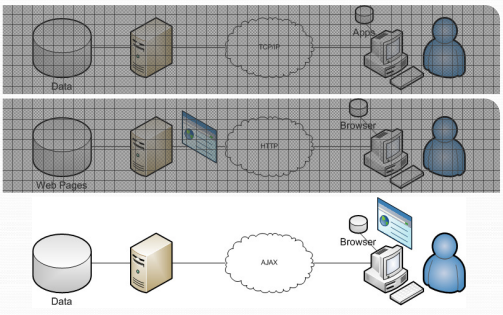
jQuery

Agenda

- Introduction To jQuery
- Selection and DOM Traversal
- DOM And Attributes Manipulation
- CSS Styling and jQuery
- Events

Evolution



Introduction To jQuery

- History :
  - Aug 22, 2005 : John Resig, creator of jQuery, floated the idea.
  - Aug 26, 2006 : First stable version of jQuery, **v1.0** released.

    . . .
  - May 3, 2011 : **v1.6** released - major rewrite leading to performance improvement
  - Popular version 1.7.1 (21-Nov 2011).

Introduction

- jQuery places a heavy emphasis on *chaining*.
- jQuery's features are organized into a handful of simple categories:
  - core functionality
  - selecting, manipulating, traversing
  - CSS, attributes
  - events
  - effects

The jQuery philosophy is "Write less, do more."

5

Useful Resources

- The jQuery docm : http://api.jquery.com
- The jQuery forum: http://forum.jquery.com/
- The jQuery mailing list archives : http://docs.jquery.com/Discussion#Archives

6

What is jQuery?

jQuery is:

- Free!
- Open-source and available under both MIT and GPL licences
- Only ~32Kb (minified and gzipped, ready for production) ⇒ lightweight footprint
- Cross-browser compatible
- Extensible! You can write your own plugins.

7

What is jQuery?

- jQuery is a JavaScript library that simplifies:
  - HTML element selection and document object model (DOM) traversal
  - Element creation, deletion, and modification
  - Event handling
  - Ajax interactions

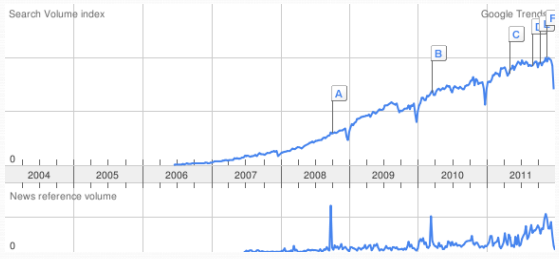
8

Why use jQuery ?

- jQuery is written entirely in JavaScript.
  - Why reinvent the wheel?
  - jQuery is tested and used by thousands of web sites.
  - jQuery is optimized for performance by JavaScript experts.
- jQuery is also designed to circumvent cross-browser compatibility problems.
- Consistent interface for maximal functionality across different browsers.
- jQuery is used by big companies such as: Google, Microsoft, Nokia, Dell, Bank of America, Drupal .
- Large community and developer pool ⇒ support is huge!

9

Usage Graph



10

A typical jQuery statement

selector	action	parameters
`jQuery('p')`	`.css`	`('color', 'blue');`
`$('p')`	`.css`	`('color', 'blue');`

11

The jQuery() Function

Four different ways to invoke it:

1. Invoke $() is to pass a CSS selector (a string)
   - It returns the set of elements from the current document that match the selector.
2. Invoke $() is to pass it an Element or Document or Window object.
   - It returns that jQuery object.
3. invoke $() to pass it a string of HTML text.
   - Returns a jQuery object representing those elements.

