

jQuery

<http://github.com/relevance/jquery-demos>

stuart halloway

<http://thinkrelevance.com>

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 License.
<http://creativecommons.org/licenses/by-nc-sa/3.0/>

1

what is jquery?

unobtrusive
standard library
open source
ajax

2

one big idea

3

wrapped sets

4

the \$ function

1. selectors

`$(selector)`

context defaults
to entire DOM

`$(context, selector)`

5

6

basic selectors

syntax	finds
<code>#foo</code>	id
<code>.bar</code>	class
<code>h2</code>	element
<code>li a</code>	a descendants of li
<code>li > a</code>	a direct children of li
<code>ol, ul</code>	ols and uls
<code>#main+p</code>	p directly preceded by sibling #main
<code>#main~p</code>	p directly preceded by sibling #main

7

attribute selectors

syntax	tests
<code>[href]</code>	presence of class
<code>[href=/foo]</code>	equality
<code>[href^=http:]</code>	starts with
<code>[href\$=.jpg]</code>	ends with
<code>[href*=relevance]</code>	contains
<code>[href!=/foo]</code>	not equality

8

dom position selectors

selectors	
<code>:first</code>	<code>:eq</code>
<code>:first-child</code>	<code>:gt</code>
<code>:last</code>	<code>:lt</code>
<code>:last-child</code>	<code>:even</code>
<code>:only-child</code>	<code>:odd</code>
<code>:nth-child(2), :nth-child(:even), :nth-child(odd), :nth-child(4n), :nth-child(3n+2)</code>	

9

state selectors

state
<code>:hidden</code>
<code>:visible</code>
<code>:enabled</code>
<code>:disabled</code>
<code>:selected</code>
<code>:checked</code>
<code>:animated</code>
<code>:contains(text)</code>
<code>:has(selector)</code>
<code>:parent</code>

10

element types

shortcuts	
<code>:button</code>	<code>:input</code>
<code>:checkbox</code>	<code>:password</code>
<code>:file</code>	<code>:radio</code>
<code>:header</code>	<code>:reset</code>
<code>:hidden</code>	<code>:submit</code>
<code>:image</code>	<code>:text</code>

11

refining wrapped sets

method	notes
<code>add(expr)</code>	adds elements to set
<code>andSelf</code>	top two sets in stack
<code>contents</code>	element
<code>end</code>	previous set in stack
<code>filter(expr)</code>	keep matching elements
<code>map(f(idx,el))</code>	calls f for each element
<code>not(expr)</code>	remove matching elements
<code>slice(b,e)</code>	elements in range

12

family relations

selectors	
children	parents
next	prev
nextAll	prevAll
offsetParent	siblings
parent	
closest(ancestor)	
find(descendants)	

13

queries and enumeration

method
index
is
get
size
each

14

selectors demo html

```
<div id="section-1">
  <p id="p1" class="vivid leader">Paragraph 1</p>
  <p id="p2">Paragraph 2</p>
  <p id="p3" class="honorificabilitudinitatibus">
    Paragraph 3 has a <span>nested span</span></p>
  <p id="p4" class="vivid">Paragraph 4</p>
  <p id="p5">Paragraph 5 has a
    <span>doubly <span> nested span</span> </span></p>
  <div>This is not a paragraph</div>
</div>
```

15

selectors demo impl

```
$(function() {
  $("#highlight").submit(function() {
    $("*").css("color", null);
    $("#selector").val().css("color", "blue");
    return false;
  });
});
```

16

finding paragraph 3

```
#p3
p:eq(2)
p:even:eq(1)
p:contains(3)
p[class^=honor]
p[class$=bus]
p:has(span):not(#p5)
p:nth-child(3)
```

be careful with **not**

```
:not(p:text(Paragraph))
```

```
p:not(text(Paragraph))
```

17

18

document ready

```
$(document).ready(function () {
    // do when document is ready
})

$(function() {
    // same as above
})
```

