Compiled from http://wiki.github.com/madrobby/scriptaculous
on March 10, 2009 by lincolnwebs@gmail.com

Table of Contents

Home

script.aculo.us Web 2.0 JavaScript

script.aculo.us is a set of JavaScript libraries to enhance the user interface of web sites. It provides an visual effects engine, a drag and drop library (including sortable lists), a couple of controls (Ajax-based autocompletion, in-place editing, sliders) and more. Be sure to have a look at the demos!

API Documentation and Reference

This wiki details Version 1.8.1 of the library, which is the most current version of the 1.x trunk of script.aculo.us.

Core Effects: Effect.Highlight, Effect.Morph, Effect.Move, Effect.Opacity, Effect.Scale, Effect.Parallel, Effect Queues

Combination Effects: Effect.Appear, Effect.BlindDown, Effect.BlindUp, Effect.DropOut, Effect.Fade, Effect.Fold, Effect.Grow, Effect.Puff, Effect.Pulsate, Effect.Shake, Effect.Shrink, Effect.SlideDown, Effect.SlideUp, Effect.Squish, Effect.SwitchOff, Effect.ScrollTo

Effect helpers: Effect.Transitions, Effect.Methods, Effect.tagifyText, Effect.multiple, Effect.toggle

Behaviours: Draggable, Droppables, Sortable, Form.Element.DelayedObserver

Controls: Ajax.InPlaceEditor, Ajax.InPlaceCollectionEditor, Ajax.Autocompleter, Autocompleter.Local, Slider

Miscellaneous: Builder, Sound, Unit Testing

script.aculo.us is open source. Read up on how to contribute by finding bugs, making bug reports and helping fixing them.

Help port the old, dead wiki to github and earn BIG BROWNIE POINTS! You can find a copy of the old wiki contents at http://script.aculo.us/docs. See some helpful hints for porting, and please follow the style guide!

Quick start

1. Download & install

Using script.aculo.us is easy! First, 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.

Next, 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`.

Third, load script.aculo.us in your web page. This is done by linking 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>
```

The **scriptaculous.js** loader script will automatically load in the other libraries.

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.:

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

The scripts that can be specified are: `builder`, `effects`, `dragdrop`, `controls`, and `slider`

Note that some of the scripts require that others be loaded in order to function properly.

2. Use it!

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

```
<script type="text/javascript" language="javascript">
  // <![CDATA[
  $('element_id').appear();
```

```
  // ]]>
</script>
```

This way, you won't have to worry about using characters like < and > in your Java Script code.

You can also use effects inside event handlers:

```
<div onclick="$(this).switchOff()">
  Click here if you've seen enough.
</div>
```

If you want to get tricky with it, you can pass extra options to the effect like `duration`, `fps` (frames per second), and `delay`.

```
<div onclick="$(this).blindUp({ duration: 16 })">
  Click here if you want this to go slooooow.
</div>
```

Next steps

Have a look at the demos to catch a glimpse of what you can achieve. Read the documentation. Create the next killer application!

Effects

The Visual Effects library (`effects.js`) includes all you need to add advanced JavaScript animation to your web site or application. All animation is time-based, not frame based. So, if you tell an effect to last exactly one second, it will do so, regardless of the rendering speed of the browser. And it's compatible, meaning all this stuff works on Firefox, Internet Explorer, Safari, the iPhone and many other browsers.

The library's Core Effects, Combination Effects (a combination of core effects) and Effect queues (a representation of timelines to create series of effects) are powerful tools to create appealing animations.

Quick Demo

- Click me for a demo!
- Reset the demo!

Source code of this demo

```
<div id="morph_demo" style="background:#cccccc; width:100px; height:100px;"></div>
<ul>
  <li><a href="#" onclick="$('morph_demo').morph('background:#00ff00; width:300px;'); return false;">Click me for a demo!</a></
li>
  <li><a href="#" onclick="$('morph_demo').morph('background:#cccccc; width:100px;'); return false;" >Reset the demo!</a></li>
</ul>
```

Core Effects

The seven core effects Effect.Opacity, Effect.Scale, Effect.Morph, Effect.Move, Effect.Highlight, Effect.Parallel, Effect.Tween are the foundation of the script.aculo.us Visual Effects JavaScript library.

Availability

script.aculo.us V1.0 and later.

Syntax

The basic syntax to start an effect is:

```
new Effect.EffectName(element, required-params, [options]);
```

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

`required-params` depend on the effect being called and may not be needed. Most effects do 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.

The `options` parameter is used to give any additional customization parameters to the effect. There are *general* and *effect-specific* options. It's called like this:

```
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	Description
duration	duration of the effect in seconds, given as a float. Defaults to `1.0`.
fps	Target this many frames per second. Default to `25`. Can't be higher than `100`.
transition	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`, `Effect.Transitions.flicker`, `Effect.Transitions.pulse`, `Effect.Transitions.spring`, `Effect.Transitions.none` and `Effect.Transitions.full`.
from	Sets the starting point of the transition, a float between `0.0` and `1.0`. Defaults to `0.0`.
to	Sets the end point of the transition, a float between `0.0` and `1.0`. Defaults to `1.0`.
sync	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	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	Sets the number of seconds to wait before the effect actually starts. Defaults to `0.0`.

Additionally, the options parameter also can be supplied with *callback* methods, so you can have JavaScript executed at various events while the effect is running. The callbacks are supplied with a reference to the effect object as a parameter.

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

Effect instances have the following useful properties and methods:

Variable	Description
effect.element	The element the effect is applied to.
effect.options	Holds the options you gave to the effect.
effect.currentFrame	The number of the last frame rendered.
effect.startOn, effect.finishOn	The times (in ms) when the effect was started, and when it will be finished.
effect.effects[]	On an Effect.Parallel effect, there's an `effects[]` array containing the individual effects the parallel effect is composed of.
effect.cancel()	Stop the effect as is.
effect.inspect()	Get basic debugging information about the instance.

```
var myEffect = new Effect.Opacity('my_element', {
  duration: 2.0,
  transition: Effect.Transitions.linear,
  from: 1.0,
  to: 0.5
});
//...
// example instance properties
myEffect.element; // $('my_element')
myEffect.currentFrame; // 12
myEffect.cancel(); // stop animation
```

Transitions Demo

Reset demo

Effect.Transitions.sinoidal

Effect.Transitions.linear

Effect.Transitions.reverse

Effect.Transitions.wobble

Effect.Transitions.flicker

Effect.Transitions.pulse

Effect.Transitions.spring

Effect.Transitions.none

Effect.Transitions.full

```
new Effect.Move(this, {
   x: 200,
   transition: Effect.Transitions.spring
});
```

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.

Syntax

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

Options

Option	Description
startcolor	Sets the color of first frame of the highlight. Defaults to "#ffff99" (a 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).
keepBackgroundImage	Unless this is set to true, any background image on the element will not be preserved.

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.

If you specify the `startcolor` or `endcolor` using short-form notation, as in #ccf, the effect will fail silently. Use the long-form, as in #ccccff.

Also be aware that applying an effect (without setting a `restorecolor`), to an element that already has an highlight effect in progress, will cause the `restorecolor` to be set to the elements background-color at the time of the new effect, and not the original background-color. For example, click the example below 4-5 times in quick succession, and the paragraph will stay yellow as opposed to the original white.

Demo

This paragraph exists for demo purposes. Click the link below and it will make your day bright!
Highlight me!

Source code of this demo

```
<p id="highlight_demo" style="padding:10px; border:1px solid #ccc; background:#ffffff;">
  This paragraph exists for demo purposes. Click the above link and it will make your day
bright!<br/>

  <a href="#" onclick="new Effect.Highlight(this.parentNode, { startcolor: '#ffff99',
endcolor: '#ffffff' }); return false;">
    Highlight me!
  </a>
</p>
```

Effect.Morph

This effect changes the CSS properties of an element.

Availability

script.aculo.us V1.7 and later.

Syntax

Simple:

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

Complex:

```
new Effect.Morph('error_message', {
  style: 'background:#f00; color: #fff;', // CSS Properties
  duration: 0.8 // Core Effect properties
});

new Effect.Morph('error_message', {
  style: {
    background: '#f00',
    color: '#fff'
  }, // CSS Properties
  duration: 0.8 // Core Effect properties
});

new Effect.Morph('message', {
  style: 'error', // CSS class name
  duration: 0.8 // Core Effect properties
});
```

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

Options

Option	Description
style	the target style of your element, as a string written with the standard CSS syntax, a hash, or a CSS class name.

Demo

- Click me for a demo!
- Reset the demo!

Source code of this demo

```
<div id="morph_demo" style="background:#cccccc; width:100px; height:100px;"></div>
<ul>
  <li><a href="#" onclick="$('morph_demo').morph('background:#00ff00; width:300px;'); return false;">Click me for a demo!</a></
li>
  <li><a href="#" onclick="$('morph_demo').morph('background:#cccccc; width:100px;'); return false;" >Reset the demo!</a></li>
</ul>
```

Details

Effect.Morph takes orginal 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.

Implementation Details

Because Effect.Morph queries the original values with Prototype's Element.getStyle API, it doesn't matter whether these styles are set inline or in an external stylesheet definition. Of course the effect supports all usual options, like `duration` or `transition`.

Effect.Move

This effect moves an element by modifying its position attributes.

Syntax

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

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

This will move the object 30px up and 20px to the right relative to its current position:

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

Options

Options	Description
x	integer value, either the new absolute target of the effect elements left value or the modifier of its current left value, depending on the `mode` option
y	integer value, either the new absolute target of the effect elements top value or the modifier of its current top value, depending on the `mode` option
mode	string, defaults to `'relative'`, can also be `'absolute'`, specifies if the element is moved absolutely or relative to its own position.

Demo

Source code of this demo

```
<style type="text/css">
  a#move_demo { background:#fa0000; color:#fff; padding:5px; border:1px solid #000; }
</style>

<div class="demo">
  <a href="#" id="move_demo" onclick="new Effect.Move(this, { x: 60, y: -30 }); return
false;">Click me for a demo!</a>
</div>
```

Effect.Opacity

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

Syntax

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

Examples

A simple example

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

This 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?).

Demo

- Hide this box
- Show this box

Source code of this demo

```
<div id="opacity_demo" style="width:100px; height:100px; background:#ccc;"></div>
<ul>
  <li><a href="#" onclick="new Effect.Opacity('opacity_demo', { from: 1, to: 0 }); return false;">Hide this box</a></li>
  <li><a href="#" onclick="new Effect.Opacity('opacity_demo', { from: 0, to: 1 }); return false;">Show this box</a></li>
</ul>
```

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.

Syntax

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

Effect Options

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`. Example: `scaleMode: { originalHeight: 400, originalWidth: 200 }`
scaleFrom	Sets the starting percentage for scaling, defaults to `100.0`.

Demo

Click me for Demo!

Source code of this demo

```
<a href='#' onclick="new Effect.Scale(this.parentNode, 200); return false;">Click me for
Demo!</a>
```

Effect.Tween

This effect tweens between two values and sets a property or calls a method on an object (including DOM elements); or allows for a callback method, which will be automatically bound to the object.

Availability

script.aculo.us V1.8 and later.

Syntax

```
new Effect.Tween(element, startVal, endVal, [options],propertyName);
new Effect.Tween(element, startVal, endVal, [options],callbackFunction);
```

Options

No custom options.

Examples

```
    new Effect.Tween(whatever, 5, 0, 'blech'); // sets property on the object
    new Effect.Tween('foo', 10, 20, 'innerHTML'); // sets property on the 'foo' DOM element
    new Effect.Tween('foo', 10, 20, 'update'); // method call on 'foo' DOM element
    new Effect.Tween('foo', 50, 0, { duration: 2.0 }, function(p){});
    new Effect.Tween(null, 0, 100, function(p){ scrollTo(0,p) }); // scrolls the window
```

Effect.Parallel

This is a special effect which allows 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.

Syntax

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

Examples

```
new Effect.Parallel([
  new Effect.Move(element, { sync: true, x: 20, y: -30, mode: 'relative' }),
  new Effect.Opacity(element, { sync: true, from: 1, to: 0 })
], {
  duration: 0.8,
  delay: 0.5
});
```

Options

Option	Description
sync	boolean, has to be `true` in order to prevent the subeffects from being started as soon as they get instantiated.

Notes

Don't forget to set the `sync` option to `true` for all of the subeffects or else all the effects will start immediately after they were instantiated.

