An Introduction To Ajax

Alex Russell
Project Lead, The Dojo Toolkit
alex@dojotoolkit.org

These Slides Are Online:

http://alex.dojotoolkit.org

Tarball Of Demo Code:

http://alex.dojotoolkit.org/wp-content/demos.tar.gz

Please Ask Questions

Odds are someone else is wondering the same thing

What We'll Cover

- What Ajax is
- What Ajax is *not*
- The fundamentals of Ajax
- How to build better experiences with Ajax
- Tools to make Ajax development easier

How We'll Get There

- Demos
- Deconstruction
- Code examples
- Introduction to debugging tools

What You'll Walk Away With

- Enough code to be dangerous
- Enough knowledge not to be
- Tools of the trade
- Places to look when you get stuck
- How it all fits together
- The lowdown on "what's next"

What Is Ajax?

- New name, old tech
- Technique for increasing UI responsiveness
- Incremental, in-place updates to a page
- Updates a result of user actions
- Preserves you reach
  - No plugins, standards based
- Browser as protocol participant

Why Ajax Now?

- The browsers stabalized
- The DOM is more-or-less implemented
- HTML and CSS are tapped out
  - At the limit of resonsiveness and usability
- REST and SOAP are handy endpoints
- What if the browser could be just another client?

When Is Ajax The Answer?

- When targeting majority browsers exclusively or can write two versions
- When you interface "feels" heavy
- When the competition is using Ajax
  - They will be soon!
- When it can make users' lives better but not worse

When Is Ajax Bad?

- When it breaks the freaking back button
- When it hinders accessibility
- When it disorients users
  - “jumping blocks”
- When it fails silently
- When it ignores indempotency

Remember Your Users

Ajax that doesn't build better experiences is *dumb*

Auto Save

Grab the Chicken!

- Surprisingly intuitive experiences
- Demo Uses:
  - Simple structural HTML elements
  - Basic event handling
  - Background REST requests
- Get out of the user's way!

Dissecting Auto Save

- XMLHTTP introduced
  - browser incompatibilities papered over
- Return XML traversed to get data
- Basic DOM manipulation
- ~150 lines of procedural JavaScript

```
var progIds = ['Msxml2.XMLHTTP',
               'Microsoft.XMLHTTP',
               'Msxml2.XMLHTTP.4.0'];
var http = null;
var last_e = null;

try{ http = new XMLHttpRequest(); }catch(e){}

// IE branch
/*@cc_on @*/
/*@if (@_jscript_version >= 5)
if(!http){
   for(var i=0; i<progIds.length; ++i){
      var objectType = progIds[i];
      try{
         http = new ActiveXObject(objectType);
      }catch(e){
         last_e = e;
      }
   }
}
@end @*/
```

Dissection contd.

- Basic DOM manipulation
- Getting “handles” to nodes
  - getElementsByTagName()
  - getElementById()
- Creating elements - example
- Deleting elements - example
- Moving elements around - example
- Changing the look of things - example

Scheduled Saves

- HTTP POST
  - We're modifying data
  - setInterval() for occasional execution
  - Events hard-wired to IDs
  - This code is brittle and hard to reuse

More on XMLHTTP

- Sync or Async communication
- Simple callback scheme for status
- Can return either text or XML doc
- Uses one of 2 available sockets
  - not x-domain
- Return order NOT guaranteed
- Interface NOT a standard (yet)

Ajax "quirks"

- Synchronous requests halt browser
- Picky about response Content-Type
- Some verbs problematic (HEAD, PUT)
- File upload not available cross-browser
  - handled w/ iframes and form hacks
- Bugs:
  - Caching (IE and FF 1.5 do, FF 1.0 doesn't)
  - IE: which MSXML did I really get?

What About Iframes?

- Used extensively by earlier toolkits
- Highly portable (works on 4.0 browsers)
- IE “Phantom Click”
- onload handler not reliable
- Only way to upload files reliably
- Async only

Design Decision: What To Send?

- XML
- plain/text
  - HTML
  - JavaScript (JSON)

“It Depends”

- HTML
  - Easy to insert into document (fast!)
  - Can return as just a string, easy to debug
  - Portability problems
  - Implies replacement, not update

contd.

- XML
  - Usually supported, MUST use “text/xml”
  - Doesn’t tie your server to your rendering
  - Need to transform on client
  - XSLT can be very fast
  - Very large on the wire

contd.