2. dom manipulation

19

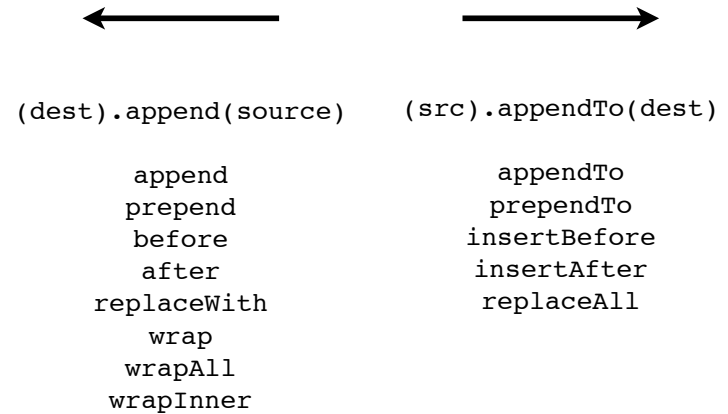
20

creating a node

```
$("#<a>").text("Clojure").attr({  
  href: "http://clojure.org"  
}).css({  
  border: "3px solid blue",  
  padding: "3px",  
  background: "#8888ff"  
}).prependTo("body");
```

21

changing nodes



22

content

method
<code>html()</code>
<code>html(markup)</code>
<code>text()</code>
<code>text(str)</code>
<code>remove</code>
<code>empty</code>
<code>clone</code>

23

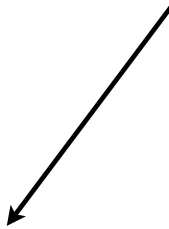
attributes

method
<code>attr()</code>
<code>attr(name)</code>
<code>attr(map)</code>
<code>css(map)</code>
<code>removeAttr(name)</code>

24

attribute normalization

```
$("#h2").attr("float", "right")
```



`<h2 cssFloat="right">...</h2>`

25

boolean normalization

```
$("#h2").attr("disabled", true)
```

```
<input value="Frak!" type="submit" disabled="*">
```

```
$("#h2").attr("disabled", false)
```

```
<input value="Frak!" type="submit">
```

26

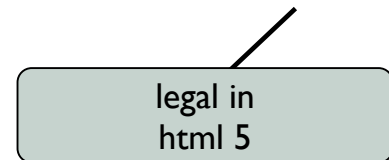
sizing things up

method	notes
<code>width</code>	wrappers for css
<code>height</code>	
<code>innerHeight</code>	height - border + padding
<code>innerWidth</code>	
<code>outerHeight(padding?)</code>	height + border ?padding
<code>outerWidth(padding?)</code>	
<code>offset</code>	relative to doc
<code>position</code>	rel to offset parent

27

nonstandard attributes

```
$("#h2").attr("data-level", "sidebar")
```



legal in
html 5

```
<h2 data-level="sidebar">Re: ...</h2>
```

28

custom data

method
<code>data()</code>
<code>data(name, value)</code>
<code>removeData(name)</code>

custom data

method
<code>addClass(sds)</code>
<code>removeClass(sds)</code>
<code>toggleClass(name)</code>
<code>hasClass(name)</code>

space-delimited string!

29

30

don't worry about

normalizing attribute names

parsing css class string

global variables

calculating width and height

3. events

31

32

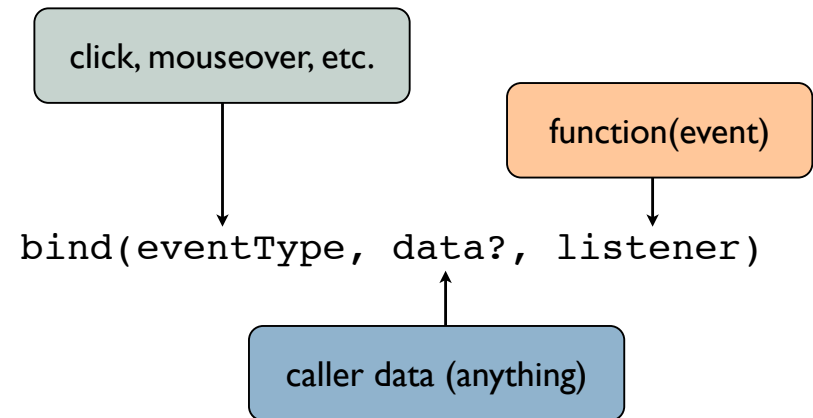
event example

```
<div id="field"></div>
X: <span id="x"></span>
Y: <span id="y"></span>
```

```
$("#field").mousemove(function(e) {
  $("#x").text(e.pageX);
  $("#y").text(e.pageY);
}).hover(null, function(e) {
  $("#x,#y").text("");
});
```

