## **Scriptaculous**

# CSE 190 M (Web Programming), Spring 2008 University of Washington

Except where otherwise noted, the contents of this presentation are © Copyright 2008 Marty Stepp and Jessica Miller and are licensed under the Creative Commons Attribution 2.5 License.



### Scriptaculous overview

Scriptaculous is another JavaScript library, built on top of Prototype, that adds:

- visual effects (animation, fade in/out, highlighting)
- drag and drop
- Ajax features:
  - Auto-completing text fields (drop-down list of matching choices)
  - In-place editors (clickable text that you can edit and send to server)
- some DOM enhancements
- other stuff (unit testing, etc.)

## Downloading and using Scriptaculous

```
<script src="http://www.cs.washington.edu/education/courses/cse190m/08sp/prototype.js"
type="text/javascript"></script>
<script src="http://www.cs.washington.edu/education/courses/cse190m/08sp/scriptaculous.js"
type="text/javascript"></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></scrip
```

- option 1: link to Scriptaculous on the CSE 190 M web site
  - notice that you must still link to Prototype before linking Scriptaculous
- option 2: download the .zip file from their downloads page, and extract the 8 . js files from its src/folder to the same folder as your project

# **Learning about Scriptaculous**

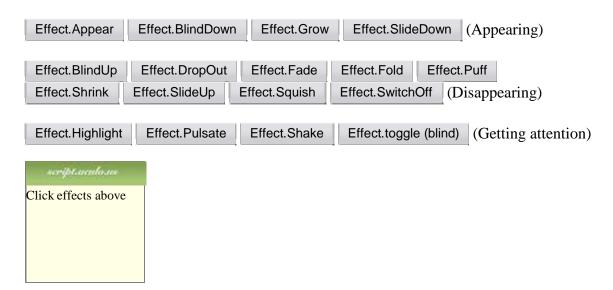
There's no complete online API documentation (argh), but the following are useful resources:

- Scriptaculous wiki documentation
  - Visuals
  - Core FX
  - Combo FX
  - Sortables
  - Drag 'n' Drop 1 | 2 | 3 | 4
  - Auto-Completion 1 | 2
  - DOM
- Scriptaculous Effects Cheat Sheet

#### Visual effects

Elements that appear, disappear, animate, grow, shrink, highlight, jiggle, ...

#### Effects demo



## Adding effects to an element

```
new Effect.name(element or id);

new Effect.Shake("sidebar");

var buttons = $$("results > button");

for (var i = 0; i < buttons.length; i++) {
   new Effect.Fade(buttons[i]);
}</pre>
```

- add an effect to an element by constructing an Effect and passing the element's DOM object or its id
- six core effects are used to implement all effects on the previous slides:
  - Effect. Highlight, Effect. Morph, Effect. Move, Effect. Opacity, Effect. Parallel, Effect. Scale

### **Effect options**

```
new Effect.name(element or id,
{
    option: value,
    ...
    option: value,
}
);

new Effect.Opacity("my_element",
    duration: 2.0,
    from: 1.0,
    to: 0.5
}
);
```

- many effects can be customized by passing additional options
- options: delay, direction, duration, fps, from, queue, sync, to, transition

#### **Effect events**

```
new Effect.Fade("my_element", {
   duration: 3.0,
   afterFinish: displayMessage
});

function displayMessage(effect) {
   alert(effect.element + " is done fading now!");
}
```

- all effects have the following events that you can handle: beforeStart, beforeUpdate, afterUpdate, afterFinish
- the afterFinish event fires once the effect is done animating
  - useful do something to the element (style, remove, etc.) when effect is done
- each of these events receives the Effect object as its parameter
  - its properties: element, options, currentFrame, startOn, finishOn

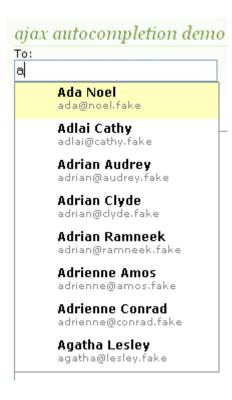
## **Auto-completion**

Text fields that let you type in partial text and suggest values that contain that text

## **Auto-completing text fields**

Scriptaculous offers ways to make a text box that auto-completes based on prefix strings:

- Autocompleter.Local: auto-completes from an array of choices
- Ajax. Autocompleter: fetches and displays list of choices using Ajax



#### Using Autocompleter.Local

```
new Autocompleter.Local(
    element or id of text box,
    element or id of div,
    array of choices,
    { options }
);
```

- you must create an (initially empty) div to store the auto-completion matches
  - it will be inserted as a ul that you can style with CSS
  - the user can select items by pressing Up/Down arrows; selected item is given a class of selected
- pass the choices as an array of strings
- pass any extra options as a fourth parameter between { }
  - options: choices, partialSearch, fullSearch, partialChars, ignoreCase

# Autocompleter.Local demo

```
<input id="bands70s" size="40" type="text" />
<div id="bandlistarea"></div>

window.onload = function() {
  new Autocompleter.Local(
    "bands70s",
    "bandlistarea",
    ["ABBA", "AC/DC", "Aerosmith", "America", "Bay City Rollers", ...],
    {}
  );
};

JS
```