- Plain text
- JavaScript/JSON
  - Smallest for large data sets
  - Fastest parsing
  - Native language of your scripting environment
  - Constructing UI can be slower
  - App more likely to “speak” XML

Improving Auto Save

Experience Improvements

- Only auto-save after things change
  - Avoid “twitchyness”
- Smoother notification
- Handle unavailable network
  - Save to cookie as backup

Code Improvements

- Less browser quirk juggling
- Less brittle event/node linkage
- Animations for transitions
- A reusable auto-save component
- Easier to instantiate

Toolkits To The Resuce!

- Dojo:
  - Save data with dojo.io.bind()
  - Generic events with dojo.event.connect()
  - Animation and component support
- Prototype:
  - Save data with Ajax.Request
  - Portable events with Event.observe()

Other Quality Toolkits

- Open Source
  - MochiKit
  - YUI
  - Scriptaculous (builds on Prototype)
- Closed Source
  - TurboWidgets (based on Dojo)
  - Bindows
  - Backbase

Auto Save With Prototype

Debugging: A Digression

Basic Debugging Tools

- Your tenacity
- Google
- Standards references
- Mozilla, Safari, Opera JavaScript consoles
- IE Script Debugger
- Rhino or WSH for basic langauge problems

Advanced Debugging Tools

- Firebug
- LiveHTTPHeaders
- Ethereal
- Venkman
- Microsoft Script Editor
- Squarefree JS shell bookmarklet
- Virtualization
- VNC/RDP/X11
- Fiddler proxy
- Web Developer Toolbar
  - Firefox
  - IE

When In Doubt,

Watch The Traffic

85+% Of Your Users

Still Use IE

Auto Save With Dojo

Making It A Component

- Auto-Save for any <textarea>
- Why widgets?
  - Automatic property binding
  - Modularization
  - Easier to re-use

Auto Save Dojo Widget

JavaScript: A Digression

It Works Now

- Not a "toy" language
- Dynamic, functional (has closures)
- Classes and inheritance
- Ubiquitous: browser, stand alone, flash, etc.
- Debuggers available
- Can be modularized/packaged
- Libraries provide packaging, AOP, etc.

The Chameleon Of Languages

- Never what you expect
- Whatever you make of it
  - Imperative to imperative programmers
  - OO (with quirks) to OO programmers
  - Functional to Functional hackers

Language Features

- Lexical scope
- Functions are objects
- Functions are closures
- Anonymous function support
- Delegation via the prototype chain
- OOP via constructor closures and prototype inheritance
- Some interpreters support continuations
- C-style syntax

OOP in JS

- Prototype-based inheritance
- Not class-based!
- Prototype chain is method and property delegation

The Prototype Chain

```
function FooClass(ctorArg1, ctorArg2){
}

FooClass.prototype.foo = “foo”;
FooClass.prototype.bar = “bar”;

var baz = new FooClass();
baz.foo = “my own foo”;

var thud = new FooClass();
thud.foo == “foo”;
```

Mixins For Multiple Inheritance

```
// “interfaces that actually do something”
namespace.mixinClass = function(){
  this.foo = “foo”;
  this.bar = function(){
    alert(this.foo);
  }
};
function thingerClass(){
  namespace.mixinClass.call(this);
}
(new thingerClass()).foo == “foo”;
```

Rant: The Open Web

The Open Web Is Under Attack

The Web Won Because It Cost Less To Build

- Markup is the fastest way to build UIs
- Text, end-to-end
- Remixable
- Open Specs allow multiple renderers
- Royalty-free licensing of IP a key tennant

Who Wouldn't Love To Control The Web?

- Now that it's valuable, there's money to be had
- Closed, markup-based systems with single renderers threaten our ability to build freely

Ajax Is Helping To Keep The Web Open

- Still text all the way
- Open protocols: HTTP
- Standards-based tools, multiple clients
- Still not an experience that's competitive with closed solutions

Browser Vendors Do What You Demand

Nothing More.

Often Much Less.

Make Your Voice Heard

- Support minority browsers in your apps
- Use better browsers
- Vote for bugs in Open Source renderers
- File bugs for missing features in closed clients
- Blog
- Comment on vendor blogs

The Open Web Can Still Win

What's Next

- Low-latency data transfer: "Comet"
- Continued JavaScript ubiquity
  - Shipping with next JDK
- E4X
- Blurred desktop/web lines
- Cross-domain API usage from the browser

Where To Look For Answers

- JavaScript, The Definitive Guide
- W3C DOM Spec
- Ecma 262-3 Spec
- Quirksmode.org
- MSDN DHTML References
- Mozilla DOM Reference
- Ajaxpatterns.org
- Toolkit-specific lists and forums
- javascript.faqts.com
- Browser source code
- Toolkit source code

Q&A

Send private questions to:
alex@dojotoolkit.org

Thanks For Showing Up!