Demo

- Click here for a demo!
- Reset

Source code of the demo

```
<div id="parallel_demo" style="width:150px; height:40px; background:#ccc;"></div>
<ul>
  <li><a href="#" id="animate_parallel_demo">Click here for a demo!</a></li>
  <li><a href="#" id="reset_parallel_demo">Reset</a></li>
</ul>
</div>

<script type="text/javascript">
$('animate_parallel_demo').observe('click', function(event) {
  event.stop();

  new Effect.Parallel([
    new Effect.Move('parallel_demo', { sync: true, x: 400, y: 0, mode: 'relative' }),
    new Effect.Opacity('parallel_demo', { sync: true, from: 1, to: 0 })
  ], {
    duration: 1.5
  });
});

$('reset_parallel_demo').observe('click', function(event) {
  event.stop();
```

```
  $('parallel_demo').setStyle({
    top: 0,
    left: 0,
    opacity: 1
  });
});
</script>
```

Effect Queues

What is a Queue

Let's examine how events in a queue occur: For a programmer a queue is an Array with events that are stacked on each other to be handled one by one. For example, a user hovers over a button with his cursor and then after half a second leaves the button area again. Two events have taken place, the mouseover and the mouseout. If you would handle the mouseover and the mouseout event takes place, it could disturb your current process. However you can't just ignore the event, because then you will be stuck in the status that is established by the actions you have performed for mouseover. In this situation you could add the event to a queue and every time you're done with processing an event, just check the queue for unprocessed events (and act accordingly).

How do Effects work and why do you need a Queue

The same happens when you are dealing with Effects and you face the situation where two effects can be called, both of which manipulate the same DOM object(s). script.aculo.us comes to the rescue with several techniques to manage your effects. To explain this further, we will show you how things work by example.

Let's start creating two effects

I created a page, with a `div[id="test1"]` and some content, on the following page called Example 1 . This page has an `onload` handler that executes two effects:

```
new Effect.SlideDown('test1');
new Effect.SlideUp('test1');
```

Click To Start the queue

LAST
FIRST

These effects are executed in parallel and thus one effects tries to slide the div down and the other to slide it up. Its doesn't work and you only see some flickering. However a similar thing occurs when you call `new Effect.Slide Down('test1');` with an `onclick` and then `new Effect.SlideUp('test1');` with an `onclick` to during the first effect. When you do this several times (quickly), you will see that the effects don't work anymore. The same problem also occurs in many other situations. That's why we need `Effect.Queue`.

Effect.Queue

The previous examples gave you an idea of the situations you could be facing without Queues. This must give you some motivation to find how Effect.Queue could improve your live and help you to manage your Effects.

Effect.Queue is an improved array of Effects (called Enumerable) that is global for the entire page. It gets created by extracting the global queue from Effect.Queues (we will discuss this later). So all effects you want to queue will be added to the *global* queue. You might be wondering how to add an effect to the queue? Well it's quite easy.

When you are creating an effect, you can pass several arguments. Let's look at How effects are created. The first argument is the DOM element you wish to manipulate, but you already know that. The second argument is actually an array of options that you can pass along, let's examine that for Effect.Queue. We will use Effect.SlideDown and Effect.SlideUp like in the previous examples.

```
new Effect.SlideDown('test1');
new Effect.SlideUp('test1', { queue: 'end' });
```

So you see I have added an array containing an entry `queue:` with the value `end`. This means that the Effect will be inserted at the end of the "global" Queue. All effects are always added to the queue, however without a position like 'end' they are always executed parallel to the other effects. Let's take a look at the improved example 1. You see that first the element slides down and then it slides back up again. There is also an option to insert the Effect at the front of the queue. You would use `front` instead of `end` in that case.

```
new Effect.SlideUp('test1', { queue: 'end' });
new Effect.SlideDown('test1', { queue: 'front' });
```

This would place the SlideDown before the SlideUp effect (same as the previous examples). The argument you pass as `queue:` is called a position and it can be used with a different syntax, but we will go into that later.

Remember that JavaScript is not multi-threaded, so blocks of code are executed together when they get parsed. This causes the effects to start in parallel if no queue is defined.

This works nicely, doesn't it? But what happens when you queue all kinds of effects in different places? You could have lots of effects that you would like to queue but they don't all relate to each other. This is where Effect.ScopedQueue? comes in.

Basic Effect.ScopedQueue

As explained before, the default Queue is 'global'. All effects are placed in the 'global' Queue unless you define a different scope. A scope is nothing more than grouping a set of Effects together in their own Queue. To do this, you need to pass an array instead of a string to the "queue:" option (so you have an array inside the outer array). This can only be done by using a different syntax to define the Queue position. Small and easy syntax:

```
new Effect.SlideUp('test1', { queue: 'end' });
```

Syntax that allows for more tuning:

```
new Effect.SlideUp('test1', { queue: { position: 'end' } });
```

The two examples are identical produce identical behaviour. However, the second syntax allows you to define a scope (a different queue) with the following syntax:

```
new Effect.SlideUp('menu', { queue: { position: 'end', scope: 'menuxscope' } });
new Effect.SlideUp('bannerbig', { queue: { position: 'end', scope: 'bannerscope' } });
new Effect.SlideDown('menu', { queue: { position: 'end', scope: 'menuxscope' } });
```

The `scope:` argument has defined a separate queue named 'menuxscope'. This queue could for example contain all menu-related effects. A second scope named "bannerscope" is defined, it could contain all banner-related effects. In what order are these separate queues executed? First both 1 and 2 get executed, and when 1 is done 3 is executed. Check example 3 for a preview

Just remember, if you don't define a scope, the effect is added to the default "global" scope, which can be accessed from Effect.Queue.

Limit

Sometimes too many effects get queued, for example when a user presses a button twice, or repeatedly enters and leaves a button hover area (and each time two effects are queued). This can cause a lot of distracting activity. To prevent this, we can add a "limit: n" option to the queue so no more than n effects are added to that ScopedQueue.

```
new Effect.SlideDown('menu', { queue: { position: 'end', scope: 'menuxscope',
limit: 2 } }); // #1

new Effect.Highlight('menu', { queue: { position: 'end', scope: 'menuxscope',
limit: 2 } }); // #2

new Effect.SlideUp('menu', { queue: { position: 'end', scope: 'menuxscope',
limit: 2 } });  // #3
```

#1 and #2 will be added to the queue but #3 will not.

This covers the queue for Effects, so let's move on to Effect.Queues.

Effect.Queues

So what is Effect.Queues and why do we need it? Well, for each scope an instance of Effect.Scoped Queue is created. That instance is stored in Effect.Queues. You can access it by using Effect.Queues.get('global') which is the same as Effect.Queue because the default 'global' queue is saved into Effect.Queue by script.aculo.us. However when you need to access a scope other than 'global, you need to fetch it using Effect.Queues.get('myscope') before you can manipulate it.

```
var queue = Effect.Queues.get('myscope');
```

ScopedQueue manages the 'effects', this is an internal system accessed by the Effect object. However you can, if you wish, add effects to the queue and remove them by yourself. Note that this requires some knowledge about the internal workings of the system.

ScopedQueue is just an Enumerable object. You can retrieve the effects one by one and manipulate them. Let's look at how we can 'empty a queue'.

```
var queue = Effect.Queues.get('myscope');
queue.each(function(effect) { effect.cancel(); });
```

Or maybe you want to change the interval between the effects in a queue:

```
Effect.Queues.get('myscope').interval = 100;
```

Combination Effects

All the combination effects are based on the Core Effects, and are thought of as examples to allow you to write your own effects. These are the combination effects which are included in script.aculo.us:

- 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.

Demos

You can see a demo of a specific combination effect on the effects site itself or get an overview of all effects by playing with the Combination Effects Demo.

Effect.Appear

Combination Effects > **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 `display` must be set *within the style attribute of an object*, and *not in the CSS* in the head of the document or a linked file. In other words, this Effect will not work if `display:none` is set within style tag or linked CSS file. Alternatively, `display:none` can be set using a `document.getElementById` script even if no style is set in the style attribute.

Examples

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

There's also a shortcut method offered which you can call on the element itself. Note, that this will only work on Prototype-extended elements (elements you extended at least once via calling `$(element)`).

```
$('id_of_element').appear();
$('id_of_element').appear({ duration: 3.0 });
```

Options

Options	Description
duration	float value, in seconds, defaults to `1.0`
from	float value, defaults to `0.0`, percent of opacity to start
to	float value, defaults to `1.0`, percent of opacity to end

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.

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.

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: This is fixed in V1.5_rc1.)

The opposite of Effect.Appear is Effect.Fade.

Demo

- Click here for a demo!
- Reset

Source code of this demo

```
<div id="appear_demo" style="display:none; width:80px; height:80px; background:#c2defb; border:1px solid #333;"></div>
<ul>
  <li><a href="#" onclick="$('appear_demo').appear(); return false;">Click here for a demo!</a></li>
  <li><a href="#" onclick="$('appear_demo').hide(); return false;">Reset</a></li>
</ul>
```

Effect.BlindDown

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

Examples

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

Options

Option	Description
scaleX	boolean, defaults to `false`
scaleY	boolean, defaults to `true`
scaleContent	boolean, defaults to `true`
scaleFromCenter	boolean, defaults to `false`
scaleMode	string, defaults to `'box'`, can also be `'contents'`
scaleFrom	integer value, percentage (0%–100%), defaults to `100`
scaleTo	integer value, percentage (0%–100%), defaults to `0`
duration	float value, in seconds, defaults to `1.0`

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. Example:

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

The opposite of Effect.BlindDown is Effect.BlindUp.

Demo

- Click here for a demo!
- Reset

Source code of this demo

```
<div id="blinddown_demo" style="display:none; width:80px; height:80px; background:#ccc;">
  This is some test content. This is some test content.
</div>
<ul>
  <li><a href="#" onclick="Effect.BlindDown('blinddown_demo'); return false;">Click here for a demo!</a></li>
  <li><a href="#" onclick="$('blinddown_demo').hide(); return false;">Reset</a></li>
</ul>
```

Effect.BlindRight

For horizontal action try this snippet:

```
Effect.BlindRight = function(element) {
  element = $(element);
```

```
  var elementDimensions = element.getDimensions();
  return new Effect.Scale(element, 100, Object.extend({
    scaleContent: false,
    scaleY: false,
    scaleFrom: 0,
    scaleMode: {originalHeight: elementDimensions.height, originalWidth: elementDimensions.width},
    restoreAfterFinish: true,
    afterSetup: function(effect) {
      effect.element.makeClipping().setStyle({
        width: '0px',
        height: effect.dims[0] + 'px'
      }).show();
    },
    afterFinishInternal: function(effect) {
      effect.element.undoClipping();
    }
  }, arguments[1] || { }));
};
```

Effect.BlindUp

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

Examples

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

Options

Option	Description
scaleX	boolean, defaults to `false`
scaleY	boolean, defaults to `true`
scaleContent	boolean, defaults to `true`
scaleFromCenter	boolean, defaults to `false`
scaleMode	string, defaults to `'box'`, can also be `'contents'`
scaleFrom	integer value, percentage (0%–100%), defaults to `100`
scaleTo	integer value, percentage (0%–100%), defaults to `0`
duration	float value, in seconds, defaults to `1.0`

Notes

Works safely with most Block Elements, except table rows, table bodies and table heads.

The opposite of Effect.BlindUp is Effect.BlindDown.

Demo

This is some
test content.
This is some
test content.

- Click here for a demo!
- Reset

Source code of this demo

```
<div id="blindup_demo" style="width:80px; height:80px; background:#ccc;">
  This is some test content. This is some test content.
</div>
<ul>
  <li><a href="#" onclick="Effect.BlindUp('blindup_demo'); return false;">Click here for a demo!</a></li>
  <li><a href="#" onclick="$('blindup_demo').show(); return false;">Reset</a></li>
</ul>
```

Effect.BlindLeft

For horizontal scrolling action try this snippet:

```
Effect.BlindLeft = function(element) {
  element = $(element);
  element.makeClipping();
  return new Effect.Scale(element, 0,
```

```
    Object.extend({ scaleContent: false,
      scaleY: false,
      scaleMode: 'box',
      scaleContent: false,
      restoreAfterFinish: true,
      afterSetup: function(effect) {
        effect.element.makeClipping().setStyle({
          height: effect.dims[0] + 'px'
        }).show();
      },
      afterFinishInternal: function(effect) {
        effect.element.hide().undoClipping();
      }
    }, arguments[1] || { })
  );
};
```

Effect.DropOut

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

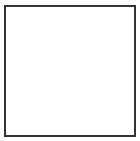
Examples

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

Notes

Works safely with most Block Elements, except tables.

Demo



- Click here for a demo!
- Reset

Effect.Fade

Combination Effects > **Effect.Fade**

Makes an element fade away and takes it out of the document flow when the effect is complete by setting the CSS `display` property to `none`. Opposite of Effect.Appear.