12

The jQuery() Function

4. Pass a function to it.
   - This function is called when the document has been loaded & the DOM is ready.

13

jQuery Terminology

- *"the jQuery function"*
- *"a jQuery object"*
- *"the selected elements"*
- *"a jQuery function"*
  - jQuery.noConflict()

14

Distinction between jQuery functions and methods

```
// Call the jQuery function each() to
// invoke the function f once for each element of array a
$.each(a,f);
```

```
// Call the jQuery() function to obtain a jQuery object that represents all
// <a> elements in the document. Then call the each() method of that
//  jQuery object to invoke the function f once for each selected element.
$("a").each(f);
```

15

Queries and Query Results

- $("body").length // => 1: doc have only one body element
- $("body")[0] // This the same as document.body
- // Find all headings, map to their ids, convert to a true array, and sort it.
- $(":header").map(function() { return this.id; }).toArray().sort();

- All jQuery objects have a property named jquery, and testing for the existence of this property is a simple way to distinguish jQuery objects from other array-like objects.

```
// Find all <script> elements in the document body
var bodyscripts = $("script", document.body);
bodyscripts.selector   // => "script"
bodyscripts.context    // =>  document  body info
bodyscripts.jquery
```

16

jQuery Getters and Setters

- Uses a single method for both.
  - Setter : set values on every element in the jQuery object, and then return the jQuery object to allow method chaining.
  - Getter : can only appear at the end of a method chain.

17

Getting and Setting CSS Attributes

```
$("h1").css("font-weight");  // Get font weight of first <h1>
$("h1").css("fontWeight");  // Camel case works, too

$("h1").css("font");  // Error: can't query compound styles

$("h1").css({ backgroundColor: "black",  // Set multiple styles at once
              textColor: "white", // camelCase names work better
              fontVariant: "small-caps", // as object properties
              padding: "10px 2px 4px 20px",
              border: "dotted black 4px" });

// Increase all <h1> font sizes by 25%
$("h1").css("font-size",  function(i,curval) {
        return Math.round(1.25*parseInt(curval));
});
```

18

How to Start

Three easy steps:

- Access the jQuery library and include it in your page.
- Make sure the page is ready.
- Execute the jQuery code!

19

Download the Library

From the http://jquery.com/ - two versions of the code:

- Production - Minified JavaScript (illegible) for use in production
  - Lightweight footprint
  - Saves bandwidth for you and speeds up page requests for your users.
- Development- Uncompressed JavaScript
  - Very useful if you need to look at how jQuery is implemented.

20

To include the jQuery library

```
<head>
  <title>My jQuery page</title>
  <script type="text/javascript" src="jquery-1.9.1-min.js">
  </script>

  <script type="text/javascript" src="myscript.js">
   </script>
</head>
```

21

jQuery library from CDN

```
<head>
  <title>My jQuery page</title>
  <script type="text/javascript"
  src="http://code.jquery.com/jquery-1.9.1.min.js">
  </script>

  <script type="text/javascript" src="myscript.js">
   </script>
</head>
```

22

The jQuery Function Object

- Loading the jQuery library creates a single *function object* named jQuery :
  - jQuery('li').css('background-color', 'yellow');
  - Alias :

```
$('li').css('background-color', 'yellow');
```

23

Is The Document Ready?

- jQuery code in the <script> element inside the <head> of your document.
- *onLoad* handler on the <body> element .
- The *ready* event occurs when the page DOM is complete.

```
$(document).ready(function() {
    //Perform here all the jQuery magic
});
```

24

Is The Document Ready?

- Register a *callback* function / *handler* :
  $(function() { /* jQuery code */ });

25

Where to Run Your Code?

- Place the <script> elements that load external JavaScript files in the <head>of a document.
- Place your <script> element(s) just before the</body> close tag.
  - Progressive enhancement

26

Selections And Dom Traversing

27

Selection

- Select elements before you can manipulate them.
  - Support for most CSS selectors, plus its own custom extensions.
- jQuery selection returns a *jQuery object*.
  - The jQuery object represents **0** or more elements in the document.
  - $('div').length returns the number of elements in the page.

