Advanced jQuery

Sergio Pereira

Sergio Who?

- http://sergiopereira.com/blog
- @sergiopereira
- Chicago ALT.NET
- Chicago Code Camp
- I'm JavaScript-infected

look us up!

The Plan

- not just Tips & Tricks but, yeah, some of those
- overlooked jQuery features
- coding the jQuery way
- new in jQuery 1.7

The Plan

Is Subject To Change

Quick Wins

end() Command

breadcrumbs for your navigation

end() Command

```
<div class=menu>
  <ul>
    <li class=current>New</li>
    <li>Open...</li>
    <li>Print...</li>
  </ul>
</div>
<div class=menu>
  ...
</div>
```

end() Command

```
$('.menu').find('li')
```

```
<div class=menu>
  <ul>
    <li class=current>New</li>
    <li>Open...</li>
    <li>Print...</li>
  </ul>
</div>
<div class=menu>
  ...
</div>
```

end() Command

```
$('.menu').find('li')
          .filter('.current')
```

```
<div class=menu>
  <ul>
    <li class=current>New</li>
    <li>Open...</li>
    <li>Print...</li>
  </ul>
</div>
<div class=menu>
  ...
</div>
```

end() Command

```
$('.menu').find('li')
          .filter('.current')
          .end()
```

```
<div class=menu>
  <ul>
    <li class=current>New</li>
    <li>Open...</li>
    <li>Print...</li>
  </ul>
</div>
<div class=menu>
  ...
</div>
```

end() Command

```
$('.menu').find('li')
          .filter('.current')
          .end().end()
```

```
<div class=menu>
  <ul>
    <li class=current>New</li>
    <li>Open...</li>
    <li>Print...</li>
  </ul>
</div>
<div class=menu>
  ...
</div>
```

Computed Values

for when a simple expression isn't enough

Computed Values

```
$('#buttons li')
  .css('margin-left', function(ix, val) {

      return ix*10;
});
```

Computed Values

- attr(name, function(i,v){ })
- prop(name, function(i,v){ })
- val(function(i,v){ })
- css(name, function(i,v){ })
- html(function(i,v){ })
- text(function(i,v){ })

Relative CSS Attributes

incremental

Relative CSS Attributes

```
$('#volumeUp').click(function(){
    var $volBar = $('#volBar');
    var width = $volBar.css('width');
    //easy mistake
    $volBar.css('width', width + 10);
});
```

Relative CSS Attributes

```
$('#volumeUp').click(function(){
    $('#vol').css('margin-left', '+=10');
});
```

data() Storage

goodbye memory leaks

data() Storage

```
var myForm = $('#myform')[0];
var formResult = {
   errors: validateForm(myForm),
   form: myForm
};
myForm.result = formResult;

//later...
var result = $('#myform')[0].result;
if(result.errors.length){
   alert('Fix the errors.');
}
```

data() Storage

```
var formResult = {
   errors: validateForm(myForm),
   form: $('#myform')
};
$('#myform').data('result', formResult);

//later...
var result = $('#myform').data('result');
if(result.errors.length){
   alert('Fix the errors.');
}
```

data() Storage

```
<ul>
  <li data-movie-id="131">Crash</li>
  <li data-movie-id="722">Up</li>
</ul>
```

```
var id = $('li:first').data('movieId');
// ==> id == 131
```

data() Storage

data attributes

- will try to use correct data type
- parses JSON values
- but is it valid/standard?
  - yes, in HTML 5
  - but does it really matter?

map() Utility

projections for your arrays

map() Utility

array-like: has most of what's needed to be used as an array

```
var arrayLike = {
   length: 2,
   0: 'first',
   1: 'second'
};
```

map() Utility

bonus utility: makeArray()

```
var arr = $.makeArray(arrayLike);
// ==> ['first','second']
```

map() Utility

```
var users = getUsersArray();

var emails =
   $.map(users, function(user, i){
      return user.profile.email;
   });
```

map() Utility

```
var users = getUsersArray();

var emails =
   $.map(users, function(user, i){
      return user.profile.email;
   });
```

map() Utility

the callback function

- return new value for the position
- return `null` to exclude that position
- return array of values (flattened)
- `this` is the global object (`window`)

Quick Wins

recap

- end(): avoid unnecessary traversals
- computed attributes
- relative CSS attributes
- data() without leaks
- map(): conversions

