I am the Director of Products at appendTo & they sponsored my time to be at the conference. appendTo rocks!
AUDIENCE

This presentation was crafted specifically for delivery at the Front End Design Conference in St. Petersburg, FL. The audience consisted of designers who use jQuery very little through developers who use jQuery regularly.

The slide portion of this presentation covers the three primary topics I covered during live coding.

If you have any questions after watching this presentation and looking at the code, please let me know: doug@dougneiner.com (Be sure to reference this presentation when emailing!)
AUDIENCE POLL

How much do you jQuery?

Results: A few have never used jQuery, a number used jQuery sparingly, many others use it a lot, and many use it every day.
“an untidy collection or pile of things”

– New Oxford American Dictionary
This presentation

Slides

Coding

On Github

QUESTIONS
What is jQuery good for now?
• DOM Manipulation (Find something, do something)
• Event Delegation
• AJAX
• Deferreds/Promises
• CSS/Animation
#1

JQUERY & CSS

In a Jumble
jQuery: What & How

- `show()` or `hide()`
  - What: I want my element to be hidden or visible
  - How: Show or hide by setting adjusting the `display`. It will appear or disappear immediately when this setting is changed.

- `slideUp()` or `slideDown()`
  - What: I want my element to be hidden or visible via a nice animation
  - How: Show or hide by animating the height property.
  - How: I want the animation to last exactly 400ms.
• show() or hide()

  • What: I want my element to be hidden or visible
  
  • How: Show or hide by setting adjusting the `display`. It will appear or disappear immediately when this setting is changed.

• slideUp() or slideDown();

  • What: I want my element to be hidden or visible via a nice animation
  
  • How: Show or hide by animating the height property.

• How: I want the animation to last exactly 400ms.
jQuery: What

```javascript
$el.addClass( "is-visible" );
$el.removeClass( "is-visible" );
```

CSS: How

```css
.my-product { display: none; }
.my-product.is-visible { display: block; }
```

This is almost equivalent to jQuery show/hide

jQuery uses classes to say "what" is happening…

…CSS controls "how" the element is shown/hidden
* left used for simplicity, transform would be a better choice. Also, use vendor prefixes as needed.
WHY?

• CSS classes can easily affect decedent elements

• Adding an `is-visible` class describes the state, but allows the CSS to manage exactly how it becomes and remains visible.

  • Visible no longer has to mean “display: block”

  • Invisible can now mean low opacity instead of completely hidden

• CSS animation via transitions can be used as part of the showing or hiding
jQuery controls the "WHAT"
CSS controls the "HOW"

jQuery applies the classes, CSS controls the rest
• Animation was how I started using jQuery

• CSS3 Transitions and animations have come a long way

• Class-based CSS3 Transitions have inherent fallbacks for older browsers
Event Delegation
Event delegation

- Event bubbling allows it to work
  (This is patched by jQuery as necessary)
- One event on a parent vs. many events on children
- Filtering vs. Traversal
- Responsible events – only run on user action
- Can be added before the document is ready
- Forces a contextual approach
EVENT DELEGATION

CONTAINER

CONTAINER

click

click

click

click

click

click

click

click

click

click

click
BOUND VS DELEGATED

- Separate handler for each element
- Executes only when event is triggered on bound elements
- Elements can be retrieved ahead of time
- Must be bound after the DOM is ready

- Single handler for all elements
- Checks event on every element in its context, and executes when a match is found
- Elements should be retrieved at run time
- Can be setup as soon as the context element is ready
SYNTAX

Traditional binding

```
$( "a" ).on( "click", function () { ... } );
```

Delegated Binding

```
$( document ).on( "click", "a", function () { ... } );
```

Older syntax for delegate...

```
$( document ).delegate( "a", "click", function () { ... } );
```
Problem with Live

Live is deprecated, but this slide explains why it wasn’t performant.

$( "a" ).live( "click", function () { ... } );

- Step 1: Search all the DOM for "a"
- Step 2: Throw away the result
- Step 3: Create a delegated event for "click" on an "a"
#3

JQUERY.DEFERRED
$.get( "/path/to", function ( data ) {
    console.log( "I got my data!!!", data );
});
$.get( "/path/to/1" , function ( data1 ) {
  $.get( "/path/to/2" , function ( data2 ) {
    $.get( "/path/to/3" , function ( data3 ) {
      console.log( "I got my data!!!" , data );
    });
  });
});
}
PROBLEM WITH CALLBACKS

• They queue up
  The first has to finish before the nested call can begin.

• Potential for Ugly, nested code

• You still need to handle caching
$.get( "/path/to" )
  .then(function ( data ) {
    console.log( "I got my data!!!", data );
  });
$.when(
  $.get( "/path/to/1" ),
  $.get( "/path/to/2" ),
  $.get( "/path/to/3" )
).then(function ( ret1, ret2, ret3 ) {
  console.log( "I got my data!!!", ret3[0] );
});

Each request is not depending on the previous request.

The callback only fires once all three are ready.
What is a Deferred?

- An object that starts in an unresolved state
- Can either be rejected or resolved with data
- Once resolved or rejected, it remains forever in that state
- It can do a LOT more as well. See http://api.jquery.com/jquery.deferred for more information.
• jQuery’s implementation is based on Promises/A Spec, but not fully compliant

• "Thenable" is the primary API for Deferreds

```javascript
$.get( "/path/to" )
  .then( passCallback, failCallback );
```
$.WHEN

Creates a new deferred that resolves a set of deferreds or values are ready. For non-deferreds, truthy values trigger a resolve, falsey trigger a reject.

```
$.when( deferredOne(), deferredTwo(), true )
.then( function ( res1, res2, res3 ) {

});
```
$.done and $.fail

Shortcut versions of one part of $.then( pass, fail );

// Done vs Then
dfd.done( function () { ... } );
dfd.then( function () { ... } );

// Fail vs Then
dfd.fail( function () { ... } );
dfd.then( null, function () { ... } );
Creating a Deferred

var dfd = $.Deferred();

dfd.then( function ( data ) {
  console.log( data ); // Logs out, It worked
});

// Some long running operation
dfd.resolve( "It worked" );
Resolving and rejecting

dfd.resolve( { successData: true } );

// OR

dfd.reject( { errorMessage: "Something is wrong!" } );
var createDfd = function () {
    var dfd = $.Deferred();
    ...
    return dfd.promise();
};

// Fails, no method named "resolve"
createDfd().resolve( "Something" );
var promise = $.Deferred( function ( d ) {
  d.resolve( "It worked!" );
}).promise();

• Create the deferred and return a promise in one statement.
• The callback will be invoked with the new Deferred as the first parameter.
If all you need are promises, there are other options than jQuery.

One option is a State Machine library Machina plus a plugin I wrote for it: Machina.Promise (Promises/A Compliant)
http://github.com/a2labs/machina.promise
LET'S CODE!

Finished code is available here:
https://github.com/dcneiner/frontend-shop
• Front End Shop (Example Project)
  http://github.com/dcneiner/frontend-shop

• Contextual jQuery Video (jQuery UK 2012)
  http://vimeo.com/40873228

• Machina Introduction Video (jQuery UK 2013)
  http://vimeo.com/67473899

• jQuery Build Tool
  http://projects.jga.me/jquery-builder
• Postman for Chrome
  http://www.getpostman.com/

• Detecting AJAX Events on the Server
  http://www.learningjquery.com/2010/03/detecting-ajax-events-on-the-server
Thank You!
@dougneiner