



JavaScript & jQuery

Tri Phan

tphan37@csc.com

02-Dec-2015

Introduction



- Your role
- Your background and experience in the subject
 - HTML, CSS, JavaScript
- What do you want from this course

Objectives



- JavaScript programming
 - Basic knowledge
 - Ajax
- jQuery and its features
 - Basic knowledge
 - \$.ajax, Defer and Promise

Prerequisite

- Have knowledge
 - HTML, CSS
 - Editor: VS, Eclipse, Notepad++, Brackets, ...
 - Developer tool: Firebug (Firefox), Chrome developer tool...



Assessment Disciplines & Timetable

- Assessment Disciplines
 - Class Participation : Required
 - Participation in class discussions: 10%
 - Assignment Completion: 40%
 - Mini test: 50%
 - Pass Criteria: $\geq 70\%$
- Course Timetable
 - Lecture Duration: 3 hours
 - Hands-on Labs: 3 hours

Agenda



I.	JavaScript (JS)	08
a.	Client side JS	09
b.	Programming JS	14
c.	JS objects, functions	24
d.	Ajax	31
II.	jQuery	40
a.	Selector	44
b.	Events, Effects	46
c.	DOM Manipulation	50
d.	\$.ajax()	54
e.	Defer and Promise	61



JavaScript Overview

What is JavaScript?

- An interpreted programming language with object-oriented capability.
- Lightweight and runs on the client side and interpreted by browser
- Loosely typed and can be embedded directly into HTML pages
- Everyone can write/run without license
- NOT JAVA

What can client-side JavaScript do?

- Contains an extended set of functionality to interface with the web browser DOM (Document Object Model)
- Event detection and handling
- Support pre-defined Objects (window, document, location etc.
- Manipulate HTML elements
- Make ajax request to server
- Others capabilities, like form validation, cookies storage, etc.



Pros and Cons of JavaScript

- Pros

- Speed: Fast executed on client-side
- Simplicity: Simple to learn and implement
- Versatility: Play nicely with other languages
- ServerLoad: Reduce the demand on web server.

- Cons

- Security: can be exploited for malicious purpose.
- Render Varies: may rendered differently in old browser engines.

Location of JS in html

- JavaScript is put inside HTML between <script>
- <script> tag could be put in any places inside a html page (at <head>, <body>...)
- Can be kept in an external JavaScript file and referenced in HTML code

```
<script>  
document.getElementById("demo").innerHTML = "My First JavaScript";  
</script>
```



Location of JS (example):

/js/script.js

```
alert("This is external JavaScript");
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>JavaScript Location Example</title>
  <script type="text/javascript" src="js/script.js"></script>
  <script type="text/javascript">
    alert("This is inline JavaScript in <head> tag.");
  </script>
</head>
<body>
  <script type="text/javascript">
    alert("This is inline JavaScript in <body> tag.");
  </script>
  <h1>This is a JavaScript Location Example</h1>
</body>
</html>
```



JavaScript™



Programming with Javascript

Variables

- Begin with a letter or \$ or _
- Case sensitive
- There are global scope and function scope
- Declare with var for global and function parameters
- No type declaration

```
//Declare Number variables
var whole = 3;
var decimal = 3.14159265;
//Variable decimal2 implicitly declared
decimal2 = decimal - whole;
//Declare String variables
var single = 'Just single quotes';
var double = "Just double quotes";
var singleEscape = 'He said \'RUN\' ever so softly.';
var doubleEscape = "She said \"hide\" in a loud voice.";
//String operators
var name = "Khanh Nguyen";
var sentence = "My name is " + name;
//Declare Boolean variables
var lying = true;
//Assign new type
whole = "3";
whole = true;
```



JavaScript™

Operators

Operator	Example	Result
>	A > B	true if A is greater than B
>=	A >= B	true if A is greater than or equal to B
<	A < B	true if A is less than B
<=	A <= B	true if A is less than or equal to B
==	A == B	true if A equals B
!=	A != B	true if A does not equal B
!	!A	true if A's Boolean value is false