Event Handling

oh, the choices

jQuery Makes It Easy

determine which element and...

bind(), live(), delegate(), on()

well, yes - choices

bind() vs. live()

```
$('#container .action')
   .bind('click', function(){
     //do stuff here
   });
//same as: $('#container .action')
//         .click(function(){...});

//prefer:
$('#container .action')
   .live('click', function(){
     //do stuff here
   });
```

bind() vs. live()

- live(): more efficient usage
- live(): works for added elements
- live(): forces you to deal with the problem generically
- bind(): for unique elements

live() vs. delegate()

```
$('#container .action')
   .live('click', doStuff);

$('#container')
   .delegate('.action','click', doStuff);
```

live() vs. delegate()

- live(): syntax is slightly easier
- live(): attaches to document
- delegate(): attaches closer
- live(): one DOM search inevitable
- live(): doesn't work in chains

Everything vs. on()

```
$('#close').bind('click', closeWindow);
$('#close').on('click', closeWindow);
```

```
$('.popup').delegate(
        '.closeBtn','click', closePopup);

$('.popup').on(
        'click','.closeBtn', closePopup);
```

Custom Events

they remove tight coupling

Custom Events

```
$('#widget').on('start', function(){
   //do stuff (react to widget's start)
});

//elsewhere, maybe in another js file:
$('#widget').on('start', function(){
   //do other stuff
});

//when appropriate:
$('#widget').trigger('start');
```

Custom Events

```
$('.editBtn').on('lock', function(){
   $(this).prop('disabled', true);
});

//elsewhere, maybe in another js file:
$('#toolbar').on('lock', function(){
   $(this).find('.icon').remove();
});

//when appropriate:
$.event.trigger('lock');
```

proxy() Utility

let's put this in context

proxy() Utility

```
var f = $.proxy(obj, 'doSomething');

var f = $.proxy(obj.doSomething, obj);
```

Event Handling

recap

- pick the right binding command
- delegate() over live()
- on() unifies it all
- decouple with custom events
- dodge context bugs with proxy()