Examples

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

There's also a shortcut method offered which you can call on the element itself. Note, that this will only work on Prototype-extended elements (elements you extended at least once via calling `$(element)`).

```
$('id_of_element').fade();
$('id_of_element').fade({ duration: 3.0, from: 0, to: 1 });
```

Options

Options	Description
duration	float value, in seconds, defaults to `1.0`
from	float value, defaults to `1.0`, percent of opacity to start
to	float value, defaults to `0.0`, percent of opacity to end

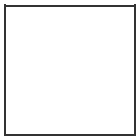
Notes

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.

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.

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: This is fixed in V1.5_rc1.

Demo



- Click here for a demo!
- Reset

Source code of this demo

```
<div id="fade_demo" style="width:80px; height:80px; background:#c2defb; border:1px solid #333;"></div>
<ul>
  <li><a href="#" onclick="$('fade_demo').fade(); return false;">Click here for a demo!</a></li>
  <li><a href="#" onclick="$('fade_demo').show(); return false;">Reset</a></li>
</ul>
```

Effect.Fold

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

Examples

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

Notes

Works safely with most Block Elements, except tables.

Demo

- Click me for a demo!
- Reset

Source code of this demo

```
<div id="fold_demo" style="width:80px; height:80px; background:#ccc;"></div>
<ul>
  <li><a href="#" onclick="Effect.Fold('fold_demo'); return false;">Click me for a demo!</a></li>
  <li><a href="#" onclick="$('fold_demo').show(); return false;">Reset</a></li>
</ul>
```

Effect.Grow

“Grows” an element into a specific direction (see demo for better understanding).

Examples

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

Options

Option	Description
direction	string, defaults to `'center'`, can also be: `'top-left'`, `'top-right'`, `'bottom-left'`, `'bottom-right'`, specifying the origin from which to “grow” the element
duration	float value, in seconds, defaults to `1.0`

Notes

Works safely with most Block Elements, except tables. You can define different durations for several div elements, and place them in a row in order to make them appear one after another.

Demo

- Click me for a demo!
- Reset

Source code of this demo

```
<div id="grow_demo" style="display:none; width:80px; height:80px; background:#ccc;"></div>
<ul>
  <li><a href="#" onclick="Effect.Grow('grow_demo'); return false;">Click me for a demo!</a></li>
  <li><a href="#" onclick="$('grow_demo').hide(); return false;">Reset</a></li>
</ul>
```

Effect.Puff

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

Examples

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

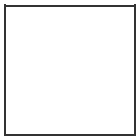
Options

Options	Description
duration	float value, in seconds, defaults to `1.0`
from	float value, defaults to `0.0`, percent of animation to start
to	float value, defaults to `1.0`, percent of animation to end

Notes

Works safely with most block elements, except tables.

Demo



- Click here for a demo!
- Reset

Source code of the demo

```
<div id="puff_demo" style="width:80px; height:80px; background:#c2defb; border:1px solid #333;"></div>
<ul>
  <li><a href="#" onclick="new Effect.Puff('puff_demo'); return false;">Click here for a demo!</a></li>
  <li><a href="#" onclick="$('puff_demo').setStyle({ display: 'block', opacity:1, width:'80px', height:'80px' }); return
false;">Reset</a></li>
</ul>
```

Effect.Pulsate

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

Examples

```
Effect.Pulsate('id_of_element');
Effect.Pulsate('id_of_element', { pulses: 5, duration: 1.5 });
```

Options

Option	Description
duration	float value, in seconds, defaults to `2.0`
from	float value, defaults to `0.0`, the minimal `opacity` during the pulsate, in a value between `0.0` and `1.0`. For example, use `0.7` for a mild pulsate
pulses	integer value, defaults to `5`, the amount of pulses within the duration time

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?.

In Firefox, at least, if the element that you Pulsate has a `:hover` CSS psuedo class and you mouse over it while it's pulsating, the effect will stop.

Demo

Click me to pulsate!

Source code of this demo

```
<div id="pulsate_demo" style="width:150px; height:40px; background:#ccc; text-align:center;">
  <a href="#" onclick="Effect.Pulsate('pulsate_demo'); return false;" style="line-height:40px;">Click me to pulsate!</a>
</div>
```

Effect.ScrollTo

Scrolls to a specific place in the page.

Examples

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

Options

Option	Description
duration	float value, in seconds, defaults to `1.0`
offset	integer value, in pixels, defaults to `0`

Notes

Demo

Click me to scroll to the top of the article

Source code of this demo

```
<a href="#" onclick="Effect.ScrollTo('article_top'); return false;">Click me to scroll to the top of the article</a>
```

Effect.Shake

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

Examples

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

Options

Option	Description
duration	float value, in seconds, defaults to `0.5`
distance	integer value, defaults to 20, the number of pixels to move horizontally

Notes

Works safely with most Block Elements, except tables.

Demo

Click me to shake!

Source code of this demo

```
<div id="shake_demo" style="width:150px; height:40px; background:#ccc; text-align:center;>
  <a href="#" onclick="new Effect.Shake('shake_demo'); return false;" style="line-height:40px;">Click me to shake!</a>
</div>
```

Effect.Shrink

“Shrinks” an element into a specific direction (see demo for better understanding), hides it when the effect is complete.

Examples

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

Options

Option	Description
direction	string, defaults to `'center'`, can also be: `'top-left'`, `'top-right'`, `'bottom-left'`, `'bottom-right'`, the direction to “shrink” the element to
duration	float value, in seconds, defaults to `1.0`

Notes

Works safely with most Block Elements, except tables. You can define different durations for several div elements, and place them in a row in order to make them disappear one after another.

Demo

- Click me for a demo!
- Reset

Source code of this demo

```
<div id="shrink_demo" style="width:80px; height:80px; background:#ccc;"></div>
<ul>
  <li><a href="#" onclick="Effect.Shrink('shrink_demo'); return false;">Click me for a demo!</a></li>
  <li><a href="#" onclick="$('shrink_demo').show(); return false;">Reset</a></li>
</ul>
```

Effect.SlideDown

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

Examples

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

Options

Option	Description
scaleX	boolean, defaults to `false`
scaleY	boolean, defaults to `true`
scaleContent	boolean, defaults to `true`
scaleFromCenter	boolean, defaults to `false`
scaleMode	string, defaults to `'box'`, can also be `'contents'`
scaleFrom	integer value, percentage (0%–100%), defaults to `100`
scaleTo	integer value, percentage (0%–100%), defaults to `0`
duration	float value, in seconds, defaults to `1.0`

Notes

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>
```

The target element should not have `padding` set, otherwise you'll see the effect "bouncing". (See discussion)

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.

The opposite of Effect.SlideDown is Effect.SlideUp.

Demo

- Click here for a demo!
- Reset

Source code of this demo

```
<div id="slidedown_demo" style="display:none; width:80px; height:80px; background:#ccc; text-align:center;">
  <div>
    This is some test content. This is some test content.
  </div>
</div>
<ul>
  <li><a href="#" onclick="Effect.SlideDown('slidedown_demo'); return false;">Click here for a demo!</a></li>
  <li><a href="#" onclick="$('slidedown_demo').hide(); return false;">Reset</a></li>
</ul>
```

Effect.SlideUp

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

Examples

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

Options

Option	Description
scaleX	boolean, defaults to `false`
scaleY	boolean, defaults to `true`
scaleContent	boolean, defaults to `true`
scaleFromCenter	boolean, defaults to `false`
scaleMode	string, defaults to `'box'`, can also be `'contents'`
scaleFrom	integer value, percentage (0%–100%), defaults to `100`
scaleTo	integer value, percentage (0%–100%), defaults to `0`
duration	float value, in seconds, defaults to `1.0`

Notes

Include a second `div` element, wrapping the contents of the outer `div`. So, if you call `new Effect.SlideUp('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.

The opposite of Effect.SlideUp is Effect.SlideDown.

Demo

This is some
test content.
This is some
test content.

- Click here for a demo!

- Reset

Source code of this demo

```
<div id="slideup_demo" style="width:80px; height:80px; background:#ccc; text-align:center;">
  <div>
    This is some test content. This is some test content.
  </div>
</div>
<ul>
  <li><a href="#" onclick="Effect.SlideUp('slideup_demo'); return false;">Click here for a demo!</a></li>
  <li><a href="#" onclick="$('slideup_demo').show(); return false;">Reset</a></li>
</ul>
```

Effect.Squish

Reduce the element to its top-left corner.

Examples

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

Notes

Works safely with most Block Elements, except tables.

Demo

- Click me for a demo!
- Reset

Source code of this demo

```
<div id="squish_demo" style="width:80px; height:80px; background:#ccc;"></div>
<ul>
  <li><a href="#" onclick="Effect.Squish('squish_demo'); return false;">Click me for a demo!</a></li>
  <li><a href="#" onclick="$('squish_demo').show(); return false;">Reset</a></li>
</ul>
```

Effect.SwitchOff

Gives the illusion of a TV-style switch off.

Examples

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

Notes

Works safely with most Block Elements, except tables.

Demo

- Click here for a demo!
- Reset

Source code of this demo

```
<div id="switchoff_demo" style="width:80px; height:80px; background:#ccc;"></div>
<ul>
  <li><a href="#" onclick="Effect.SwitchOff('switchoff_demo'); return false;">Click here for a demo!</a></li>
  <li><a href="#" onclick="$('switchoff_demo').show(); return false;">Reset</a></li>
</ul>
```

Effect.Transitions

Put simply, a transition in script.aculo.us is a function which transforms an input value to another value and returns it. `Effect.Transitions` is a collection of 9 of those functions which can be used to achieve interesting variations on any effect.

Example

A transition can be specified by using an effects `transition` option.

```
new Effect.Move('id_of_element', {
  x: 200, y: 0, mode: 'relative',
  transition: Effect.Transitions.spring
});
```

Demo

To get a better understanding of how each of script.aculo.us' included transitions work, play around with the following demo.

Choose a transition: linear and animate the demo!

Effect.Methods

Effect.Methods is a mixin of various effects and helper functions for DOM elements. These methods can be accessed through the `$()` function or through any `Element` object.

Syntax

```
$(element).methodName(arguments);
```

The following methods are included within `Effect.Methods` and can be used as a shortcut for effects, helpers, etc you might want to call on an element:

Method Name	Description
morph(element, style, options)	starts an `Effect.Morph` on the element, takes `style` (the styles string or hash for the effect) as the first parameter, takes an optional second parameter which is the options hash for the effect
visualEffect(element, effect, options)	specify any of the supported effects and pass options
getInlineOpacity	a shortcut for Element#getInlineOpacity?
forceRerendering	a shortcut for Element#forceRerendering?
setContentZoom	a shortcut for Element#setContentZoom?
collectTextNodes	a shortcut for Element#collectTextNodes?
collectTextNodesIgnoreClass	a shortcut for Element#collectTextNodesIgnoreClass?
getStyles	a shortcut for Element.getStyles?

Additional to this set of helper function, also all of scriptaculous' Combination Effects and Core Effects are available within `Effect.Methods` and take the an `options` hash as an optional parameter:

Method Name	Description
fade	a shortcut for Effect.Fade
appear	a shortcut for Effect.Appear
grow	a shortcut for Effect.Grow
shrink	a shortcut for Effect.Shrink
fold	a shortcut for Effect.Fold
blindUp	a shortcut for Effect.BlindUp
blindDown	a shortcut for Effect.BlindDown
slideUp	a shortcut for Effect.SlideUp
slideDown	a shortcut for Effect.SlideDown
pulsate	a shortcut for Effect.Pulsate
shake	a shortcut for Effect.Shake
puff	a shortcut for Effect.Puff
squish	a shortcut for Effect.Squish
switchOff	a shortcut for Effect.SwitchOff
dropOut	a shortcut for Effect.DropOut
highlight	a shortcut for Effect.Highlight

Examples

```
$('id_of_element').highlight();
$('id_of_element').visualEffect('Opacity', { from: 1.0, to: 0.7, duration: 0.5 });
$('id_of_element').morph({ height: '50px', width: '200px' }, { duration: 0.5 });
$('id_of_element').fade({ delay: 0.3, duration: 0.8 });
```

Effect.multiple

`Effect multiple` takes an array of elements and performs a given effect for each element. If one specific element is passed instead of an array of elements, the specific elements child nodes will be used for the effect. Each subsequent effect will start by default with a slight delay depending on the `speed` option.

Syntax

```
Effect.multiple([element1, element2, element3, …], Effect); // takes an array of elements
Effect.multiple(element, Effect); // also takes a specific element and will use its childNodes
```

Options

Additional to a typical effects options, `Effect.multiple` also takes these options:

Option	Description
speed	float value, defaults to `0.1`, a delay offset for each subsequent effect
delay	float value, defaults to `0.0`, the effects start delay

Examples

