script.aculo.us

javascrip ui library

tutorialspoint

SIMPLYEASYLEARNING

About the Tutorial

script.aculo.us is a JavaScript library built on the Prototype JavaScript Framework. It provides dynamic visual effects and other functionalities via the Document Object Model (DOM). This tutorial gives you a complete understanding on how to use script.aculo.us.

Audience

This tutorial is aimed for JavaScript programmers and experienced Ajax programmers who want to improve their understanding of the library features and want to enhance their knowledge.

Prerequisites

In order to make the most of this tutorial, you should have a good understanding of JavaScript.

Copyright & Disclaimer

Table of Contents

1. SCRIPT.ACULO.US – OVERVIEW

What is script.aculo.us?

script.aculo.us is a JavaScript library built on top of the *Prototype JavaScript Framework*, enhancing the GUI and giving Web 2.0 experience to the web users.

script.aculo.us was developed by Thomas Fuchs and it was first released to the public in June 2005.

script.aculo.us provides dynamic visual effects and user interface elements via the Document Object Model (DOM).

The Prototype JavaScript Framework is a JavaScript framework created by Sam Stephenson that provides an Ajax framework and other utilities.

How to Install script.aculo.us?

It is quite simple to install the script.aculo.us library. It can be set up in three simple steps:

1. Go to the download page to download the latest version in a convenient package.
2. Unpack the downloaded package and you will find the following folders:
    - **lib**: contains prototype.js file.
    - **src**: contains the following 8 files:
        - builder.js
        - controls.js
        - dragdrop.js
        - effects.js
        - scriptaculous.js
        - slider.js
        - sound.js
        - unittest.js
    - **test**: contains files for testing purpose.
    - **CHANGELOG**: File that contains the history of all the changes.
    - **MIT-LICENSE**: File describing the licensing terms.

    - **README**: File describing the installation package including the installation instructions.

- Now put the following files in a directory of your website, e.g. /javascript.
    - builder.js
    - controls.js
    - dragdrop.js
    - effects.js
    - scriptaculous.js
    - slider.js
    - prototype.js

NOTE: The sound.js and unittest.js files are optional.

How to Use script.aculo.us Library?

Now you can include *script.aculo.us* script as follows:

```
<html>
<head>
<title>script.aculo.us examples</title>
   <script type="text/javascript"
   src="/javascript/prototype.js"></script>
   <script type="text/javascript"
   src="/javascript/scriptaculous.js"></script>
</head>
<body>
........
</body>
</html>
```

By default, scriptaculous.js loads all of the other JavaScript files necessary for effects, drag-and-drop, sliders, and all the other script.aculo.us features.

If you don't need all the features, you can limit the additional scripts that get loaded by specifying them in a comma-separated list, e.g.:

```
<html>
<head>
<title>script.aculo.us examples</title>
   <script type="text/javascript"
   src="/javascript/prototype.js"></script>
   <script type="text/javascript"
   src="/javascript/scriptaculous.js?load=effects,dragdrop">
   </script>
</head>
<body>
........
</body>
</html>
```

The scripts that can be specified are:

- effects
- dragdrop
- builder
- controls
- slider

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

How to Call a script.aculo.us Library Function?

To call a script.aculo.us library function, use HTML script tags as shown below:

```
<html>
<head>
<title>script.aculo.us examples</title>
   <script type="text/javascript"
   src="/javascript/prototype.js"></script>
   <script type="text/javascript"
   src="/javascript/scriptaculous.js?load=effects,dragdrop">
   </script>

   <script type="text/javascript" language="javascript">
   // <![CDATA[
   function action(element){
    new Effect.Highlight(element,
        { startcolor: "#ff0000",
          endcolor: "#0000ff",
          restorecolor: "#00ff00",
          duration: 8
        });
   }
   // ]]>
   </script>

</head>
<body>
<div id="id" onclick="action(this);">
Click on this and see how it change its color.
</div>
</body>
</html>
```

Here we are using the Effect module and we are applying *Highlight* effect on an element. To understand it in a better way, you can use our online compiler by clicking the 'Try it' option in the tutorial.

Another easy way to call any module's function is inside event handlers as follows:

```
<html>
<head>
<title>script.aculo.us examples</title>

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

</head>
<body>
<div onclick="new Effect.BlindUp(this, {duration: 5})">
    Click here if you want this to go slooooow.
</div>
</body>
```

```
</html>
```

Use our online compiler for practical understanding of the same.

2. SCRIPT.ACULO.US – MODULES

script.aculo.us is divided into modules, each with its own JavaScript file. These modules are explained here:

Effects

The effects module comes with more than twenty-five visual effects and seven transition modes.