Extensions the jQuery way

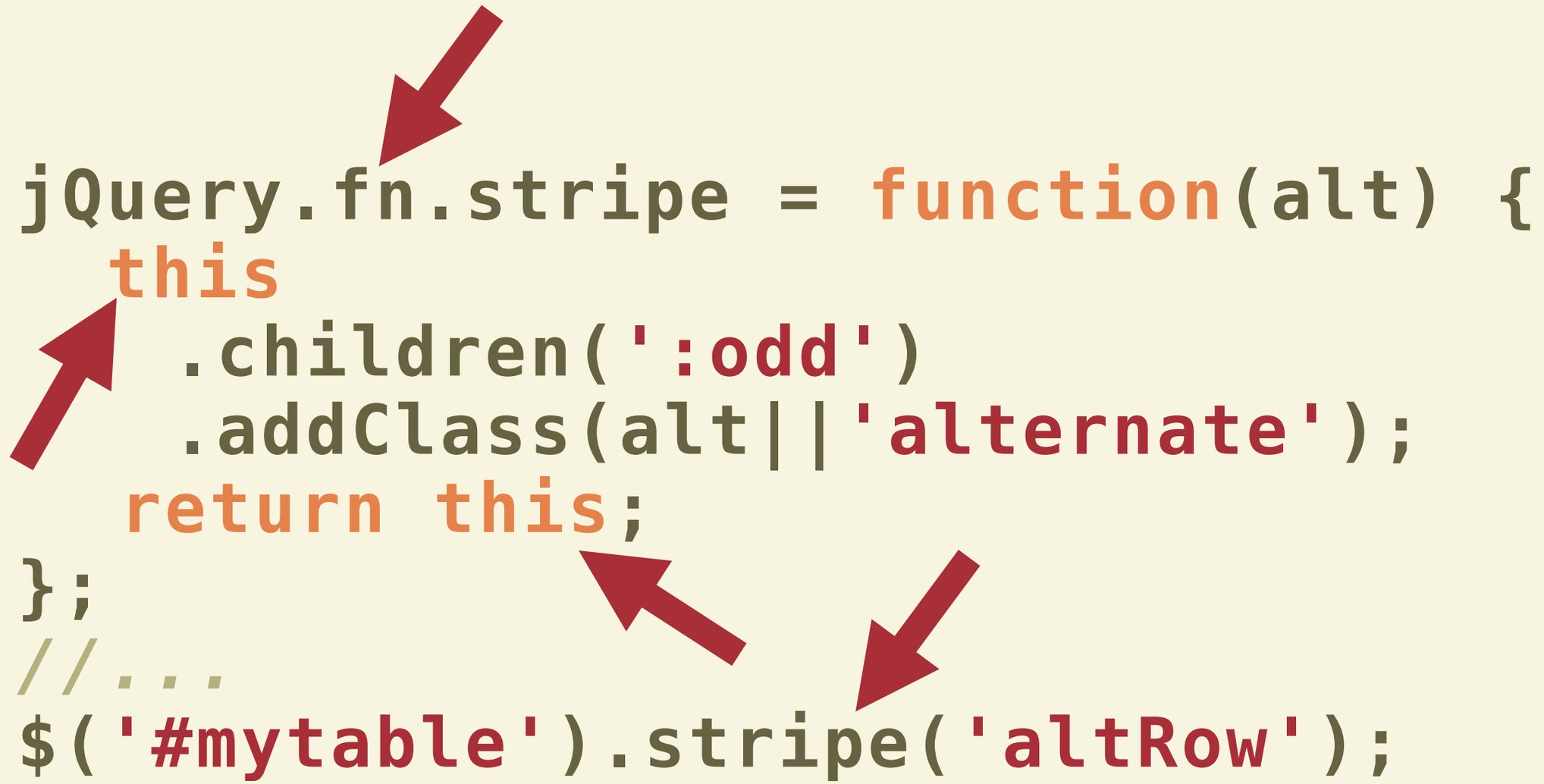
Custom Commands

the abc of jQuery mastery

Custom Commands

```
function stripeTable(table, alt) {
  $(table)
    .find('tr:odd')
      .addClass(alt||'alternate');
}
//...
var table = $('#mytable')[0];
stripeTable(table, 'altRow');
```

Custom Commands



```
jQuery.fn.stripe = function(alt) {
   this
     .children(':odd')
     .addClass(alt||'alternate');
   return this;
};
//...
$('#mytable').stripe('altRow');
```

Custom Expressions

expressive means readable

Custom Expressions

```
$.expr[':'].myfilter = function(el) {
   //return true if el(ement) matches
}
//...
$('div:myfilter').slideDown();
```

Extensions

recap

- commands
- expressions
- events (shown earlier)

1.5

Deferred Object

Futures, Promises

decouple the task from what you do with its outcome

Tidy Callbacks

```
startLongProcess(
   function(result){
      useResult(result);
      logCompletion(result);
   },
   function(error){
      logCompletion(result);
      logFailure(error);
   }
);
```

Tidy Callbacks

```
startLongProcess(
   function(result){
      useResult(result, function(res2){
         updateScreen(res2);
      });
      logCompletion(result);
   },
   //...
);
```

Tidy Callbacks

```
var task = startLongProcess()
    .done(useResult)
    .always(logCompletion)
    .fail(logFailure);

doSomethingElse();

//time passes...
task.fail(function(){
    alert('error');
});
```

Tidy Callbacks

```
var task = startLongProcess()
    .always(logCompletion)
    .fail(logFailure);

$.when(task, otherTask)
  .then(function(args1, args2){
      useResult(args1[0])
        .done(updateScreen);
  });
```

Deferreds in jQuery

time to relearn some features

Ajax

```
var fetch = $.get('/data.aspx')
      .done(validateResponse)
      .done(doSomething)
      .done(doMoreStuff)
      .fail(reportError);

fetch.fail(resetUI);
```

when() Utility

```
$.when(
    $.get('/somedata'),
    $.get('/moredata'),
    $.post('/updatedata', data),
    someOtherDeferred)

.then(function(){
    alert('all done');
});
```

promise() Command

```
$('.widget')
   .first().fadeOut(500).end()
   .last().fadeOut(1000).end()
   .promise('fx') //default queue
   .done(function(){
     //do something...
   });
```

Deferred Object

recap

- common way for callbacks
- returned by jQuery Ajax
- for command queues too
- framework for your own
- composable

The True Source

github

https://github.com/jquery

Build It, Run Tests

check out QUnit

Questions?

Reference

- http://api.jquery.com
- Paul Irish's articles and videos
- slides, code: http://bit.ly/advjquery

Thank You