
```

Microsoft Internet Explorer can only set opacity on elements that have a 'layout'. To let an element have a layout, you must set some CSS positional properties, like 'width' or 'height'. See [Giving Elements Layout](#). (Note: fixed in 1.5_rc1.)

On Microsoft Internet Explorer, this effect may display a bold/ghosting artifact on elements that don't have a defined background. It's unclear if this is a feature or a bug. See bug [#4806](#).

3.2.2 **Effect.BlindDown**

This effect simulates a window blind, where the contents of the affected elements stay in place.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.BlindDown('id_of_element');
```

Options

Items in **bold** are default values.

- scaleX:**true**, **false**
- scaleY:**true**, **false**
- scaleContent:**true**, **false**
- scaleFromCenter:**true**, **false**
- scaleMode:**'box'**, 'contents'
- scaleFrom:**100.0** (0%-100%)
- scaleTo:**0** (0%-100%)
- duration:**1** (0.0-1)

Examples

Make the transition longer by adding options.

```
Effect.BlindDown('id_of_element', {duration:3});
```

Notes

Works safely with most [Block Elements](#), except table rows, table bodies and table heads.

Also, if you would like the block hidden when someone first lands on your page, you must use the "display: none" property within the style attribute of the div/block tag, and not in the CSS class for the div. For example:

```
<div style="display: none" id = "id of element">  
Blind content  
</div>
```

and not:

```
#id of element{  
display: none;  
etc...  
}
```

This is the opposite of [Effect.BlindUp](#)

3.2.3 **Effect.BlindUp**

This effect simulates a window blind, where the contents of the affected elements stay in place.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.BlindUp('id_of_element');
```

Options

Items in **bold** are default values.

- scaleX:**true**, **false**
- scaleY:**true**, false
- scaleContent:**true**, false
- scaleFromCenter:**true**, **false**
- scaleMode:**'box'**, 'contents'
- scaleFrom:**100.0** (0%-100%)
- scaleTo:**0** (0%-100%)
- duration:**1** (0.0-1)

Examples

Make the transition longer by adding options.

```
Effect.BlindUp('id_of_element', {duration:3});
```

Notes

Works safely with most [Block Elements](#), except table rows, table bodies and table heads.

This is the opposite of [Effect.BlindDown](#)

3.2.4 **Effect.DropOut**

Makes the element drop and fade out at the same time.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.DropOut('id_of_element');
```

Notes

Works safely with most [Block Elements](#), except tables.

3.2.5 **Effect.Fade**

Makes an element fade away and takes it out of the document flow at the end of the effect by setting the CSS display property to none. Opposite of [Effect.Appear](#)

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.Fade('id_of_element');
```

Options

- from: (defaults to current opacity or 1.0)
- to: (defaults to 0.0)

Examples

```
Effect.Fade('id_of_element', { transition: Effect.Transitions.wobble })
```

Notes

Can take an options parameter, to control the underlying [Effect.Opacity](#) effect.

Works safely with most HTML elements, except table rows, table bodies and table heads.

Microsoft Internet Explorer can only set opacity on elements that have a 'layout'. To let an element have a layout, you must set some CSS positional properties, like 'width' or 'height'. See [Giving Elements Layout](#). (Note: fixed in 1.5_rc1.)

On Microsoft Internet Explorer, this effect may display a bold/ghosting artifact on elements that don't have a defined background. It's unclear if this is a feature or a bug. See bug [#4806](#).

3.2.6 **Effect.Fold**

Reduce the element to its top then to left to make it disappear.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.Fold('id_of_element');
```

Notes

Works safely with most [Block Elements](#), except tables.

3.2.7 **Effect.Grow**

This effect grows a element to a specified size.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.Grow('id_of_element');
```

Notes

Works safely with most [Block Elements](#), except tables.

Options

Effect.Grow has an optional parameter: **direction**.
This parameter can accept the following values:

- top-left
- top-right
- bottom-left
- bottom-right
- center

Examples

```
new Effect.Grow('content', {direction: 'top-left'});  
new Effect.Grow('content', {direction: 'center', duration: 2.0});
```

You can define different durations for several DIV elements, and place them in a row in order to make them appear one after another.

3.2.8 **Effect.Puff**

Gives the illusion of the element puffing away (like a in a cloud of smoke).

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.Puff('id_of_element');
```

Options

Items in bold are default values.

- duration: **1.0** (in seconds)
- from: **0.0**-1.0 (percent of animation to start)
- to: 0.0-**1.0** (percent of animation to end)

Examples

```
Effect.Puff('id_of_element', {duration:3});
```

Notes

Works safely with most [Block Elements](#), except tables.

3.2.9 **Effect.Pulsate**

Pulsates the element, loops over five times over fading out and in.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.Pulsate('id_of_element');
```

Options

- duration: Number of seconds after which to stop the effect.
- from: The minimal opacity during the pulsate, in a value between 0 and 1. For example, use 0.7 for a mild pulsate.
- pulses: The amount of pulses with-in the duration time (default is 5).

Notes

Works safely with most HTML elements, except table rows, table bodies and table heads.

Microsoft Internet Explorer can only set opacity on elements that have 'layout'. To let an element have layout, you must set some CSS positional properties, like 'width' or 'height'. See [Giving Elements Layout](#).

3.2.10 **Effect.Shake**

Moves the element slightly to the left, then to the right, repeatedly.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.Shake('id_of_element');
```

Notes

Works safely with most [Block Elements](#), except tables.

3.2.11 **Effect.Shrink**

Reduce the element to its top-left corner.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.Shrink('id_of_element');
```

Notes

Works safely with most [Block Elements](#), except tables.

3.2.12 Effect.SlideDown

This effect simulates a window blind, where the contents of the affected elements scroll down accordingly.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.SlideDown('id_of_element');
```

Options

Items in **bold** are default values.

