CSCI 1320
Creating Modern Web Applications
Lecture 6: Dynamic Web Pages
Final Projects

• Team and project assignments were posted

• This week
  • Team should meet as a group
  • Decide responsibilities
  • Discuss project ideas and understanding
    • Make sure you are all on the same page
  • Contact sponsor
    • Introduce yourselves
    • Set up a meeting for this weekend or early next week
Resize Experience

• What pages did you try?
  • What happened when you resized them
  • When did this “work”
  • When did it “fail”

• Do web sites use the same pages for phone & browser?
  • Why or why not?

• How might this be done?
Interactive Applications

• Respond to user inputs
• Change the display (e.g. add fields, show errors, ...)
• Dynamically check and verify inputs
• Allow direct manipulation (drag and drop)
• Use animation to highlight or emphasize or show things
• Display external changes in real time
• Provide input help (e.g. text completion)
Achieving Interactivity

- Handling HTML events using JavaScript
  - Dynamically check and verify inputs
  - Handle direct manipulation
- Changing styles and the content of the page
  - Change the display (e.g. add fields, show errors, ...)
  - Providing input help (e.g. text completion)
  - Adding graphs, etc. to show output
- With animation/drawing/multimedia packages
- By talking to the server continually
  - Displaying external changes in real time
Dynamic Web Page Examples

• http://fred4.cs.brown.edu:5000/ (spheree)
• http://fred4.cs.brown.edu:8800/ (sign)
• http://conifer.cs.brown.edu/s6
HTML is a Tree

• When it is written
• When it is displayed
• Internally
Changing the Style and Content

- Document Object Model or DOM
  - Representation of HTML in the browser
  - As a set of objects representing the nodes
  - You can observe it using the debugger
- JavaScript can access the DOM
  - Get access to objects
    - Values of object attributes
    - Child objects & child text
    - Styles
  - Set values of objects
    - Setting attributes, children, styles, etc.
    - Setting text
  - Add/remove whole new subtrees
DOM Modifications

- Changing the DOM changes the display
  - When JavaScript returns control to the browser
  - As if the new DOM were the original HTML
- This provides most of what you need to do interaction
  - Change text and input values
  - Much can be done by changing classes
    - Hide/show, changing styles
  - Can do limited animation as well
    - Changing position, size, etc. dynamically
    - Changing things based on timers
DOM Modification Syntax

• Standard notation
  • document.getElementById("id")
  • document.id1.nestedid1.nestedid2 …
  • <element>.attribute
• Want this to be easier
  • Simple element selection and setting
  • Doing it for a set of elements at once
  • Not requiring ids for all elements
• Where have you seen selectors?
  • CSS
  • Why not use the same selectors
JQuery DOM Access

- JQuery is a library to simplify DOM access/modification
  - Plus make it easier to do standard manipulations
  - Plus build on what you already know from CSS
- $(“selector”)
  - Selector is effectively a CSS selector
  - What follows applies to ALL matching elements
  - $(“.test”).hide(), $(“#Sum”).val(sum)
  - $(“#sample”).html(“<em>This is sample text</em>”);
  - $(“.error”).attr(“color”, “red”);
- Using JQuery is pretty standard
  - And easier than using pure JavaScript
  - Cover in homeworks, lab
Using jQuery

• \$\ldots$.onChange(function() { ... })
• $("<div>....</div>")
• $(\ldots).html("<....>")
• $(\ldots).text(" string")
• $(function() { ... })
• $(\ldots).animate({height:300},"slow")

• <script type='text/javascript'
src='https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js'> </script>
Changing the DOM

• Easiest
  • Put all text on page, then hide/show as needed
  • Add or remove classes to change display properties
  • Set text or html or value for computed items
  • Moving items around within a list or table

• More difficult
  • Create new html for items in a list or table
    • Cloning original or just creating from scratch
    • Libraries exist for this
  • Actually creating new html for the page
    • Better done elsewhere
DOM Update and Accessibility

- Changing the DOM can cause accessibility problems
  - Updates can confuse screen readers
  - Updates might not be visible in high magnification
  - Updates might come too fast (before page was read)

- These should be addressed
  - If content updates for more than 5 seconds, provide the ability to pause, stop or hide the updates
  - Inform the user of changes (live region, alert, setting focus, highlight)
  - Inform user that the page is dynamic
  - Work without JavaScript, provide HTML alternative pages

- These need to be tested carefully (i.e. screen reader testing)
Question

Which of the following are not built-in to HTML(5)?