33

binding event handler



34

interaction events

click
dblclick
mousedown
mousemove
mouseout
mouseover
mouseup

keydown
keypress
keyup

blur
change
focus
select
submit
hover
toggle
unload
unblur

```
submit(...)
↓
bind("submit", ....)
```

35

normalized event props

altKey	screenX
ctrlKey	screenY
currentTarget	shiftKey
data	result
metaKey	target
pageX	timestamp
pageY	type
relatedTarget	which

36

normalized event fns

<code>preventDefault</code>	<code>isDefaultPrevented</code>
<code>stopPropagation</code>	<code>isPropagationStopped</code>
<code>stopImmediatePropagation</code>	<code>isImmediatePropagationStopped</code>

hovering

~~`mouseover/mouseout`~~

`.hover(enter, leave)`



`.mouseenter(enter).mouseleave(leave)`

37

38

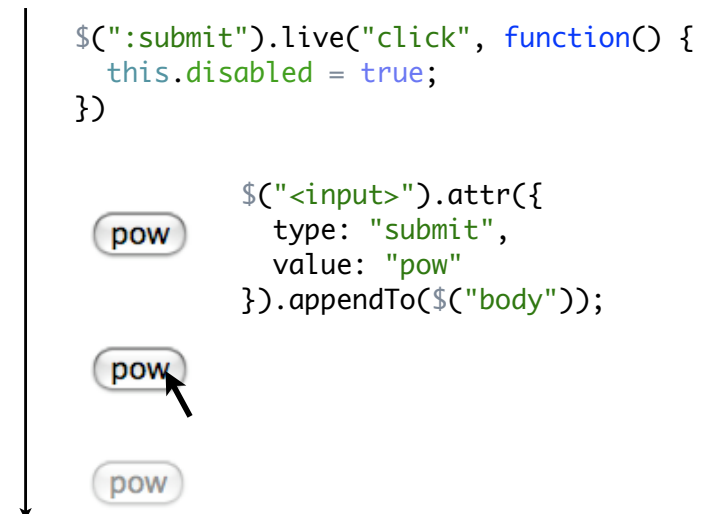
unbinding

```
foo.bind("submit.new", f1)
baz.bind("submit", f2)
bind.bind("submit", f3)
```

<code>\$("#*").unbind("submit.new")</code>	unbind namespace event
<code>\$("#*").unbind("submit")</code>	unbind "default" namespace
<code>\$("#*").unbind("submit", f2)</code>	unbind specific fn

live events

time



39

40

and there's more!

method	notes
one	like bind , but one-shot
trigger	trigger like user would
triggerHandler	fire code only
toggle	bind n listeners round-robin

41

don't worry about

dom level 0

bubbling

event instance

multiple handlers

dom level 2

capture

ie event model

42

\$.grep

4. utility functions

```
$.grep([1,2,3,4,5], isOdd)
```

```
=> 1,3,5
```

```
$.grep([1,2,3,4,5], isOdd, true)
```

```
=> 2,4
```

```
$.grep([1,2,3,4,5], isOdd, "blam")
```

```
=> 2,4
```

43

44

\$.extend

```
x = {floorWax: "lemon"}  
y = {dessertTopping: "chocolate"}
```

```
$.extend(x,y)  
=> floorWax: lemon  
    dessertTopping: chocolate
```

45

\$.map

```
jQuery.map([1,2,3],  
  function(x) { return x*x; }  
)  
=> [1,4,9]
```

```
jQuery.map("attack", function(x) {  
  return x.charCodeAt(0)+1;  
})  
=> [98,117,117,98,100,108]
```

