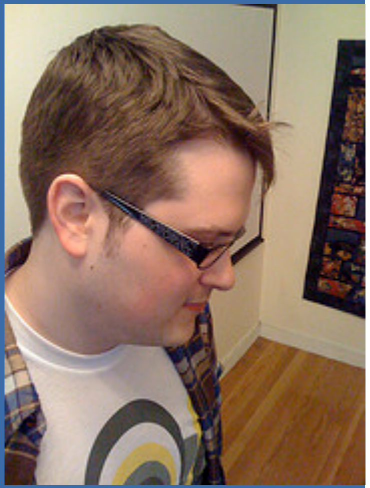
jQuery

CSCB20

What is jQuery?



- A JavaScript library created by John Resig that simplifies client-side programming tasks and solves messy issues such as cross-browser code compatibility (write once, deploy across many browsers)
- Implemented as a JavaScript file; to use jQuery include a reference to its definition file in your html document head element using a `<script>` element.
- Delivers huge boost in productivity and sophistication over coding in native JavaScript

Why jQuery?

JavaScript code is a tedious, time-consuming, and error-prone process, e.g.:

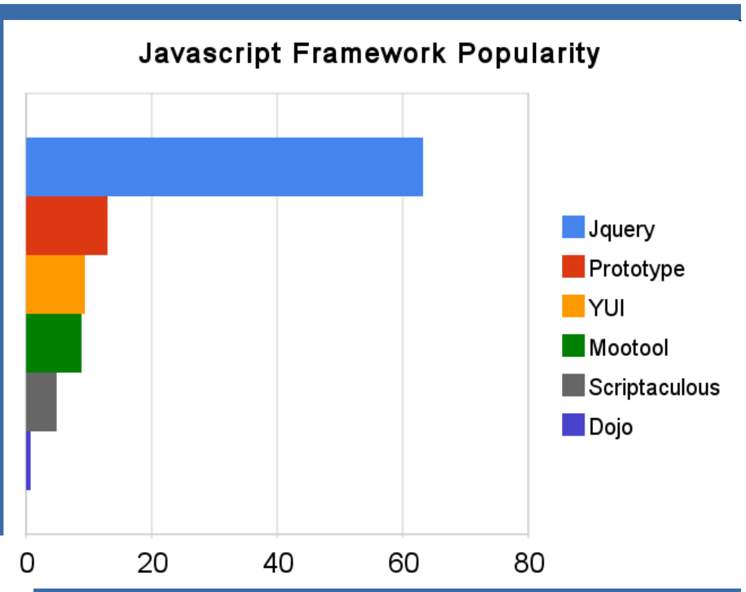
- extra code to accommodate browser differences
- unexpected effects due to browser-loading order
- lack of mature tools to support development
- significant effort required for DOM-API navigation

