

Advanced Features of Dojo Toolkit

Sang Shin
Java Technology Architect
Sun Microsystems, Inc.
sang.shin@sun.com
www.javapassion.com

Disclaimer & Acknowledgments

- Even though Sang Shin is a full-time employee of Sun Microsystems, the contents here are created as his own personal endeavor and thus does not necessarily reflect any official stance of Sun Microsystems on any particular technology
- Many slides are created from the contents posted in dojotoolkit.org website

Topics Covered

- Dojo Drag and Drop
- Dojo Animation
- Dojo Storage
- Dojo Performance Tuning

Dojo Drag and Drop

Dojo Toolkit's Drag and Drop

- Dojo's Drag-and-Drop facility is implemented as a set of extensible classes whose interaction is mediated by a manager object
 - > DragSource
 - > DragObject
 - > DropTarget
 - > DragManager
- The classes are implemented for each rendering environment.

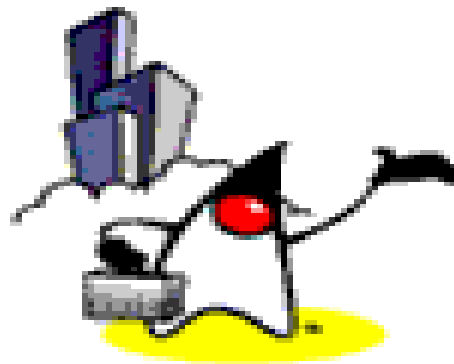
Example: Drag and Drop

// note that these examples are for the HTML environment

// make a list item draggable, but give it the catch-all type
new dojo.dnd.HtmlDragSource(dojo.byId("item1"), "*");

// make a element something that can be dropped into and let it accept
// DragSources of any type
new dojo.dnd.HtmlDropTarget(dojo.byId("list1"), ["*"]);

// we can also create "typed" DragSources, which DropTargets can choose to
// accept (or not) via the array of types given as the second parameter to
// the DropTarget constructor. Here we create incompatible DragSource and
// DropTarget objects
new dojo.dnd.HtmlDragSource(dojo.byId("item2"), "foo");
new dojo.dnd.HtmlDropTarget(dojo.byId("list2"), ["bar"]);



Demo: Dojo Drag and Drop

Demo Scenarios

- Go to demo page (or locally deployed demo page if there is no internet) and select **Effects**
 - > <http://localhost:8084/dojo-json-solution2/dojo-0.4.3-ajax/demos/demoEngine.html>
 - > Try **Drag And Drop**
- See the source
 - > Click **source** button on the page

Dojo Animation

Dojo Animation

- Provides a powerful, event-based animation library for developers

Change the property

- In order to change the duration property
 - > `anim.duration = 1000; // change duration to be 1 second`
- Do not change these values during an animation as the behavior for such an action isn't defined

Call `dojo.animation.Animation()` to Create an Animation object

- `var anim = new dojo.animation.Animation(curve, duration, accel, repeatCount);`
 - > `curve` - a special object detailed below used to give the animation points
 - > `duration` - the length of the animation in milliseconds?
 - > `accel` - defines whether or not an animation accelerates (1 = max accel), decelerates (-1 = max decel), or moves at a constant speed (0 = no accel).
 - > `repeatCount` - if set to a number greater than 0, the animation will repeat that many times. If it is set to -1, the animation will repeat forever (or until paused/stopped)

All Animations are Time-driven

- You specify the duration of an animation and it runs in that time frame
- The benefit of this system is that animations will look the same regardless of CPU speed

All Animations Run Off of Curves

- A curve is a special object whose only requirement is that it has a `getValue(n)` method, where `n` is a number between 0 and 1, that returns an array of numbers
- Dojo has some pre-built curves available in `dojo.math.curves`

Example: Animation on Curves

```
function Line(start, end) {
  this.start = start;
  this.end = end;
  this.dimensions = start.length;

  //simple function to find point on an n-dimensional, straight line
  this.getValue = function(n) {
    var retVal = new Array(this.dimensions);
    for(var i=0;i<this.dimensions;i++)
      retVal[i] = ((this.end[i] - this.start[i]) * n) + this.start[i];
    return retVal;
  }

  return this;
}
```

Event-based

- Dojo animation library allows developers to hook into all parts of an animation which enables them to be able to make advanced animations in a simple and intuitive manner

