Putting Them Together: Asset Presentation

- The task of effectively presenting multimedia assets over the Web involves all three component technologies: XHTML, CSS, and JavaScript
- We will cover the following:
  - Image presentation as both a gallery and slideshow
  - Invoking plug-ins for other media files (audio, video)
- Note that what is covered here are just general guidelines; in the end, there are infinite variations for doing the same thing with Web technologies

Embedding Media Files

- Interestingly, presenting audio or video media assets is somewhat simpler than presenting individual images: we let *Web browser plug-ins* do the work
- As mentioned, a Web browser plug-in is an additional piece of software that "takes over" for the browser for certain types of files
- The main trick is to use the `<object>` or `<embed>` tag to tell the browser where to get the media file; this file is then passed to the appropriate plug-in

Plug-In Gotchas

While, in principle, "that's all there is" with media files, there are a few things to remember:

- Remember that plug-ins are *add-on software* — there is a chance that a user will not have these installed in their Web browser
  - Fortunately, many plug-ins, such as Flash, and at times for QuickTime, frequently come pre-installed
  - In any case, special attributes can be tacked onto the XHTML tags to help the user find and download a plug-in that they don't have

- Officially, the `<object>` tag is the XHTML 1.1 way for invoking plug-ins; unfortunately, this is not well-supported by many browsers yet — they recognize the older `<embed>` tag only
  - Thus, here, unfortunately, we must compromise: for a working plug-in, we must use both tags, and are thus forced to create a page that won't validate as XHTML; in time, we hope, this issue will fade away
  - An alternative to this exists for the QuickTime Plug-In, which handles many movie and audio formats: the plug-in's vendor provides a JavaScript file that allows a page to be valid XHTML 1.1; the "bad stuff" is inserted dynamically (see http://developer.apple.com/internet/ieembedprep.html for details)

Presenting Images

- Because of the "native" `<img>` tag in XHTML, we don't need plug-ins to display images; on the other hand, there are virtually infinite possibilities to present images to the user
- Here, we discuss two common mechanisms, which frequently co-exist: an image *gallery* and a *slideshow*
- A gallery is conceptually simple but may take some time to do: simply create XHTML that has an `<img>` tag for each image you want to show (don't forget that the `alt` property is required in XHTML 1.1)

- Common details regarding galleries:
  - Typically, the gallery itself only displays *thumbnails*, or smaller versions, of the images — for speed purposes, it helps to pre-create these thumbnail files
  - An anchor (`<a>`) tag then surrounds the thumbnails, so that when the user clicks on the thumbnail, the full size image appears
- The alternative mechanism, a *slideshow*, presents the images one at a time, allowing the user to move forward and backward at his or her own pace
  - The simplistic way is to write a pile of XHTML pages, each looking similar, with just the central image changing and with links to go from one page to another — however, this creates a *lot* of extra files that are very similar to each other except for a few minor details (image file, previous/next links)…this is a signal that there must be a better way
  - An easier-to-manage way, but requiring a little more technical knowledge, is to add JavaScript into the mix — first, create a single XHTML page, giving the central image its own *id* property (so that it can easily be seen in JavaScript); then, write JavaScript functions that replace the image displayed by that tag with either the previous or next image in your slideshow sequence

More to Do

- Remember that the guidelines here are very cursory; there are *lots* of possible variations on these themes
- Things to look up on the Web, if you're interested:
  - Plug-ins generally come with their own controls; however, sometimes you want to customize these controls to your liking — depending on the plug-in, you may be able to do this; look up the plug-in documentation on the Web to find out how
  - JavaScript has the ability to perform functions *based on a timer* instead of a user action — this facilitates asset presentation functions such as autoplay and animation; again, lookups on the Web will yield instructions in case you want to learn more
  - JavaScript can not only modify the properties of XHTML tags, but *can modify the XHTML itself* via the *document.write()* method and the *innerHTML* properties — when used well, this can also yield additional asset presentation techniques