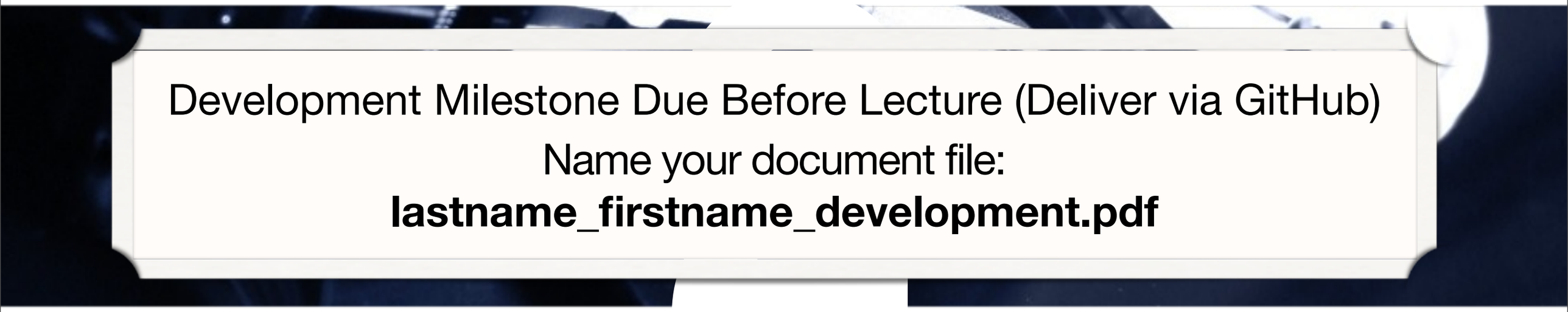
FULL SAIL
UNIVERSITY

scripting for web applications



Development Milestone Due Before Lecture (Deliver via GitHub)

Name your document file:

lastname_firstname_development.pdf

jQuery plugin development

PWA-2 DUE Dates

Item	Due Dates
Branding / Logo	~~08/05/13 - After Lab on the First Day~~
Project Pitch	~~08/09/13 - Before Lecture on 3~~
Creative Brief - Finished Document	~~08/12/13 - Before Lecture 4~~
Site Prototype *(html/css)*	~~08/16/13 - After Last Lab of the 2nd Week~~
Development Milestone *(javascript)*	08/21/13 - Due Before Lecture 8
Inclusion of 5 media center items	08/28/13 - Last Day of Class After Lab
Aesthetics & Usability *(finished site)*	08/28/13 - Last Day of Class After Lab
Functionality *(finished site)*	08/28/13 - Last Day of Class After Lab
Professionalism	The duration of the course
Class Participation	The duration of the course

things to keep in mind

- If you need a method and can't remember how it works, you have tools:
  - ***The cheat-sheet***
  - *Official documentation:* ***api.jquery.com***
    dochub.io

- We haven't looked at all of the *utility methods,* because we haven't needed most of them yet. Remember that they exist, and read up on them when you can.
- Today we are going to start some plugin development, and we'll see some utility methods as a result.

Some jQuery Plugins Resources

Website	*Description*
harvesthq.github.com/chosen/	Dropdown selects with autocompleter
christophercliff.github.com/sausage/	Page anchoring generator
craigsworks.com/projects/qtip2/	Advanced tooltips
danpalmer.me/jquery-complexify	Password strength meter
fancybox.net/	Easy modal lightboxes
www.zurb.com/playground/reveal-modal-plugin	Simple modal
projects.nickstakenburg.com/lightview	Another modal plugin
needim.github.com/noty/	Notification popups
isotope.metafizzy.co/	Grid re-arranger
podio.github.com/jquery-mentions-input/	Twitter Style Autocompleter
twitter.github.com/bootstrap/	HTML/CSS/JS Boilerplate by Twitter
nicolahibbert.com/demo/liteAccordion/	Horizontal accordion
github.com/mathiasbynens/jquery-placeholder	Input Placeholder Plugin
jamesflorentino.com/jquery.nanoscroller/	Custom Scrollbar
demo.mobiscroll.com/	Date and time picker (phone styled)
jquery.malsup.com/cycle/	Slideshow Plugin

lecture activity

Accordion Plugin

download the following file from the FSO “References”
“Course Material Follow Along Files”

