Web Programming Step by Step

Chapter 12
Web 2.0 and Scriptaculous

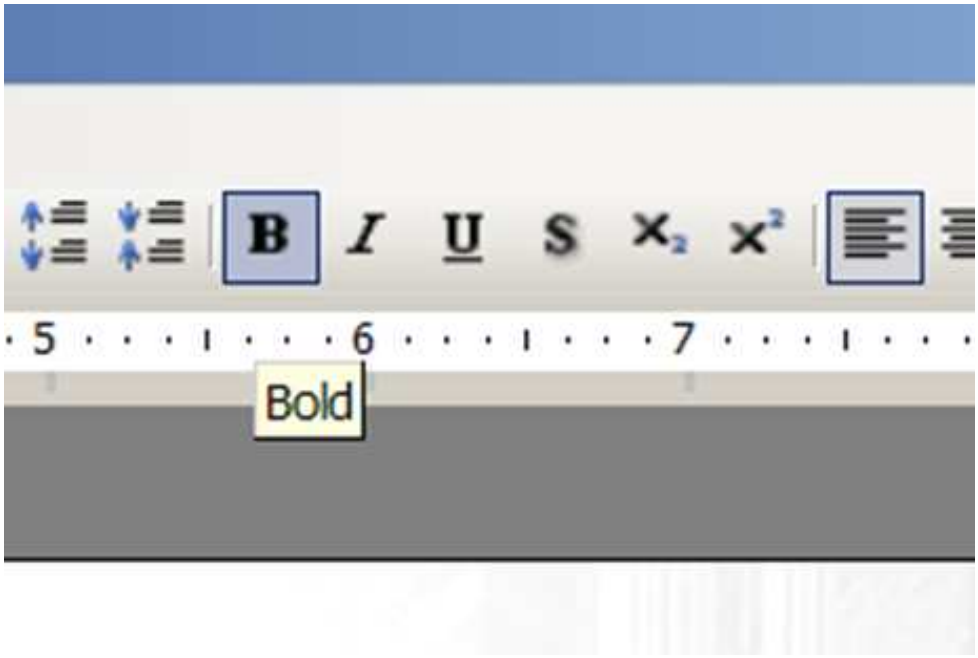
12.1: Designing for Web 2.0

What is usability?

- **usability**: the effectiveness with which users can achieve tasks in one software environment
- studying and improving usability is part of **Human-Computer Interaction** (HCI)

Visibility and feedback

- **visibility**: ability for user to find controls that are meant to be interacted with
  - Where are they?
  - What is their state? ("Is this setting on or off?")
- **feedback**: response from the control to the user before, during, or after an interaction