```
Effect.multiple('id_of_element', Effect.Fade); // performs an Effect.Fade for each childNode of the given element
Effect.multiple(['id_one', 'id_two'], Effect.Puff); // performs an Effect.Puff for each element in the given array
Effect.multiple('id_of_element', Effect.Fade, { speed: 0 }); // instantely performs an Effect.Fade for each childNode of the
given element
```

Demo

Click somewhere on the list for a demo. Reset the demo.

- This is
- what you
- can do
- with
- Effect.multiple

Source code of this demo

```
<style type="text/css">
  ul#multiple_demo { cursor:pointer; }
  ul#multiple_demo li { font-size:16px; }
</style>

<div class="demo">
  Click somewhere on the list for a demo. <a href="#" id="reset_link">Reset the demo</a>.
  <ul id="multiple_demo" class="on">
    <li>This is</li>
    <li>what you</li>
    <li>can do</li>
    <li>with</li>
    <li>Effect.multiple</li>
  </ul>
</div>

<script type="text/javascript">
  (function() {
    $('multiple_demo').observe('click', fadeListItems);
    $('reset_link').observe('click', reset);

    var listItems = $('multiple_demo').select('li');

    function fadeListItems() {
```

```
      Effect.multiple(listItems, Effect.Fade);
    }

    function reset(event) {
      event.stop();
      Effect.multiple(listItems, Effect.Appear);
    }
  })();
</script>
```

Effect.tagifyText

Effect.tagifyText transforms any text string contained in a specific element into a chain of `span` elements, each containing one character of the string.

Syntax

```
Effect.tagifyText(element);
```

Demo

Tagify the text which is contained in the following `div`.

Go! Click on the button! Tagify me!

Source code of this demo

```
<style type="text/css">
  div#tagify_demo { padding:10px 0; }
  button#tagify_button { padding:3px; }
  div#tagify_element { border:1px solid #3071cc; padding:10px; margin-top:10px;}
  div#tagify_element span { border:1px solid #df7418; padding:5px; }
</style>

<div class="demo" id="tagify_demo">
  <button id="tagify_button" onclick="Effect.tagifyText('tagify_element'); return false;">Tagify the text</button> which is
contained in the following <code>div</code>.
  <div id="tagify_element">Go! Click on the button! Tagify me!</div>
</div>
```

Effect.toggle

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

Syntax

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

`element` can be either a string containing the `id` of the element, or a JavaScript 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.

Examples

```
Effect.toggle('id_of_element', 'appear');
Effect.toggle('id_of_element', 'slide', { delay: 0.5 });
Effect.toggle('id_of_element', 'blind', { duration: 2.0 });
```

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>
```

Demo

Toggle via `'blind'`

Toggle the following paragraph with a 'blind'-effect

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Toggle via `'slide'`

Toggle the following paragraph with a 'slide'-effect

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Toggle via `'appear'`

Toggle the following paragraph with an 'appear'-effect

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Source code of the demo

```
<a href="#" onclick="Effect.toggle('toggle_appear', 'appear'); return false;">Toggle the following paragraph with an 'appear'-
effect</a>
<div id="toggle_appear" style="background:#ccc;">
  <div>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute
irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
  </div>
</div>
```

```
  </div>
</div>
```

Draggable

Behaviours > **Draggable**

To make an element draggable, create a new instance of class *Draggable*. For additional built-in functionality, make a Sortable instead.

There is also a class named Draggables that exposes functions for observing drag actions.

Draggables become much more useful when you use them with droppables, which are the areas that you can drag draggables to.

Syntax

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

Options

Option	Since	Description
handle	1.0	string or DOM reference, not set by default. Sets whether the element should only be draggable by an embedded handle. The value must be an element reference or element id.
handle	1.5.	string or DOM reference, not set by default. 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	1.0	boolean, defaults to `false`. If set to `true`, the element returns to its original position when the drags ends.
revert	1.5	boolean or function reference, defaults to `false`. Revert can also be an arbitrary function reference, called when the drag ends. Specifying 'failure' will instruct the draggable not to revert if successfully dropped in a droppable.
snap	1.5	If set to false no snapping occurs. Otherwise takes one of the following forms – `Δi`: one delta value for both horizontal and vertical snap, `[Δx, Δy]`: delta values for horizontal and vertical snap, `function(x, y, draggable_object) { return [x, y]; }`: a function that receives the proposed new top left coordinate pair and returns the coordinate pair to actually be used.
zindex	1.5	integer value, defaults to 1000. The css `z-index` of the draggable item.
constraint	1.0	string, not set by default. If set to `'horizontal'` or `'vertical'` the drag will be constrained to take place only horizontally or vertically.
ghosting	??	boolean, defaults to `false`. Clones the element and drags the clone, leaving the original in place until the clone is dropped.
starteffect	??	Effect, defaults to Effect.Opacity. Defines the effect to use when the draggable starts being dragged.
reverteffect	??	Effect, default to Effect.Move. Defines the effect to use when the draggable reverts back to its starting position.
endeffect	??	Effect, defaults to Effect.Opacity. Defines the effect to use when the draggable stops being dragged.
scroll	??	string or DOM reference, not set by default. Specifies the element which will scroll when you get to the boundary. By default this is turned off.
scrollSensitivity	??	integer value, defaults to 20 pixels. Minimum distance from the element boundary to start scrolling.

Additionally, the options parameter accepts any of the following callback functions:

Callback	Description
onStart	Called when a drag is initiated.
onDrag	Called repeatedly a mouse moves, if mouse position changed from previous call.
change	Called just as onDrag (which is the preferred callback). Gets the Draggable instance as its parameter.
onEnd	Called when a drag is ended.

Except for the `change` callback, each of these callbacks accepts two parameters: the Draggable object, and the mouse event object.

Examples

```
// revert
new Draggable('product_1', { revert: true });
```

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

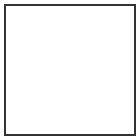
To disable draggables later on, store it in a variable like:

```
var mydrag = new Draggable('product_1', { revert: true });
// then destroy it when you don't need it anymore
mydrag.destroy();
```

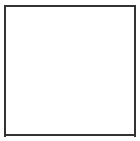
This way, you can enable and disable dragging at will.

Demo

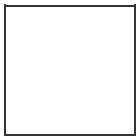
A demo with the default options



A demo with `{ revert: true, snap: [40, 40] }` set as options



A demo with `{ scroll: window }` set as options



Source code of the demo

Demo 1 (default options)

```
<div id="drag_demo_1" style="width:100px; height:100px; background:#7baaed; border:1px solid #333;"></div>
  <script type="text/javascript">
    new Draggable('drag_demo_1');
  </script>
```

Demo 2 (with `revert` and `snap` set)

```
<div id="drag_demo_2" style="width:100px; height:100px; background:#fff85d; border:1px solid #333;"></div>
  <script type="text/javascript">
    new Draggable('drag_demo_2', { revert: true, snap: [40, 40] });
  </script>
```

Demo 3 (with `scroll` set)

```
<div id="drag_demo_3" style="width:80px; height:80px; cursor:move; background:#88da5d; border:1px solid #333;"></div>
  <script type="text/javascript">
    new Draggable('drag_demo_3', { scroll: window });
  </script>
```

Draggables Object

Are you looking for how to make an element draggable?

Draggables Object

The Draggables object is a global helper object. In most cases you will not need to use or modify the Draggables object, except when adding/removing custom drag observers.

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

onStart()	called after drag begins
onDrag()	called on each mousemove during a drag
onEnd()	called when drag is finished

The parameters passed to these three events are

- `eventName`, one of `'onStart'`, `'onEnd'`, `'onDrag'`
- `draggable`, a reference to the Draggable?. The `draggable.element` property is a reference to the DOM element being dragged.
- `event`, the DOM Event object

Droppables

Behaviours > **Droppables**

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

Options	Description
accept	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	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	if set, the Droppable will have this additional CSS class when an accepted Draggable is hovered over it.
overlap	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	boolean, defaults to `true`, stops processing hovering (don't look for other Droppables that are under the Draggable)

Callbacks

Callback	Description
onHover	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	Called whenever a Draggable is released over the Droppable and the Droppable 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.

Examples

```
Droppables.add('shopping_cart', {
  accept: 'products',
  onDrop: function(element) {
    $('shopping_cart_text').update('Dropped the ' + element.alt + ' on me.');
  }
});
```

Removing Droppables

When you delete a Node in the HTML Code that was droppable you will not be able to use any draggable Element. Before you delete a droppable element be sure to remove it first from the Droppables list:

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

Notes

Nested Droppables

If you're adding droppable elements that have other droppable elements inside of them, make sure that you add the droppables in reverse order of the nesting (i.e. most inner droppable first, then second most inner droppable second). For example you have a nested list:

```
<ul>
  <li>Parent
```

```
    <ul>
      <li>Child 1</li>
      <li>Child 2</li>
      <li>Child 3</li>
    </ul>
  </li>
</ul>
```

Make sure that the children get defined as droppables *before* the parent is.

Demo

Drag me!

Drop here!

Source code of this demo

```
<style type="text/css">
  div#droppable_container {
    height: 140px;
    width: 400px; }
  div#draggable_demo {
    width: 60px;
    height: 60px;
    cursor: move;
    background: #9fcfba;
    border: 1px solid #666;
    text-align: center;
    position: relative;
    top: 30px;
    line-height: 50px; }
  div#droppable_demo {
    width: 160px;
    height: 120px;
    background: #fff;
    border: 5px solid #ccc;
    text-align: center;
    position: relative;
    top: -60px;
    left: 140px;
    line-height: 100px; }
  div#droppable_demo.hover {
    border: 5px dashed #aaa;
    background:#efefef; }
</style>

<div class="demo" id="droppable_container">
  <div id="draggable_demo" class="draggable">
    Drag me!
  </div>

  <div id="droppable_demo">
    Drop here!
  </div>
</div>

<script type="text/javascript">
  new Draggable('draggable_demo', {
    revert: true
  });

  Droppables.add('droppable_demo', {
    accept: 'draggable',
```

```
    hoverclass: 'hover',
    onDrop: function() { $('droppable_demo').highlight(); }
  });
</script>
```

Sortable

Behaviours > **Sortable**

A Sortable is a quick way to initialize many Draggable elements in a container element. When you create a new Sortable, it takes care of the creation of the corresponding draggable Droppables.

Syntax

Use `Sortable.create('id_of_container',[options]);` to create new Sortables. See Sortable.create.

Object

_Property/Method	Description
SERIALIZE_RULE	`(RegExp) /^[^_\-](?:[A-Za-z0-9\-\_]*)[_](.*)$/`
sortables	(Object) { }
options (element)	Internal function
destroy (element)	Destroys sortable
create (element, options)	Creates sortable
findElements (element, options)	Internal function
onHover (element, dropon, overlap)	Internal function, which may be overridden through the options parameter on creation
onEmptyHover (element, dropon, overlap)	Internal function
unmark()	Internal function
mark()	Internal function
tree (element)	
sequence (element)	
setSequence (element, new_sequence)	
serialize (element)	

Demos

See Sortable Lists Demo, Ghostly Sortable Demo? and Sortable Floats Demo?.

Creating sortables

See Sortable.create.

Disabling sortables

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

A call to `Sortable.create` implicitly calls `Sortable.destroy` if the referenced element was already a Sortable.

Functions

Function	Description
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. See Sortable.serialize
Sortable.sequence	The Sortable object also provides a function to get the values in an sequence array object. See Sortable.sequence

Tutorials

Use PHP and Ajax call to interact with Sortable

A short tutorial on using Sortables prepared for the Linux Users Group, Villafranca, Italy.

Sortable.create

Behaviours > Sortable > **create**

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

Syntax

Use `Sortable.create('id_of_container',[options]);` to create new Sortables.