- * scaleX:**true**, **false**
- * scaleY:**true**, **false**
- * scaleContent:**true**, **false**
- * scaleFromCenter:**true**, **false**
- * scaleMode:**'box'**, 'contents'
- * scaleFrom:**100.0** (0%-100%)
- * scaleTo:**0** (0%-100%)
- * duration:**1**

Examples

```
Effect.SlideDown('id_of_element', {duration:3});
```

Notes

You must include a second DIV element, wrapping the contents of the outer DIV. So, if you call new Effect.SlideDown('x'), your element must look like this:

```
<div id="x"><div>contents</div></div>
```

Because of a bug in Internet Explorer 6 (overflow not correctly hidden), an additional wrapper div is needed if you want to use these effects on absolutely positioned elements (wrapper is the absolutely positioned element, x has position:relative set):

```
<div id="wrapper"><div id="x"><div>contents</div></div></div>
```

Works only on [block elements](#).

In Internet Explorer 6.0 there's a problem where floated block level elements don't animate. If you add a position: relative to the element it all works though.

3.2.13 **Effect.SlideUp**

This effect simulates a window blind, where the contents of the affected elements scroll up accordingly.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.SlideUp('id_of_element');
```

Options

Items in **bold** are default values.

- * scaleX: true, **false**
- * scaleY: **true**, false
- * scaleContent: **true**, false
- * scaleFromCenter: true, **false**
- * scaleMode: '**box**', 'contents'
- * scaleFrom: **100.0** (0%-100%)
- * scaleTo: **0** (0%-100%)
- * duration: **1**

Examples

```
Effect.SlideUp('id_of_element', {duration:3});
```

Notes

You must include a second DIV element, wrapping the contents of the outer DIV. So, if you call new Effect.SlideDown('x'), your element must look like this:

```
<div id="x"><div>contents</div></div>
```

Because of a bug in Internet Explorer 6 (overflow not correctly hidden), an additional wrapper div is needed if you want to use these effects on absolutely positioned elements (wrapper is the absolutely positioned element, x has position: relative set):

```
<div id="wrapper"><div id="x"><div>contents</div></div></div>
```

Works only on [block elements](#).

In Internet Explorer 6.0 there's a problem where floated block level elements don't animate. If you add a position: relative to the element it all works though.

3.2.14 **Effect.Squish**

Reduce the element to its top-left corner.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.Squish('id_of_element');
```

Notes

Works safely with most [Block Elements](#), except tables.

3.2.15 **Effect.SwitchOff**

Gives the illusion of a TV-style switch off.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Effect.SwitchOff('id_of_element');
```

Notes

Works safely with most [Block Elements](#), except tables.

3.2.16 **Effect.toggle**

Effect.toggle allows for easily toggling elements with an animation.

Availability

script.aculo.us V1.5.0 and later.

Syntax

```
Effect.toggle(element, ['appear' | 'slide' | 'blind'], [options] );
```

Options

element can be either a string containing the id of the element, or a Java Script DOM element object.

The **options** parameter is used to give any additional customization parameters to the effect. There are general and effect-specific options. See the individual effects for more information.

Notes

Keep in mind, like individual Effects, you must include a second DIV element, wrapping the contents of the outer DIV. So, if you call `new Effect.Slide Down?('x')`, your element must look like this:

```
<div id="x"><div>contents</div></div>
```

4 Controls

4.1 Drag and Drop

The Drag-and-drop Java Script library enables easy-to-do dragging and dropping of elements in your web application and to do sortable lists and floats.

Draggables & Droppables

Use [Draggables](#) (elements that can be dragged) and [Droppables](#) (elements that can be dropped on) to add rich user interactivity to your web site or web application.

Sortable lists and floats

Use [Sortable](#) (container elements whose child elements can be sorted) to make lists (HTML elements LI and OL) and floating images or DIVs inside a container div sortable.

4.1.1 Draggables

The Draggables object is a global helper object.

| Property/Method | Description |
|-------------------------------|--|
| <code>drags</code> | Array of all Draggables on the page |
| <code>observers</code> | Array of drag observers. Use <code>Draggables.addObserver()</code> and <code>Draggables.removeObserver()</code> to add/remove observers, respectively |
| <code>register()</code> | <code>function(draggable)</code> . Called when you create a new Draggable. If this is the first Draggable on the page, starts observing mouse events necessary for dragging. |
| <code>unregister()</code> | <code>function(draggable)</code> . Called by <code>Draggable.destroy()</code> . Stops observing window mouse events if <code>Draggable.drag</code> is empty. |
| <code>activate()</code> | Marks a particular Draggable as the activeDraggable |
| <code>deactivate()</code> | Sets <code>Draggables.activeDraggable</code> to null |
| <code>updateDrag()</code> | Passes the window mousemove event to the activeDraggable's <code>updateDrag</code> function. |
| <code>endDrag()</code> | Caught by the window's mouseup, stops dragging the activeDraggable, if any, via its <code>endDrag</code> function. |
| <code>keyPress()</code> | Passes the window keypress event to the activeDraggable's <code>keyPress</code> function. |
| <code>addObserver()</code> | Adds an observer to <code>Draggables.observers</code> |
| <code>removeObserver()</code> | Removes an observer from <code>Draggables.observers</code> . Takes the observer's element property as a parameter |

notify() Calls the observers' onStart(), onEnd(), and onDrag() functions as necessary

Draggable Observers

A draggable observer, as used in **Draggables.addObserver()**, is an object with an element property defined, and one or more of the following functions defined:

| Property/Method | Description |
|-----------------|--|
| onStart() | Called after dragging begins |
| onDrag() | Called on each mousemove during a drag |
| onEnd() | Called when dragging is finished |

The parameters passed to these three events are **eventName**, **draggable**, and **event**. The **draggable.element** method gives us the html element being dragged.

4.1.1.1 Draggable

To make an element draggable, you create a new instance of class Draggable.

Syntax