Goal8 JACC.zip

5

Plugin Development

- Start building your own collection of re-usable code. Whenever you tackle a feature that could have use in other applications, try to build it more dynamically.
- Benefits of plugins:
    - Smaller application code
    - Abstraction: less clutter of variables in scopes
    - Re-usability
    - Dynamic: forces you to tackle a solution dynamically

Naming Practices

- Best practices for plugin filenames:
    - *all lowercase*
        - *framework - first*
        - *plugin name - second*
        - *versioning - third*

jquery.jacc.1.6.js

jquery.easing.min.1.3.js

Compatibility

- Next, we should make sure our plugin will work even if the page has multiple javascript libraries included.
    - *Keep in mind that Prototype and Mootools also use $ dollar sign as functions.*

- (Excluding plugin development for a second...) When working on a site that is using Prototype or Mootools.. good old jQuery gives a compatibility function:

$.noConflict( )	jQuery backs up a save of the $ name before installing itself. By calling this method, jQuery reverts the $ global to whatever it was before.

- So, make sure jQuery is included after any other libraries, and then call this method.

Compatibility

- noConflict is not a magic wand, we can no longer use $, we'd have to use the jQuery global instead.

```
jQuery.ajax()
```

instead of

```
$.ajax()
```

- This is the same problem with plugins, we can't assume that $ means jQuery.
- So, we need to create a privatized version of $. And how can we create private scopes? ***A closure.***

9

Compatibility

- Here's a reminder of the self-executing function closure:

```
( function(){} )();
```

- And here's how we can use it to privatize the $ name

```
(function($){
   // privatized $ scope
})(jQuery);
```


10

Plugin Type

- Next we need to consider what type of plugin we want to build.

Utility Methods

```
$.plugin( options )
```

- These are called directly from the jQuery object, and usually do not involve any targeting of DOM elements.

Factory Methods

```
$(target).plugin( options )
```

- These allow us to use the factory to make a DOM selection, and then manipulate them.

Utility Plugins

- These methods are assigned directly to the jQuery namespace

Template

```
(function($){
   $.myplugin = function(options){
      // plugin code
   };
})(jQuery);
```


12

Factory Plugins

- In order to gain access to the factory, these are created on a **fn** object of the jQuery namespace.
- Inside our plugin function, the context ***this*** becomes the jQuery factory of DOM elements.

Template

```
(function($){
    $.fn.myplugin = function(options){
        // plugin code
        console.log(this);
    };
})(jQuery);
```

```
$(“div”).myplugin();
```


13

.each( )

- This utility method allows us to loop through the elements in a factory set, using a function.

$(target).each(*fn*)	*function:* is run once per element in the set (like a loop)

- The provided function will receive 2 arguments, and it will also gain ***context*** to each element in the set

```
$(“.myclass”).each(function(i, elem){
   console.log(this);
});
```


14

Factory Plugins

- Since our context ***this*** is the factory set, we can loop through those elements using jQuery's .each method.
- Finally, to make our plugin *chainable*, it needs to return jQuery, so let's combine...

Template

```
(function($){
   $.fn.plugin = function(options){
      return this.each(function(){
         // plugin code
      });
   };
})(jQuery);
```


15

Plugin Options

- The last thing our plugin need is to define a set of default options.
- For example, let's say our plugin has 5 options, but the user only passes in 2

```
$("div").plugin({
    opt1: true,
    opt4: true
});
```

opt1: *boolean*
opt2: *boolean*
opt3: *boolean*
opt4: *boolean*
opt5: *boolean*

- We need a way to set up default values for opt2, opt3, & opt5

16

Plugin Options

$.extend({}, *injector, original*)	*injector:* object of new properties *original:* object to inject into

- Extend allows us to provide an object and add properties into it. Any properties that the original object already has are not added. A new object is then returned.

```
var obj = {opt1: true, opt2: true};

obj = $.extend({
    opt1: false,
    opt2: false,
    opt3: false,
    opt4: false
}, obj);
```

=

```
{
  opt1: true,
  opt2: true,
  opt3: false,
  opt4: false
}
```


17

Factory Plugin Template

- Here's our general starting template for any factory plugin

```
(function($){
   $.fn.myplugin = function(options){
      options = $.extend({
         // default properties here
      }, options);
      return this.each(function(){
         // plugin code
      });
   };
})(jQuery);
```


18

Lab 8

- **Next Milestone:** FINAL Deliverable
  - ***lastname_firstname_FINAL.zip***

19