Event Handlers

- `onBegin` - fired when animation is played (from the beginning)
- `onAnimate` - fired every iteration of the animation
- `onEnd` - fired when the animation has completed
- `onPlay` - fired when the animation is played (from any point, including beginning)
- `onPause` - fired whenever the animation is paused
- `onStop` - fired whenever the animation is stopped (by calling `stop()`, not by finishing)
- `handler` - catch-all for events. You could potentially just define one handler to handle all types of events. You'll have to look at the `type` property of the event object to see what kind of event has been fired

Example: Slide a node across the screen

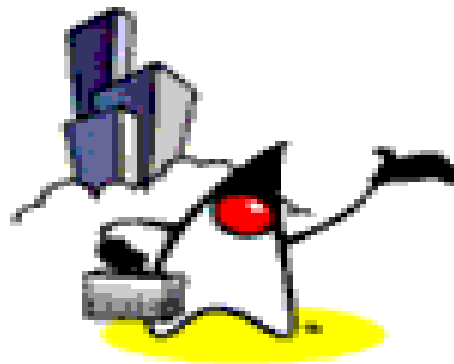
```
var anim = new dojo.animation.Animation(  
    // linear points from (0,0) to (400,200)  
    new dojo.math.curves.Line([0,0], [400,200]),  
    1. , // do it over a 3 second duration  
    2. // with no acceleration  
);  
  
dojo.event.connect(anim, "onAnimate", function(e) {  
    node.style.left = e.x + "px";  
    node.style.top = e.y + "px";  
});
```

Example: Slide a node across the screen

```
dojo.event.connect(anim, "onBegin", function(e) {  
    var div = document.createElement("div");  
    div.appendChild(document.createTextNode("Animation  
        started!"));  
    document.body.appendChild(div);  
});
```

```
dojo.event.connect(anim, "onEnd", function(e) {  
    var div = document.createElement("div");  
    div.appendChild(document.createTextNode  
        ("Animation done!"));  
    document.body.appendChild(div);  
});
```

```
anim.play();
```



Demo: Dojo Animation

Demo Scenarios

- Go to demo page (or locally deployed demo page if there is no internet) and select **Effects**
 - > <http://archive.dojotoolkit.org/nightly/demos/demoEngine.html>
 - > Try **Fade out, Fade in, Slide, Explode, Implode, Wipe out**
- See the source
 - > Click **source** button on the page

Dojo Storage

What is `dojo.storage`?

- `Dojo.storage` is a unified API to provide JavaScript applications with storage
- It is a generic front-end to be able to provide all JavaScript applications a consistent API for their storage needs
- The storage backends can use whatever mechanism is appropriate
 - > `dojo.storage` automatically detects its environment and available storage options and selects the most appropriate one.

Future Storage Providers

- Cookie Storage Provider - uses cookies to persist the hash table
- Flash Storage Provider - uses Flash's SharedObjects to persist data
- ActiveX Storage Provider - uses COM's File APIs to persist data
- XPCOM Storage Provider - uses XPCOM's File APIs to persist data

Future Storage Providers

- Form Storage Provider - uses the text autosave features of a hidden form to save transient data (the Really Simple History library uses this trick)
- WHAT WG Storage Provider - uses native browser persistence to store data, as defined by the WHAT Working Group.
- IE Storage Provider - uses IE's proprietary abilities to store up to 60K of data.

Dojo Performance Tuning

Areas of Performance Tuning

- Download
- Parsing
- Instantiating
- Deferred download

Download

- Web server setting
 - > set your web server so that javascript files etc. have "Cache-content" set so the browser doesn't try to reload them every time
 - > use web server compression (mod_deflate on apache)
- Build a custom dojo.js file
 - > to speedup download you should build a custom "edition" of dojo.js containing the modules your app needs

Parsing

- Dojo searches all the nodes to see if they are widgets
- Specify which nodes to search for dojoType widget tag
- Reduce number of tags on a page

Deferred Download

- Instead of downloading your whole page at once, defer portions until the user needs them.

```
<div dojoType="Tooltip" href="tooltip.jsp"></div>
```

```
<div dojoType="TabContainer">  
  <a dojoType="LinkPane" href="tab1.jsp">Tab #1</a>  
  <a dojoType="LinkPane" href="tab2.jsp">Tab #2</a>  
</div>
```

Questions

Sang Shin
Java Technology Architect
Sun Microsystems, Inc.
sang.shin@sun.com
www.javapassion.com