Conditionals

if... statement

if...else statement

if...else if...else statement

```
if (condition) {  
    conditional code;  
}
```

```
if (condition1) {  
    conditional 1 code execute;  
} else if (condition2){  
    conditional 2 code execute;  
} else {  
    conditional 3 code execute if 2 above  
    not match;  
} 
```

```
if (condition) {  
    conditional code;  
} else {  
    alternative conditional  
    code;  
}
```



Conditionals (cont)

switch statement

```
switch (n)
{
    case 1:
        conditional 1 code execute;
        break;
    case 2:
        conditional 1 code execute;
        break;
    default:
        code execute if n is different from 1 and 2;
}
```

Loops

for and for...in loop

```
for (var i=0; i <= 8; i++) {  
    code;  
}
```

```
var x;  
var txt="";  
var person={fname:"John",lname:"Doe",age:25} :  
for (x in person) {  
    txt=txt + person[x];  
}
```



JavaScript™

Loops (cont)

while and do... while loop

```
while (var i < 8) {  
    code to be executed;  
}
```

```
do {  
    code to be executed;  
} while (var i < 8)
```

Functions

- A function will be executed by an event or by a call to the function.
- A function can be declared nested with child function inside.

```
function function_name(var1, var2,..., varN)
{
    // code to be executed inside function;
}
function function_name(var1, var2,..., varN)
{
    // code to be executed inside function
    function sub_function_name(...) {
        // code to be executed inside functi
    }
}
```



JavaScript™

Events

- Html event can be something browser does, or something user does.
- Syntax:

```
<some-HTML-element some-event="some JavaScript">
```

```
<button onclick="displayDate()">The time is?</button>
```

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

Error handling

- The **try...catch...** statement
 - The **try** block contains the code to be run, and the **catch** block contains the code to be executed if an error occurs.

```
try {  
    // code to be executed and can be  
    generated errors;  
} catch (err) {  
    // Handle errors here  
}
```





Javascript object

Object literals

- You don't declare the types of variables in JavaScript
- Javascript object literal syntax:
 - { name1 : value1 , ... , nameN : valueN }

```
var person= {  
    name: "John",  
    skills: ["Java", "NodeJS"],  
    status: getMaritalStatus()  
}
```


3 ways to create object

- Use object literal

```
var course = { number: "CIT597", teacher="Dr. Dave" }
```

- Use new to create blank object, then add fields to it later

```
var course = new Object();  
course.number = "CIT597";  
course.teacher = "Dr. Dave";
```

- Use a constructor

```
function Course(n, t) {  
    this.number = n;  
    this.teacher = t;  
}  
var course = new Course("CIT597", "Dr. Dave");
```



JavaScript™

Array literals

- No variable type declaration
- Array literal syntax:
 - [value1, ... , valueN]
- Arrays are zero-based

```
var color= ["red", , , "blue", "green"]  
=> Color has 5 elements
```

4 ways to create array

- Use array literals

```
var color = ["red", "green", "blue"];
```

- Use new Array() to create empty array

```
var color = new Array();  
color[0] = "red";
```

- Use new Array(n) with numeric argument to create empty array with that size

```
var color = new Array(3);
```

- Use new Array(...) with initial values

```
var color= new Array("red", "blue", "green")
```



JavaScript™

Arrays and objects

- Arrays are objects.

```
var person = {name: "John", age: 30};
```

So, person.name is the same as person["name"].

- If you know the name of a property, use dot notation

```
person.name
```

- If you don't know the name of property, but you have it in variable, you must use array notation.

```
var prop= "name"; → person[prop]
```



Array functions

Functions	Description
<code>sort()</code>	Sort the array alphabetically
<code>reverse()</code>	Reverse the array elements
<code>push(object)</code>	Add any number of new elements to the end of the array, and increase the array's length
<code>pop()</code>	Remove and return the last element of the array, and decrease the array's length
<code>toString()</code>	Return a string containing the values of the array elements, separated by commas.





Ajax

Problem with traditional web?