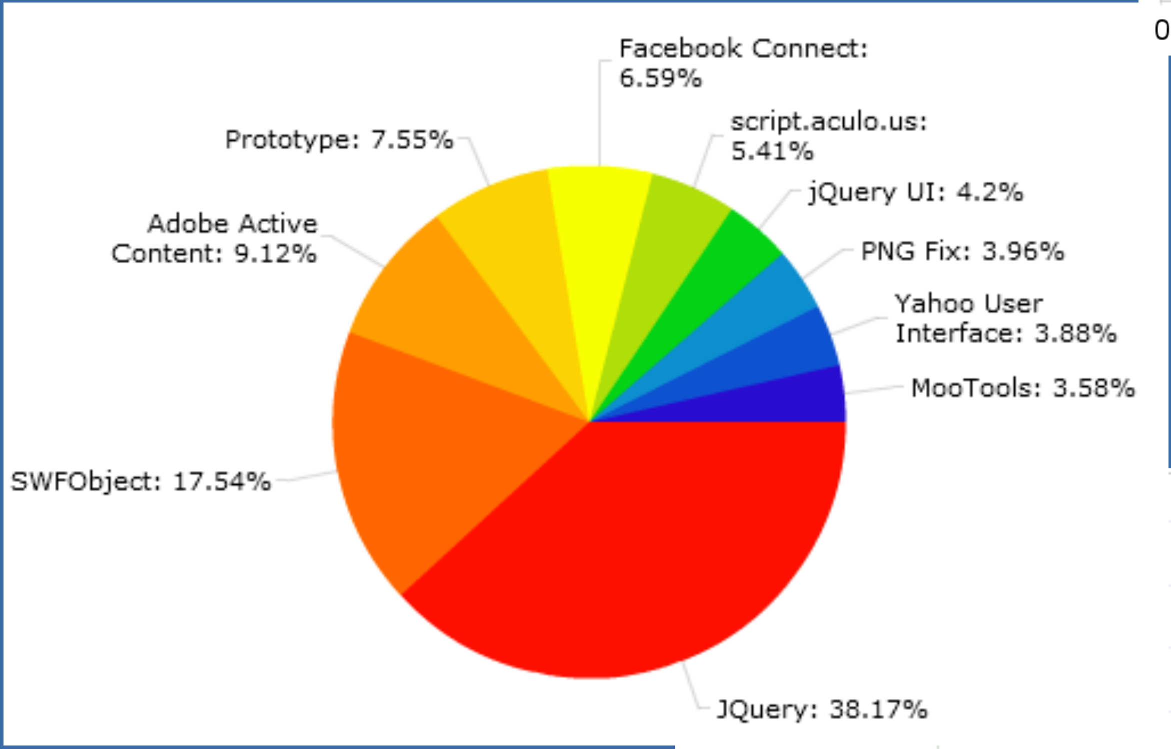
jQuery simplifies this process.

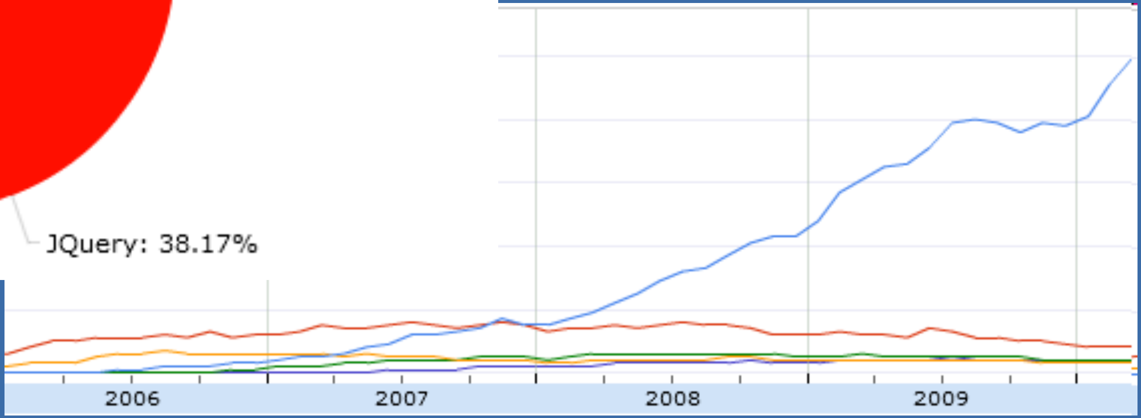
Why jQuery?

Why jQuery, rather than Prototype, Scriptaculous, or … ?

jQuery most widely adopted, lots of momentum







•

Why jQuery?

- Simplifies coding, making it easier to get right
- With a single consistent interface you can control:
  - DOM interaction (to manipulate Web page content)
  - Event handlers
  - Style properties
- Consistency across browsers; a major headache when using plain JS + DOM, one of the Achilles' heels of Web-app development
- Takes care of error-prone hard-to-debug race conditions between object creation and object use (JS referring to a page element that isn't yet loaded – its value is undefined)
- Good for your resume ;-)

jQuery Resources

- These slides provide an overview of most of the jQuery structures you need for assignment 3, but you may wish to use additional online documentation and API references:
- Getting started with jQuery: http:/docs.jquery.com/Tutorials
- jQuery API reference: api.jquery.com
- Visual jQuery – indexed examples: visualjquery.com
- jQuery Pocket Reference: http://books.google.com/books/about/Jquery_Pocket_Reference.html?id=qPCCUdDefdkC

jQuery Advantages

A few things that make jQuery powerful and easy to use:

- queries can be chained together to perform complex tasks
  - each jQuery operation returns a jQuery value so you can chain them together (like a pipeline)
- result sequences can be referenced as a single unit e.g.: `$('.tab').hide()` to hide all elements of class tab

jQuery Advantages

A few things that make jQuery powerful and easy to use:

- API, including its use of CSS selector notation, is intuitive, consistent, and common-sense, so even with only a little knowledge you can usually achieve your desired result

- jQuery has an extensible plugin architecture that has spawned a large community of plugin features useful for building more sophisticated RIA/Web 2.0 type Web apps with much less time and effort

Some of these are matters of opinion, but there's no disputing jQuery is now the most popular JS library

How do you use jQuery?

Download and put in the same location as webpage:

```
<head>
<script src="jquery-1.12.0.min.js"></script>
</head>
```

Include it from an external source such as google:

```
<head>
<script src="http://ajax.aspnetcdn.com/ajax/
jQuery/jquery-1.12.0.min.js">
</script>
</head>
```

jQuery vs POJS

Let's consider a few simple examples to compare how things are done in jQuery vs plain old JavaScript (POJS) with the Document Object Model (DOM) API

Suppose we want to associate a click event with each link within a particular area of a page:

Using JavaScript and DOM:

```
// confirm user hypertext links upon click
var ext_links = document.getElementById('ext_links');
var links = ext_links.getElementsByTagName('a');
for (var i=0;i < links.length;i++) {
  var link = links.item(i);
  link.onclick = function() {
   return confirm('You are going to visit: ' + this.href);
   };
}
```

Basic jQuery Format

jQuery uses the format:

```
$(select).action()
```

Where

- A `$` sign is used to define/access jQuery
- A `(selector)` to "query" HTML elements
- A jQuery `action()` to be performed on the elements

Examples

```
$("#test").hide()
$(".test").hide()
```

jQuery Functions

Function go inside a document ready event:

```
$(document).ready(function(){
     // jQuery methods go here…
});
```

There is a short cut notation:

```
$(function(){
     // jQuery methods go here…
});
```

•

•

jQuery vs POJS

Using JavaScript and DOM (from prior slide):

```
var external_links =
    document.getElementById('external_links');
var links =
   external_links.getElementsByTagName('a');
for (var i=0;i < links.length;i++) {
  var link = links.item(i);
  link.onclick = function() {
    return confirm('You are going to visit: ' +
       this.href);
  };
}
```

Using jQuery:

```
$('#external_links a').click(function() {
  return confirm('You are going to visit: ' +
         this.href);
});
```

•

•

DOM-Selection Using CSS Selectors

same Selectors
we use for CSS

- Element Type `E`
- Grouping `E, F, G`
- Universal `*`
- Class `[E].classvalue`
  (element name E optional – meta brackets)
- Id `[E]#myID`
- Contextual
  - Descendent `E F` (prior jQuery example)
  - Child `E > F`
  - Adjacent `E + F`
- Pseudo-element `E:pseudo-element`
- Attribute `E[foo="hi"]` (literal brackets)

•

•

jQuery vs POJS

- Suppose we want to create a new paragraph <p> and add it at the end of the existing page:
- using JavaScript DOM:

```
var new_p = document.createElement("p");
var new_text = document.createTextNode("Hello World");
new_p.appendChild(new_text);
var bodyRef =
   document.getElementsByTagName("body").item(0);
bodyRef.appendChild(new_p);  // new_p becomes visible
```

- Typical POJS pattern: incrementally build elements and glue (append) them together; only when glued to a currently-displayed document element do the new elements become visible

jQuery vs POJS

Using JavaScript DOM (from prior slide):

```
var newPP = document.createElement("p");
var newTxt = document.createTextNode("Hello World");
newPP.appendChild(newTxt);
var bodyRef =
      document.getElementsByTagName("body").item(0);
bodyRef.appendChild(newPP);
```