B
I
U
S
5
6
7
Bold

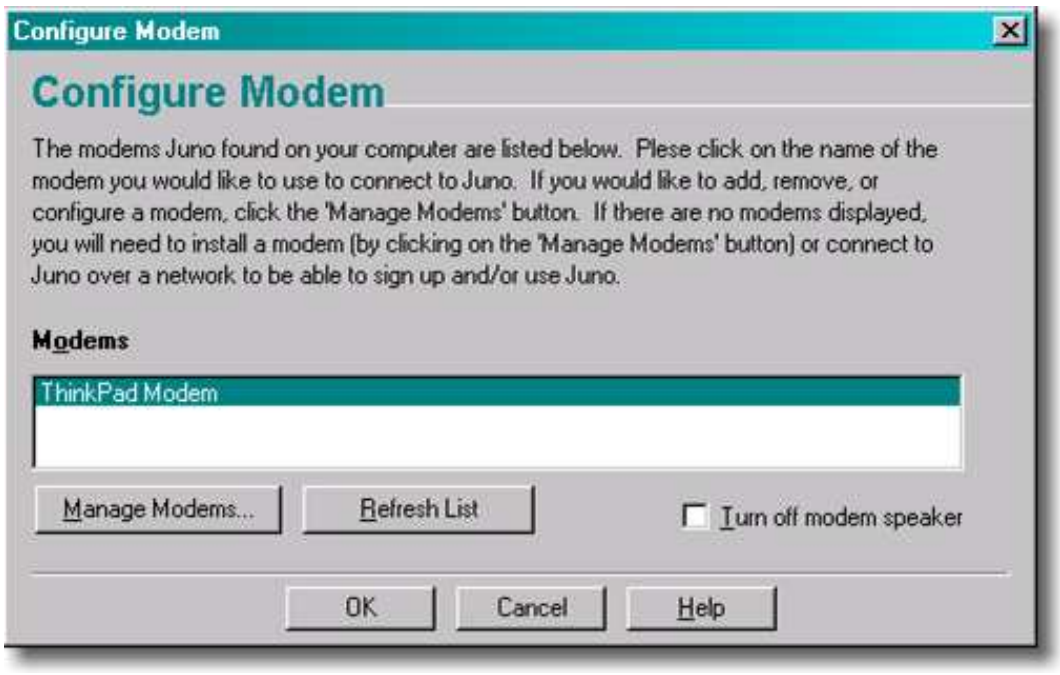
Common web usability problems

- cluttered or otherwise poor layout
- requires horizontal scrolling, or makes assumptions about user's screen size
- poorly chosen colors
- uses frames
- uses splash screen(s)
- poor or missing navigation controls (Back, Forward, Home)
- text is not scannable (can't be read quickly)

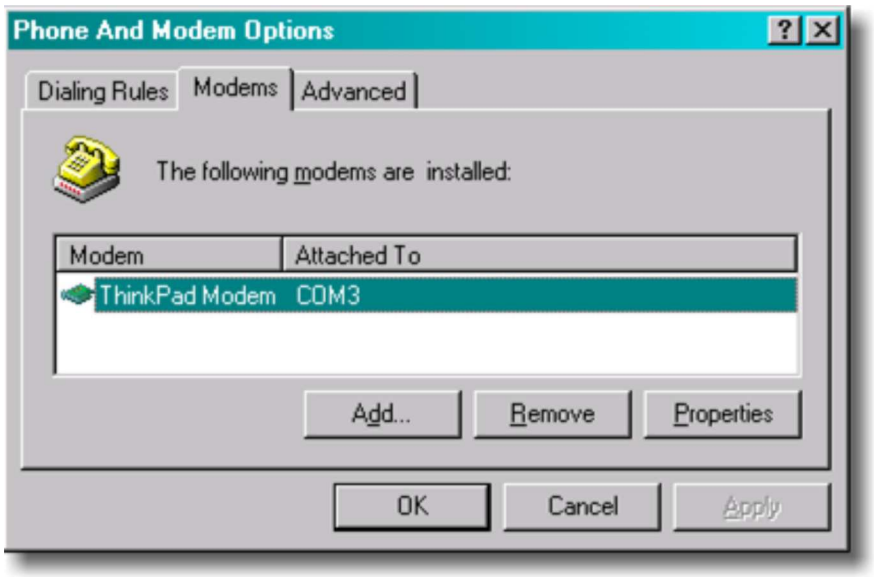
Content usability problems

- most important content isn't on the first page / screenful
- nondescript headings
- too many ads (or things that appear to be ads)
- important site content is contained in PDF documents
- isn't designed to be easily indexed by a search engine (HTML title, meta tags, page text, link text, etc.)
- tiny thumbnails of detailed large photos: 

Users do not read



vs.



- this also often applies to CSE students

Link usability problems

- links that don't say where they go
- badly chosen link text (such as "Click here for more info")
- links that forcibly open a new browser window
- links opened by complex Javascript needlessly

- visited links don't appear in a different color

Feature usability problems

- poorly performing site search
- having a web search feature (why??)
- not having a site map or other means to navigate the site
- relying on non-standard plugins or browser versions(e.g. Overly reliant on Flash, Java applets, etc.)

Web design suggestions

- place your name and logo on every page and make the logo a link to the home page
- provide search if the site has more than 100 pages
- write straightforward and simple headlines and page titles that clearly explain what the page is about
- structure the page to facilitate scanning and help users ignore large chunks of the page in a single glance: for example, use grouping and subheadings to break a long list into several smaller units
- instead of cramming everything about a product or topic into a single, infinite page, use hypertext to structure the content space into a starting page that provides an overview and several secondary pages that each focus on a specific topic
- use link titles to provide users with a preview of where each link will take them, before they have clicked on it

More web design suggestions

- Use relevance-enhanced image reduction when preparing small photos and images: instead of simply resizing the original image to a tiny and unreadable thumbnail, zoom in on the most relevant detail and use a combination of cropping and resizing.
- Ensure that all important pages are accessible for users with disabilities, especially blind users
- Do the same as everybody else: if most big websites do something in a certain way, then follow along since users will expect things to work the same on your site
- Jakob's Law of the Web User Experience: users spend most of their time on other sites, so that's where they form their expectations for how the Web works
- Test your design with real users as a reality check. People do things in odd and unexpected ways, so even the most carefully planned project will learn from usability testing.