A. 2D and 3D drawing areas
B. Input from a web or phone camera
C. Drag and drop
D. Geolocation with arbitrary updating
E. All are built into HTML5
HTML = HTML5

• HTML5 is designed to support modern web apps
  • More interaction
  • More devices

• Multimedia and animations are more common
  • A large fraction of web sites are using them
  • They shouldn’t require plugins to be usable
  • These should be standard in all browsers

• Other features have similar properties
  • Simple databases, cookie management, ...

• HTML doesn’t provide enough context information
  • About the page (for search, readers, ...)
  • About forms (numbers, dates, ...)
HTML5 Forms

• Do forms work on your smart phone/tablet?
• Forms are the basis for much HTML interaction
  • But they are quite limiting
  • And not well-oriented to tablets / smart phones
  • And require significant JavaScript to validate
• HTML5 significantly expands the input types in forms
  • Text, password, submit, radio, checkbox, button
  • Color, date, datetime, email, month, number, range, search, tel, time, url, week
  • With built-in validation
  • Even regular-expression based validation
HTML5 Canvas

- A canvas is a drawing area on the page
  - Use JavaScript to draw on that canvas
  - Drawing is similar to Java2D drawing
    - Similar primitives, transformations, coordinates, etc.
    - Rectangles, paths, arcs, text
    - Java Graphics2D maps to HTML5 Context
  - Can be used for static graphics and animations
  - [http://www.youtube.com/watch?v=xnAiEJEBLJg](http://www.youtube.com/watch?v=xnAiEJEBLJg)
  - [http://www.youtube.com/watch?v=oZInfZ0wecw](http://www.youtube.com/watch?v=oZInfZ0wecw)
  - [http://www.youtube.com/watch?v=UVcLTm-FEBQ](http://www.youtube.com/watch?v=UVcLTm-FEBQ)
SVG Graphics

- Different approaches to graphics
  - Procedural calls to draw everything
  - Structure representing what should be drawn

- SVG takes the second approach
  - The structure is part of the DOM
    - Can be manipulated by JavaScript
  - Objects correspond to various primitives
  - Often easier than functional drawing
    - Refresh handled automatically

- http://www.youtube.com/watch?v=6SDKN-Amlyo
HTML5 Multimedia

• `<audio>` and `<video>` tags
  • Controls
  • Multiple formats can (and have to) be provided

• Examples
  
  `<video width="320" height="240" controls="controls">`
  
  `<source src="movie.mp4" type="video/mp4" />`
  `<source src="movie.ogg" type="video/ogg" />`
  
  Your browser does not support the video tag.

  `</video>`

  `<audio controls="controls">`
  
  `<source src="song.ogg" type="audio/ogg" />`
  `<source src="song.mp3" type="audio/mpeg" />`
  
  Your browser does not support the audio element.

  `</audio>`
HTML5 Drag and Drop

• Interactive interfaces are based on drag and drop
  • That’s what users have come to expect
• HTML5 lets any element be dragged
  • And any element can be a drop target
• HTML5 also provides JavaScript events to support
  • On drag start (set the content of the drag)
  • On drag over (allow/disallow drop)
  • On drop (use the contents)
• Much simpler to use than Java drag and drop
Drag and Drop Example

```html
<!DOCTYPE HTML>
<html>  
<head>
    <script type="text/javascript">

    function allowDrop(ev) { ev.preventDefault(); }
    
    function drag(ev) { ev.dataTransfer.setData("Text",ev.target.id); }

    function drop(ev) {
        var data=ev.dataTransfer.getData("Text");
        ev.target.appendChild(document.getElementById(data));
        ev.preventDefault();
    }

</script> 
</head>  
<body>  

<svg id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></svg>

<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69"/>

</body>  
</html>
```
HTML5 Web Storage

• Cookies are not efficient or secure
  • Have to be sent with each HTTP request
• HTML5 offers several new facilities
  • Local storage (name-value) of arbitrary data
    • Permanent, fixed length, or session-based
  • Generation of public-private keys
    • Offers secure communication
HTML5 Geolocation

• HTML5 enables using the current location
  • Accurate from a device with GPS
  • Approximate from other computers
• Can use this using JavaScript
  • Locally (place on a map)
  • Globally (send to server)
• Can also get automatic updates
  • JavaScript code that is invoked as the position changes
Geolocation Example

```javascript
var x=$('#demo');
function getLocation()
{
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
    } else { x.html("Geolocation is not supported.");}
}
function showPosition(position)
{
    x.html("Latitude: "+ position.coords.latitude +
"<br />Longitude: "+ position.coords.longitude);
}
```
HTML5 Messaging

• Mashups
  • Web pages composed of information from multiple sources
  • Browsers limit where requests can be sent based on URLs
    • Make mash-ups difficult to implement
  • Plug-in JavaScript for various applications
    • Maps: Google maps, leaflet.js
    • Payments: Stripe, Paypal
    • Other: addresses, weather, ...