```
new Draggable('id_of_element', [options]);
```

Options

| Option | Since | Default | Description |
|--------------|-------|---------|--|
| handle | V1.0 | (none) | Sets whether the element should only be draggable by an embedded handle. The value must be an element reference or element id. |
| handle | V1.5 | (none) | As above, except now the value may be a string referencing a CSS class value. The first child/grandchild/etc. element found within the element that has this CSS class value will be used as the handle. |
| revert | V1.0 | false | If set to true, the element returns to its original position when the drags ends. |
| revert | V1.5 | false | Revert can also be an arbitrary function reference, called when the drag ends. |
| snap | V1.5 | false | If set to false no snapping occurs. Otherwise takes the forms – xy or [x,y] or function(x,y){ return [x,y] }. |
| zindex | V1.5 | 1000 | The css zindex of the draggable item. |
| constraint | V1.0 | (none) | If set to 'horizontal' or 'vertical' the drag will be constrained to take place only horizontally or vertically. |
| ghosting | ?? | false | Clones the element and drags the clone, leaving the original in place until the clone is dropped. |
| starteffect | ?? | Opacity | Defines the effect to use when the draggable starts being dragged. |
| reverteffect | ?? | Move | Defines the effect to use when the draggable reverts back to its starting position. |
| endeffect | ?? | Opacity | Defines the effect to use when the draggable stops being dragged. |

Additionally, the options parameter can take the following callback function:

| Callback | Description |
|----------|---|
| change | Called whenever the Draggable is moved by dragging. The called function gets the Draggable instance as its parameter. |

Examples

```
// from the shopping cart demo
new Draggable('product_1',{revert:true});

// constrain direction and give a handle
new Draggable('my_div',{constraint:'horizontal',handle:'handle'});
```

To disable draggables later on, store it in a variable like:

```
var mydrag = new Draggable('product 1', {revert:true})
  (... do stuff ..)
mydrag.destroy();
```

This way, you can enable and disable dragging at will.

4.1.2 Droppables

Droppables are elements which fire a callback function when a Draggable element is released over them.

Creating droppables

See `Droppables.add`.

Disabling droppables

See `Droppables.remove`.

4.1.2.1 Droppables.add

To make an element react when a [Draggable](#) is dropped onto it, you'll add it to the [Droppables](#) of the page with the `Droppables.add` class method.

Syntax

```
Droppables.add('id_of_element',[options]);
```

Options are:

| Option | Since | Default | Description |
|--------|-------|---------|---|
| accept | V1.0 | (none) | Set accept to a string or an array of strings describing CSS classes. The Droppable will only accept Draggables that have one or more of these CSS classes. |

| | | | |
|-------------|--------|--------|--|
| containment | V1.0 | (none) | The droppable will only accept the Draggable if the Draggable is contained in the given elements (or element ids). Can be a single element or an array of elements. This option is used by Sortables to control Drag-and-Drop between Sortables. |
| hoverclass | V1.0 | (none) | If set, the Droppable will have this additional CSS class when an accepted Draggable is hovered over it. |
| overlap | V1.0 | (none) | If set to 'horizontal' or 'vertical' the droppable will only react to a Draggable if its overlapping by more than 50% in the given direction. Used by Sortables. |
| greedy | V1.1b1 | true | If true stops processing hovering (don't look for other Droppables that are under the Draggable) |

Additionally, following callbacks can be used in the option parameter:

| Callback | Since | Description |
|----------|-------|--|
| onHover | V1.0 | Called whenever a Draggable is moved over the Droppable and the Droppable is affected (would accept it). The callback gets three parameters: the Draggable, the Droppable element, and the percentage of overlapping as defined by the overlap option. Used by Sortables. |
| onDrop | V1.0 | Called whenever a Draggable is released over the Droppable and the Droppable is accepts it. The callback gets three parameters: the Draggable element, the Droppable element and the Event. You can extract additional information about the drop – like if the Ctrl or Shift keys were pressed – from the Event object. |

Example

```
// from the shopping cart demo
Droppables.add('shopping cart', {
  accept: 'products',
  onDrop: function(element)
    { $('shopping cart text').innerHTML =
      'Dropped the ' + element.alt + ' on me.'; });
```

4.1.2.2 Droppables.remove

Will remove any droppable capabilities from 'element'.

Syntax

```
Droppables.remove(element)
```

4.1.3 Sortables

A Sortable consists of item elements in a container element. When you create a new Sortable, it takes care of the creation of the corresponding [Draggables](#) and [Droppables](#).

Availability

script.aculo.us V1.0 and later.

Creating sortables

See [Sortable.create](#).

Disabling sortables

See [Sortable.destroy](#).

Serializing

The Sortable object also provides a function to serialize the Sortable in a format suitable for HTTP GET or POST requests. This can be used to submit the order of the Sortable via an Ajax call.

See [Sortable.serialize](#)

4.1.3.1 Sortable.create

Use `Sortable.create` to initialize a sortable element.

Availability

script.aculo.us V1.0 and later.

Syntax