28

Basic Selectors

- $('p');
- $('*');
- $('.status');
- $('#header');
- $('p, li, .status, #nav');

29

jQuery Usage Example

```
$("li:odd").prepend('<span>Changed</span>').css({background:"red"});
```

```
<ul>
  <li>
    First item
  </li>
  <li>
    Second item
  </li>
  <li>
    Third item
  </li>
</ul>
```

```
<ul>
  <li>
    <span>Changed</span>
    First item
  </li>
  <li>
    Second item
  </li>
  <li>
    <span>Changed</span>
    Third item
  </li>
</ul>
```

```
<ul>
  <li style="background:red;">
    <span>Changed</span>
    First item
  </li>
  <li>
    Second item
  </li>
  <li style="background:red;">
    <span>Changed</span>
    Third item
  </li>
</ul>
```

Hierarchy Selectors

- $('#footer span');
- $('ul > li');
- $('h2 + p');
- $('h2 ~ p');

31

Selection By Attribute

- $('a[href]');
- $('a[href="http://google.com/af/zaz"]');
- $('a[href*="goo.gl"]');
- $('a[href^="http://goog.co"]');
- $('a[href$=".pdf"]');
- $('p[lang|="en"]');

32

Form Selectors

- $(':button'); , $(':checkbox'); , $(':file'); , $(':image'); , $(':password'); ,$(':radio'); , $(':reset'); , $(':submit'); , $(':text');
- $(':input');
- $(':checked');
- $(':selected');

33

Position Filters

- $('.article:eq(2)');
- $('.article:gt(1)');
- $('.article:lt(3)');
- $('.article:first');
- $('.article:last');
- $('.article:odd');
- $('.article:even');

34

Other Filters

- $(p:contains('Google'));
- $('div:has(h2)');
- $('option:not(:selected)');
- $('p:hidden');

35

jQuery Method Chaining

- The value returned from a jQuery selection is an instance of a jQuery object.

```
$('.status')
      .css('backgroundColor','yellow')
      .attr('title', 'My Web page');
```

36

jQuery Usage Example

```
$("div:hidden").find(".foo").empty().text("Changed").end().show();
```

DOM Traversal

- Methods for selecting the children, the parents, and the siblings of each element in the wrapped set.

Traversal methods:

- $('.article').children('p');
- $('.article').find('p');
- $('h2').prev();
- $('h2').next('p');
- $('#form-button').siblings('fieldset');
- $('.menu-li2').parent();

38

Filter Methods

- $('#content h2').first();
- $('#menu > li').eq(1);
- $('li').filter(':even');
- $('div').not('.article');
- $('li').has('ul');

39

Advanced Method Chaining

- Combining method chaining with traversal methods :

```
$('#selected')
    .css('backgroundColor', 'yellow')
    .parent()
    .addClass('current');
```

40

Advanced Method Chaining

```
$('.article')
    .find('p')
    .css('color','green')
    .end()
    .find('h2').css('color','red');
```

41

Dom Manipulation

42

DOM Manipulation

- Most common operation.
- Handling of elements in the DOM :
  - create, insert, move, clone, remove, and replace.
- To create :
  - Provide a string with HTML markup of the element as the sole argument:

```
var $menuli1 = $('<li class="menu-li1“>Food</li>');
```

43

Creating Elements

- To create :
  - Provide a string with the basic HTML markup as the first arg, and a JavaScript object as the second arg.

```
var $menuli1 = $('<li/>', {
    'class': 'menu-li1',
    'text': ‘Food‘
  });
```

44

Inserting Elements

Many ways to insert elements into a page:

- Before an existing element.
- After an existing element.
- Inside a container element, at its beginning.
- Inside a container element, at its end.

- Also, there are two approaches to each method of inserting elements into a page:
  - Place the selected element(s) relative to another element
  - Place an element relative to the selected element(s)

45

Before an Existing Element

- .insertBefore()

```
var $menuli1 = $('<li class="menu-li1">Italian Food</li>');
$menuli1.insertBefore('.menu-li1:first');
```

- .before()

```
$('.menu-li1:first').before('<li class="menu-li1">Italian Food</li>');
```

46

CSS Styling

47

Reading And Modifying CSS

- Accessing CSS properties in plain old JavaScript : painful.
- jQuery handles these browser inconsistencies.
  - .css() method
  - CSS names (e.g., background-color) or
  - DOM name (e.g., backgroundColor)

  - E.g. `$('div').css('backgroundColor');`
    `$('.header').css('backgroundColor', 'red');`

