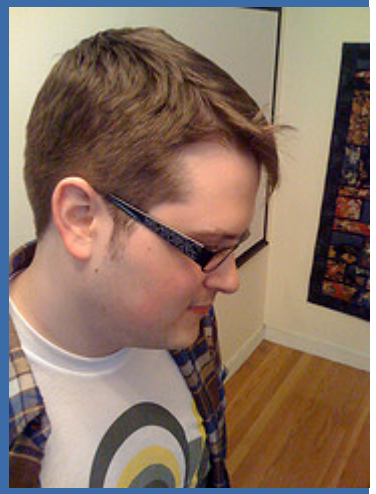
jQuery

CSCB20

What is jQuery?

- A JavaScript library created by John Resig that simplifies client-side
- 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?

- 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-3.1.1.min.js"></script>
</head>
```

Include it from an external source such as google:

```
<head>
<script src="http://ajax.aspnetcdn.com/ajax/
jQuery/jquery-3.1.1.min.js">
</script>
</head>
```

Basic jQuery Format

jQuery uses the format:

```
$(selector).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

Functions 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 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);
  };
}
```

Using jQuery:

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

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 (‘ ’)
- 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 elements 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>"`

Review - using jQuery

Link to the jQuery source -- local copy or remote master URL:

```
<script type ="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/
  3.1.1/jquery.min.js" >
</script>
```

or

```
https://code.jquery.com/jquery-3.1.1.min.js
```

Use "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();`

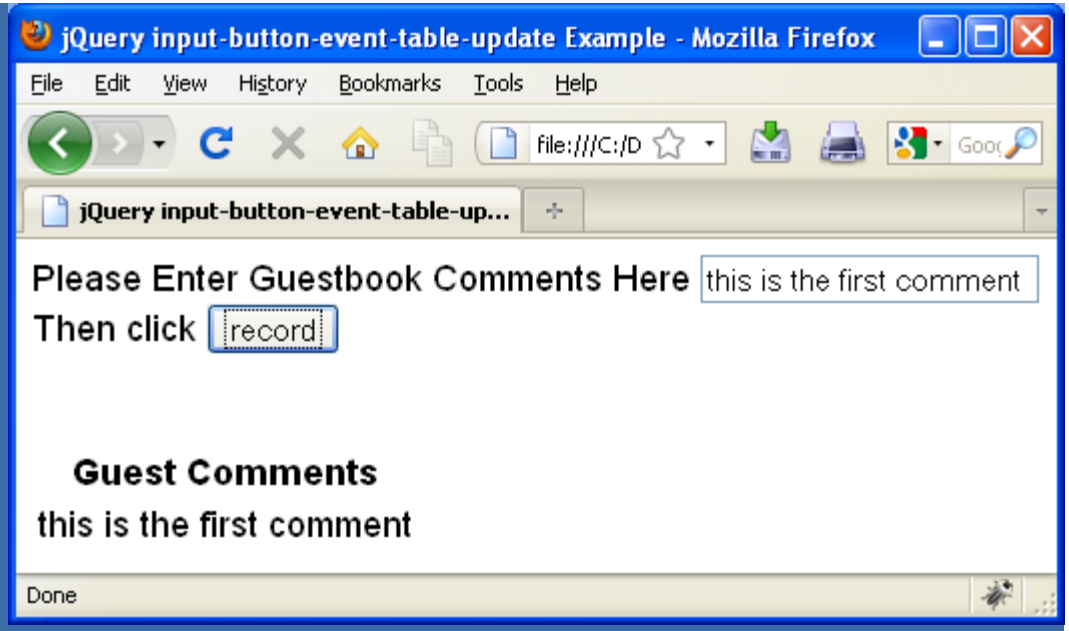
jQuery, Event Handlers

```
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

jQuery, a Larger Example



- Let's raise the bar a bit by doing something .
- Create a Web page with an input text field for users to enter brief guestbook comments
- Provide a button that when clicked will retrieve the currently-entered comment text and write it to an HTML table where comments are collected.

jQuery, a Larger Example

```
<html>
  <head>
    <title>jQuery input-button-event with table-update Example</title>
    <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1
.8.3/jquery.min.js">
    </script>
    <script>
      $(document).ready(function(){
          $('#record').click (function() {
                var sig = $('#comment').val();
                $('#guestbook').before('<tr><td>' + sig + '</td></tr>');
            });
      });
    </script>
  </head>
  <body>
    Please Enter Guestbook Comments Here <input id="comment" type="text"/>
    Then click <input type="button" id="record" value="record"/>
    <br/> <br/> <br/>
    <table>
      <tr><th>Guest Comments</th></tr>
      <tr style="display:none" id="guestbook"><td/> </tr>
    </table>
  </body>
</html>
```

Detecting event types

Suppose a user makes a choice from a `<select>` element; how do you detect that they made a choice, and how can you determine what they chose?

```
... header with script tags ...
$(”#mymenu").change(showMe);
function showMe() {
  alert(this.value + "was chosen");
}
</script>
<select id="mymenu">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```

Dealing with forms

- An HTML form element wraps a set of input types, such as buttons, input-text fields, "radio" buttons, checkboxes