Using jQuery:

```
$('<p></p>').html('Hello World!').appendTo('body');
```

- Create a new element by quoting its HTML (‘<p></p>’)
- Call html() with new-element content as parameter-value
- Add to existing document, using appendTo() on body element

jQuery excels at “chaining” together actions in a fluid way

jQuery vs POJS

- Suppose we want to bind a handler-function to an event associated with a particular element.
- using JavaScript DOM we use an on-event attribute to bind built-in function alert to a click event:

```
<a href="" onclick="alert('Hello world')">hey</a>
```

- `using jQuery:`

```
 $(document).ready(function() {
    $("a").click(
        function() { alert("Hello world!"); }
    );
 });
```

- wait a minute, how is that an improvement?!

jQuery vs POJS

```
<a href="" onclick="alert('Hello world')">hey</a>
```

versus

```
$(document).ready(function() {
    $("a").click(
        function() { alert("Hello world!"); }
    );
});
```

- The difference is that the jQuery version is handling every single `<a>` element, POJS just one single element
- Just as CSS separates presentation from structure, jQuery separates behavior from structure.
- This is one of the key insights that has made jQuery such a powerful & successful JavaScript library.

POJS DOM Element Selection

Getting Elements using JavaScript DOM:

```
    document.getElementById("idval")
```

Returns unique DOM object with an id attribute of idval (id values must be unique within documents)

```
    getElementsByTagName("tagname”)
```

Returns an array of DOM objects, all of type tagname.

Powerful, but must process the array using a loop.

Problems with JavaScript DOM API:

- Browser inconsistencies; much too verbose; hard to read

jQuery Element Selection

Getting Elements with jQuery (use CSS selectors!):

```
$("#idval")
```

Get unique element by idval

```
$(".classname")
```

Selects all elements with matching classname

```
$(".classname div < p")
```

p children of div within elts with matching classname

```
$("tagname, #myid")
```

All tagname elements and element with id "myid"

- returns array of pointers to jQuery objects for all the HTML elements that match the selection criteria

jQuery get/set Element Values

`object.html()`

gets (returns) object's innerHTML (the HTML content of the element)

`object.html("<p>my html content</p>")`

sets object' s html content to the parameter string

```
$(".important").html("<h1>Note</h1>");
```

Sets the html of all elements with a class of 'important' to `"<h1>Note</h1>"`

•

•

Using jQuery

Link to the jQuery source

- local copy or remote master URL:

```
<script type ="text/javascript" src="http://
   code.jquery.com/jquery-1.8.3.js">
</script>
```

or

```
http://code.jquery.com/jquery-1.8.3.min.js
```

"minified" version to reduce size and download-time)

To execute your jQuery code, put it in the ready() function (within a script element)

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

Can abbreviate ready() line as just: `$(function() {…});`

•

•

jQuery, Try This

Create an HTML page with this element in the body

```
<div id="test"></div>
```

What does this mean?

Placeholder: common pattern when we need a destination to place some Ajax or DOM content

Now in the body of the ready function, use jQuery to:

select the div with an id of "test"

set the HTML of that element to value "Hello World!"

jQuery, Event Handlers

JavaScript provides a large selection of events

We can capture user interactions via these events (e.g. mouse clicks, key presses, form field focus, form submission)

We can also trigger these events using jQuery code

abort	**blur**	**change**	**click**	**dblclick**	**error**	**focus**
keydown	keypress	keyup	load	mousedown	mousemove	mouseout
mouseover	mouseup	reset	resize	select	submit	unload

•

•

jQuery, Event Handlers

Suppose you want to trigger execution of a JavaScript (or jQuery) function when an event is triggered

```
function makeInvisible(event) {
    /* typically do something with the
       element associated with the event */
}
// call makeInvisible func when .hide clicked
$(".hide").click(makeInvisible);
```

Suppose you need to trigger a click event from jQuery: `$(".hide").click();`

jQuery, Event Handlers

```
function handler(event) { // do something }
```

An event handler can take an event-type parameter, which has the following methods and properties defined

method / property name	description
type	what kind of event, such as "click" or"mousedown"
target	the element on which the event handler was registered
preventDefault()	prevents browser from performing its usual action in response to the event
stopPropagation()	prevents the event from bubbling up further
stopImmediatePropagation()	prevents the event from bubbling and prevents any other handlers from being executed