中山大學
SUN YAT-SEN UNIVERSITY

# Lecture 22
# Rich Client Side and RIA

**SE-805 Web 2.0 Programming (supported by Google)**

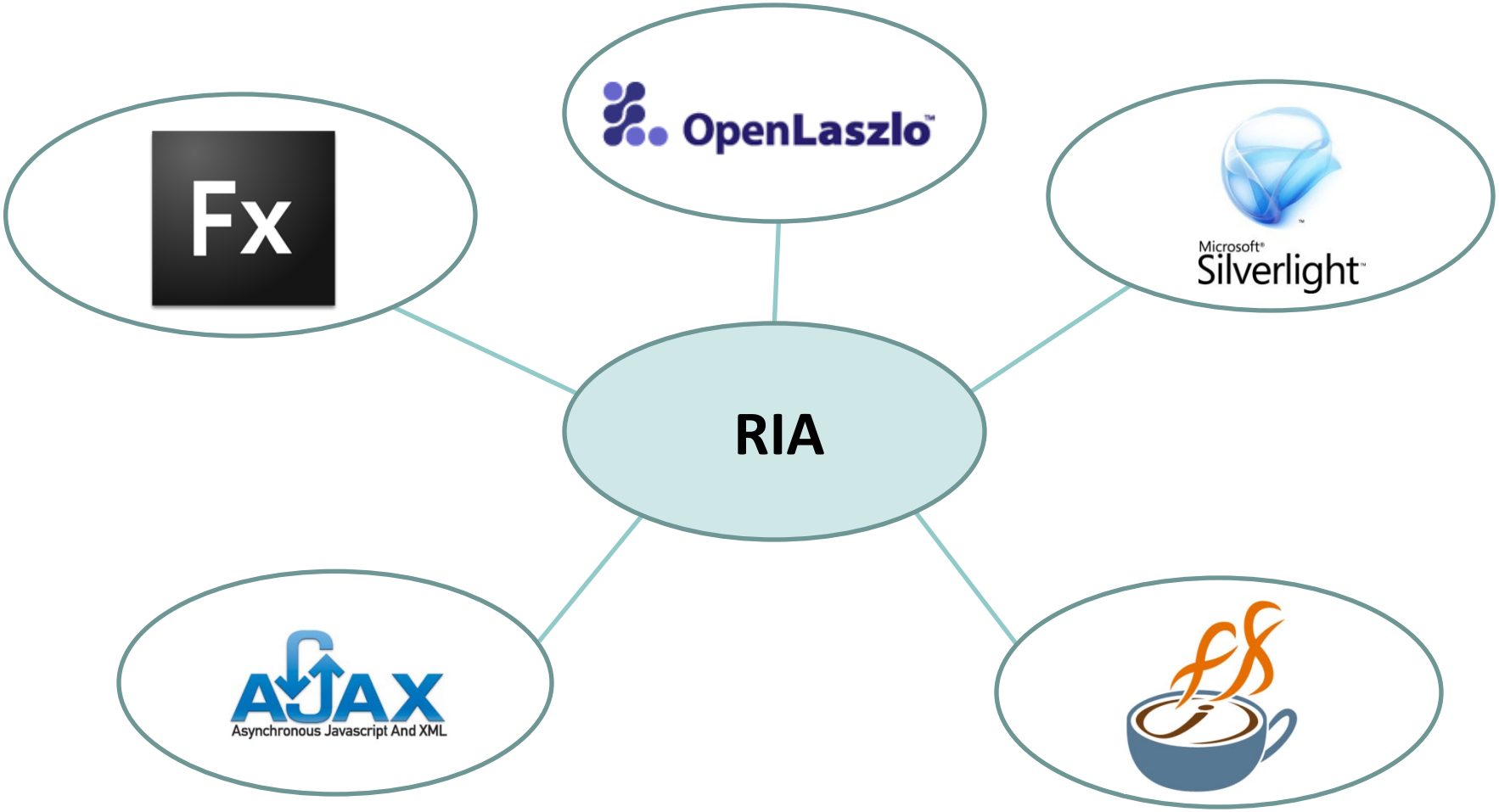
http://my.ss.sysu.edu.cn/courses/web2.0/

**School of Software, Sun Yat-sen University**

# Outline

- **RIA Survey**
- script.aculo.us
  - Visual effects
  - Drag and drop; sortable lists
  - Auto-completing text fields
  - Other features