# Using Ajax. Autocompleter

```
new Ajax.Autocompleter(
    element or id of text box,
    element or id of div,
    url,
    { options }
);
```

- when you have too many choices to hold them all in an array, you can instead fetch subsets of choices from the server using Ajax
- instead of passing choices as an array, pass a URL from which to fetch them
  - the choices are sent back from the server as an HTML ul with li elements in it
- options: paramName, tokens, frequency, minChars, indicator, updateElement, afterUpdateElement, callback, parameters

## **Drag and Drop**

Elements that can be moved by dragging them with the mouse

## Drag and drop facilities

Scriptaculous provides several classes for supporting drag-and-drop functionality:

- Draggable: an element that can be dragged
- Draggables: manages all Draggable objects on the page
- Droppables: elements on which a Draggable can be dropped
- Sortable: a list of items that can be reordered

#### Draggable

```
new Draggable(element or id,
{ options }
);

JS
```

- specifies an element as being able to be dragged
- options: handle, revert, snap, zindex, constraint, ghosting, starteffect, reverteffect, endeffect
- event options: onStart, onDrag, onEnd
  - each callback accepts two parameters: the Draggable object, and the mouse event

#### Draggable example

```
window.onload = function() {
   new Draggable("draggabledemo1");
   new Draggable("draggabledemo2", {revert: true, snap: [40, 40]});
};
```

Draggable demo.
Default options.

Draggable demo. {snap:[40, 40], revert:true}

#### Draggables

- a global helper for accessing/managing all Draggable objects on a page
- (not needed for this course)
- properties: drags, observers
- methods: register, unregister, activate, deactivate, updateDrag, endDrag, keyPress, addObserver, removeObserver, notify

#### Droppables

```
Droppables.add(element or id,
{ options }
);

JS
```

- specifies an element as being able to be dragged
- options: accept, containment, hoverclass, overlap, greedy
- event options: onHover, onDrop
  - each callback accepts three parameters: the Draggable, the Droppable, and the event
  - Shopping Cart demo

## Drag/drop shopping demo

```
<img id="product1" src="images/shirt.png" alt="shirt" />
<img id="product2" src="images/cup.png" alt="cup" />
<div id="droptarget"></div>
HTML
```

```
window.onload = function() {
  new Draggable("product1");
  new Draggable("product2");
  Droppables.add("droptarget", {onDrop: productDrop});
}

function productDrop(drag, drop, event) {
  alert("You dropped " + drag.id);
}
```





#### Sortable

```
Sortable.create(element or id of list,
{ options }
);

JS
```

- specifies a list (ul, ol) as being able to be dragged into any order
- implemented internally using Draggables and Droppables
- options: tag, only, overlap, constraint, containment, format, handle, hoverclass, ghosting, dropOnEmpty, scroll, scrollSensitivity, scrollSpeed, tree, treeTag
- event options: onChange, onUpdate
  - each callback receives the affected element as its parameter
  - NOTE: for onUpdate to work, each li must have an id attribute
- to make a list un-sortable again, call Sortable.destroy on it

## Sortable demo

```
    Homer
    Marge
    Bart
    Lisa
    Maggie

HTML
```

```
window.onload = function() {
   Sortable.create("simpsons");
};
```

- 1. Homer
- 2. Marge
- 3. Bart
- 4. Lisa
- 5. Maggie

## **Events on rearranged items**

```
window.onload = function() {
    Sortable.create("simpsons", {
        onUpdate: listUpdate
    });
};

function listUpdate() {
    // I can do anything I like here; create an Ajax.Request, etc.
    new Effect.Shake("simpsons");
}
```

- 1. Homer
- 2. Marge
- 3. Bart
- 4. Lisa
- 5. Maggie

#### Persistent saved items

problem: rearranged items are not "remembered"; they return to their original order when we revisit the page

- a Sortable has events you can handle when the list order changes:
  - onChange: during a drag, each time the list order changes
  - onUpdate: when a drag is done and the order has changed
- in a handler for a Sortable's event, POST the data to the server to save it

### **Subtleties of sortable lists**

- if the elements of the list change after you make it sortable (if you add or remove an item using the DOM, etc.), the Sortable-ness breaks
  - symptom: some elements will not be draggable, or can't be dragged past
  - must call Sortable. create on the list again to fix it
- the onUpdate event will not work unless each li has an id of the form listID\_index, e.g. "simpsons\_0"

```
    Homer
    Marge
    Bart
    Lisa
    Maggie

HTMI
```

## In-place editing

Elements whose text content can be changed dynamically (and saved to a server)

#### Ajax.InPlaceEditor

- options: okButton, okText, cancelLink, cancelText, savingText, clickToEditText, formId, externalControl, rows, onComplete, onFailure, cols, size, highlightcolor, highlightendcolor, formClassName, hoverClassName, loadTextURL, loadingText, callback, submitOnBlur, ajaxOptions
- event options: onEnterHover, onLeaveHover, onEnterEditMode, onLeaveEditMode

#### Ajax.InPlaceCollectionEditor

```
new Ajax.InPlaceCollectionEditor(element or id,
    url,
    {
       collection: array of choices,
       options
    }
);
```

- a variation of Ajax. InPlaceEditor that gives a collection of choices
- requires collection option whose value is an array of strings to choose from
- all other options are the same as Ajax. InPlaceEditor