Options

Option	Since	Description
tag	v1.0	**Default:** 'li' 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	**Default:** (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	**Default:** 'vertical' Either 'vertical' or 'horizontal'. For floating sortables or horizontal lists, choose 'horizontal'. Vertical lists should use 'vertical'.
constraint	v1.0	**Default:** 'vertical' Restricts the movement of Draggables, see the constraint option of Draggables.
containment	v1.0	**Default:** (only within container) Enables dragging and dropping between Sortables. Takes an array of elements or element-ids (of the containers). Important note: To ensure that two way dragging between containers is possible, place all Sortable.create calls after the container elements.
format	v?	**Default:** `/^[^_\-](?:[A-Za-z0-9\-\_]*)[_](.*)$/` The format that the id is computed from each item-id
handle	v1.0	**Default:** (none) Makes the created Draggables? use handles, see the handle option on Draggables.
hoverclass	v1.1b1	**Default:** (none) Gives the created Droppables a hoverclass (see there).
ghosting	v1.5	**Default:** 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	**Default:** 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	v1.5.2	**Default:** 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; Update: Scrolling the whole document does work (at least on Safari 3.2 (Mac), IE7 and Firefox). Use scroll: window
scrollSensitivity	v?	**Default:** 20 Will start scrolling when element is x pixels from the bottom, where x is the scrollSensitivity.
scrollSpeed	v?	**Default:** 15 Will scroll the element in increments of scrollSpeed pixels.
tree	v1.6.1	**Default:** false If true, sets sortable functionality to elements listed in treeTag

treeTag	v1.6.1	**Default:** ul The element type tree nodes are contained in.

Callbacks

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.

If you want your sortable list to be scrollable, wrap the list in a div and set the div to scrollable as apposed to making the ul element scrollable. Also, in IE you must set "position:relative" on the scrollable div.

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.

Patches

Marking the Drop Zone : Having a marker in the empty place where you will drop (for versions: 1.6.x, 1.7.0, 1.8.0 and 1.8.1)

Have a look at this page for the patch and how to modify scriptaculous to have a drop zone marker.

Tankut Koray

Sortable.sequence

The Sortable object also provides a function to get an sequence array of the id's.

Syntax

```
id_array = Sortable.sequence('id_of_container');
```

Example

Taken from the Puzzle Demo

```
(function() {
  var p = $('puzzle'), info = $('puzzleinfo'), moves = 0;

  Sortable.create('puzzle', {
    tag: 'img', overlap: 'horizontal', constraint: false,
    onUpdate: function() {
      info.update('You\'ve made ' + (++moves) + ' move' + (moves > 1 ? 's' : ''));
      if (Sortable.sequence('puzzle').join('') == '123456789')
         info.update('You\'ve solved the puzzle in ' + moves + ' moves!').morph('congrats');
    }
  });
})();
```

Sortable.serialize

Behaviours > 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:

```
poststring = Sortable.serialize('id_of_container',[options]);
// poststring now contains key[]=value pairs separated by &
```

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

```
id="string_identifier"

//Example
<ol id="container_id">
  <li id="image_1">Item 1</li>
  <li id="image_2">Item 1</li>
  <li id="image_3">Item 1</li>
</ol>
```

Only the identifier part of the id attribute will be serialized. If you want to use an other form of id attributes, you need to implement your own serialization.

Options

Option	Since	Description
tag	v1.0	The kind of tag (of the child elements of the container) that will be serialized.
name	v1.0	The name of the key that will be used to create the key/value pairs for serializing in HTTP GET/POST format (that is, key[] =value&key[]=value ...)

Example (from the Sortable Ghosted example)

```
<style>
  #content #testlist {
      list-style-type:none;
      margin:0;
      padding:0;
   }
   #content #testlist li {
     width:200px;
     font:13px Verdana;
     margin:0;
     margin-left:20px;
     padding-left:20px;
     padding:4px;
     cursor:move;
   }
  div.dropmarker {
      height:6px;
      width:200px;
      background: url(/images/dropmarker.png) left top;
      margin-top:-3px;
      margin-left:-5px;
      z-index:1000;
      overflow: hidden;
   }
</style>

<ol id="testlist">
  <li id="image_1">Lorem ipsum dolor</li>
  <li id="image_2">sit amet</li>
  <li id="image_3">consectetur adipisicing</li>
  <li id="image_4">elit</li>
  <li id="image_5">sed do eiusmod</li>
  <li id="image_7">ut labore et dolore</li>
  <li id="image_6">tempor incididunt</li>
```

```
  <li id="image_8">magna aliqua</li>
</ol>

<script type="text/javascript" language="javascript">
  Sortable.create('testlist',{ghosting:true,constraint:false})
  alert(Sortable.serialize('testlist'));
</script>
```

Clarification: In this example, the output of the serialization will only give the numbers after the underscore in the list item IDs.

Tutorials

A short tutorial on using Sortables.serialize

Sortables

Behaviours > **Sortable**

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.

Creating Sortables

Syntax

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

Options

Option	Description
tag	string, represents a tagname, defaults to `'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	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	string, defaults to `'vertical'`, can also be `'horizontal'`. For floating sortables or horizontal lists, choose `'horizontal'`. Vertical lists should use `'vertical'`.
constraint	string, defaults to `'vertical'`, restricts the movement of Draggables, see the constraint option of Draggables?.
containment	Enables dragging and dropping between Sortables. Takes an array of elements or element-ids (of the containers). Important note: To ensure that two way dragging between containers is possible, place all Sortable.create calls after the container elements.
handle	Makes the created Draggables use handles, see the handle option on Draggables?.
hoverclass	Gives the created Droppables a hoverclass (see Droppables).
ghosting	boolean, defaults to `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.
dropOnEmpty	boolean, defaults to `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	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.
scrollSensitivity	Will start scrolling when element is x pixels from the bottom, where x is the scrollSensitivity.
scrollSpeed	Will scroll the element in increments of scrollSpeed pixels.
tree	boolean, defaults to `false`, if `true`, sets sortable functionality to elements listed in `treeTag`
treeTag	string, defaults to `'ul'`, the element type tree nodes are contained in.

Callbacks

Callback	Description
onChange	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	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

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.

If you want your sortable list to be scrollable, wrap the list in a `div` and set the `div` to scrollable as apposed to making the `ul` element scrollable. Also, in *IE* you must set `position:relative` on the scrollable `div`.

Serializing Sortables

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.

Syntax

```
var poststring = Sortable.serialize('id_of_container',[options]);
// poststring now contains key[]=value pairs separated by &
```

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

```
id="string_identifier"
// example: id="image_1"
```

Only the *identifier* part of the `id` attribute will be serialized. If you want to use an other form of id attributes, you need to implement your own serialization.

Options

Option	Description
tag	string, defaults to the `tag` originally used when calling `Sortable.create`, Sets the kind of tag (of the child elements of the container) that will be serialized.
name	string, defaults to `id` of the 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, `key[]=value&key[]=value` ...)

Disabling Sortables

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.

Syntax

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

Notes

A call to `Sortable.create` implicitly calls on `Sortable.destroy` if the referenced element was already a *Sortable*.

Form.Element.DelayedObserver

A delayed observer works like a normal observer, but the triggered callback is delayed. Every time the observed event is observed the internal timer is reset. Once the internal timer reaches the time set in the second parameter the callback is fired.

Syntax

```
Form.Element.DelayedObserver(element, time, callback)
```

element: the form element to observe time: the internal timer after which the callback is fired (in seconds) callback: the callback function.

Example code

```
new Form.Element.DelayedObserver($('inputbox'), 0.5, function(){
     //do your stuff here, like an ajax request
});
```

Controls

The in-place “text edit” testing allows for Flickr-style AJAX-backed “on-the-fly” textfields. See the documentation

- Ajax.InPlaceEditor
- Ajax.InPlaceCollectionEditor

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

- Ajax.Autocompleter
- Autocompleter.local
- Slider

Ajax.InPlaceCollectionEditor

Introduction

This constructor generates a Flickr-style AJAX-based “on-the-fly” fields with a select box instead of Ajax.InPlaceEditor text fields.

Syntax

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

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. Within that hash of options should be a field called collection that is an array of values to place inside your select box.

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.

If the `collection` parameter (or the result of the `loadCollectionURL`) is a one-dimensional array, the `option` tag’s `value` attribute will be the same as the label. For explicit value attributes, use a two dimensional array for the `collection` where the second dimension is a key value pair such as `[['key', 'value'], ['key', 'value']]`.

Options

The Ajax.InplaceCollectionEditor takes all the options as Ajax.InPlaceEditor plus:

Name	*since*	*default*	*Description*
`collection`	V??	none	Array of static elements to be displayed as options (in JSON format)
`loadCollectionURL`	V1.5	null	Loads values and tag texts for the `<option>` tags
`loadingCollectionText`	V??	null	Text to be displayed while the collection is loading
`loadingClassName`	V??	null	Class applied to form while the collection is loading

Static Collection

The values for the collection of options are specified as an array when you invoke the Ajax.InPlaceCollectionEditor. Each element in the array will be rendered as an `<option>` in a `<select>` element for the user to choose from.

```
new Ajax.InPlaceCollectionEditor( element_1, '/url_to_validate_and_save_selection/', {
  collection: ['value one', 'value two', 'value three']
});

new Ajax.InPlaceCollectionEditor( element_2, '/url_to_validate_and_save_selection/', {
  collection: [['key_1', 'value one'], ['key_2', 'value two'], ['key_3', 'value three']]
});
```

Dynamic Collection

The collection is loaded in the same format but is loaded as the response from a URL in the `loadCollectionURL` parameter. The response from that URL should be an array in JSON format.

In Rails, it can be done like this:

```
# /url_to_load_collection /
def my_collection
  @collection = SomeModel.all
  respond_to do |format|
    format.json { render :json => @collection.map{ |c| [c.id, c.title] } }
  end
end
```

```
new Ajax.InPlaceCollectionEditor( element_3, '/url_to_validate_and_save_selection/', {
  loadCollectionURL: '/url_to_load_collection/'
});
```

```
  loadCollectionURL: '/url_to_load_collection/'
});
```

Demo

```
<p id="editme">Click me, click me!</p>
<script type="text/javascript">
 new Ajax.InPlaceCollectionEditor(
   'editme',
   '/demoajaxreturn.html',
   { collection: ['one','two','three'],
     ajaxOptions: {method: 'get'} }
  );
</script>
```

three

(should autoselect “three”)

Examples

You may want to send the Ajax request directly to `update` method in Rails to be RESTful. You can do it like this, using the `callback` option:

```
<script>
new Ajax.InPlaceCollectionEditor(
  '<%= dom_id(person) %>_role_title',
  '<%= person_path(person, :authenticity_token => form_authenticity_token) %>',
  { loadCollectionURL: '<%= formatted_roles_people_path(:json,
                           :authenticity_token => form_authenticity_token) %>',
    callback: function(form, value) {
      return 'value=' + encodeURIComponent(value) +
             '&person[role_id]=' + encodeURIComponent(value) +
             '&_method=PUT' // Fix the HTTP METHOD for the update action!!!
    },
    ajaxOptions: { method: 'post' }
  }
);
</script>
```

Ajax.InPlaceEditor

Introduction

The in-place “text edit” testing allows for Flickr-style AJAX-backed “on-the-fly” textfields. See the documentation on Ajax.InPlaceEditor and Ajax.InPlaceCollectionEditor

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.
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.

Options

Name	*since*	*default*	*Description*
`okControl`	V??	“button”	What type of ok button to use in edit mode, or none at all (button, link, false)
`cancelControl`	V??	“link”	What type of cancel button to use in edit mode, or none at all (button, link, false)
`okText`	V1.5	“ok”	The text of the submit button that submits the changed value to the server
`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 plus ‘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.
`externalControlOnly`	V1.5	false	Whether or not to disable onclick editing so that only an external control can activate editable mode
`rows`	V1.5	1	The row height of the input field (anything greater than 1 uses a multiline textarea for input)
`onComplete`	V1.6	“function(transport, element) {new Effect.Highlight (element, {startcolor: this.options.highlightcolor});}”	Code run if update successful with server
`onFailure`	V1.6	“function(transport) {alert(“Error communicating with the server: ” + transport.responseText.stripTags());}”	Code run if update failed with server
`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 in_place_edit form when the input tag loses focus.
`ajaxOptions`	V1.5	{}	Options specified to all AJAX calls (loading and saving text), these options are passed through to the prototype AJAX classes.

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.

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 on here.

If this doesn’t work, you can use iconv as outlined here.

Removing the behavior

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

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

Demo1 (Single Line Edit Mode)

note: the demo doesn’t actually save its changes, as `demoajaxreturn.html` is not a valid url.

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

Click me, click me!

Demo2 (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>
```

Click me to edit this nice long text.

How to specify a different name for the parameter sent to the server

Add a callback function which is supposed to return the parameters that is sent to the server. Like this:

```
new Ajax.InPlaceEditor('id', 'url', {
callback: function(form, value) { return 'myparam=' + encodeURIComponent(value) }
})
```

The `encodeURIComponent()` makes sure values containing special characters like “&” or ”=” don’t cause problems. Use `encodeURIComponent()` instead of `escape()` from previous examples to get UTF-8 encoded data (and not “malformed URI” errors in Firefox). This function can also be used to pass additional parameters, such as what item or field to edit:

```
new Ajax.InPlaceEditor('id', 'url', {
callback: function(form, value) { return 'item=123&field=description&myparam='+escape(value) }
})
```

Note: this parameters name may change (to “parameters” or “with”) before the final release of 1.5.

How to execute custom code using the InPlaceEditor’s callbacks.