# Introduction of RIA Techniques



# Introduction to Flex

- **Programming languages:** MXML & ActionScript 3.0
- **Development environments:** Flex Builder (Eclipse based), SEPY, Notepad, etc…
- **Plugins required:** Flash Player 9.0.16 or higher

# Introduction to Silverlight

- **Programming languages:** XAML, C#, JS, Ruby
- **Development environments:** Microsoft Expression Blend 2, Microsoft Visual Studio, etc…
- **Plugins required:** Silverlight Runtime 1.0 or higher

# Introduction to OpenLaszlo

- **Programming languages:** LZX, ECMAScript
- **Development environments:** IDE4Laszlo, Notepad, etc…
- **Plugins recommended:** Flash Player 7 or higher

# Introduction to AJAX

- **Programming languages:** JavaScript, XML, HTML
- **Development environments:** Notepad, Eclipse, GWT Designer, etc…
- **Plugins required:** none (only JS library downloading at runtime)

# Introduction to JavaFX

- **Programming languages:** JavaFX Script, Java
- **Development environments:** JavaFXPad, NetBeans 6.0, JFXBuilder, Notepad, etc…
- **Plugins required:** Java Runtime Environment (JRE) 1.5 or higher

# Showcase

**Flex sample applications (Flash Player 9+ required):**

Scrapblog

Sumeco

**Silverlight sample applications (Silverlight runtime required):**

Tafiti

Sprawl

**OpenLaszlo sample applications (Flash Player sometimes required):**

World Clock

Pandora

**AJAX sample applications:**

PageBakery CMS

Netvibes, Google Maps, Gmail, Flickr

**JavaFX sample applications (Java Runtime Environment 1.5 or higher required):**

JavaFX Script Studiomoto Demo

JavaFX Script JavaFXPad Demo

# Development environments

- **Flex**: Adobe Flex Builder
- **Silverlight**: Microsoft Expression Blend 2
- **OpenLaszlo**: IDE4Laszlo (Eclipse plugin)
- **AJAX**: Notepad, MyEclipse (Eclipse plugin)
- **JavaFX**: JFXBuilder

# Differences

- Community size (Flex & Ajax: Huge – Silverlight & OpenLaszlo: upcoming – JavaFX: small)
- Plugin requirement
- Silverlight is using hardware acceleration, which makes it stronger on with graphics
- Flex apps are easily ported to the desktop via Adobe AIR
- Different video format support (FLV in the lead)
- Database support

# Pros and cons

Pros:

- Only tiny portions of the screen need to be refreshed after user interaction
- New and highly innovative things are shown every day

Cons:

- Browser back-button functionality often lost
- Search Engine Optimalisation
- Plugin install-base
- RIA designers/developers forget about usability standards

# Recommendations

1. Try the techniques for yourself.
2. Find out which technique will be the most suitable for your project/work.
3. Keep updated about the work of others.
4. Keep informed about new techniques/frameworks/usability aspects, etc…

# Outline

- RIA Survey
- script.aculo.us
  - **Visual effects**
  - Drag and drop; sortable lists
  - Auto-completing text fields
  - Other features

# script.aculo.us Overview

- script.aculo.us : a 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://ssw2p.3322.org/public/scripts/prototype/prototype.js"
type="text/javascript"></script>

<script src="http://ssw2p.3322.org/public/scripts/scriptaculous/scriptaculous.js"
type="text/javascript"></script>
```
JS

- Option 1: link to Scriptaculous on the our 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
- Documentation available on their wiki
- script.aculo.us Effects Cheat Sheet

# Visual Effects

appear | blindDown | grow | slideDown

(appearing)

blindUp | dropOut | fade | fold | puff

shrink | slideUp | squish | switchOff

(disappearing)

highlight | pulsate | shake | morph

Effect.Move | Effect.Scale | Effect.toggle (blind)

(Getting attention)

# Adding Effects to an Element

```
element.effectName();   // for most effects

// some effects must be run the following way:
new Effect.name(element or id);
```
JS

```
$("sidebar").shake();

var buttons = $$("results > button");
for (var i = 0; i < buttons.length; i++) {
  buttons[i].fade();
}
```
JS

- The effect will begin to animate on screen (asynchronously) the moment you call it
- 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

```JS
element.effectName(
  {
    option: value,
    option: value,
    ...
  }
);
```