```
<form method="GET" action="search.php">
… <input type="submit" value="go"> … other fields
</form>
```

- When the user clicks on the input element with `type "submit"`, which is displayed as a button, the values of the form's fields are submitted to the URL named by the "`action`" parameter.
- Sometimes you may need to interrupt that normal submit behavior, e.g. to perform validation or some other action using the form fields.

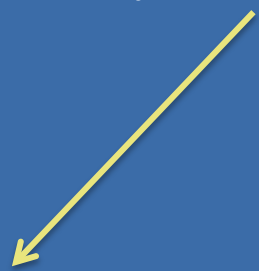
Dealing with forms

```
<form id="f1" method="GET" action="search.php">
… <input type="submit" value="go"> … other fields
</form>

$(function() {
  $("#f1").submit(getActor);
});

function getActor(event) {
   event.preventDefault(); // suspend submit
   /* code to invoke actors.php script to get
      actor names, create <select> menu */
}
```

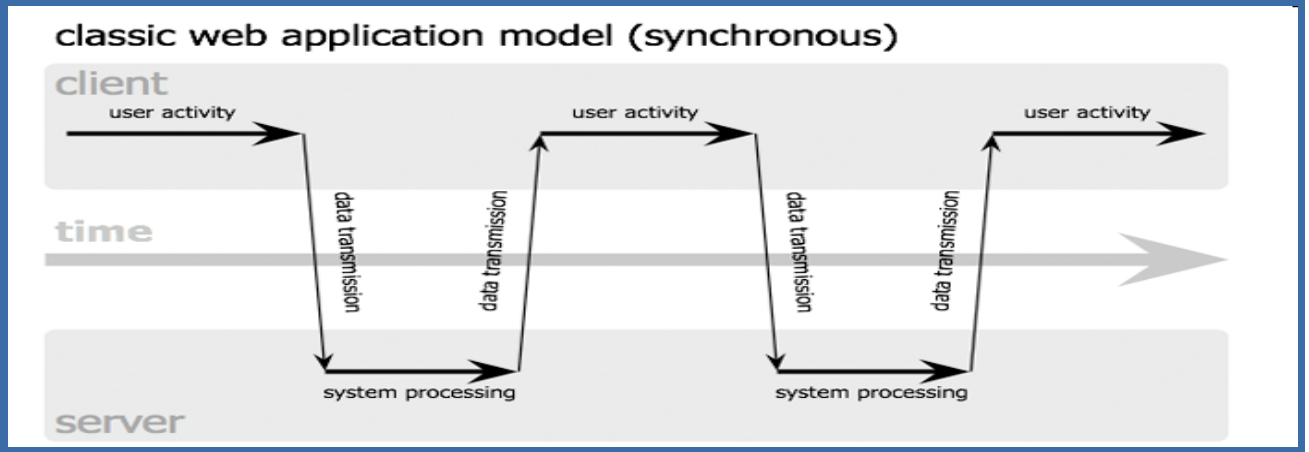
You will need this in your A3



Ajax: Asynchronous Requests

- Traditional Web applications follow the underlying HTTP protocol model of *request and response* page-flow.
- A client Web page makes a request to the server, which results in a new Web page being returned by the server, replacing the original page in the browser.

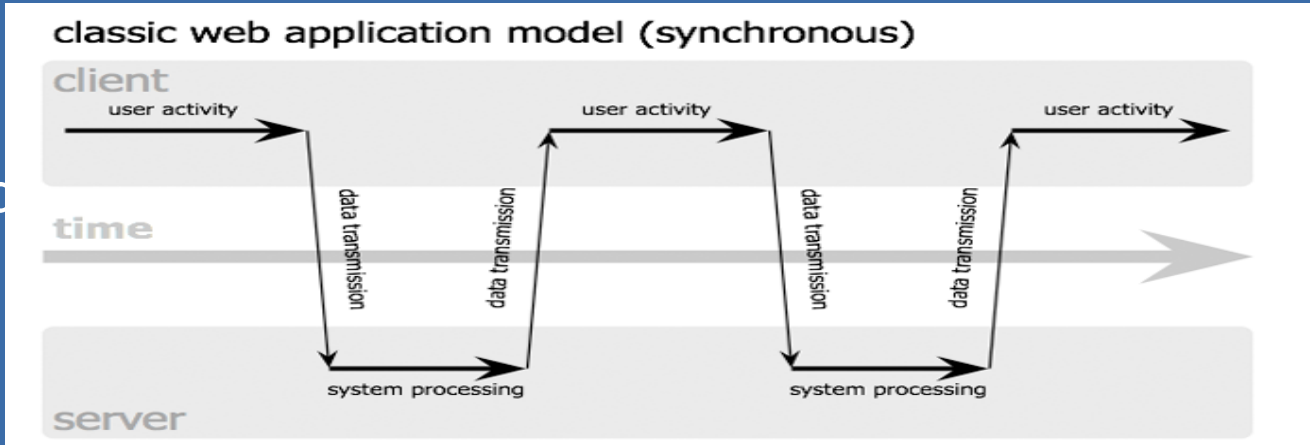
