So You Wanna be a DOM Star

Aaron Gustafson
Easy Designs, LLC

Familiarity > Memorization

- Many resources on the web to get syntax
- Understand the concepts first… the syntax will follow
- Don't jump on the library bandwagon too soon

Keys to the kingdom

- JavaScript:
  - objects (strings, numbers, arrays, functions, etc.)
  - conditionals (`if`, `if...else`, `switch`, etc.)
  - loops (`for`, `while`, `do...while`, etc.)
- The DOM
  - node walking (`getElementById`, `getElementsByTagName`, `parentNode`, `childNodes`, etc.)
  - builders (`createElement`, `createElement`, etc.)
  - getters (`getAttribute`, `innerHTML`, etc.)
  - setters (`setAttribute`, `innerHTML`, etc.)

Read, read, read...

- Read the documentation *and* the source
- Read the blogs of luminaries in the field
- Read articles on ALA, DW & *Vitamin*
- Read the source of your favorite websites

Ask questions

- Most developers love to talk about their work, ask them about the decisions they made
- Ask for help
- Ask for a critique

Develop a methodology

- Consider “scripting” your script before you write a single line of code
- Write an outline in code comments to help keep you on track
- Break difficult tasks into simpler steps
- Write pseudo-code and then write the real thing

Develop a methodology


```
// create a storage array


// collect the links


// loop through each

  // collect the href


…
```

Develop a methodology


```
// create a storage array
var arr = [];

// collect the links
var links = document.getElementsByTagName( 'a' );

// loop through each
for( var i=0; i<links.length; i++){
  // collect the href
  arr.push( links[i].href );
}
…
```

Experiment

- Try new things and see if they work
- Discover how methods interact
- Build discrete test cases
- Find someone to challenge you

Fall on your face

- Don't be afraid of failing
- Learn from your mistakes

Take notes

- "Working" copy
- "Production" copy

Get to know your tools

- Firefox is a DOM scripter's best friend
  - JavaScript Console
  - DOM Inspector
  - Web Developer Toolbar
  - Firebug
  - Leak Monitor
- Consider using a tracing script to watch your script in action
  - jsTrace
  - jsTracer

Think beyond *your* application

- Keep your scripts markup agnostic whenever possible
- Look for ways to improve script flexibility

Think beyond *your* application


```
function collectLinks(){
  // create a storage array
  var arr = [];

  // collect the links in the content block
  var links = document.getElementById( 'content' ).getElementsByTagName( 'a' );

  // loop through each
  for( var i=0; i<links.length; i++){
    // collect the href
    arr.push( links[i].href );
  }

  // return the array
  return arr;
}
```

Think beyond *your* application


```
function collectLinks( source ){
  // create a storage array
  var arr = [];

  // collect the links in the content block
  var links = document.getElementById( source ).getElementsByTagName( 'a' );

  // loop through each
  for( var i=0; i<links.length; i++){
    // collect the href
    arr.push( links[i].href );
  }

  // return the array
  return arr;
}
```

Keep your scripts *unobtrusive*

- Drop the inline event handlers

```
<a href="#" onclick="newWin('path/to/somewhere');">link</a>
```

```
<a href="javascript:void(null);" onclick="newWin('path/to/somewhere');">link</a>
```

Keep your scripts *unobtrusive*

- Drop the inline event handlers

```
<a href="path/to/somewhere" onclick="return newWin(this.href);">link</a>
```

Keep your scripts *unobtrusive*

- Drop the inline event handlers

```
function newWinSetup(){
  // create a storage array
  var arr = [];

  // collect the links in the content block
  var links = document.getElementsByTagName( 'a' );

  // loop through each
  for( var i=0; i<links.length; i++){
    // collect the href
    links[i].onclick = function(){
                         return newWin( this.href );
                       };
  }
}
addEvent( window, 'load', newWinSetup );
```

Keep your scripts *unobtrusive*

- Drop the inline event handlers

```
function eventDelegator(){
  var el = Event.element( e );

  // external links
  if( el.nodeName.toLowerCase() === 'a' &&
      el.getAttribute( 'href' ) ){
      // Abort if a modifier key is pressed
      if( e.shiftKey || e.altKey || e.ctrlKey || e.metaKey ) return;
      var newWindow = window.open( el.getAttribute( 'href' ), '_blank' );
      if( newWindow ){
        if( newWindow.focus ) newWindow.focus();
        Event.stop( e );
      }
  }
}
document.getElementsByTagName( 'body' )[0].onclick = eventDelegator;
```

Keep your scripts *unobtrusive*

- Work the DOM
- `class`-ify and `id`-entify
- Pay attention to your markup

Keep your scripts *unobtrusive*


```
function collectLinks( source ){
  // test for support & existance
  if( !document.getElementById ||
      !document.getElementsByTagName ||
      !document.getElementById( source ) ) return;

  // create a storage array
  var arr = [];

  // collect the links in the content block
  var links = document.getElementById( source ).getElementsByTagName( 'a' );

  …

}
```

Follow the Golden Rule

> Do unto others as you would have them do unto you.

- Graceful Degradation & Progressive Enhancement
- What happens when JavaScript is turned off?
- What happens when methods you use aren't available?
- Consider the experience sans JavaScript *first*
- Constantly test your scripts
- Don't force errors on your users

Think in layers

- Build the “lo-fi” experience first
- Add in the bells & whistles after
- Test for support before you use a technique
    1. JavaScript
    2. cookies
    3. DOM methods
    4. Ajax

Learn “appropriateness”

- Does your page *really* need to autoscroll?
- Is your use of Ajax really a usability enhancement or simply showing off?
- Is it for you or them?

Get familiar with OOP

- Object Notation
- Object Literals/Singletons vs. classes
- Methods & properties

Get familiar with OOP

- Singleton

```
var SelectBuilder = {
  property1: 'value',
  property2: true,
  method1:   function(){
    return this.property2;
  },
  method2:   function( arg ){
    if( this.method1 ) return 'the argument was a ' + typeof( arg );
  }
};
alert( SelectBuilder.property1 );       // 'value'
alert( SelectBuilder.method2( 'hi' ) ); // 'the argument was a string'
```

Get familiar with OOP

- Object

```
function SelectBuilder(){
  var _property1 = 'value';
  var _property2 = true;

  // exposed method
  this.publicMethod = _method2;

  function _method1(){
    return _property2;
  }
  function _method2( arg ){
    if( _method1 ) return 'the argument was a ' + typeof( arg );
  }
};
var builder = new SelectBuilder();
alert( builder._property1 );           // undefined because it is not exposed
alert( builder.publicMethod( 'hi' ) ); // 'the argument was a string'
```

Iterate your scripts

- Refactor
- Test alternate routes to the same result
- Check times
- Test for memory leaks

Release your work into the wild

- Solicit feedback
- Encourage participation
- Create an API

Share your knowledge with others