Time on Your Side

• As another exploration of what you can do with XHTML and JavaScript, let’s walk through the concept of a self-playing slideshow of images

• If you think about it, the idea is actually fairly straightforward: we need to call the same next() function used in the manual slideshow at a regular timed interval, without user intervention

• As it turns out, JavaScript provides just such a function: setInterval(), which tells JavaScript to perform some work repeatedly at a preset interval of real time

• Turns out to be fairly simple: setInterval() takes two parameters — the script to run, and the amount of time to wait between runs (in milliseconds)
  ◦ For example, setInterval("next();", 2000) will call the next() function every 2 seconds
  ◦ setInterval() gives back a value — an ID, really — that we can use with its partner, clearInterval(): invoking clearInterval(ID) will stop the repeated execution

• The setInterval/clearInterval pair turns out to be quite useful for adding one more feature to an autoplay slideshow — the ability to pause

• So, the overall approach is: use setInterval() to “play” the images automatically, then use clearInterval() to pause

• Typically, we give the user some HTML block which, when clicked, toggles between playback and pausing
Communicating with a Plug-In

- One final wrinkle (and what a wrinkle it turns out to be) regarding an autoplay slideshow: can we set things up so that background music plays/pauses along with the slideshow?
- It turns out to be possible, but not completely straightforward; the current discrepancy between `<object>` and `<embed>` support gets us again here.
- The problem is that `<object>` tags have an `id` just like any other tag, but `<embed>` tags can’t — instead, they have a `name`.

- The trick boils down to trying to “get to” the object in the DOM tree that represents the plug-in; in the sample files, this entails trying to locate the object by ID first (`getElementById`), then, if that doesn’t work, trying to locate it by name (`getElementsByName`).
- Then, once we have the object, we can call functions on that object just like any other JavaScript object; in the case of the QuickTime Plug-In, we have `Play()` and `Stop()` functions (among many others) that we simply invoke once we get to the plug-in object.
- To play it safe, we should still present the plug-in’s native interface (e.g., the QuickTime controller in the case of that plug-in), just in case our fancy JavaScript approach doesn’t work in the user’s current browser.