```
new Ajax.InPlaceEditor(id, url, {
callback: function(form, value) { return 'item=123&field=description&myparam='+escape(value) },
onEnterEditMode: function(form, value) { // custom code goes here... },
onLeaveEditMode: function(form, value) { // custom code goes here... }
});
```

You can find all the possible callbacks by looking for lines in the source like this:

```
this.triggerCallback('onEnterEditMode');
```

(Should these callback options be documented in the Options list above?)

How to add a spinning in progress indicator

Get an animated .gif that shows progress (the spinning image used on the Macintosh for example). Add a CSS class for ‘inplaceeditor-saving’ with the spinning image as background. The class will be added to the element during communication with the server and removed afterwards.

```
<style type="text/css" media="screen">
  .inplaceeditor-saving { background: url(/images/wait.gif) bottom right no-repeat; }
</style>

<p id="editme3">Click me, click me!</p>
<script type="text/javascript">
 new Ajax.InPlaceEditor('editme3', '/demoajaxreturn.html');
</script>
```

How to customize the look and feel of the form

Use the id or class (‘inplaceeditor-form’) of the form in a CSS selector to apply the desired style. The text field is always named ‘value’, the ok button is an input with type ‘submit’ and the cancel link is a normal anchor (‘a’) element.

```
form.inplaceeditor-form { /* The form */
}

form.inplaceeditor-form input[type="text"] { /* Input box */
}

form.inplaceeditor-form textarea { /* Textarea, if multiple columns */
}

form.inplaceeditor-form input[type="submit"] { /* The submit button */
```

```
  margin-left:1em;
}

form.inplaceeditor-form a { /* The cancel link */
  margin-left:1em;
}
```

How to edit server side formatted text (formatted with eg. textile)

Format the text on the server when the initial page is rendered. Add a new server side action that returns the unformatted text in the reponse and specify the load Text URL? option as a URL to this action. The control will load the text from the server instead of using the text on the page. The action that receives the updated text should save it and return the formatted text as an HTML snippet in the reponse.

Example using Ruby on Rails

```
#controller
def my_action
    #get the text to display initially
    @fancy_text = "Some fancy *bold* and _emphasized_ text."
end

def update_text
    #save your text here, and return the saved value
    @fancy_text = params[:value]
    render :layout => false, :inline => "<%= textilize( @fancy_text ) %>"
end

def return_unformatted_text
    #get your text and return it without the formatting
    @fancy_text = "Some fancy edited, *bold* and _emphasized_ text."
    render :layout => false, :inline => "<%= @fancy_text %>"
end

#view ( of my_action.rhtml )

<p id="editme2"><%= textilize( @fancy_text ) %></p>

<script language="JavaScript">
 new Ajax.InPlaceEditor('editme2', 'update_text',
 {rows:15,cols:40,loadTextURL:'return_unformatted_text'});
</script>
```

In-Place edits with select lists

In my (java) application, I have a need for in-place editing in the form of select lists. Given a server-side ajax helper that provides the necessary options for the current user, I was able to write the following:

```
function setupCategoryEditor(el, url) {
    var editor=new Ajax.InPlaceEditor(el, url);
    Object.extend(editor, {
        createEditField: function() {
            var text=this.getText();

            var field=document.createElement("select");
            field.name="value";

            this.editField=field;
            this.form.appendChild(this.editField);

            new Ajax.Request('/url/to/option/list', {
                onSuccess: function(req) {
                    // Get the text from an XML tag.
                    var getData=function(el, which) {
                        stuff=el.getElementsByTagName(which);
                        return stuff[0].firstChild.nodeValue;
                    };
                    var cats=req.responseXML.getElementsByTagName("cat");
                    $A(cats).each( function(cat, idx) {
                        var op=document.createElement("option");
```

```
                        op.value=getData(cat, "value");
                        op.text=getData(cat, "key");
                        if(window.ActiveXObject) {
                            field.options.add(op);
                        } else {
                            field.appendChild(op);
                        }

                        // Select the current item
                        if(op.text == text) {
                            field.selectedIndex=idx;
                        }
                    });
                }
                });
        }
    });
}
```

Obviously, one could write something smaller with JSON or innerHTML if you're into that kind of thing.

There is another implementation of In Place Editor for select lists called Ajax. In Place Select? available here.

You can change getText like so:

```
getText: function() {
  return this.element.childNodes[0] ? this.element.childNodes[0].nodeValue : '';
},
```

to avoid e.g. "Me & Myself" being represented in the input box as "Me & amp; Myself" (use nodeValue because innerText does not work with Safari when the element is hidden).

I just added inplace editor inside a sortable list array. I can drag and after, the editor appears, but ideally I would not want that to happen and instead activate the editor specifically, not as a result of dragging my list items. What is suggested here? I tried using a drag handle, which seemed like an obvious solution, except the editor then shows the span class code as well, which is a caveat I suppose. Unless I can use a drag handle without the method in example, then that won't be a good solution. How else can I drag or how else can I activate the editor besides writing redundant for each instance? ~~ms~~

I just spent a good deal of time figuring that out myself so I'll share my result:

```
<style>
div.handle {
    display: none;
    float: left;
    width: 25px;
    height: 11px;
    margin-top: 2px;
    padding-right: 3px;
    background: url('/images/drag.gif');
    background-repeat: no-repeat;
    clear: both;
}
</style>
<ul id="field">
    <li id="field_1">

        <div class="handle"></div><div class="delete"></div>
        <span class="data" id="field_name_1">Brand</span></li>
    <li id="field_3">
        <div class="handle"></div><div class="delete"></div>

        <span class="data" id="field_name_3">SKU Number</span></li>
    <li id="field_5">
        <div class="handle"></div><div class="delete"></div>
        <span class="data" id="field_name_5">Color</span></li>

</ul>
<a id="field_sort" href="javascript:;" onclick="makeSortable('field');">Reorder</a>
<a id="field_donesort" href="javascript:;" onclick="doneSorting('field');" style="display:none;">Done Reordering</a>
<script>
function makeSortable(myid,mytag){
    Sortable.create(myid,{ tag: mytag || 'li', handle:'handle'});
    setDisplayByClass('handle','inline',$(myid));
    $(myid+'_sort').style.display = 'none';
```

```
    $(myid+'_donesort').style.display = '';
    $(myid).setAttribute('reordering','yes');
    return false;
}
function doneSorting(myid){
    new Ajax.Updater('reply',page,{parameters:'a=reorder&table='+myid+'&'+Sortable.serialize(myid)});
    Sortable.destroy(myid);
    setDisplayByClass('handle','none',$(myid));
    $(myid+'_sort').style.display = '';
    $(myid+'_donesort').style.display = 'none';
    $(myid).setAttribute('reordering','no');
    return false;
}
var list = getElementsByClass('data',$('fields'));
var ipeOptionsStd = {
        callback:    function(form,value){
                        var id=form.id;
                        id = id.substring(0,id.indexOf('-inplaceeditor'));
                        var t = id.substring(0,id.indexOf('_'));
                        var c = id.substring(0,id.lastIndexOf('_'));
                        var i = id.substring(id.lastIndexOf('_')+1,id.length);
                        return 'a=update&table='+t+'&column='+c+'&id='+i+'&value='+escape(value);
                    }
        };
for(var i=0;i<list.length;i++){
    new Ajax.InPlaceEditor(list[i], page, ipeOptionsStd );
}

</script>
```

Hope that helps, that is a much simplified version. I've also done this with tables where each cell is an In Place Editor and the rows are Sortable.

- Colin

Using HTML within the In Place Editor

If you are using Markdown or Textile, you can include HTML within your text. And if you follow the instructions above, you will be able to load this text in from your database with `loadTextURL`. But the stripTags function is invoked by the In Place Editor, and that strips out all the code before it can be displayed in the editor field.

To get around this, add the following two extensions anywhere after the rest of the library has loaded. I use the extensions.js method noted elsewhere on this site.

```
Object.extend(Ajax.InPlaceEditor.prototype, {
    onLoadedExternalText: function(transport) {
        Element.removeClassName(this.form, this.options.loadingClassName);
        this.editField.disabled = false;
        this.editField.value = transport.responseText;
        Field.scrollFreeActivate(this.editField);
    }
});

Object.extend(Ajax.InPlaceEditor.prototype, {
    getText: function() {
        return this.element.childNodes[0] ? this.element.childNodes[0].nodeValue : '';
    }
});
```

If you do this, make sure that your `loadTextURL` does not return entity-encoded text (using `htmlentities()` for example) or you will end up with double-encoded text.

In-Place edits only by externalControl

i searched for a way to allow editing of certain fields only by clicking the external control field. currently its implemented to allow a user to click either on the element itself or on the external control element.

to only allow editing by clicking on the external control, simply change a few lines:

```
Event.observe(this.element, 'click', this.onclickListener);
Event.observe(this.element, 'mouseover', this.mouseoverListener);
Event.observe(this.element, 'mouseout', this.mouseoutListener);
```

```
if (this.options.externalControl) {
  Event.observe(this.options.externalControl, 'click', this.onclickListener);
  Event.observe(this.options.externalControl, 'mouseover', this.mouseoverListener);
  Event.observe(this.options.externalControl, 'mouseout', this.mouseoutListener);
}

change to:

if (this.options.externalControl) {
  Event.observe(this.options.externalControl, 'click', this.onclickListener);
  Event.observe(this.options.externalControl, 'mouseover', this.mouseoverListener);
  Event.observe(this.options.externalControl, 'mouseout', this.mouseoutListener);
} else {
  Event.observe(this.element, 'click', this.onclickListener);
  Event.observe(this.element, 'mouseover', this.mouseoverListener);
  Event.observe(this.element, 'mouseout', this.mouseoutListener);
}
```

maybe someone can add a flag like ‘only External Control ? to add this to the trunk ..

Small extension to editor to add a text in case field is empty.

```
/*
 * InPlaceEditor extension that adds a 'click to edit' text when the field is
 * empty.
 */
Ajax.InPlaceEditor.prototype.__initialize = Ajax.InPlaceEditor.prototype.initialize;
Ajax.InPlaceEditor.prototype.__getText = Ajax.InPlaceEditor.prototype.getText;
Ajax.InPlaceEditor.prototype.__onComplete = Ajax.InPlaceEditor.prototype.onComplete;
Ajax.InPlaceEditor.prototype = Object.extend(Ajax.InPlaceEditor.prototype, {

    initialize: function(element, url, options){
        this.__initialize(element,url,options)
        this.setOptions(options);
        this._checkEmpty();
    },

    setOptions: function(options){
        this.options = Object.extend(Object.extend(this.options,{
            emptyText: 'click to edit...',
            emptyClassName: 'inplaceeditor-empty'
        }),options||{});
    },

    _checkEmpty: function(){
        if( this.element.innerHTML.length == 0 ){
            this.element.appendChild(
                Builder.node('span',{className:this.options.emptyClassName},this.options.emptyText));
        }
    },

    getText: function(){
        document.getElementsByClassName(this.options.emptyClassName,this.element).each(function(child){
            this.element.removeChild(child);
        }.bind(this));
        return this.__getText();
    },

    onComplete: function(transport){
        this._checkEmpty();
        this.__onComplete(transport);
    }
});
```

To use, copy this into a extensions.js and load it in your app after the other scriptaculous stuff.

Here is the style I use with this:


```
/* ----  InPlaceEditor style ---------------------------------------------- */
```

```
.inplaceeditor-empty {
    font-style: italic;
    color: #999;
}
```

- Pedro

This doesn't seem to work for me in Prototype 1.6, so I wrote an updated version. (Blog Post).