The client blocks while awaiting the server response. We refer to this kind of request as synchronous



Ajax: Asynchronous Requests

- There's nothing wrong with synchronous requests when building Web apps that consist of a sequence of page views (traditional view-click-view-click cycle)
- But if instead you're trying to create a Web app, that does not block (freeze) while waiting for the server, perhaps to provide interactivity, this is a poor model.

While waiting for server response, the client is blocked – no interaction with the user can take place.



Web 2.0, RIA (Rich Internet Apps)

RIA/Web-2.0 apps attempt to bridge gap between "native" (desktop) apps and traditional Web apps

Goals:

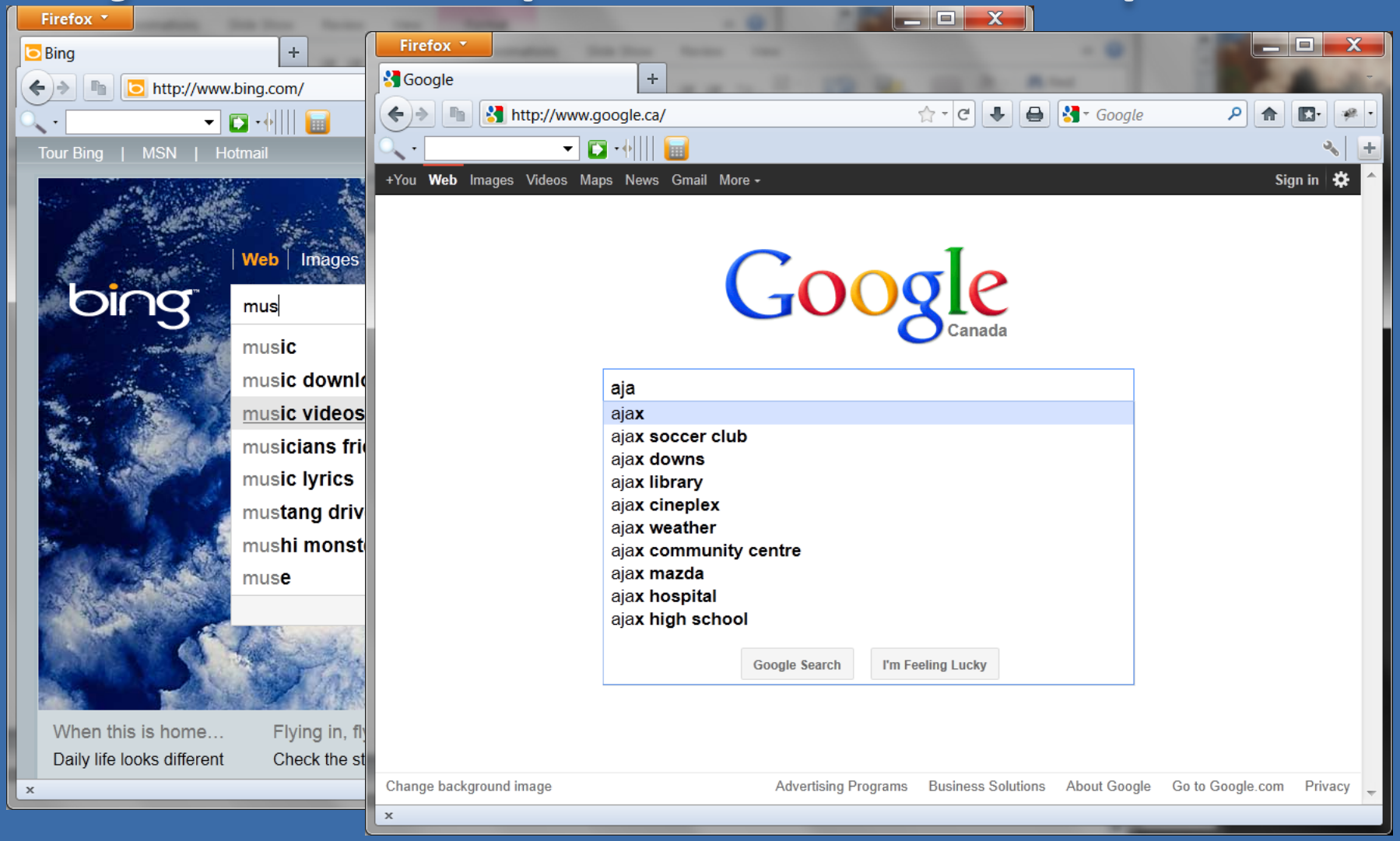
- more interactivity (can achieve some of this with JS and the DOM without server-side interaction)
- would like to achieve performance similar to native (desktop) applications

A major change is a shift from page-oriented interaction to "*event-driven*" programming where events result in updates to *portions* of pages.

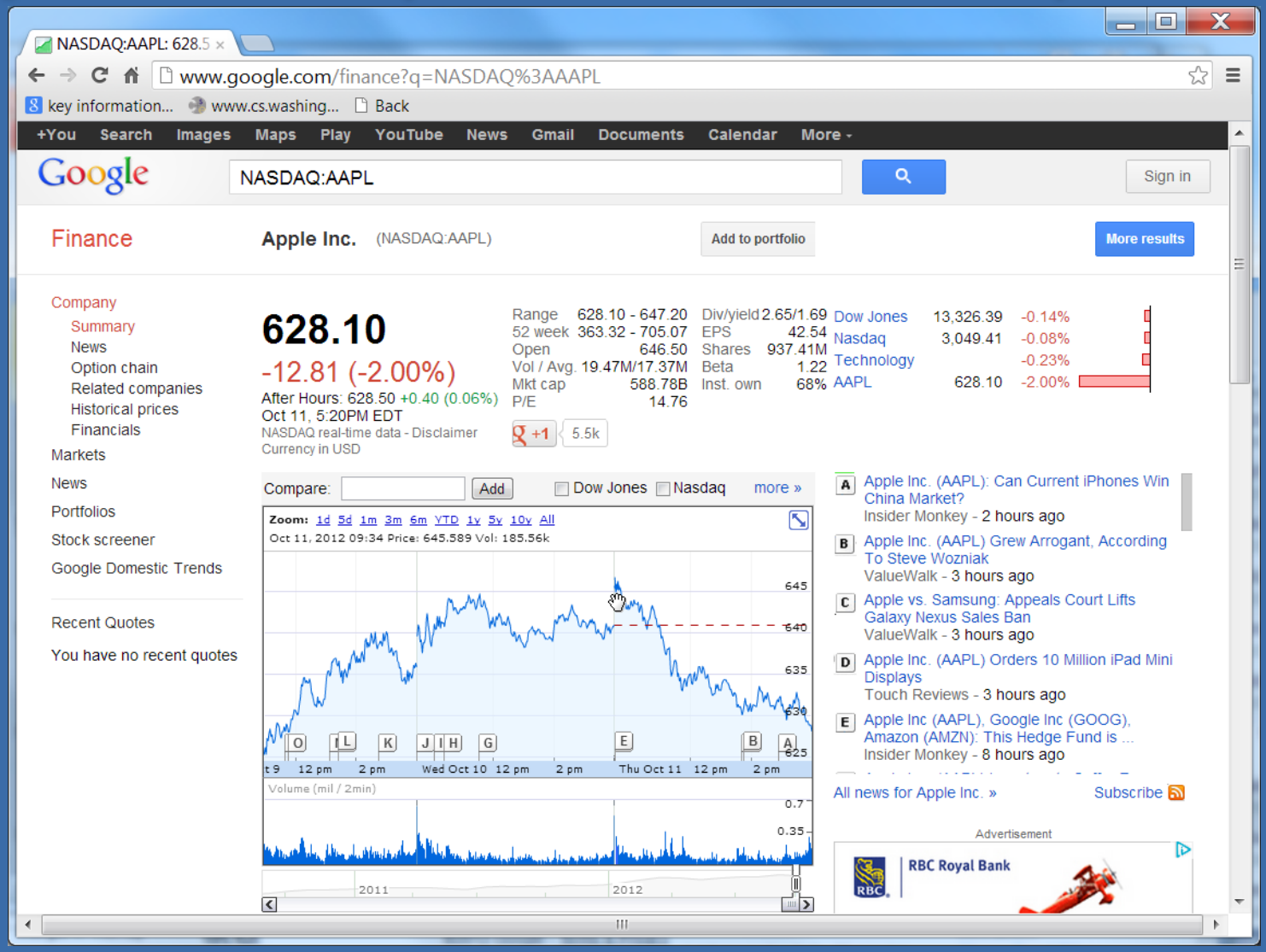
Ajax Use

- Many mainstream apps now utilize Ajax to improve the user experience, e.g. GMail, Google maps, Google finance page, Yahoo search, Flickr