```
Sortable.create('id_of_container', [options]);
```

Options

| Option | Since | Default | Description |
|-------------|-------|--|---|
| tag | V1.0 | li | Sets the kind of tag (of the child elements of the container) that will be made sortable. For UL and OL containers, this is 'LI', you have to provide the tag kind for other sorts of child tags. |
| only | V1.0 | (none) | Further restricts the selection of child elements to only encompass elements with the given CSS class (or, if you provide an array of strings, on any of the classes). |
| overlap | V1.0 | vertical | vertical Either 'vertical' or 'horizontal'. For floating sortables or horizontal lists, choose 'horizontal'. Vertical lists should use 'vertical'. |
| constraint | V1.0 | vertical | Restricts the movement of Draggables, see the constraint option of Draggables. |
| containment | V1.0 | (only within container) | Enables dragging and dropping between Sortables. Takes an array of elements or element-ids (of the containers). |
| format | ?? | <code>/^[^_\-](?:[A-Za-z0-9\-_])*(.*)\$/</code> | The RegEx to get the correct values from the IDs of all sortable child elements. This can be important for the output of Sortable.serialize |
| handle | V1.0 | (none) | Makes the created Draggables use handles, see the handle option on Draggables. |

| | | | |
|-------------------|-------|--------|--|
| hoverclass | V1.1 | (none) | Gives the created Droppables a hoverclass (see there). |
| | b1 | | |
| ghosting | V1.5 | false | If set to true, dragged elements of the Sortable will be cloned and appear as “ghost”, i.e. a representation of their original element, instead of directly dragging the original element. See below for more details. |
| dropOnEmpty | V1.5 | false | If set to true, the Sortable container will be made into a Droppable, that can receive a Draggable (as according to the containment rules) as a child element when there are no more elements inside. |
| scroll | V1.5. | (none) | When the sortable is contained in an element with style overflow:scroll, this value can be set to the ID of that container (or the container’s DOM object). The scroll position of the container will now move along when the sortable is dragged out of the viewable area. The container must have overflow:scroll set to include scroll bars. Does not yet work for scrolling the entire document. To get this to work correctly, include this line in your code before creating the sortable: Position.includeScrollOffsets = true; |
| | 2 | | |
| scrollSensitivity | ?? | 20 | Will start scrolling when element is x pixels from the bottom, where x is the scrollSensitivity. |
| scrollSpeed | ?? | 15 | Will scroll the element in increments of scrollSpeed pixels. |
| tree | V1.6. | false | If true, sets sortable functionality to elements listed in treeTag |
| | 1 | | |
| treeTag | V1.6. | ul | The element type tree nodes are contained in. |
| | 1 | | |

You can provide the following callbacks in the options parameter:

| Callback | Since | Description |
|----------|-------|---|
| onChange | V1.0 | Called whenever the sort order changes while dragging. When dragging from one Sortable to another, the callback is called once on each Sortable. Gets the affected element as its parameter. |
| onUpdate | V1.0 | Called when the drag ends and the Sortable’s order is changed in any way. When dragging from one Sortable to another, the callback is called once on each Sortable. Gets the container as its parameter. Note that the id attributes of the elements contained in the Sortable must be named as described in Sortable.serialize |

Notes

Important: You can use Sortable.create on any container element that contains [Block Elements](#), with the exception of TABLE, THEAD, TBODY and TR. This is a technical restriction with current browsers.

A sortable nested somewhere inside a table won’t work well under IE unless the table has a “position:relative” style. If you use the css display: table property, sortable lists will work a little, but doesn’t allow true drag and drop of the elements.

Got it working using tbody as container and TR as the sortables (IE6 (pc) and Firefox (mac/pc)).

A call to Sortable.create implicitly calls on [Sortable.destroy](#) if the referenced element was already a Sortable.

4.1.3.2 Sortable.destroy

Use `Sortable.destroy` to completely remove all event handlers and references to a `Sortable` created by [Sortable.create](#). It does not remove or alter the referenced element in any other way.

Availability

script.aculo.us V1.1b1 and later.

Syntax

```
Sortable.destroy( element );
```

Notes

A call to [Sortable.create](#) implicitly calls on `Sortable.destroy` if the referenced element was already a `Sortable`.

4.1.3.3 Sortable.serialize

The `Sortable` object also provides a function to serialize the `Sortable` in a format suitable for HTTP GET or POST requests. This can be used to submit the order of the `Sortable` via an Ajax call:

Availability

script.aculo.us V1.0 and later.

Syntax

```
Sortable.serialize('id_of_container', [options]);
```

Options

| Option | Since | Default | Description |
|--------|-------|---|---|
| tag | V1.0 | tag originally used on <code>Sortable.create</code> | Sets the kind of tag (of the child elements of the container) that will be serialized. |
| name | V1.0 | id of container | Sets the name of the key that will be used to create the key/value pairs for serializing in HTTP GET/POST format (that is, <code>key[]=value&key[]=value ...</code>) |

Example

```
var userTopTen = Sortable.serialize('top10');  
// userTopTen now contains key[]=value pairs separated by &  
new Ajax.Request('/users/mytop10/saveorder', {parameters: userTopTen});
```

Notes

For this to work, the elements contained in your Sortable must have id attributes in the following form:

```
<li id="image_1">Something</li>
```

You can tweak this by modifying the format parameter in the [Sortable.create](#) call

4.2 Autocompletion

The Autocompleter controls allow for Google-Suggest style local and server-powered autocompleting text input fields.

AJAX Autocompletion

See [Ajax.Autocompleter](#)

Local Autocompletion

See [Autocompleter.Local](#)

Classes

- [Autocompleter.Base](#)
- [Ajax.Autocompleter](#)
- [Autocompleter.Local](#)

4.2.1 Autocompleter.Local

The local array autocompleter.

Used when you'd prefer to inject an array of autocompletion options into the page, rather than sending out Ajax queries.

Syntax

```
new Autocompleter.Local(inputElement, choicesElement, dataArray, [options])
```

The constructor takes four parameters. The first two are, as usual, the id of the monitored textbox, and id of the autocompletion menu. The third is an array of strings that you want to autocomplete from, and the fourth is the options block.

Extra local autocompletion options:

- **choices**: How many autocompletion choices to offer
- **partialSearch**: If false, the autocompleter will match entered text only at the beginning of strings in the autocomplete array. Defaults to true, which will match text at the beginning of

any word in the strings in the autocomplete array. If you want to search anywhere in the string, additionally set the option `fullSearch` to `true` (default: `off`).

- **fullSearch**: Search anywhere in autocomplete array strings.
- **partialChars**: How many characters to enter before triggering a partial match (unlike `minChars`, which defines how many characters are required to do any match at all). Defaults to 2.
- **ignoreCase**: Whether to ignore case when autocompleting. Defaults to `true`.

It's possible to pass in a custom function as the 'selector' option, if you prefer to write your own autocompletion logic. In that case, the other options above will not apply unless you support them.

Example

```
<p><label for="bands from the 70s">Your favorite rock band from the 70's:</label><br />
<input id="bands from the 70s" autocomplete="off" size="40" type="text" value="" /></p>