46

\$.inArray

```
$.inArray("b", "foobar")  
=> 3
```

```
$.inArray("c", "foobar")  
=> -1
```

47

\$.unique is evil

WTF?

```
$.unique([1,2,3,2,3,2,1])  
=> [1,2,3,2,3,2,1]
```

```
$.unique(["foo", "bar", "foo"])  
=> [foo,bar,foo]
```

unique works
only on node sets

48

\$.merge

```
$.merge([1,2], ["a", "b"])
=> 1,2,a,b

$.merge([],null)
=> Result of expression
'second' [null] is not an object.

$.merge([1,2], {name: "Bill"})
=> 1,2
```

\$.makeArray

```
$.makeArray(1)
=> [1]

$.makeArray([1])
=> [1]

$.makeArray(1,2)
=> [1]

$.makeArray({name: "Bo"})
=> [[object Object]]
```

49

50

type testing

```
$.isArray([1,2,3])
=> true

$.isFunction(alert)
=> true

$.isFunction($)
=> true
```

\$.trim

```
$.trim("    foo    ")
=> foo

$.trim("\nbar\n")
=> bar
```

51

52

\$.param

```
$.param({a: 1})  
=> a=1
```

```
$.param({a: 1, a: 2})  
=> a=2
```

```
$.param({a: {nested: 1}, b: 2})  
=> a=%5Bobject+Object%5D&b=2
```

53

\$.support

```
$.support  
=> leadingWhitespace: true  
    tbody: true  
    objectAll: true  
    htmlSerialize: true  
    style: true  
    hrefNormalized: true  
    opacity: true  
    cssFloat: true  
    scriptEval: true  
    noCloneEvent: true  
    boxModel: true
```

54

preventing conflict

5. conflict avoidance

```
$  
=> function (selector, context) { ... }  
  
jQuery.noConflict()  
=> function (selector, context) { ... }  
  
$  
=> null
```

55

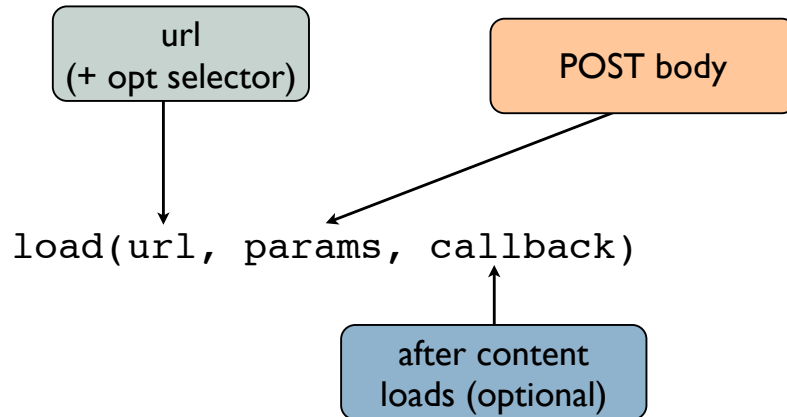
56

have it both ways

```
// $ not defined  
  
(function($) {  
    // $ == jQuery  
  
})(jQuery);  
  
// $ not defined
```

6. ajax

load



utility functions

serialize → query string

serializeArray → array

```
$.get(url, params, callback)  
$.getJSON(url, params, callback)  
$.post(url, params, callback, type)
```

html | text | xml | json | script | jsonp

under the hood

```
$.ajax(options)
$.ajaxSetup(options)

options =
  url
  type data dataType contentType
  timeout async
  beforeSend success error
  ifModified processData
  global
```