- What is Ajax used for?
  - real-time *semantic form-data validation* (not just regex checks, but actual value checks against e.g. a server DB
  - Auto-completion (e.g. Google suggest, Yahoo search)
  - dynamic page interaction, e.g. user drag/pan results in new data display (e.g. Google maps, Google finance)
  - auto-refresh of page data (e.g., real-time stock quotes embedded within document)
  - interactive user-to-user communications, e.g. Google talk

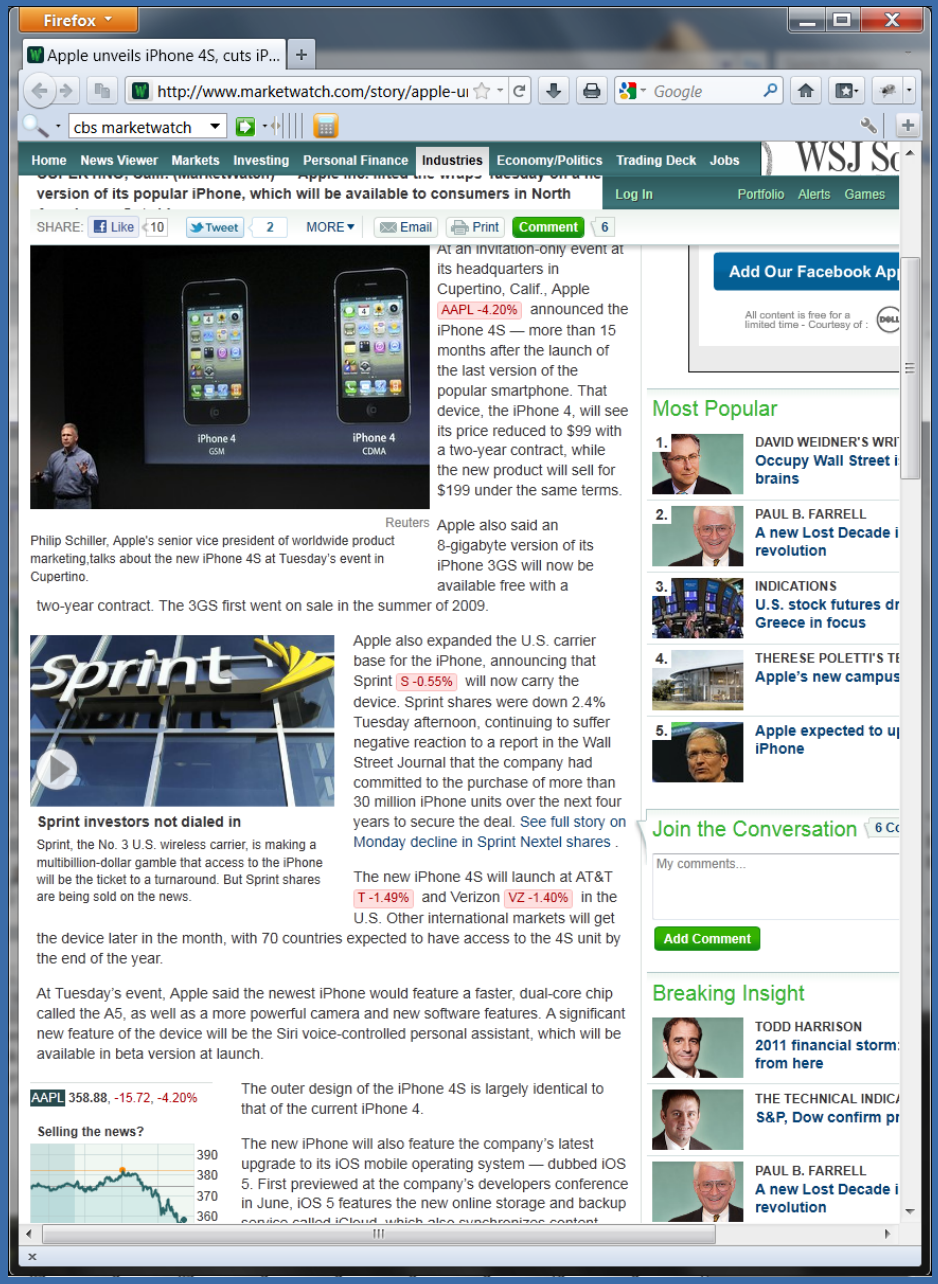
Ajax example: autocomplete



Ajax example: live data updates



Ajax example: live document updates



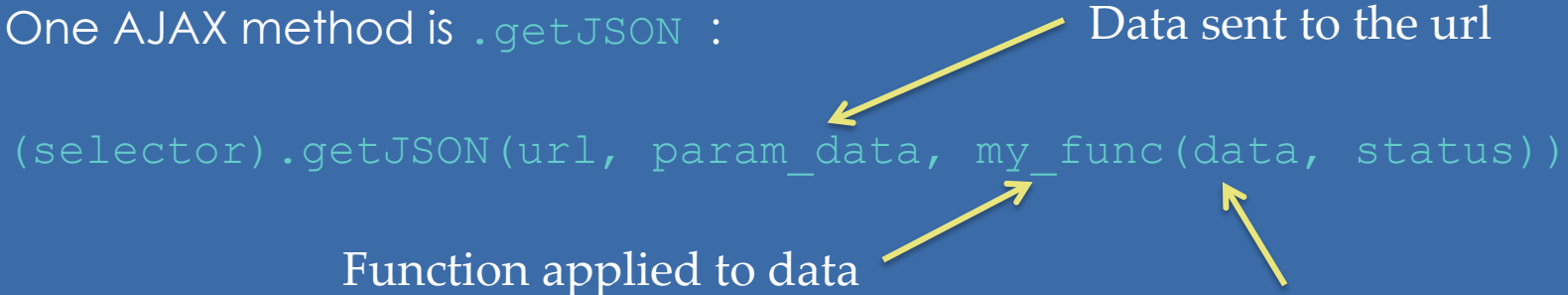
Ajax Interaction

- What's different with Ajax?
- Traditional web
  - user-initiated HTTP requests
    - typing on navigation window, or clicking on form or hyperlink
  - response from server replaces existing page
  - low request rate, and random amount of time between requests
- Ajax application
  - new type of request that does not trigger page reload
  - not initiated by user (at least not directly)
  - requests are typically small, but more frequent

jQuery AJAX

We will only use a few specific *jQuery Ajax* methods related to managing JSON data.

One AJAX method is `.getJSON` :



Example:

```
$(document).ready(function(){
    $("button").click(function(){
        $.getJSON("demo_ajax_json.js", function(result){
            $.each(result, function(i, field){
                $("div").append(field + " ");
            });
        });
    });
});
```

jQuery AJAX

What is `$.each`?

```
$(document).ready(function(){
    $("button").click(function(){
        $.getJSON("demo_ajax_json.js",function(result){
            $.each(result, function(i, field){
                $("div").append(field + " ");
            });
        });
    });
});
```

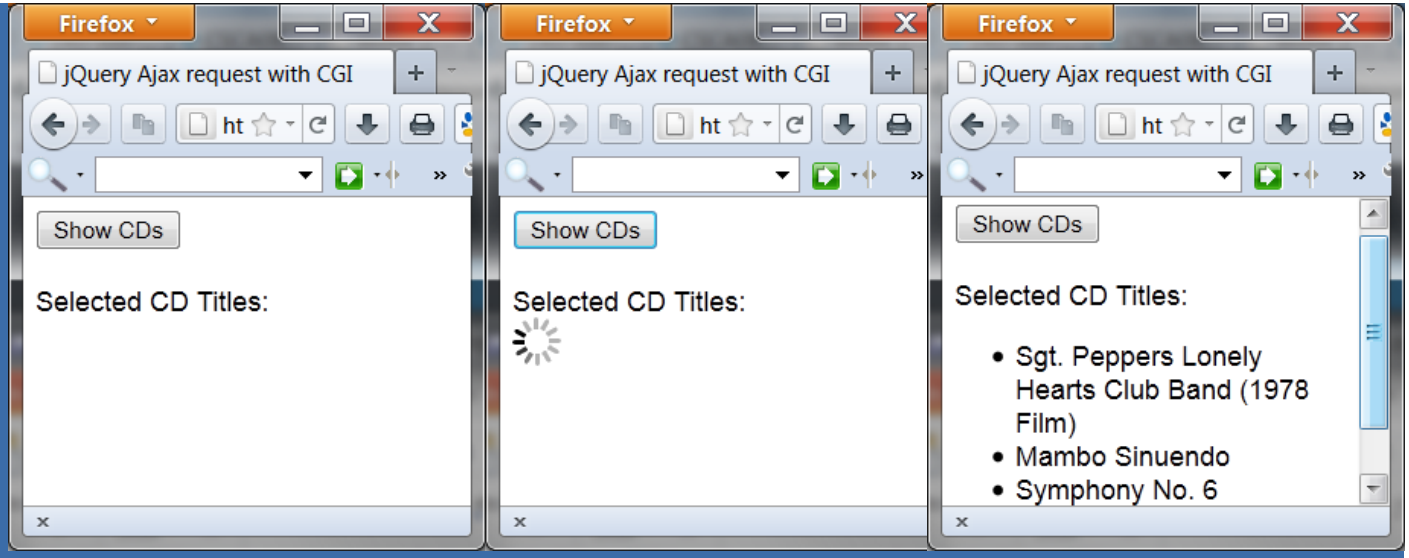
The `$.each(result, func())` method specifies a function to run for each element in `result` .

Another variation is:

```
$(selector).each(function(index, element))
```

Where `function` is applied to each element matching the `selector`.

jQuery : events



- Another task, possibly useful for assignment 3: provide the user with a button, that when clicked will initiate an Ajax request to a server-side PHP service to retrieve JSON data
- We use jQuery's getJSON() method to call the server, retrieve the JSON result, and define a "callback" function to handle this JSON.

jQuery Ajax example

```
    <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1
.8.3/jquery.min.js">
    </script>
    <script type="text/javascript">
      $(document).ready(function(){
         $('.getcds').click(function() {
            $("#cdcat").html('<img src="wait.gif"/>');
            $.getJSON('getCDs.php',
                { 'myparam': 'testing' },   // pass extra param values
                function(data) {   // "callback" function
                    var items = [];   // array to contain HTML <li>'s
                    $.each(data, function(key, val) {
                        items.push('<li id="' + key + '">' + val.title + '</li>');
                    });
                    $('<ul/>', { html: items.join('') }).replaceAll('#cdcat');
                });
         });
      });
    </script>
  </head>
  <body>
    <input type="button" class="getcds" value="Show CDs"/>
    <br/> <br/>
        Selected CD Titles: <br/>
        <div id="cdcat"></div>
  </body>
</html>
```

jQuery Ajax callbacks

- A typical jQuery Ajax callback function binds the Ajax result value to the callback parameter, as in:

```
$.getJSON("… URL …", function(data) { … });
```

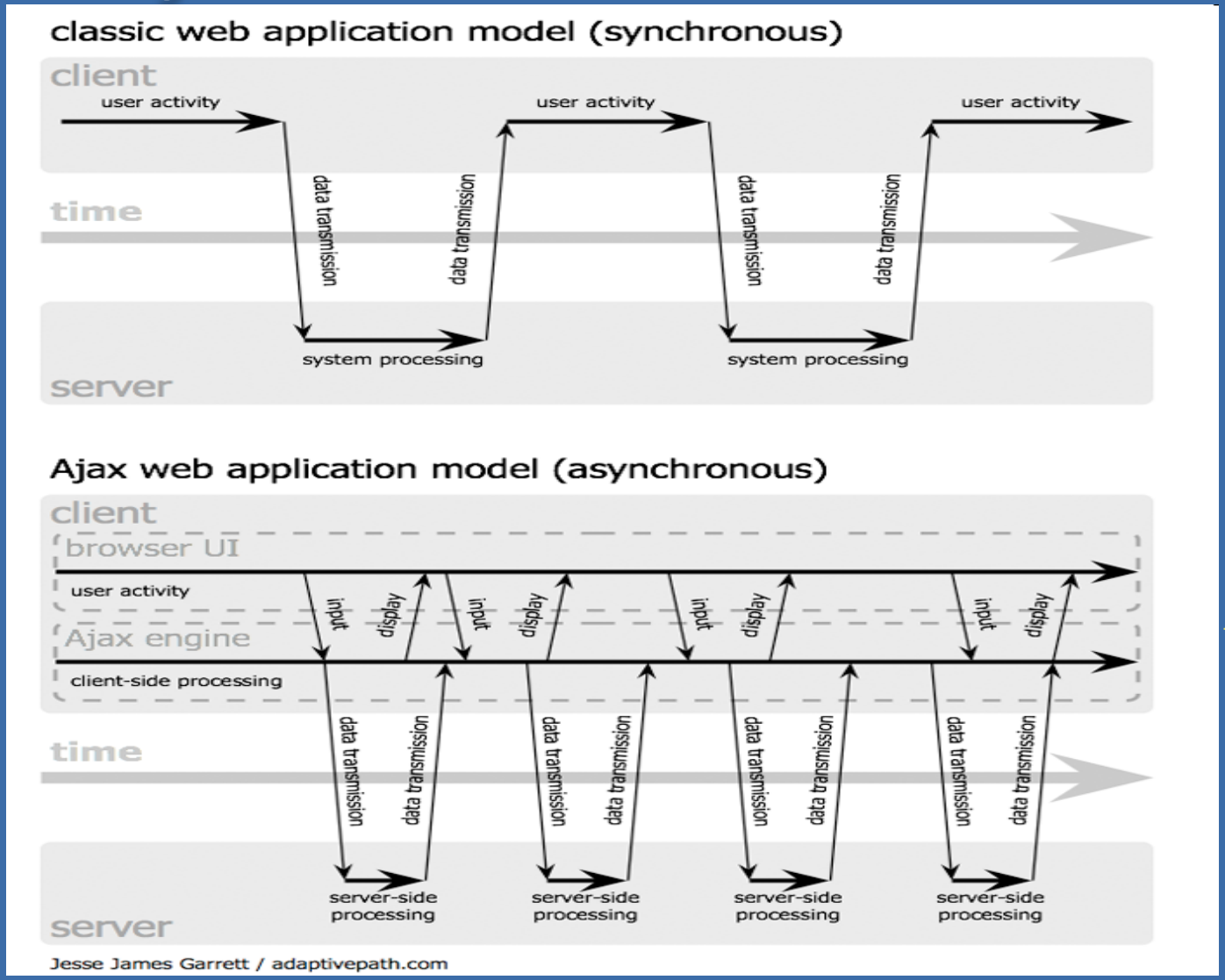
  - Within the anonymous callback function, **`data`** refers to the value returned by the getJSON() Ajax call.
- If the Ajax result is a dictionary or array, you can iterate over the items in this dictionary/array using $.each():

```
$.each(data, function(key, value) { … });  // dict
$.each(data, function(idx, value) { … });  // array
```

- Putting the parts together:

```
$.getJSON("… URL …", function(data) { ...
    $.each(data, function(idx, value) {
     ... // here idx refers to current array index });
  ...
});
```

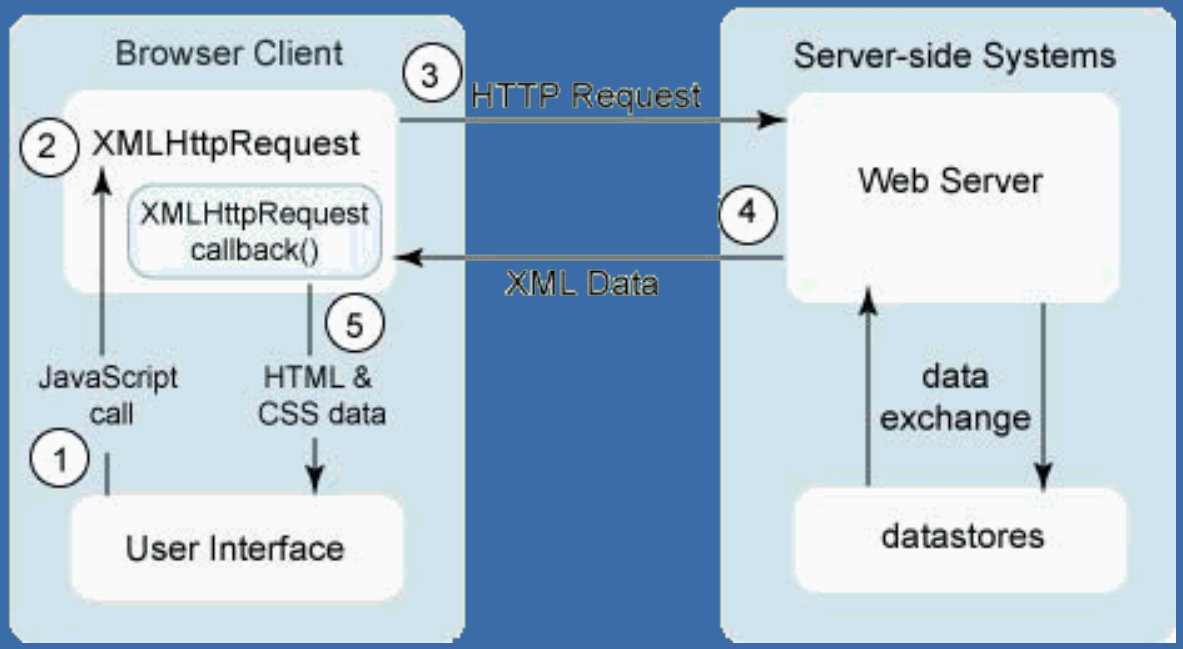
Ajax-Server Interaction



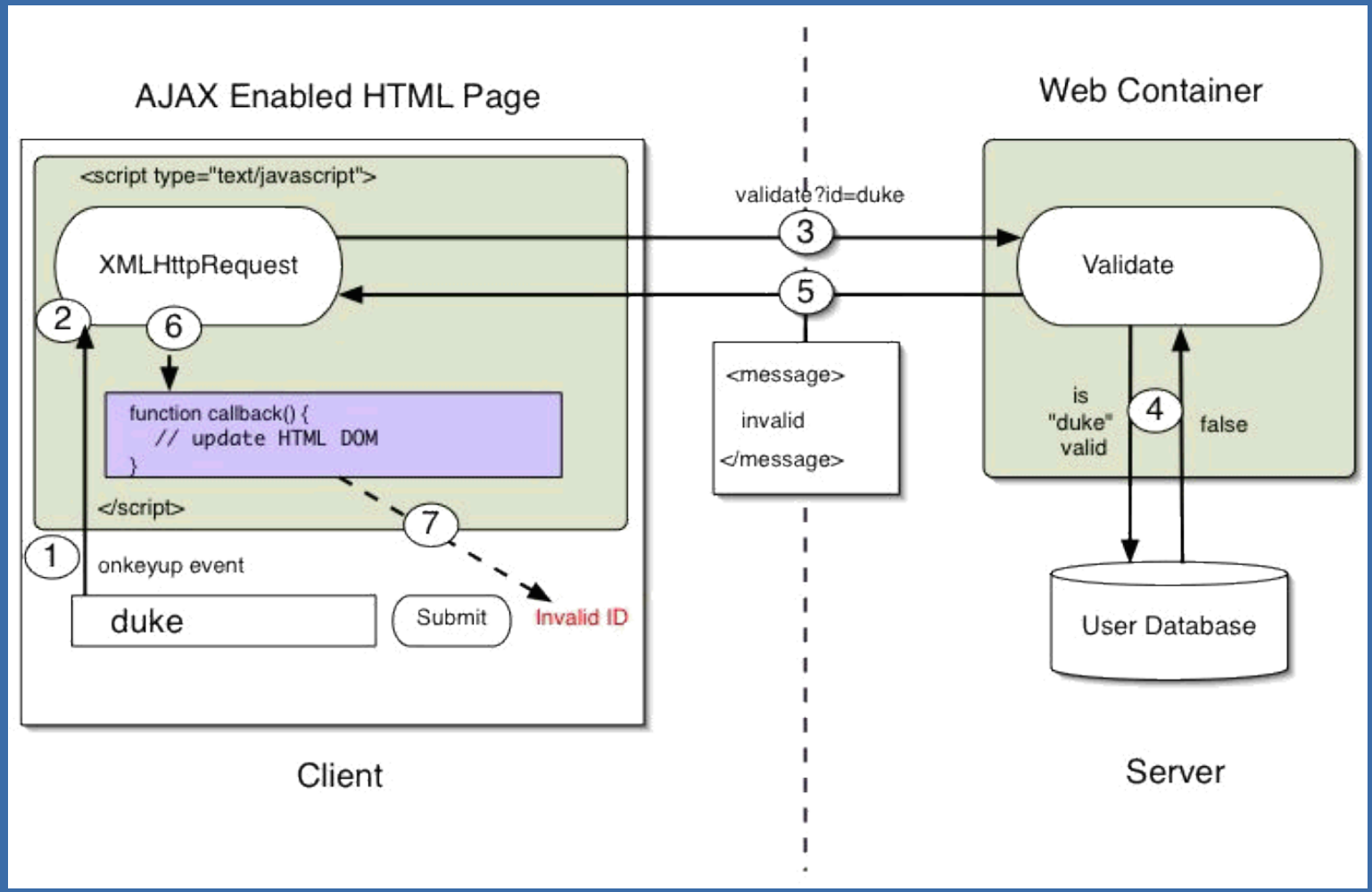
traditional Web app

Ajax-based Web app

Ajax Control Flow



Ajax Control Flow Example



Old slides

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);
   };
}
```

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