48

Setting Properties

```
$('.article')
    .css({ 'backgroundColor': '#A2C5CC',
           'borderColor': 'green',
           'marginBottom': '12px'
});
```

```
$('#mydiv').css('color', 'green');   =>
document.getElementById('mydiv').style.color = 'green';
```

49

Removing CSS Properties

- Setting the value of a style property to an empty string removes it:
  - In the HTML style attribute
  - Through jQuery's .css() method
  - Through direct DOM manipulation of the style property

50

Events

51

Events Overview

When an user is interacting with a web page, lots of events are fired:

- Clicking on an element
- Typing in a text box
- Scrolling on the page
- Hovering on an element
- etc…

- JS can detect & handle these :
  - *event handler* or a *callback*

52

Common Event Types

blur	change	click	dblclick	error	focus
focusin	focusout	keydown	keypress	keyup	load
mousedown	mouseenter	mouseleave	mousemove	mouseout	mouseover
mouseup	resize	scroll	select	submit	unload

53

Performing Tasks on Page Load

JavaScript's built-in onload event.

- window.onload : Either of:
  - Within our HTML markup:

```
function doStuff() {
    // Perform a task...
}
<body onload="doStuff();">
```

  - Within JavaScript code:

```
window.onload = doStuff;
```

54

Performing Tasks on Page Load

Using jQuery:

```
$(document).ready(function() {
    // Our code here...
});
```

Or, we can write:

```
$().ready(function() {
    // Our code here...
});
```

Or, make an implicit call :

```
$()(function() {
    // Our code here...
});
```

55

Simple Events

User initiated:

- Mouse clicks (onclick)
- Form fields being modified (onchange)
- Windows changing size (onresize).

56

Binding an Event

```
$(document).ready(function() {
    $('#switcher-large').bind('click', function() {
        $('body').addClass('large');
    });
});
```

57

Event Handler Context

- Using **$(this)** within the event handler: creates a jQuery object corresponding to the element.
  - Can act on it just as if located it with a CSS selector.

```
$(this).addClass('selected');
```

58

Sample html

```
<div id="switcher">
    <h3>Style Switcher</h3>
    <div class="button selected" id="switcher-
      normal">Normal</div>
    <div class="button" id="switcher-narrow">Narrow
      Column</div>
    <div class="button" id="switcher-large">Large Print</div>
</div>
```

59

Optimization

```
$('#switcher-large').bind('click', function() {
    $('body').removeClass('narrow').addClass('large');
});

$('#switcher .button').bind('click', function() {
    $('#switcher .button').removeClass('selected');
    $(this).addClass('selected');
    });
});
```

60

Shorthand Events

```
$(document).ready(function() {
  $('#switcher .button').click(function() {
    $('body').removeClass();
    if (this.id == 'switcher-narrow') {
      $('body').addClass('narrow');
    }
    else if (this.id == 'switcher-large') {
      $('body').addClass('large');
    }
    $('#switcher .button').removeClass('selected');
    $(this).addClass('selected');
  });
});
```

61

Compound Events

- These intercept combinations of user actions, and respond to them using more than one function.
  - .toggle()
  - .hover()
- The .toggle() method takes two arguments, both of which are functions.
  - The first click on the element causes the first function to execute.
  - The second click triggers the second function.
  - The two functions continue to alternate every other click thereafter.

62

.toggle()

```
$(document).ready(function() {
  $('#switcher h3').toggle(function() {
    $('#switcher .button').addClass('hidden');
  }, function() {
    $('#switcher .button').removeClass('hidden');
  });
});
```

63

Alt. approach

```
$(document).ready(function() {
    $('#switcher h3').click(function() {
        $('#switcher .button').toggleClass('hidden');
    });
});
```

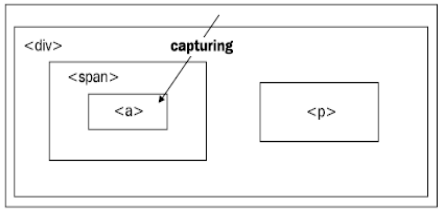
64

.hover()

```
$(document).ready(function() {
    $('#switcher .button').hover(function() {
        $(this).addClass('hover');
    }, function() {
        $(this).removeClass('hover');
    });
});
```