61

don't worry about

creating the xhr instance
ready states
tracking xhr across request/response

63

events summary

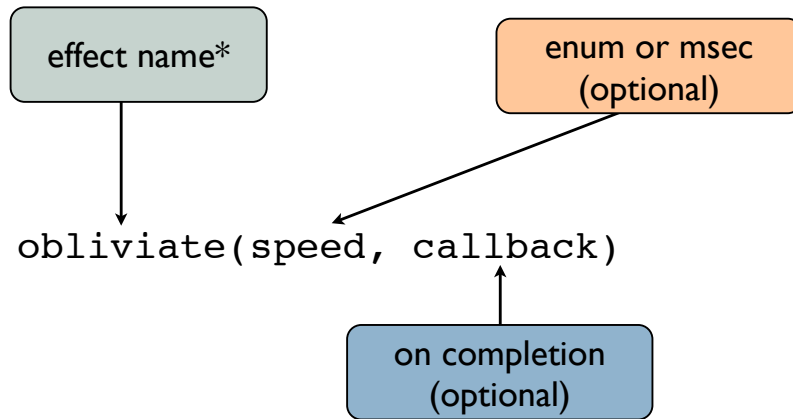
local	global	notes
	ajaxStart	sent when starting & norequests active
beforeSend	ajaxSend	outbound hook
success	ajaxSuccess	HTTP success
error	ajaxError	HTTP error
complete	ajaxComplete	regardless of HTTP status
	ajaxStop	sent when last active request completes

62

7. effects and animations

64

anatomy of an effect



*there is no function named obliviate

65

some basic effects

effect	modifies
show	size & opacity
hide	size & opacity
toggle	size & opacity
fadeIn	opacity
fadeOut	opacity
slideDown	reveal
slideUp	cover

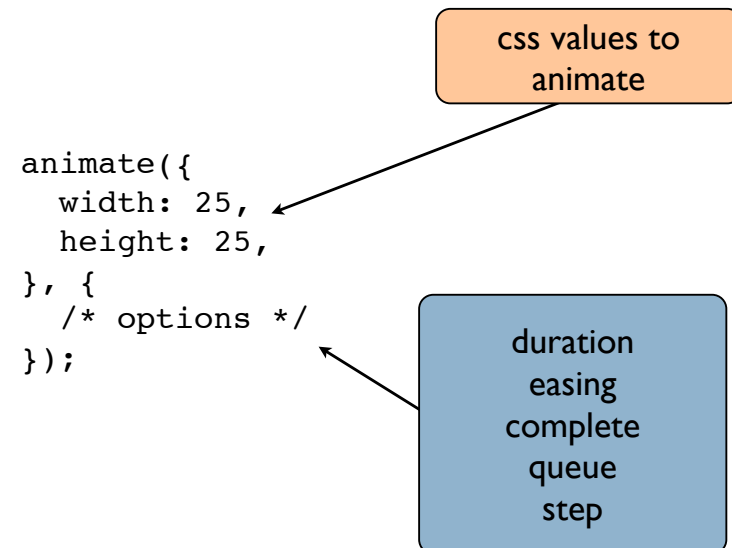
66

effects demo code

```
$(function() {  
  $(".call-value").click(function() {  
    $("#target")[$(this).val()]();  
  });  
  $(".eval-value").click(function() {  
    eval($("#target")." +  
      $(this).val());  
  });  
});
```

67

animate



68

8. extending jquery

get text from anything

```
if (jQuery(this).is(":input")) {  
    return jQuery(this).val();  
} else {  
    return jQuery(this).text();  
}
```

69

this would be easier

```
if (jQuery(this).is(":input")) {  
    return jQuery(this).val();  
} else {  
    return jQuery(this).text();  
}
```



jQuery(this).valOrText();

71

add to jquery.fn

```
jQuery.fn.valOrText = function() {  
    if (this.is(":input"))  
        return this.val();  
    else  
        return this.text();  
};
```

70

not done yet...

72

support all aritys

```
jQuery.fn.valOrText = function() {  
  if (this.is(":input"))  
    return this.val.apply  
      (this, arguments);  
  else  
    return this.text.apply  
      (this, arguments);  
};
```

not done yet...

73

dry it up

```
jQuery.fn.getTextFn = function() {  
  return this.is(":input") ?  
    this.val : this.text;  
}  
  
jQuery.fn.valOrText = function() {  
  this.getTextFn().apply(this, arguments);  
};
```

not done yet...