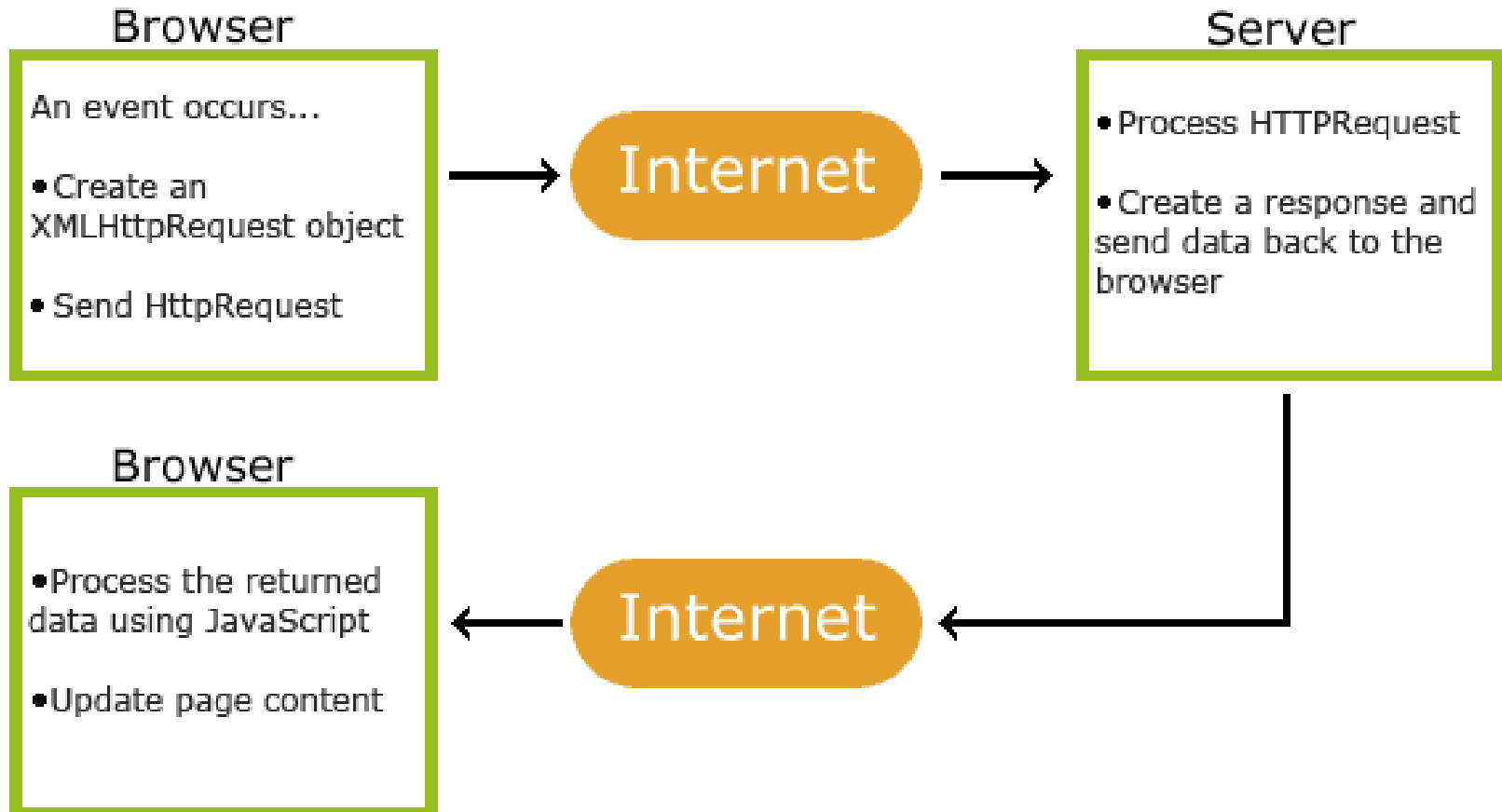
- In traditional web, to get information from server:
 - Make html form
 - GET or POST data to the server (like Submit button)
 - The browser loads a result page.
- Traditional web app run slowly and tend to be less user friendly.

Why Ajax come in?