<div class="page name auto complete" id="band list" style="display:none"></div>

<script type="text/javascript">
new Autocompleter.Local('bands from the 70s', 'band list', ['ABBA', 'AC/DC', 'Aerosmith', 'America',
  'Bay City Rollers', 'Black Sabbath', 'Boston', 'David Bowie', 'Can',
  'The Carpenters', 'Chicago', 'The Commodores', 'Crass', 'Deep Purple',
  'The Doobie Brothers', 'Eagles', 'Fleetwood Mac', 'Haciendo Punto en Otro Son',
  'Heart', 'Iggy Pop and the Stooges', 'Journey', 'Judas Priest',
  'KC and the Sunshine Band', 'Kiss', 'Kraftwerk', 'Led Zeppelin', 'Lindisfarne (ba
  'Lipps, Inc', 'Lynyrd Skynyrd', 'Pink Floyd', 'Queen', 'Ramones', 'REO Speedwagon',
  'Rhythm Heritage', 'Rush', 'Sex Pistols', 'Slade', 'Steely Dan', 'Stillwater',
  'Styx', 'Supertramp', 'Sweet', 'Three Dog Night', 'The Village People',
  'Wings (fronted by former Beatle Paul McCartney)', 'Yes'], {});
</script>
```

4.2.2 Autocompleter.Base

`Autocompleter.Base` handles all the autocompletion functionality that's independent of the data source for autocompletion. This includes drawing the autocompletion menu, observing keyboard and mouse events, and similar.

Availability

script.aculo.us V1.1b1

Extending

Specific autocompleters need to provide, at the very least, a `getUpdatedChoices` function that will be invoked every time the text inside the monitored textbox changes. This method should get the text for which to provide autocompletion by invoking `this.getToken()`, NOT by directly accessing `this.element.value`. This is to allow incremental tokenized autocompletion. Specific auto-completion logic (AJAX, etc) belongs in `getUpdatedChoices`.

Tokenized autocompletion

Tokenized incremental autocompletion is enabled automatically when an autocompleter is instantiated with the 'tokens' option in the options parameter:

```
new Ajax.Autocompleter('id','upd', '/url/', { tokens: ',' });
```

will incrementally autocomplete with a comma as the token.

The `,` in the above example can be replaced with an array, e.g. `{ tokens: [',', '\n'] }` which enables autocompletion on multiple tokens. This is most useful when one of the tokens is `\n` (a newline), as it allows smart autocompletion after linebreaks.

4.2.3 Ajax.Autocompleter

The `Ajax.Autocompleter` class allows for server-powered autocompleting text fields.

Availability

script.aculo.us V1.1 ???

Syntax

```
new Ajax.Autocompleter(id_of_text_field, id_of_div_to_populate, url, options);
```

Options (inherited from [Autocompleter.Base](#))

| Option | Default | Description |
|---------------------------------|-------------------------------|---|
| <code>paramName</code> | 'name' of the element | Name of the parameter for the string typed by the user on the autocompletion field |
| <code>tokens</code> | (empty array) <code>[]</code> | See Autocompleter.Base |
| <code>frequency</code> | 0.4 | |
| <code>minChars</code> | 1 | Minimum characters required to query the server |
| <code>indicator</code> | null | When sending the Ajax request <code>Autocompleter</code> shows this element with <code>Element.show</code> . You can use this to e.g. display an animated "please-wait-while-we-are-working" gif. See here for examples. When the request has been completed it will be hidden with <code>Element.hide</code> . |
| <code>updateElement</code> | null | Hook for a custom function called after the element has been updated (i.e. when the user has selected an entry). This function is called instead of the built-in function that adds the list item text to the input field. The function receives one parameter only, the selected item (the <code></code> item selected) |
| <code>afterUpdateElement</code> | null | Hook for a custom function called after the element has been updated (i.e. when the user has selected an entry). This function is called after the built-in function that adds the list item text to the input field (note difference from above). The function receives two parameters, the autocompletion input field and the selected item (the <code></code> item selected) |

Examples

HTML

```
<input type="text" id="autocomplete" name="autocomplete parameter"/>
<div id="autocomplete_choices" class="autocomplete"></div>
```

Javascript

```
new Ajax.Autocompleter("autocomplete", "autocomplete_choices", "/url/on/server", {});
```

HTML with indicator

```
<input type="text" id="autocomplete" name="autocomplete parameter"/>  
<span id="indicator1" style="display: none"></span>  
<div id="autocomplete_choices" class="autocomplete"></div>
```

(As with any element destined to work with Prototype's Element.toggle/show/hide, your indicator element should use an inline style attribute with a display property, here initially set to none. If you need to assign it CSS rules, put the class attribute before the style attribute to avoid override.)

Javascript with options

```
new Ajax.Autocompleter("autocomplete", "autocomplete choices", "/url/on/server",  
{paramName: "value", minChars: 2, updateElement: addItemToList, indicator: 'indicator1'});
```

CSS Styles

The styling of the div and the returned UL are important.

Applying a visual cue that an item is selected allows the user to take advantage of the keyboard navigation of the dropdown and adding background colors, borders, positioning, etc to the div (as the demo does) allows the UI element to stand out. The CSS from the demo applied to the examples would be

```
div.autocomplete {  
  position:absolute;  
  width:250px;  
  background-color:white;  
  border:1px solid #888;  
  margin:0px;  
  padding:0px;  
}  
div.autocomplete ul {  
  list-style-type:none;  
  margin:0px;  
  padding:0px;  
}  
div.autocomplete ul li.selected { background-color: #ffb;}  
div.autocomplete ul li {  
  list-style-type:none;  
  display:block;  
  margin:0;  
  padding:2px;  
  height:32px;  
  cursor:pointer;  
}
```

Server return

Look through your POST environment variable for the current entry in the text-box.

The server-side will receive the typed string as a parameter with the same name as the name of the element of the autocompletion (name attribute). You can override it setting the option paramName.

The server must return an unordered list.

For instance this list might be returned after the user typed the letter "y"