74

enumerate wrapped set!

```
// "set" case  
var args = arguments;  
this.each(function() {  
  $(this).getTextFn().apply($(this), args);  
});  
return this;
```

done (set)

75

get returns first item

```
var elem = this[0];  
if (elem) {  
  return $(elem).getTextFn()  
    .apply($(elem), arguments);  
} else {  
  return undefined;  
}
```

done (get)

76

valOrText

```
jQuery.fn.valOrText = function() {  
  if (arguments[0] === undefined) {  
    var elem = this[0];  
    if (elem) {  
      return $(elem).getTextFn()  
        .apply($(elem), arguments);  
    } else {  
      return undefined;  
    }  
  } else {  
    var args = arguments;  
    this.each(function() {  
      $(this).getTextFn().apply($(this), args)  
    });  
    return this;  
  }  
};
```

77

9. testing

78

screw.unit example

unit testing: screw.unit

```
Screw.Unit(function(){  
  describe("Your application javascript", function(){  
    it("does something", function(){  
      expect("hello").toEqual("hello");  
    });  
  });  
});
```

79

80

smoke example

mocking: smoke

```
it("can stub with Smoke!", function() {
  stub(Foo, "bar").and_return(7);
  expect(Foo.bar()).to(equal, 7);
});

it("can mock with Smoke!", function() {
  mock(Foo).should_receive("bar")
    .with_arguments(10).exactly(1, "time").and_return(42);
  expect(Foo.bar(10)).to(equal, 42);
});
```

81

82

headless builds

putting it all together: blue-ridge

```
rake test:javascripts
(in /Users/stuart/presentations/refactoring-javascript)
Running application_spec.js with fixture 'fixtures/application.html'...
..

2 test(s), 0 failure(s)
0.456 seconds elapsed
Running numberformatter_spec.js with fixture 'fixtures/numberformatter.html'...
.....

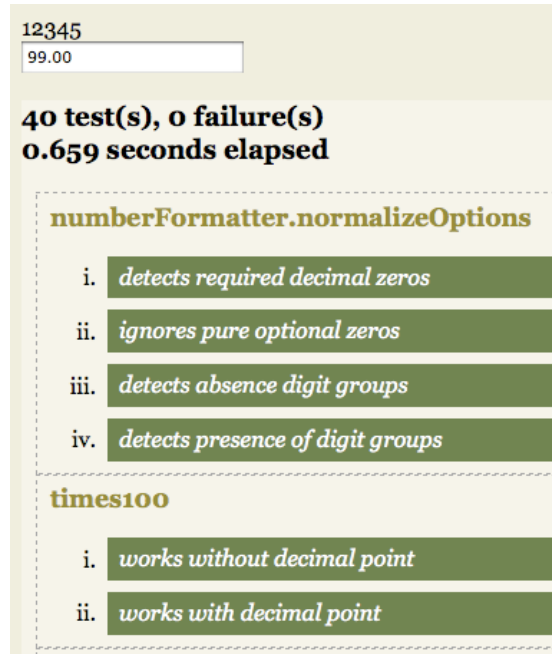
40 test(s), 0 failure(s)
0.518 seconds elapsed
```

<http://github.com/relevance/blue-ridge>

83

84

in-browser builds



85

relevance's choices

screw.unit (bdd/unit testing)
smoke (mocking)
rhino (headless test runs)
env.js (stub browser environment)
blue ridge (framework around the above)

86

where we have been...

1. wrapped sets
2. dom manipulation
3. events
4. utility functions
5. conflict avoidance
6. ajax
7. effects and animations
8. extending jquery
9. testing

jQuery
in Action: <http://www.manning.com/bibeault/>

Slides &
Samples: <http://github.com/relevance/jquery-demos>

Email: stu@thinkrelevance.com
Office: 919-442-3030
Twitter: twitter.com/stuarthalloway
Facebook: [stuart.halloway](https://www.facebook.com/stuarthalloway)
Github: [stuarthalloway](https://github.com/stuarthalloway)

Talks: <http://blog.thinkrelevance.com/talks>
Blog: <http://blog.thinkrelevance.com>
Book: <http://tinyurl.com/clojure>

How we work: <http://howwework.thinkrelevance.com>
Biblio: <http://tinyurl.com/agile-biblio>

87

88