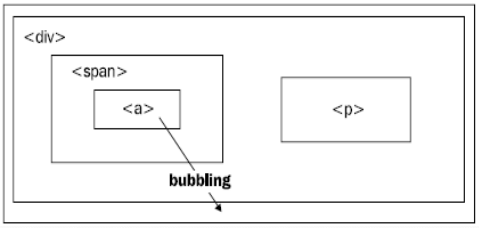
65

Event Capturing



66

Event Bubbling



67

Side Effects of Event Bubbling

- Event bubbling can cause unexpected behaviour - the wrong element responds to a *mouseover* or *mouseout*.
- When the cursor exits the <a> element, a mouseout event is sent to that.
  - This event will then bubble up to the and then to the , firing the same event handler.
    - This bubbling sequence is not likely to be desired.

68

Event Object

```
$(document).ready(function() {
    $('#switcher').click(function(event) {
        if (event.target == this) {   // the first receiver
            $('#switcher .button').toggleClass('hidden');
        }
    });
});
```

69

Stopping Event Propagation

- .stopPropagation() : eliminates bubbling completely for the event.

```
$('#switcher .button').removeClass('selected');
$(this).addClass('selected');
event.stopPropagation();
```

70

Default Actions

- .preventDefault() : stops the event in its tracks before the default action is triggered.
- Event propagation and default actions are **independent** mechanisms.
- If we wish to halt both, we can return **false** from our event handler, which is a shortcut for calling both.

71

Removing an Event Handler

```
$(document).ready(function() {
  var toggleStyleSwitcher = function() {
    $('#switcher .button').toggleClass('hidden');
  };

  $('#switcher').click(toggleStyleSwitcher);

  $('#switcher-normal').click(function() {
    $('#switcher').click(toggleStyleSwitcher);
  });
  $('#switcher-narrow, #switcher-large').click(function() {
    $('#switcher').unbind('click', toggleStyleSwitcher);
  });
});
```

72

One shot trigger

- Unbind an event handler immediately after the first time it is triggered.

```
$(document).ready(function() {
    $('#switcher').one('click', toggleStyleSwitcher);
});
```

73

Simulating User Interaction

- .trigger() method

```
$(document).ready(function() {
    $('#switcher').trigger('click');
});
```

74

jQuery Event Object

- Commonly used properties and methods :
  - pageX , pageY
  - target
  - timestamp
  - type
  - preventDefault()
  - stopPropagation()

75

Event Delegation

- Handles dynamic addition of new similar elements.
- It lets an event "bubble up" the DOM tree to a parent element.
- Convenient methods:
  - .live()
  - .on()

76

live()

- Lets you bind a handler.
- It works for all current and future matched elements.

```
$('#description').live('mouseover', function() {
        $(this).css('background-color', 'yellow');});
```

- To stop (unbind):

```
$('p#description').die('mouseover');
```

77

.on()

```
$("#dataTable tbody tr").on("click", function(event){
        alert($(this).text());
});
```

78

AJAX

79

Ajax

- Coined in 2005
- Acronym for Asynchronous JavaScript and XML.
- Mimics the feel of a desktop application, and gives the user a more responsive and natural experience.
- Earlier, each web browser supplied slightly different methods for achieving this.

80

Loading Remote HTML

- load()
  - Magically grabs an HTML file off the server and insert its contents within the current web page.
  - You can load either static HTML files, or dynamic pages that generate HTML output.
    - same origin
  - So, its different from the window load event.