```
<ul>
  <li>your mom</li>
  <li>yodel</li>
</ul>
```

Display additional information

If you wish to display additional information in the autocomplete dropdown that you don't want inserted into the field when you choose an item, surround it in a `` (could work with others but not tested) with the class of "informal".

For instance the following shows a list of companies and their addresses, but only the company name will get inserted:

Response

```
<ul>
  <li>ACME Inc <span class="informal"> 5012 East 5th Street</span></li>
  <li>Scriptaculous <span class="informal"> 1066 West Redlands Parkway</span></li>
</ul>
```

Working with Callback

Another way to get additional information, just send an ID in every list, and redefine `afterUpdateElement`?

Response

```
<ul>
  <li id="1">your mom</li>
  <li id="2">yodel</li>
</ul>
```

Javascript

```
new Ajax.Autocompleter("autocomplete", "autocomplete choices", "/url/on/server",
  {afterUpdateElement : getSelectionId});
```

```
function getSelectionId(text, li) {
  alert (li.id);
}
```

Notes

When a choice is highlighted the `Autocompleter` applies a class of "selected" to the `li` element. This allows the end user to style the selected element as desired.

4.3 InPlace Editing

Enter topic text here.

4.3.1 **Ajax.InPlaceEditor**

The in-place "text edit" testing allows for Flickr-style AJAX-backed "on-the-fly" textfields.

Availability

script.aculo.us V1.5 and later.

Syntax

```
new Ajax.InPlaceEditor( element, url, [options]);
```

The constructor takes three parameters. The first is the element that should support in-place editing. The second is the url to submit the changed value to. The server should respond with the updated value (the server might have post-processed it or validation might have prevented it from changing). The third is a hash of options.

Options

| Option | Since | Default | Description |
|-------------------|-------|--|--|
| okButton | V1.6 | true | If a submit button is shown in edit mode (true,false) |
| okText | V1.5 | "ok" | The text of the submit button that submits the changed value to the server |
| cancelLink | V1.6 | true | If a cancel link is shown in edit mode (true,false) |
| cancelText | V1.5 | "cancel" | The text of the link that cancels editing |
| savingText | V1.5 | "Saving..." | The text shown while the text is sent to the server |
| clickToEditText | V1.6 | "Click to edit" | The text shown during mouseover the editable text |
| formId | V1.5 | id of the element to edit +'InPlaceForm' | The id given to the element |
| externalControl | V1.5 | null | ID of an element that acts as an external control used to enter edit mode. The external control will be hidden when entering edit mode and shown again when leaving edit mode. |
| rows | V1.5 | 1 | The row height of the input field (anything greater than 1 uses a multiline textarea for input) |
| cols | V1.5 | none | The number of columns the text area should span (works for both single line or multi line) |
| size | V1.5 | none | Synonym for 'cols' when using single-line (rows=1) input |
| highlightcolor | ?? | Ajax. InPlaceEditor. defaultHighlightColor | The highlight color |
| highlightendcolor | ?? | "#FFFFFF" | The color which the highlight fades to |
| savingClassName | V1.5 | "inplaceeditor-saving" | CSS class added to the element while displaying "Saving..." (removed when server responds) |
| formClassName | V1.5 | "inplaceeditor-form" | CSS class used for the in place edit form |
| hoverClassName | ?? | ?? | ?? |
| loadTextURL | V1.5 | null | Will cause the text to be loaded from the server (useful if your text is actually textile and formatted on the server) |
| loadingText | V1.5 | "Loading..." | If the loadText URL option is specified then this text is displayed while the text is being loaded from the server |
| callback | V1.5 | function(form) {Form.serialize(form)} | A function that will get executed just before the request is sent to the server, should return the parameters to be sent in the URL. Will get two parameters, the entire form and the value of the text control. |

| | | | |
|--------------|------|-----------------|--|
| submitOnBlur | V1.6 | false | This option if true will submit the <code>in_place_edit</code> form when the input tag loses focus. |
| ajaxOptions | V1.5 | (empty hash) {} | Options specified to all AJAX calls (loading and saving text), these options are passed through to the prototype AJAX classes. |

Since version 1.6 you can provide the following callbacks in the options parameter:

| Callback | Default | Description |
|------------|---|---|
| onComplete | <code>"function(transport, element) {new Effect.Highlight(element, {startcolor: this.options.highlightcolor});}"</code> | Code run if update successful with server |
| onFailure | <code>"function(transport) {alert("Error communicating with the server: " + transport.responseText.stripTags());}"</code> | Code run if update failed with server |

Examples

Single Line Edit Mode

```
<p id="editme">Click me, click me!</p>
<script type="text/javascript">
  new Ajax.InPlaceEditor('editme', '/demoajaxreturn.html');
</script>
```

Multi Line Edit Mode

```
<p id="editme2">Click me to edit this nice long text.</p>
<script type="text/javascript">
  new Ajax.InPlaceEditor('editme2', '/demoajaxreturn.html', {rows:15,cols:40});
</script>
```

The server side component gets the new value as the parameter 'value' (POST method), and should send the new value as the body of the response.

Notes

Character encoding

The form data is sent encoded in UTF-8 regardless of the page encoding. This is as of the prototype function `Form.serialize`. [More info here](#) (web). If this doesn't work, you can use `iconv` [as outlined here](#) (web).

Remove and force InPlaceEditor

To disable the `InPlaceEditor` behavior later on, store it in a variable like:

```
var editor = new Ajax.InPlaceEditor('product_1',...);
```

Later you can then remove the behaviour by calling:

```
editor.dispose();
```

This way, you can enable and disable In Place Editing at will. You can also arbitrarily force it into edit mode like so:

```
editor.enterEditMode('click');
```

4.3.2 **Ajax.InPlaceCollectionEditor**

Generates a selectbox from the values in the **collection** array.

Syntax