Drag and Drop

You will use the drag and drop module to make any element *draggable*, turn it into a drop zone, or even make an entire series of elements sortable so that you can rearrange them by dragging and dropping.

Sliders

A slider is a sort of small rail or track, along which you can slide a handle. It translates into a numerical value. With script.aculo.us, you can create such sliders with a lot of control.

Autocompletion

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

In-place Editing

You can make any text or collection of items editable in-place by simply clicking it.

Builder

A helper to build DOM fragments in JavaScript. This is a developer tool that eases DOM creation considerably.

Sound

Version 1.7.1 introduced a sound system that lets you play sounds easily, queue them up, use multiple tracks, and so on.

3. SCRIPT.ACULO.US – EFFECTS

script.aculo.us effects are divided into two groups:

- **Core Effects**: The following six core effects are the foundation of the script.aculo.us Visual Effects JavaScript library.

    - Effect.Opacity
    - Effect.Scale
    - Effect.Morph
    - Effect.Move
    - Effect.Highlight
    - Effect.Parallel

    All the core effects support various common parameters as well as effect-specific parameters and these effect names are case-sensitive.

    All the effect-specific common parameters have been discussed in this tutorial along with the effects.

- **Combination Effects**: All the combination effects are based on the five Core Effects, and are thought of as examples to allow you to write your own effects.

    Usually these effects rely on the parallel, synchronized execution of other effects. Such an execution is readily available, hence creating your own combined effects is very easy. Here is a list of Combination Effects:

    - Effect.Appear
    - Effect.Fade
    - Effect.Puff
    - Effect.DropOut
    - Effect.Shake
    - Effect.SwitchOff
    - Effect.BlindDown
    - Effect.BlindUp
    - Effect.SlideDown
    - Effect.SlideUp
    - Effect.Pulsate
    - Effect.Squish
    - Effect.Fold

- Effect.Grow
- Effect.Shrink

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

- Effect.toggle

Common Parameters

The following common options can be set for all the core effects:

Option	Description
duration	Duration of the effect in seconds, given as a float. Defaults to 1.0.
fps	Target these many frames per second. Defaults 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
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, it 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}.

delay	Sets the number of seconds to wait before the effect actually starts. Defaults to 0.0.
direction	Sets the direction of the transition. Values can be either 'top-left', 'top-right', 'bottom-left', 'bottom-right' or 'center' (Default). Applicable only on Grow and Shrink effects.

Here is an example to apply one or more of the above-mentioned parameters. All the parameters are put in a {} and they are separated by comma (,).

```
<html>
<head>
<title>script.aculo.us examples</title>

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

</head>
<body>
<p>Try by giving different parameters</p>
<div onclick="new Effect.Opacity(this,
      {
        duration: 2.0,
        transition: Effect.Transitions.linear,
        from: 1.0, to: 0.5
      });">
    Click here to see the result:
</div>
</body>
</html>
```

To understand it better, use the **Try it** option available online.

Callback Methods

You can apply any of the above-mentioned parameters to any element at various events while the effect is running. This is done by writing a callback method in JavaScript for that element.

To use a callback method, you have additional four parameters as listed below:

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 places.
afterUpdate	Called on each iteration of the effects rendering loop, after the redraw takes places.
afterFinish	Called after the last redraw of the effect was made.

Within the effect object, there are several useful variables you can access and use them in your callback functions:

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	The times (in ms) when the effect was started.
effect.finishOn	The times (in ms) when the effect will be finished after starting an effect.
effect.effects[]	On an Effect.Parallel effect, there's an effects[] array containing the individual effects the parallel effect is composed of.

Example

The following example shows how to use a callback:

```
<html>
<head>
<title>script.aculo.us examples</title>

   <script type="text/javascript"
   src="/javascript/prototype.js"></script>
   <script type="text/javascript"
   src="/javascript/scriptaculous.js?load=effects"></script>
   <script type="text/javascript">
   function OnFinish(obj){
     alert("Finishing - I'm element :" + obj.element.id);
   }
   function OnStart(obj){
     alert("Starting - I'm element :" + obj.element.id);
   }
   function myEffect(myObject){
      new Effect.Highlight(myObject,
                        { startcolor:'#ffffff',
                          endcolor:'#ffffcc',
                          duration: 0.5,
                          afterFinish: OnFinish,
                          beforeStart: OnStart
                         });
    }
    </script>
</head>
<body>
<p>Click following line to see the result:</p>
<div onclick="myEffect(this)" id="bestdiv">
    Click me to see the result!
</div>
</body>
</html>
```

NOTE: Here *startcolor* and *endcolor* are effect-specific parameters. We will discuss these parameters in *Effect.Highlight*.