81

Picking up only parts

- Be specific using selectors

```
$('#biography').load('computadors.html div:first');
```

82

Advanced loading

- A common requirement is to specify some data to pass along to the server, e.g. search for info.

```
$('div#results').load('search.php',
  'q=jQuery&maxResults=10');
```

- Alt.: adding a callback fn.

```
$('div#result').load('feed.php', function(data, status,
  response) { // Post-processing time!
});
```

83

Fetching Data with $.getJSON

- Creating mashups
  - Grab data from multiple third-party web sites and squish it together in a new and interesting way.
  - XML
  - JSON

```
$.getJSON(
  'http://feeds.delicious.com/v2/json/tag/celebs?call
  back=?', function(data) {
      alert('Fetched ' + data.length + ' items!'); });
```

84

The jQuery Ajax Workhorse

- $.ajax method
  - All of jQuery's Ajax functions are simply wrappers around the $.ajax method.
  - Each designed to provide with a simple interface for a specific type of task.

85

$.ajax

```
$.ajax({
   type: 'GET',
   url: 'getDetails.php',
   data: { id: 142 },
   success: function(data) {
         // grabbed some data!
   };
});
```

86

Common Ajax Settings

```
$.ajaxSetup({
   type: 'POST'
   url: 'send.php',
   timeout: 3000
});

$.ajax({
   data: { id: 142 }
});
```

87

Loading External Scripts with $.getScript

- To make the application to be as snappy as possible.
- This function will load and execute a JavaScript file via Ajax.
- We can load a minimal amount of our code when the page loads
- Then pull in further files as we need them.
  - This is particularly useful if we require any plugins that have a sizeable footprint,

88

Loading External Scripts with $.getScript

```
$.getScript('http://view.jquery.com/trunk/plugins/color/
jquery.color.js', function() {
    $('body').animate({'background-color': '#fff'}, 'slow');
});
```

89

GET and POST Requests

- Simple wrapper functions around the $.ajax method
  - But are more convenient to use.
  - Just select a URL and any data you want to push.

```
$.get(url, data, callback, dataType);
$.post(url, data, callback, dataType);
```

- dataType : xml, html, json, etc. (optional)

90

GET and POST Requests

```
$.get("getInfo.php", function(data) {
      alert("got your data:" + data);
});

$.post("setInfo.php", {id: 2, name: "Equator  and land"});
```

91

jQuery Ajax Events

- When making Ajax requests with jQuery, a number of events are fired.
  - Show "loading ..." message
  - Error can occur.
- There are two types of Ajax events in jQuery:
  - local events: apply only to individual Ajax requests.
    - You handle local events as callbacks.
  - Global

92

Local Error

```
$.ajax({
   url: "test.html",
   error: function() {
         alert('an error occurred!');
   }
});
```

93

Global Error

- Are attached to the DOM node where you'd like to show a response to the user.

```
$("#msg").ajaxError(function(event, request, settings) {
   $(this).html("Error requesting page " + settings.url +
   "!");
});
```

94

Ajax Events

- The success (local) and ajaxSuccess (global) events let us know when an event has completed successfully.
- The complete(local) and ajaxComplete(global) events tell us when an Ajax request has concluded, regardless of its success or failure.
- You will only receive either a success or error event for each request.
- But you'll always receive a complete event.

95

Ajax Events

- The beforeSend(local) and ajaxSend(global) events let us react just before an Ajax request is sent into the world.
- The ajaxStart and ajaxStop global events occur when an Ajax request fires and no others are already running, and when all requests are finished, respectively.

96

jQuery : Best Practices

97

Best Practices

- Always Descend From an #id
  - The fastest selector in jQuery is the ID selector
    E.g. ($('#my_button')).
  - Selecting a single element : direct.
  - Selecting Multiple Elements:
    - Descend from the closest parent ID
- Use Tags Before Classes
  - The second fastest selector in jQuery is the Tag selector.
    E.g. ($('head'))

98

Best Practices

- Cache jQuery Objects
  - Save your jQuery objects to a variable & continue.
- Harness the Power of Chaining
  - Less code & so, lightweight.
- jQuery allows us to run additional selector operations on a wrapped set.
  - This reduces performance overhead on subsequent selections, since already stored.
- Limit direct DOM manipulation

99

Best Practices

- Compressing Your JS
  - Gzipping + minifying
- Use of CDN vs local hosted copy
  - Specify the library version explicitly.
  - Sharing of bandwidth
  - More users likely to have it already
  - Else, it downloads in parallel
  - Google could collect details about your site (traffic profile, etc.)

100