```
new Ajax.InPlaceCollectionEditor( element, url, { collection: [array], [moreOptions] } );
```

The constructor takes three parameters.

1. **element**: The element that should support in-place editing.
2. **url**: The url to submit the changed value to. The server should respond with the updated value (the server might have post-processed it or validation might have prevented it from changing).
3. **hash**: The third is a hash of options. Within that hash of options should be a field called **collection** that is an array of values to place inside your select box.

For **moreOptions** see [Ajax.InPlaceEditor](#)

Examples

HTML

```
<p title="Click to edit" id="tobeedited">Click me, click me!</p>
```

Javascript

```
new Ajax.InPlaceCollectionEditor(  
  'tobeedited', 'ajax_inplaceeditor_result.php', {  
    collection: ['one', 'two', 'three']  
  });
```

The server side component gets the new value as the parameter 'value' (POST method), and should send the new value as the body of the response.

Notes

Character encoding

The form data is sent encoded in UTF-8 regardless of the page encoding. This is as of the prototype function `Form.serialize`. [More info here](#) (web). If this doesn't work, you can use `iconv` [as outlined here](#) (web).

Remove and force InPlaceEditor

To disable the InPlaceEditor behavior later on, store it in a variable like:

```
var editor = new Ajax.InPlaceEditor('product_1',...);
```

Later you can then remove the behaviour by calling:

```
editor.dispose();
```

This way, you can enable and disable In Place Editing at will. You can also arbitrarily force it into edit mode like so:

```
editor.enterEditMode('click');
```

4.4 Slider

To make a slider element, you create a new instance of class **Control.Slider**.

Availability

script.aculo.us V1.5 and later.

Syntax

```
new Control.Slider('id_of_slider_handle','id_of_slider_track', [options]);
```

Options

| Option | Since | Default | Description |
|----------------|-------|---|---|
| axis | V1.5 | 'horizontal' | Sets the direction that the slider will move in. It should either be horizontal or vertical. |
| increment | V1.5 | 1 | Defines the relationship of value to pixels. Setting this to 1 will mean each movement of 1 pixel equates to 1 value. |
| maximum | V1.5 | length of track in pixels adjusted by increment | The maximum value that the slider will move to. For horizontal this is to the right while vertical it is down. |
| minimum | V1.5 | 0 | The minimum value that the slider can move to. For horizontal this is to the left while vertical it is up. Note: this also sets the beginning of the slider (zeroes it out). |
| alignX | V1.5 | 0 | This will move the starting point on the x-axis for the handle in relation to the track. It is often used to move the 'point' of the handle to where 0 should be. It can also be used to set a different starting point on the track. |
| alignY | V1.5 | 0 | This will move the starting point on the y-axis for the handle in relation to the track. It is often used to move the 'point' of the handle to where 0 should be. It can also be used to set a different starting point on the track. |
| sliderValue | V1.5 | 0 | Will set the initial slider value. The handle will be set to this value, assuming it is within the minimum and maximum values. |
| handleImage | V1.5 | (none) | The id of the image that represents the handle. This is used to swap out the image src with disabled image src when the slider is enabled. |
| disabled | V1.5 | (none) | This will lock the slider so that it will not move and thus is disabled. |
| handleDisabled | V1.5 | (none) | The id of the image that represents the disabled handle. This is used to change the image src when the slider is disabled. |
| values | V1.5 | (none) | Accepts an array of integers. If set these will be the only legal values for the slider to be at. Thus you can set specific slider values that the user can move the slider to. |
| range | ?? | (none) | Use the $\$(min,max)$, provided by Prototype Library |

The slider control offers some functions to let javascript update its state:

| Function | Description |
|-------------|--|
| setValue | Will update the slider's value and thus move the slider handle to the appropriate position. If you're using multiple handles, then the id should be the second paramater (the 'active' (last-used?) handle is used by default) NOTE: when using setValue, the callback functions (below) are called. |
| setDisabled | Will set the slider to the disabled state (disabled = true). |
| setEnabled | Will set the slider to the enabled state (disabled = false). |

Additionally, the options parameter can take the following callback function:

| Callback | Description |
|----------|--|
| onSlide | Called whenever the Slider is moved by dragging. The called function gets the slider value as its parameter. |
| onChange | Called whenever the Slider has finished moving or has had its value changed via the setValue function. The called function gets the slider value as its parameter. |

With both of the above, using multiple handles causes an array of their respective values to be passed to the callback. Both receive the Slider object as a second paramater.

Examples

From the author's first demo of a vertical slider. It begins disabled.

```
var s2 = new Control.Slider('slider 2','track 2', {axis:'vertical',
  minimum: 60, maximum:288, alignX: -28, alignY: -5, disabled: true,
  handleImage: 'slider_2handle', handleDisabled: 'images/vsliderhandle_gray.gif'});
```

Example of a horizontal slider that allows only 4 possible values

```
var sliderLimited = new Control.Slider('slider Limited','track Limited', {
  minimum:2, maximum:30, increment:9, alignX: -5, alignY: -5,
  values: [2, 10, 15, 30]});
```

Setting the callbacks for the first example slider, referenced by "s2"

```
s2.options.onChange = function(value){
  activeProfile.height = value;
  updateBankDescription();
  setResizeDesc();
  $('height value').innerHTML = value;
};
s2.options.onSlide = function(value){
  vidFrame1.setHeight(value);
  $('height value').innerHTML = value;
  setResizeDesc();
};
```

Here are some examples of disabling and setting the values outside of the slider

```
// continued from the above example
s2.setValue(60);
s2.setDisabled();
```

An easy way to convert the standard output decimal value to whole integers:

```
(value*100).toFixed();
```

Leaving minimum, maximum and increment undefined, adding this to your value handling will produce the integers 0-100.

Notes

The handle and track elements have to be loaded into the browser before the Slider instance is created. You should either place the script tags creating the slider after your elements in your HTML document, or use the `body.onload` callback to create the slider after everything has finished rendering.

See also - Online Slider Demo:

<http://wiki.script.aculo.us/scriptaculous/show/SliderDemo>

5

Tools

5.1 Builder

Use Builder to easily create DOM elements dynamically.

Availability

script.aculo.us V1.5 and later.

Synopsis