```JS
$("my_element").pulsate({
  duration: 2.0,
  pulses: 2
});
```

- Many effects can be customized by passing additional options (note the {})
- Options (wiki): delay, direction, duration, fps, from, queue, sync, to, transition
- Q: How would we show two effects in a row on the same element?

# Effect Events

```JS
$("my_element").fade({
  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
  - Some effects (e.g. Shrink) are technically "parallel effects", so to access the modified element, you write effect.effects[0].element rather than just effect.element

# Outline

- RIA Survey
- script.aculo.us
  - Visual effects
  - **Drag and drop; sortable lists**
  - Auto-completing text fields
  - Other features

# Drag and Drop

- Scriptaculous provides several objects 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**
- Shopping Cart demo

# 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 handler function accepts two parameters: the Draggable object, and the mouse event

# Draggable Example

```HTML
<div id="draggabledemo1">Draggable demo. Default options.</div>
<div id="draggabledemo2">Draggable demo.
 {snap: [40,40], revert: true}</div>
```

```JS
document.observe("dom:loaded", function() {
  new Draggable("draggabledemo1");
  new Draggable("draggabledemo2", {revert: true, snap: [40, 40]});
});
```

script.aculo.us

Draggable demo. Default options.

script.aculo.us

Draggable demo.

{snap:[60, 60], revert:true}

# Draggables

- A global helper for accessing/managing all Draggable objects on a page
- 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

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

HTML

```js
document.observe("dom:loaded", function() {
  new Draggable("product1");
  new Draggable("product2");
  Droppables.add("droptarget", {onDrop: productDrop});
});

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

JS


# 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
- To make a list un-sortable again, call Sortable.destroy on it

# Sortable Demo

```HTML
<ol id="simpsons">
  <li id="simpsons_0">Homer</li>
  <li id="simpsons_1">Marge</li>
  <li id="simpsons_2">Bart</li>
  <li id="simpsons_3">Lisa</li>
  <li id="simpsons_4">Maggie</li>
</ol>
```

```JS
document.observe("dom:loaded", function() {
  Sortable.create("simpsons");
});
```

1. Homer
2. Marge
3. Bart

# Sortable list events

| Event | Description |
|---|---|
| onChange | when any list item hovers over a new position while dragging |
| onUpdate | when a list item is dropped into a new position (more useful) |

```JS
document.observe("dom:loaded", function() {
  Sortable.create("simpsons", {
     onUpdate: listUpdate
  });
});
```

- onChange handler function receives the dragging element as its parameter
- onUpdate handler function receives the list as its parameter

# Sortable list events example

```JS
document.observe("dom:loaded", function() {
  Sortable.create("simpsons", {
     onUpdate: listUpdate
  });
});

function listUpdate(list) {
  // can do anything I want here; effects, an Ajax request, etc.
  list.shake();
}
```

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

# Subtleties of Sortable events

- For onUpdate to work, each li **must** have an id of the form *listID_index*

```HTML
<ol id="simpsons">
  <li id="simpsons_0">Homer</li>
  <li id="simpsons_1">Marge</li>
  <li id="simpsons_2">Bart</li>
  <li id="simpsons_3">Lisa</li>
  <li id="simpsons_4">Maggie</li>
</ol>
```

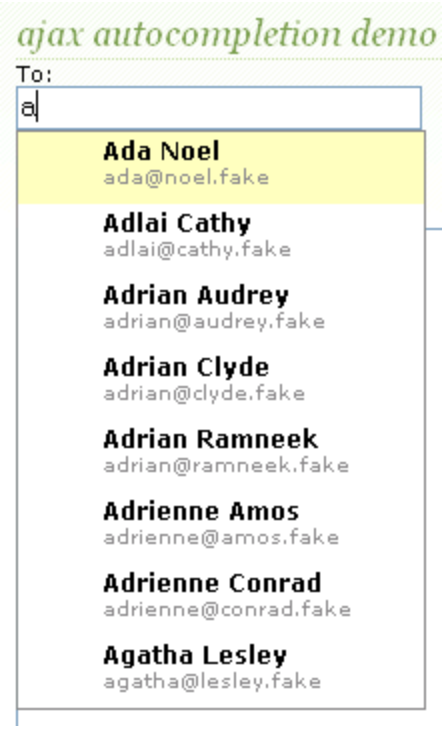
- If the elements of the list change after you make it sortable (if you add or remove an item using the DOM, etc.), the new items can't be sorted
  - Must call Sortable.create on the list again to fix it

# Outline

- RIA Survey
- script.aculo.us
  - Visual effects
  - Drag and drop; sortable lists
  - **Auto-completing text fields**
  - Other features

# Auto-completing Text Fields

- script.aculo.us 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 to show completions,
  array of choices,
  { options }
);
```
JS