• Implemented using HTML5 messaging
• Web Sockets
Animation on Web Pages

• Is animation a good idea in a web application?
• Something moving (changing) on the screen

• Properties
  • Can be one-time or continuous
  • Can be smooth or jerky
    • All animation is jerky, why does it appear smooth
    • Persistence of vision, frames per second

• Types of animation
  • Movies
  • Sound
  • Bitmap animation
  • Vector animation
Question

What is a good rule of thumb for the minimum number of frames per second for a video or animation to look smooth to the human eye?

A. 10  
B. 30  
C. 80  
D. 240  
E. 1080
Bitmap Animation

• Animation as a sequence of bitmaps

• Techniques
  • Animated GIFs
    • All images stored in a single GIF file
    • Browser takes care of the animation
  • JavaScript
    • Change the image associated with a region using timers
    • Let the browser then redraw the image

• Pros/cons
  • Simple to use, built into browser, tool support
  • Limited in what it can do
Vector Animation

• Consider a drawing program
  • You place objects on the screen
    • Circles, rectangles, lines, text
    • Each of these is parameterized
      • Position, size, angle, string, color, line style, fill style, ...
  • The result is an image
• Can create a sequence of images drawn this way
  • Next image is going to be similar to the previous
    • What is going to change
  • Change expressed in terms of changes to parameters
    • This makes it easy to define such a sequence
• Show the images at the appropriate speed
• Can be done using canvas/SVG
Data Visualization

• Canvas/SVG
  • http://www.youtube.com/watch?v=B2LNOU2e1mM
• D3
  • http://vimeo.com/29862153
  • http://www.youtube.com/watch?v=OoOC2FYNo1M
Next Class

- JavaScript LAB
  - Be sure to have done the prelab
Videos

• Full bitmap images with automatic sequencing
• Typically **encoded** to save space
  • Data doesn’t change much from one image to another
  • Code key frames completely, otherwise just differences
  • Decoding should be fast and cheap
Sound

• Sound is nearly continuous
  • Time between values depends on frequency
    • To get a frequency response of 20K, need 40K sample/second
    • This means every 25 microseconds
    • But this is only one value, not a whole image
    • 8, 16, or 32 bits of data

• Again a variety of encodings are possible and used

• Synchronizing sound and image can be tricky
  • Handled by movie players
  • Handled by multimedia languages

• Sound in web applications can be annoying

• HTML5 <audio> tag
Making This Easier

• Key Frames
  • Suppose one takes 2 images K-frames apart
    • Parameters/positions change from one to the next
  • Have the computer construct the intermediate frames
    • Parameter values need to change from one value to the other
    • Interpolate values based on the end points
  • Types of interpolation
    • Linear: simplest to do
    • Path-based: user specifies a path (virtual or real)

• Example
  • https://www.youtube.com/watch?v=jJlAcTc5HUw
Making This Easier

- **Layers**
  - Split the drawing into different layers
  - Layers can be animated independently
    - Bottom layer(s) might be fixed (scenery/background)
    - Top layer(s) might be animated (person)
  - Can reuse the top layer or change its properties easily
Languages for Vector Animations

• Flash is the most widely known
• Microsoft Silverlight, Adobe AIR, JavaFX are alternatives
• These are implemented as browser plugins
  • Pretty much trusted
  • Provide a sandbox for executing programs
  • Include a scripting language for writing animations
  • Generally provide lots of other multimedia features
  • But not always available
• Java via Applets
• JavaScript implementations of these
• JavaScript using SVG
JQuery DOM Traversal

Strap on your climbing gear! DOM traversal is all about moving up, down, and sideways across the DOM.

To climb up the DOM, we can use the jQuery `parent` method.

To climb down the DOM, we can use the jQuery `children` method.

To climb across the DOM, we can use the jQuery `prev` and `next` methods.