```
Builder.node( elementName )
Builder.node( elementName, attributes )
Builder.node( elementName, children )
Builder.node( elementName, attributes, children )
```

- **elementName** *string* The tag name for the element.
- **attributes** *object* Typical attributes are id, className, style, onclick, etc.
- **children** *array* List of other nodes to be appended as children.

If an element of the children array is plain text or numeric, it will be automatically appended as a text node.

Instead of an array, children can also be a Java Script String or numeric, to ease usage.

Special cases

- **class**: When specifying the class attribute for the node, use `className` to circumvent a Firefox bug.
- **for**: To set a for attribute (in labels) use `htmlFor`, since 'for' is a reserved word in javascript.

Examples

Creating TR and TD elements

```
table = Builder.node('table', {width:'100%',cellpadding:'2',cellspacing:'0',border:'0'});
tbody = Builder.node('tbody');
tr = Builder.node('tr',{className:'header'});
td = Builder.node('td',[ Builder.node('strong','Category')]);

tr.appendChild(td);
tbody.appendChild(tr);
table.appendChild(tbody);

$('divCat').appendChild(table);
```

You can also combine them but you need to make sure you create the tbody tag or it will not work in IE. I have tested this in FF 1.5 and IE 6. I don't have access to other browsers. The one problem that I have found is that with TR and TD elements you can not put a style tag on them as it just makes IE stop doing the Builder Function.

Simple Example

```
var element = Builder.node('p',{className:'error'},'An error has occurred');
```

creates following element:

```
<p class="error">An error has occurred</p>
```

Complex Example

```
element = Builder.node('div',{id:'ghosttrain'},[
  Builder.node('div',{className:'controls',style:'font-size:11px'},[
    Builder.node('h1','Ghost Train'),
    "testtext", 2, 3, 4,
    Builder.node('ul',[
      Builder.node('li',{className:'active', onclick:'test()'},'Record')
    ]),
  ]),
]);
```

creates (without newlines):

```
<div id="ghosttrain">
  <div class="controls" style="font-size:11px">
    <h1>Ghost Train</h1>
    testtext234
    <ul>
      <li class="active" onclick="test()">Record</li>
    </ul>
  </div>
</div>
```

Usage

In javascript code, if you want to use your new element, you can add it to an existing dom element by calling

```
$('#myExistingDomElement').appendChild(element);
```

If you want to be able to call any of prototype's extension-methods on the created node, then you need to pass it through the \$() function:

```
var new_el = Builder.node('div',{id:'some id'});
new_el = $(new_el);
new_el.hide();
```

5.2 Sound

Plays a sound file. The sound player uses native sound support on Internet Explorer, and falls back to using <embed> on other browsers, which means it uses QuickTime for most cases.

Availability

script.aculo.us V1.7.1 and later.

Syntax

```
Sound.play(soundfileUrl, [options])
```

Options

| Option | Default | Description |
|---------|----------|---|
| track | 'global' | Name of the sound container |
| replace | false | If true, will stop all currently playing sounds and play the new sound. If no track is given, it will stop all sounds in 'global' |

Examples

Play sound immediately in 'global' track

```
Sound.play('alert.mp3');
```

Stop playing all sounds in 'global' track and play the new sound

```
Sound.play('notice.mp3', {replace:true});
```

Place the given sound into 'chat' track, and play all sounds in 'chat'

```
Sound.play('info.mp3', {track:'chat'});
```

Stop playing all sounds in 'chat' track and play the new sound

```
Sound.play('error.mp3', {track:'chat', replace:true});
```

Notes

The recommended format to use is MP3.

6 Appendix

6.1 Block Elements

Block elements are HTML elements that by default are displayed as a block in the page flow. Normally, if no styles change the default behaviour, elements of the following types are considered block elements:

- * ADDRESS
- * BLOCKQUOTE
- * BODY
- * DD
- * DIV
- * DL
- * DT
- * FIELDSET
- * FORM,
- * FRAME
- * FRAMESET
- * H1, H2, H3, H4, H5, H6
- * IFRAME
- * NOFRAMES

- * OBJECT
- * P
- * OL, UL, LI
- * APPLET
- * CENTER
- * DIR
- * HR
- * MENU
- * PRE

Additionally, TABLE, TR, THEAD, TBODY, TFOOT, COL, COLGROUP, TD, TH and CAPTION are to be considered block-level elements, although they use special values for their display attributes. More detailed info available through the CSS spec.

6.2 Inline Elements

Inline elements are HTML elements that by default are placed side-by-side into lines. (Paragraphs are blocks, an italicized word is inline.)

Normally, if no styles change the default behaviour, elements of the following types are considered inline elements (some of these types are not allowed in strict HTML 4 and XHTML):

Fontstyle (use style sheets instead)

- * T
- * I
- * B
- * BIG
- * SMALL

Phrase

- * EM
- * STRONG
- * DFN
- * CODE
- * SAMP
- * KBD
- * VAR
- * CITE
- * ABBR
- * ACRONYM

Special

- * A
- * IMG
- * OBJECT
- * BR
- * SCRIPT
- * MAP
- * Q
- * SUB
- * SUP
- * SPAN
- * BDO

Form controls

```
* INPUT
* SELECT
* TEXTAREA
* LABEL
* BUTTON
```

It's not too common to cause these to be rendered as blocks, but you can do it simply by adding the CSS declaration `display: block`.

6.3 Giving Elements Layout

For [Effect.Opacity](#) based effects to work in **Internet Explorer**, you need to have give the affected elements the so-called 'layout' quality. The object that the filter is applied to must have layout before the filter effect displays.

You can give the object layout by doing one of the following:

- Setting the height or width property (doesn't work on inline elements when the browser is not in quirksmode but in standards-compliant mode)
- Setting the display property to inline-block
- Setting the position property to absolute
- Setting the writingMode property to tb-rl
- Setting the contentEditable property to true

Source: <http://msdn.microsoft.com/workshop/author/filter/reference/filters/alpha.asp>