- AJAX = Asynchronous JavaScript and XML
- Ajax does not require the page to be reloaded.
 - Use XmlHttpRequest object
 - The scripts requested to the server are executed asynchronously in the background.
- Ajax call is asynchronous,
 - The browser is not locked up.
 - The page is reloaded partially.



How Ajax works?



Ajax – send request



- Create XMLHttpRequest object

```
variable = new XMLHttpRequest();
```

- Send request to server

Method	Description
<code>open(method, url, async)</code>	Specifies the type of request <i>method</i> : the type of request: GET or POST <i>url</i> : the server (file) location <i>async</i> : true (asynchronous) or false (synchronous)
<code>send()</code>	Sends the request to the server (used for GET)
<code>send(string)</code>	Sends the request to the server (used for POST)

Ajax – get or post ?



- Get is simpler and faster than POST

```
xhttp.open("GET", "demo_get.asp", true);  
xhttp.send();
```

- Use POST when

- Send large amount of data to server (no size limit)
- Sending user input (robust and secure than GET)

```
xhttp.open("POST", "ajax_test.asp", true);  
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
xhttp.send("fname=Henry&lname=Ford");
```

Ajax – server response



- Use `responseText` or `responseXML` property of `XMLHttpRequest` object.

Property	Description
<code>responseText</code>	get the response data as a string
<code>responseXML</code>	get the response data as XML data

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

Ajax – event & callback



- When a request to server is sent, perform some actions based on the response.

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

Ajax – example



```
<div id="demo"><h2>Let AJAX change this text</h2></div>
```

```
<button type="button" onclick="loadDoc()">Change Content</button>
```

```
function loadDoc() {  
    var xhttp = new XMLHttpRequest();  
    xhttp.onreadystatechange = function() {  
        if (xhttp.readyState == 4 && xhttp.status == 200) {  
            document.getElementById("demo").innerHTML = xhttp.responseText;  
        }  
    }  
    xhttp.open("GET", "ajax_info.txt", true);  
    xhttp.send();  
}
```



Jquery

What is jQuery?

- jQuery is a fast and concise JavaScript Library
- Features:
 - that simplifies DOM traversing, event handling, animating, and Ajax interactions for rapid web development.
- jQuery is designed to change the way that you write JavaScript
- References:
 - api.jquery.com/jquery.ajax



jQuery features

- jQuery is a library of JavaScript functions
- Write less, do more
- Awesome features:
 - HTML element selections
 - HTML element manipulation
 - CSS manipulation
 - HTML event functions
 - JavaScript Effects and animations
 - HTML DOM traversal and modification
 - AJAX



Jquery – How to add?

- The jQuery library is stored as a single JavaScript file, containing all the jQuery methods.
- It can be added to a web page with the following mark-up:

```
<head>
```

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

```
</head>
```



jQuery – how to use? (cont)

- Basic syntax is: `$(selector).action()`
 - A dollar sign (\$) to define jQuery
 - A (selector) to query (or find) elements
 - A jQuery action() to be performed on the element(s)

```
$ ("p") .hide ()
```

- Entry point `ready()` function

```
$(document) .ready ( function () {  
    // Handler for .ready() called.  
} ) ;
```



Jquery – selector

- Start with `$()`

Selectors	Descriptions
<code>\$("*")</code>	selects all elements.
<code>\$("p")</code>	selects all <code><p></code> elements.
<code>\$("p.intro")</code>	selects all <code><p></code> elements with <code>class="intro"</code> .
<code>\$("p#intro")</code>	selects the first <code><p></code> elements with <code>id="intro"</code> .
<code>\$(":button")</code>	selects all <code><button></code> and <code><input></code> elements of <code>type="button"</code> .
<code>\$(":even")</code>	selects even elements
<code>\$(":odd")</code>	selects odd elements.

Jquery – event

- The jQuery event handling methods are core functions in jQuery.
- Event handlers are methods that are called when something happens in HTML.
- The term triggered (or fired) by an event is often used.