```
Ajax.InPlaceEditorWithEmptyText = Class.create(Ajax.InPlaceEditor, {

  initialize : function($super, element, url, options) {

    if (!options.emptyText)        options.emptyText      = “click to edit…“;
    if (!options.emptyClassName)   options.emptyClassName = “inplaceeditor-empty“;

    $super(element, url, options);

    this.checkEmpty();
  },

  checkEmpty : function() {

    if (this.element.innerHTML.length == 0 && this.options.emptyText) {

      this.element.appendChild(
          new Element(“span“, { className : this.options.emptyClassName }).update(this.options.emptyText)
        );
    }

  },

  getText : function($super) {

    if (empty_span = this.element.select(“.“ + this.options.emptyClassName).first()) {
      empty_span.remove();
    }

    return $super();

  },

  onComplete : function($super, transport) {

    this.checkEmpty();
    return $super(transport);

  }

});
```

- Nik

Create your own Callback

I wanted to have a custom callback that would fire after the InPlaceEditor form was added to the DOM. `onEnterEditMode` fires as soon as you click, but the form doesn't exist yet.

This works by copying an existing method `enterEditMode()`, which builds the form and writes it to the page, into a new method name so that I can edit the real one. In mine, the original is called via `__enterEditMode()` and then my custom code is run:

```
Ajax.InPlaceEditor.prototype.__enterEditMode = Ajax.InPlaceEditor.prototype.enterEditMode;
Object.extend(Ajax.InPlaceEditor.prototype, {
  enterEditMode:function(e) {
    this.__enterEditMode(e);
    this.triggerCallback('onFormReady',this._form);
  }
});
```

Now when I create my InPlaceEditor I just add a the `onFormReady` callback as an option:

```
new Ajax.InPlaceEditor('location', '/change_location',
  {
    onFormReady: function(obj,form) {
      form.getInputs().first().value = '';
    }
  });
```

In my case I wanted to clear out the value in the text box so that the user had to type something from scratch. If they cancel it will switch back to the original text.

- Rob

Validation.

Information for implementing validation with ajax. In Place Editor can be found here. Link Dead!

Ajax.Autocompleter

Introduction

Ajax.Autocompleter allows for server-powered autocompleting text fields.

Syntax

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

Options (Inherited from Autocompleter.Base, as well as all options for Prototype's Ajax.Request)

Option	*Default Value*	*Description*
`paramName`	the 'name' of the element	Name of the parameter for the string typed by the user on the autocompletion field
`tokens`	[]	Tokenized incremental autocompletion is enabled automatically when an autocompleter is instantiated with the 'tokens' option in the options parameter: See Also Autocompleter.Base
`frequency`	0.4	How frequently (in seconds) the input field should be polled for changes before firing an Ajax request.
`minChars`	1	The minimum number of characters that must be entered in the input field before an Ajax request is made.
`select`	null	The element that contains the text to be placed into the input box. By default all text will be used.
`indicator`	null	The HTML `id` of an element to display (using Element.show) while the Ajax request is in progress. This element will be hidden with Element.hide when the request is completed. This is useful for displaying an animated spinner during processing. See Ajaxload for some image examples.
`updateElement`	null	Hook for a custom function to replace the built-in function that adds the list item text to the input field. The custom function is called after the element has been updated (i.e. when the user has selected an entry). The function receives one parameter only: the selected item (the `<li>` item selected)
`afterUpdateElement`	null	Hook for a custom function that's called following the execution of the built-in function that adds the list item text to the input field, which happens after a user has selected an entry. (The difference between updateElement and afterUpdateElement is that updateElement replaces the built-in Autocompleter function; afterUpdateElement supplements that built-in function.) The function receives two parameters, the input field specified for autocompletion, and the selected item (the `<li>` item selected)
`callback`	null	This function is called just before the Request is actually made, allowing you to modify the querystring that is sent to the server. The function receives the completer's input field and the default querystring ('value=XXX') as parameters. You should return the querystring you want used, including the default part.
`parameters`	null	To send additional parameters to the server, add them here in the format: `'field=value&another=value'`. Errata/bug: The hash format `{field: 'value',another: 'value'}` was noted in this wiki to work, but it does not, as the Ruby on Rails project has separately documented and fixed in their codebase.

Example

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">
  <img src="/images/spinner.gif" alt="Working..." />
</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 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:0;
  padding:0;
}
div.autocomplete ul {
  list-style-type:none;
  margin:0;
  padding:0;
}
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;
}
```

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.

Additionally, `','` in the above example can be replaced with a token 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.

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 text field element of the autocompletion (its `name` attribute) . You can override it by 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>
```

If your search returns no results, it is critical that your server immediately return an empty list, rather than nothing. You will only notice this need if you use an indicator to show server activity—the indicator will never go away if your server doesn't return an empty list.

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:

```
<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>
```

Another way to get additional information is to encode an ID into each list item, and redefine the afterUpdate Element to handle that ID:

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

```
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 use of CSS to style the selected element.

Tutorials

A short tutorial on using Autocompleter, together with Ruby and WEBrick, prepared for the Linux Users Group, Villafranca, Verona, Italy.

Autocompleter.Local

Introduction

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(id_of_text_field, id_of_div_to_populate, array_of_strings, 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

Option	*Default Value*	*Description*
`choices`	10	How many autocompletion choices to offer
`partialSearch`	off	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
`fullSearch`	false	Search anywhere in autocomplete array strings.
`partialChars`	2	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).
`ignoreCase`	true	Whether to ignore case when autocompleting

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

HTML

```
<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="autocomplete" id="band_list" style="display:none"></div>
```

Javascript

```
var bandsList = [
  '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 (band)',
  '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'
];

new Autocompleter.Local('bands_from_the_70s', 'band_list', bandsList, { });
```

CSS

Styling the `div` and the returned `ul` is very 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 user interface elements to stand out. The CSS from the demo applied to the examples would be:

```
div.autocomplete {
  margin:0px;
  padding:0px;
  width:250px;
  background:#fff;
  border:1px solid #888;
  position:absolute;
}

div.autocomplete ul {
  margin:0px;
  padding:0px;
  list-style-type:none;
}

div.autocomplete ul li.selected {
  background-color:#ffb;
}

div.autocomplete ul li {
  margin:0;
  padding:2px;
  height:32px;
  display:block;
  list-style-type:none;
  cursor:pointer;
}
```

Notes

You can also have your options provided via Ajax callbacks. For more information regarding this implementation, see Ajax.Autocompleter.

Advance extended version that mimics Facebook to: field.

- Tutorial
- Git repository
- Live demo

Slider

Controls > **Slider**

Introduction

A slider control which can be used to select a single or multiple values from a given range, or even set of values.

Syntax

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

```
new Control.Slider('handles','track', [options]);
```

`handles` can either be a single id (or element) or, if you want more than one handle, an array of ids (or elements). `track` is either id or element.

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	(none)	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).
range	0	(none)	Use the $R(min,max)
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 maxium values.
disabled	V1.5	(none)	This will lock the slider so that it will not move and thus is disabled.
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.
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.
spans	??	(none)	An array of ids or elements which are positioned between handles. This is used only when slider has more than one handle.
restricted	??	false	Used only for multiple handles, when restricted is true, handle(s) with greater indexes are not allowed to have values less than handles with smaller indexes. When restricted is false, handles can be moved independently from others.

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

Function	Parameters	Description
setValue	`value`, `handleIndex`	Will update the slider's value and thus move the slider handle to the appropriate position. `handleIndex` is optional, when it is not passed then 'active' (last-dragged/used) handle is used. **NOTE:** when using setValue, the callback functions (below) are called.
setDisabled	(none)	Will set the slider to the disabled state (disabled = true).
setEnabled	(none)	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 (or array if slider has multiple handles) as its parameter.
onChange	Called whenever the Slider has finished moving or has had its value changed via the setSlider Value function. The called function gets the slider value (or array if slider has multiple handles) 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

Single handle

```
// 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 later on
s2.options.onChange = function(value) {
  // ...
  $('height_value').update(value);
};

s2.options.onSlide = function(value) {
  // ...
  $('height_value').update(value);
};
```

Multiple handles

```
<div id="square_slider" class="slider">
  <div id="square_slider_handle_min" class="handle left"></div>
  <div id="square_slider_handle_max" class="handle right"></div>

  <div id="square_slider_span" class="span"></div>
</div>
```

```
var handles = ['square_slider_handle_min', 'square_slider_handle_max'];
var square_slider = new Control.Slider(handles, 'square_slider', {
    range: $R(0, 100),
    values: $R(0, 100),
    sliderValue: [20, 80],
    spans: ["square_slider_span"],
    restricted: true
});
```

Demo

Use the slider to resize the box

And this to change its color

Source Code of the Demo

```
<style type="text/css">
  div.slider { width:256px; margin:10px 0; background-color:#ccc; height:10px; position: relative; }
  div.slider div.handle { width:10px; height:15px; background-color:#f00; cursor:move; position: absolute; }

  div#zoom_element { width:50px; height:50px; background:#2d86bd; position:relative; }
</style>

<div class="demo">
  <p>Use the slider to resize the box</p>
  <div id="zoom_slider" class="slider">
    <div class="handle"></div>
  </div>

  <p>And this to change its color</p>
  <div id="rgb_slider" class="slider">
    <div class="handle" style="background-color: #f00;"></div>
    <div class="handle" style="background-color: #0f0;"></div>
    <div class="handle" style="background-color: #00f;"></div>
  </div>

  <div id="zoom_element"></div>
</div>

<script type="text/javascript">
  (function() {
    var zoom_slider = $('zoom_slider'),
        rgb_slider = $('rgb_slider'),
        box = $('zoom_element');

    new Control.Slider(zoom_slider.down('.handle'), zoom_slider, {
      range: $R(40, 160),
      sliderValue: 50,
      onSlide: function(value) {
        box.setStyle({ width: value + 'px', height: value + 'px' });
      },
      onChange: function(value) {
        box.setStyle({ width: value + 'px', height: value + 'px' });
      }
    });

    new Control.Slider(rgb_slider.select('.handle'), rgb_slider, {
      range: $R(0, 255),
      sliderValue: [45, 134, 189],
      onSlide: function(values) {
        box.setStyle({ backgroundColor: "rgb("+ values.map(Math.round).join(',') +")" });
      },
      onChange: function(values) {
        box.setStyle({ backgroundColor: "rgb("+ values.map(Math.round).join(',') +")" });
      }
    });
  })();
</script>
```

Builder

Introduction

Use `Builder` to easily create DOM elements dynamically.

Syntax

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

Arguments	Description
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 a String or Number, it will be automatically appended as a text node. Instead of an array, `children` can also be a JavaScript String or Number, 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

```
var table = Builder.node('table', {
  width: '100%',
  cellpadding: '2',
  cellspacing: '0',
  border: '0'
});

var 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 occured</p>
```

Complex Example

```
var 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 using the browsers native `appendChild` function:

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

… or by using Protoypes built-in `Element#insert` method which allows you to specify a position the new node gets inserted into:

```
$('myExistingDomElement').insert(element); // appends the new node to @myExistingDomElement@-element
$('myExistingDomElement').insert({ after: element }); // inserts the new node right after the @myExistingDomElement@-element
$('myExistingDomElement').insert({ top: element }); // inserts the new node as the first element in @myExistingDomElement@
```

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

Sound

Introduction

Use `Sound` to play audio directly from the browser.

Syntax

```
Sound.play( url, [ options ])
Sound.enable()
Sound.disable()
```

Play Options

Option	Description
replace	replace the current active track, defauts to `false`

Notes

- Sound.disable does not disable the playback immediately, it just prevents the next tracks from being played. Watch the demo to see how to stop the sound immediately.
- Sound.enable does not resume the playback, just allows the next track to be played.
- In Firefox 3, Sound does not work from local files, but works if served (even from localhost). As an alternative, you can change the template within sound.js from “object” to “embed” and from “data” to “src”.

Demo

URL: http://www.joshwoodward.com/mp3/TheSimpleLife /JoshWoodward-T

Sound: On

- Play
- Disable sound
- Disable now
- Enable sound

Source code of the demo

```
<div>
    URL: <input type = "text" id = "sound_demo_track_url" style = "width:400px;"
    value = "http://www.joshwoodward.com/mp3/TheSimpleLife/JoshWoodward-TheSimpleLife-101-HeritagePlace.mp3"/>
    <br/>
    Sound: <span id = "sound_status">On</span>
    <ul class = "sound_controls">
      <li><a href="javascript:void(0)" id="sound_demo_play">Play</a></li>
      <li><a href="javascript:void(0)" id="sound_demo_disable">Disable sound</a></li>
      <li><a href="javascript:void(0)" id="sound_demo_disable_now">Disable now</a></li>
      <li><a href="javascript:void(0)" id="sound_demo_enable">Enable sound</a></li>
    </ul>
</div>
```

```
<script type="text/javascript">
  $('sound_demo_play').observe('click', function(event) {
       event.stop();
       Sound.play($('sound_demo_track_url').value,{replace:true});
   });

  $('sound_demo_enable').observe('click', function(event) {
       event.stop();
       Sound.enable();
     $('sound_status').innerHTML = 'On';
   });
```

```
  $('sound_demo_disable').observe('click', function(event) {
       event.stop();
       Sound.disable();
     $('sound_status').innerHTML = 'Pending Off';
   });