- 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

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

```JS
document.observe("dom:loaded", function() {
  new Autocompleter.Local(
    "bands70s",
    "bandlistarea",
    ["ABBA", "AC/DC", "Aerosmith", "America", "Bay City Rollers", ...],
    {}
  );
});
```

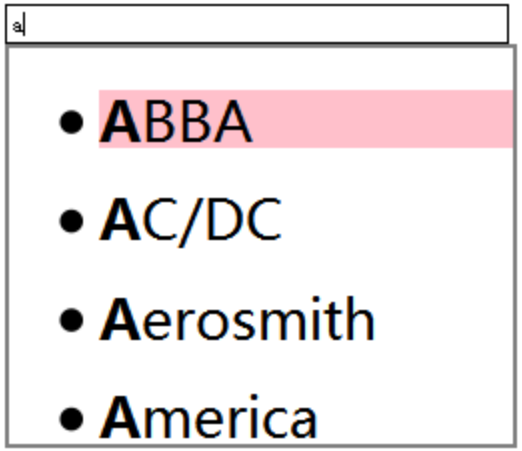
a

- **A**BBA
- **A**C/DC
- **A**erosmith

# Autocompleter Styling

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

```CSS
#bandlistarea {
  border: 2px solid gray;
}
/* 'selected' class is given to the autocomplete item currently chosen */
#bandlistarea .selected {
  background-color: pink;
}
```



# Using Ajax.Autocompleter

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

- 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

# Ajax.InPlaceEditor

```JS
new Ajax.InPlaceEditor(element or id,
  url,
  { options }
);
```

- 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
  }
);
```
JS

- 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

# Outline

- RIA Survey
- Script.aculo.us
  - Visual effects
  - Drag and drop; sortable lists
  - Auto-completing text fields
  - **Other features**

# Playing Sounds (API)

| method | description |
|---|---|
| Sound.play("*url*"); | plays a sound/music file |
| Sound.disable(); | stops future sounds from playing (doesn't mute any sound in progress) |
| Sound.enable(); | re-enables sounds to be playable after a call to Sound.disable() |

```
Sound.play("music/java_rap.mp3");
Sound.play("music/wazzaaaaaap.wav");
```
JS

- To silence a sound playing in progress, use Sound.play('', {replace: true});
- Cannot play sounds from a local computer (must be uploaded to a web site)

# Other Neat Features

- slider control:

```
new Control.Slider("id of knob", "id of track", {options});
```
JS

- Builder - convenience class to replace

```
var img = Builder.node("img", {
  src: "images/lolcat.jpg",
  width: 100, height: 100,
  alt: "I can haz Scriptaculous?"
});
$("main").appendChild(img);
```
JS

- Tabbed UIs

# Summary

- RIA Survey
  - Flex, OpenLaszlo, Silverlight, JavaFX, Ajax
- script.aculo.us
  - Visual effects
  - Drag and drop; sortable lists
  - Auto-completing text fields
  - Other features

# Exercises

- Write a simple **RIA** to-do list application as a Web page.
  - A  element wraps all html elements
  - A form for adding new to-do items
  - A sortable list of all to-do items
  - Buttons of "select all", "deselect all", and "remove", and all actions in a RIA way
    - Selected/Deselected items will be highlighted
    - Adding animations to removals of items
  - When the "add" button is clicked the new to-do item will be inserted to the bottom of the list in a RIA way
    - Adding animations to the insertion of a new item

# Further Readings

- Introduction of RIA http://en.wikipedia.org/wiki/Rich_Internet_application
- Adobe Flex http://www.adobe.com/products/flex/
- OpenLaszlo http://www.openlaszlo.org/
- Microsoft Silverlight http://www.silverlight.net/
- JavaFX http://javafx.com/
- script.aculo.us http://script.aculo.us/
- script.aculo.us wiki http://wiki.github.com/madrobby/scriptaculous/
- script.aculo.us tutorial http://www.tutorialspoint.com/script.aculo.us/

# Thank you!