Sites about web design

- A List Apart
- CSS Play
- css/edge
- Design by Fire
- Jeffrey Zeldman Presents
- QuirksMode

Writing for the web

- People read web page text differently than they read books, etc.
- Writing for the web includes:
  - subheads
  - bulleted lists
  - highlighted keywords
  - short paragraphs
  - the "inverted pyramid"
  - (put the most newsworthy information at the top, and then the remaining information follows in order of importance, with the least important at the bottom)
  - a simple writing style

Web pages that suck

What's wrong with each of these web sites?

- http://www.envy-hair.co.uk/index.html
- http://www.corvalliscommunitypages.com/
- http://www.pigletscatering.co.uk/
- http://www.bigbearparties.com/
- http://www.developingwebs.net/
- http://www.bobmarshall.com/
- http://www.orchy.com/dictionary/
- http://www.delmarvadatacenter.com/main.html
- http://www.videosphotosanddjs.com/
- credit: webpagesthatsuck.com

Ajax usability

- since Ajax requests happen in the background, users may not know the page is loading
- well-designed web sites give visual cues to the user so they know to wait

Forms and usability

- client-side validation
- lighting up required elements left blank or filled out incorrectly
- avoiding `alert` unless absolutely necessary

Search Engine Optimization (SEO)

- get people to link to your site (particularly popular sites!)
- use relevant keywords in link text
    - example: My friend Marty Stepp is a swell guy!
- set descriptive `meta` tags
- use a site URL and page title that contains the keywords you want to match
- don't do "black-hat" stuff (link farms, hidden text, etc.)
- use Google Webmaster Tools: https://www.google.com/webmasters/tools/

12.2: Scriptaculous

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/scriptaculou
 type="text/javascript"></script>
```

- 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

Effect.Appear | Effect.BlindDown | Effect.Grow | Effect.SlideDown (Appearing)

Effect.BlindUp | Effect.DropOut | Effect.Fade | Effect.Fold | Effect.Puff
Effect.Shrink | Effect.SlideUp | Effect.Squish | Effect.SwitchOff (Disappearing)

Effect.Highlight | Effect.Pulsate | Effect.Shake | Effect.toggle (blind) (Getting attention)

Adding effects to an element

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

```
new Effect.Shake("sidebar");

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

- 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,
  }
);
```
JS

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

- 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!");
}
```
JS

- 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`

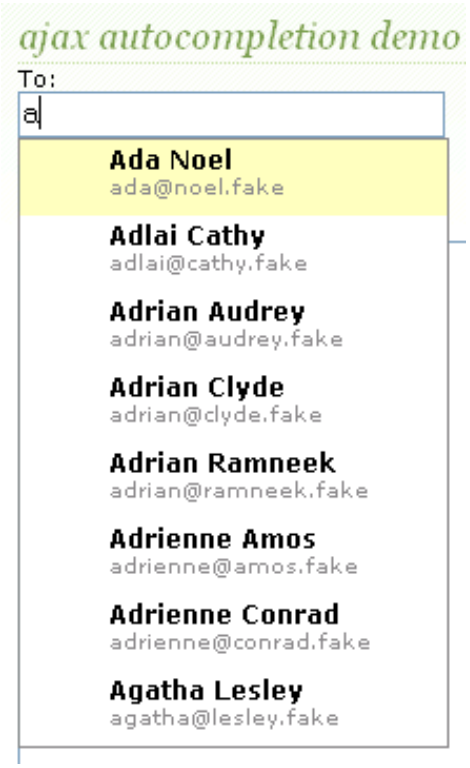
Auto-completion

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 }
);
```
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

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

```
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 }
);
```
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`

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

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

```JS
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);
}
```
JS

Sortable

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

- 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

```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
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");
}
```
JS

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"`

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

In-place editing

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

Ajax.InPlaceEditor

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

- 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`

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");
```
PHP

- 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 `document.createElement` :

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

- Tabbed UIs