  $('sound_demo_disable_now').observe('click', function(event) {
       event.stop();
       Sound.enable();
       Sound.play('',{replace:true});
       Sound.disable();
     $('sound_status').innerHTML = 'Off';
   });

</script>
```

Unit Testing

Introduction

script.aculo.us provides a complete set of classes and methods for Javascript unit testing.

To use the unit test library, you must include `unittest.js` in your HTML file. See Test.Unit.Runner for a complete example on how this works.

script.aculo.us “eats its own dog food,” and includes unit and functional tests for itself, inside the `test` folder of the repository.

Note that as of August 2005 some of the tests will fail in browsers other than Firefox (due to implementation differences and some of the DOM functions used not implemented in some browsers).

Classes

- Test.Unit?
- Test.Unit.Logger
- Test.Unit.Runner
- Test.Unit.Assertions?
- Test.Unit.Testcase?

Examples

A short tutorial (second half of the page) demonstrating the use of the script.aculo.us test harness for Javascript, prepared for the Linux Users Group, Villafranca, Verona, Italy.

Test.Unit.Runner

Introduction

`Test.Unit.Runner` is a utility class for writing unit test cases with an easy to use syntax.

Syntax

```
new Test.Unit.Runner(functions [, options] );
```

The `functions` and `options` parameters are both `{key: value}` collections.

Examples

Using `Test.Unit.Runner` requires you to use a specific XHTML page template, that looks as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Page title</title>
  <script src="path/to/prototype.js" type="text/javascript"></script>
  <script src="path/to/util.js" type="text/javascript"></script>
  <script src="path/to/unittest.js" type="text/javascript"></script>
  <!-- other JavaScript includes -->
  <link rel="stylesheet" href="path/to/test.css" type="text/css" />
</head>
<body>

<!-- Log output -->
<div id="testlog"> </div>

<!-- here go any elements you do the testing on -->

<!-- Tests -->
<script type="text/javascript" language="javascript">
// <![CDATA[
  new Test.Unit.Runner({
    // optional setup function, run before each individual test case
    setup: function() { with(this) {
      // code
    }},
    // optional teardown function, run after each individual test case
    teardown: function() { with(this) {
      // code
    }},

    // test cases follow, each method which starts
    // with "test" is considered a test case
    testATest: function() { with(this) {
      // code
    }},
    testAnotherTest: function() { with(this) {
      // code
    }}
  }, { options });
// ]]>
</script>
</body>
</html>
```

Notice how the `with(this) { ... }` syntax allows for easily calling on methods of the Test.Unit.Testcase? class, which includes the assertions, as defined in Test.Unit.Assertions?.

More Examples

Probably the best examples of Test.Unit.Runner are the actual tests used by script.aculo.us. Don't be afraid to dive in to the code! See the test folder in the repository root.

Ruby on Rails

Ruby On Rails features complete Prototype, and script.aculo.us integration.

Demo site and example code

Visit the Rails/script.aculo.us demo site for live demos and example code.

Usage

First, you need to include the libraries in your app, which is easy, as Prototype and script.aculo.us come prepackaged with Rails.

In your layout or view add this to the `<head>` section of your layout:

```
<%= javascript_include_tag :defaults %>
```

You can now use the Rails JavaScript helpers or use do-it-yourself `<script>` tags and inline events as documented in this Wiki.

As of Rails 2.0, for certain functionality, like autocompleting text fields and in place editing, you'll need to install the correspoding plugins.

Related Articles

- Using the Scriptaculous autocomplete with Rails

Tabs

Here is another Script that I use quite often, and I think is rather handy.

About

This script will allow you to create interactive tabs for use in an application.

The CSS and HTML:

```
<style>
    #tabs{
        margin-left: 4px;
        padding: 0;
        background: transparent;
        voice-family: "\"}\"";
        voice-family: inherit;
        padding-left: 5px;
    }
    #tabs ul{
        font: bold 11px Arial, Verdana, sans-serif;
        margin:0;
        padding:0;
        list-style:none;
    }
    #tabs li{
        display:inline;
        margin:0 2px 0 0;
        padding:0;
        text-transform:uppercase;
    }
    #tabs a{
        float:left;
        background:#A3BBE6 url(images/tabs_left.gif) no-repeat left top;
        margin:0 2px 0 0;
        padding:0 0 1px 3px;
        text-decoration:none;
    }
    #tabs a span{
        float:left;
        display:block;
        background: transparent url(images/tabs_right.gif) no-repeat right top;
        padding:4px 9px 2px 6px;
    }
    #tabs a span{float:none;}
    #tabs a:hover{background-color: #7E94B9;color: white;}
    #tabs a:hover span{background-color: #7E94B9;}
    #tabHeaderActive span, #tabHeaderActive a { background-color: #42577B; color:#fff;}
    .tabContent {
        clear:both;
        border:2px solid #42577B;
        padding-top:2px;
        background-color:#FFF;
    }
</style>

<div id="tabs">
    <ul>
        <li style="margin-left: 1px" id="tabHeader1" class="currenttab"><a href="javascript:void(0)" onClick="toggleTab(1,6)
"><span>Tab 1</span></li>
        <li id="tabHeader2"><a href="javascript:void(0)" onClick="toggleTab(2,6)" ><span>Tab 2</span></li>
        <li id="tabHeader3"><a href="javascript:void(0)" onclick="toggleTab(3,6)"><span>Tab 3</span></li>
        <li id="tabHeader4"><a href="javascript:void(0)" onClick="toggleTab(4,6)"><span>Tab 4</span></li>
        <li id="tabHeader5"><a href="javascript:void(0)" onclick="toggleTab(5,6);"><span>Tab 5</span></li>
        <li id="tabHeader6"><a href="javascript:void(0)" onclick="toggleTab(6,6);"><span>Tab 6</span></li>
    </ul>
    </div>
    <div id="tabscontent">
        <div id="tabContent1" class="tabContent" style="display:none;">
```

```
            <br /><div>First Tab Content goes here</div>
        </div>

        <div id="tabContent2" class="tabContent" style="display:none;">
            <br /><div>Second Tab Content goes here</div>
        </div>

        <div id="tabContent3" class="tabContent" style="display:none;">
            <br /><div>Third Tab Content goes here</div>
        </div>

        <div id="tabContent4" class="tabContent" style="display:none;">
            <br /><div>Fourth Tab Content goes here</div>
        </div>

        <div id="tabContent5" class="tabContent" style="display:none;">
            <br /><div>Fifth Tab Content goes here</div>
        </div>

        <div id="tabContent6" class="tabContent" style="display:none;">
            <br /><div>Sixth Tab Content goes here</div>
        </div>
    </div><!--End of tabscontent-->
</div><!--End of tabs-->
```

The Javascript:

```
/*-------------------------------------------------------------
    Toggles element's display value
    Input: any number of element id's
    Output: none
    -------------------------------------------------------------*/
function toggleDisp() {
    for (var i=0;i<arguments.length;i++){
        var d = $(arguments[i]);
        if (d.style.display == 'none')
            d.style.display = 'block';
        else
            d.style.display = 'none';
    }
}
/*-------------------------------------------------------------
    Toggles tabs - Closes any open tabs, and then opens current tab
    Input:     1.The number of the current tab
                    2.The number of tabs
                    3.(optional)The number of the tab to leave open
                    4.(optional)Pass in true or false whether or not to animate the open/close of the tabs
    Output: none
    -------------------------------------------------------------*/
function toggleTab(num,numelems,opennum,animate) {
    if ($('tabContent'+num).style.display == 'none'){
        for (var i=1;i<=numelems;i++){
            if ((opennum == null) || (opennum != i)){
                var temph = 'tabHeader'+i;
                var h = $(temph);
                if (!h){
                    var h = $('tabHeaderActive');
                    h.id = temph;
                }
                var tempc = 'tabContent'+i;
                var c = $(tempc);
                if(c.style.display != 'none'){
                    if (animate || typeof animate == 'undefined')
                        Effect.toggle(tempc,'blind',{duration:0.5, queue:{scope:'menus', limit: 3}});
                    else
                        toggleDisp(tempc);
                }
            }
        }
        var h = $('tabHeader'+num);
        if (h)
            h.id = 'tabHeaderActive';
```

```
            h.id = 'tabHeaderActive';
        h.blur();
        var c = $('tabContent'+num);
        c.style.marginTop = '2px';
        if (animate || typeof animate == 'undefined'){
            Effect.toggle('tabContent'+num,'blind',{duration:0.5, queue:{scope:'menus', position:'end', limit: 3}});
        }else{
            toggleDisp('tabContent'+num);
        }
    }
}
```

Example:

TAB 1 TAB 2 TAB 3 TAB 4 TAB 5 TAB 6

FAQ

If you have questions, look here first! If you feel like you're having a questions (and maybe an answer) that will interested many people, please add it here.

General questions

1.1 Where to ask for help if I can't find an answer here?

First, use the search function. If that won't help, subscribe to the Mailing List and ask there.

1.2 Which browsers are supported?

See Supported Browsers.

1.3 Where can I see this in action?

See Demos.

1.4 Cool. So how to get started…?

See Usage.

1.5 So can I use this in my commercial app?

Yes. See License.

1.6 How do I extend/override without modifying the library code?

See HowtoExtendAndOverride.

1.7 How much overhead does script.aculo.us add to a page?

The complete library is about 228KB (including Prototype). Unofficially released compressed versions are available that can bring the files size down below 100KB, try looking on the mailing lists and support groups listed on the MailingList page for sources.

Effects

2.1 On Internet Explorer, opacity-based effects don't work!

Please see GivingElementsLayout. Starting with Vh2. 1.5_rc1 this issue should be automatically handled by script.aculo.us.

2.2 On Internet Explorer (yes, there's a pattern here!), Effect.SlideUp/Effect.SlideDown are broken…?

That's an Internet Explorer CSS bug, see Effect.SlideDown for a workaround. See http://css.nu/pointers/bugs-ie.html for more on this.

2.3 Some effects cause Firefox to flicker once

That's a bug with the Gecko rendering engine in Firefox h2. 1.0.x. Starting with Firefox h2. 1.5b1 the flicker is gone. You can set the 'overflow:hidden' CSS style on the element you run the effect on as a workaround.

2.4 Effect.* doesn't work when object's display is none!

The problem lies with class / id definitions. use the inline property style="display:none;" instead

2.4.1 Why?

script.aculo.us is based off of prototype.js, and calls the 'show' function to make an element visible. This works by setting the element's style.display = '' (undef). This is intended to set it to the default, which is visible. However, if you have a style for display defined higher up in the CSS than the inline element level (which prototype is overwriting), it will look at the undefined style on your element and cascade up. Your stylesheet probably has display='none', so it looks like nothing is happening.

2.5 Effect.Appear doesn't work from onload. Any Ideas? h1. Controls

3.1 I get weird Java Script errors, what do I do?

The most common cause of this is that you don't include all script.aculo.us libraries. If you use anything from controls.js you must include effects.js, too. Otherwise, double check if you have typos regarding your id attribute names.

Drag and drop

4.1 Dragging/Dropping doesn't work?

You may run in a browser limitation here if you specify CSS properties that are needed by the libraries in an external CSS file. Please try specifying properties in the inline style attribute of the affected element(s) instead.

4.2 Why can't I make sortable TABLE elements?

Because of technical restrictions. See Sortable.create for more info on this.

4.3 The onUpdate callback on Sortable.create doesn't seem to work!

You're probably missing the requirements for naming the id attributes in the elements contained in your sortable element. See Sortable.serialize for more on this.

4.4 How do I tell a Sortable that I added new elements to it?

Call Sortable.create again.

Autocompletion

5.1 Why does my input box show a blank result after selecting something from the autocompletion drop down list?

The value selected is everything not inside an element with a class="informal". Remove any whitespace from the result of your auto complete responder, to prevent this from being selected.

5.2 Nothing happens when I type something in?

See question h2. 3.1 for a possible solution.

Contribute

Want to help make script.aculo.us better? You can!

Found a bug?

script.aculo.us uses Lighthouse for bug tracking: http://prototype.lighthouseapp.com/projects/8887-script-aculo-us

(First, be sure that there isn't a ticket already, then file a new bug!)

If you found a bug

When creating a bug report, be sure to include as much relevant information as possible. Post an example that shows off the problem. Preferably, alter the unit tests and show through either changed or added tests how the expected behavior is not occuring.

Source code repository

script.aculo.us uses Git to manage development. Check out the current development trunk with:

```
git clone git://github.com/madrobby/scriptaculous.git
```

As script.aculo.us 1.xx is feature-frozen, this development trunk is for bugfixes only (new features will only happen for script.aculo.us 2).

JavaScript is not your thing?

You may also contribute by writing documentation, making tutorials and general pimping of scriptaculous! :)