Jquery – event (example)

- jQuery Submit is triggered whenever a visitor submits a form.

```
$("form").submit(function() {})
```

- jQuery Click is triggered when you click and release the mouse button

```
$("#id1").click(function() {})
```

- jQuery Mouseup and Mousedown Event

```
$("#id1").mousedown(function() {})
```



Jquery – effect

- Hide() and show()
 - `$(selector).hide(speed, callback)`
 - `$(selector).show(speed, callback)`
- The toggle()
 - `$(selector).toggle(speed, callback)`
- Slide methods (slideDown, slideUp, slideToggle)
 - `$(selector).slideUp(speed, callback)`
 - `$(selector).slideDown(speed, callback)`
 - `$(selector).slideToggle(speed, callback)`



Jquery – effect (cont)

- Fade methods (fadeIn, fadeOut, fadeTo).
 - `$(selector).fadeIn(speed,callback)`
 - `$(selector).fadeOut(speed,callback)`
 - `$(selector).fadeTo(speed,opacity,callback)`
- Custom animate() function.
 - `$(selector).animate({params},[duration],[easing],[callback])`



jQuery – html manipulation

- Changing HTML content of an element
 - `$(selector).html(content)`: set the HTML contents of each element in the set of matched elements.
- Add contents to an element
 - `$(selector).append(content)`: inserts specified content at the end of (but still inside) the selected elements.
 - `$(selector).prepend(content)`: inserts specified content at the beginning of (but still inside) the selected elements.



jQuery – html manipulation (cont)

- More methods for HTML manipulation:
 - `$(selector).before(content)`
 - `$(selector).after(content)`
 - `$(content).insertAfter(selector)`
 - `$(content).insertBefore(selector)`
 - `$(selector).attr(attribute)`
 - `$(selector).attr(attribute,value)`
 - `$(selector).remove()`
 - `$(selector).removeAttr(attribute)`



jQuery – css manipulation

- CSS Manipulation
 - `$(selector).css(name)`
 - `$(selector).css(name,value)`
- Size manipulation
 - `$(selector).width()` and `$(selector).width(value)`
 - `$(selector).height()` and `$(selector).height(value)`
- Add/remove a css class
 - `$(selector).addClass(class)`
 - `$(selector).removeClass(class)`
- Check existing css class
 - `$(selector).hasClass(class)`



jQuery – css manipulation (cont)

- More methods for CSS manipulation:
 - `$(selector).scrollLeft()`
 - `$(selector).scrollLeft(position)`
 - `$(selector).scrollTop()`
 - `$(selector).scrollTop(position)`
 - `$(selector).position()`
 - `$(selector).offset()`
 - `$(selector).offset(value)`
 - `$(selector).toggleClass(class)`





Jquery Ajax

jQuery ajax - introduction

- Perform an AJAX (asynchronous HTTP) request.
- Methods: GET / POST / PUT / DELETE
- Syntax:

```
$.ajax({name:value, name:value, ... })
```

- The parameters specifies one or more name/value pairs for the AJAX request.



\$.ajax options : URL

- Address of the server-side resource
- Can contain query string

```
$.ajax({  
  url: '/url/to/serverResource'  
});
```

- Can be passed as a string to first argument

```
$.ajax('/url/to/serverResource');
```



\$.ajax options : data

- To send information from client to server
 - with GET: Appended to url as query string
 - with POST: Sent as POST body

```
$.ajax({  
  url: '/url/to/serverResource',  
  data: {  
    key1: 'value1',  
    key2: 'value2'  
  }  
});
```



\$.ajax options : data (cont)

- When submitting a form, use .serialize()

```
$( '#myform' ).submit( function( event ) {  
    event.preventDefault();  
  
    var formUrl = $(this).attr('action'),  
        formData = $(this).serialize();  
  
    $.ajax({  
        url: formUrl,  
        data: formData  
    });  
});
```



\$.ajax options : datatype

- Specifies expected response
- Default based on MIME Type of the response
- Available data type: html, xml, json , jsonp.

```
$.ajax({  
  url: '/url/to/serverResource',  
  dataType: 'json'  
});
```



\$.ajax options : more...

- Read all about 'em at
 - api.jquery.com/jQuery.ajax/



\$.ajax responses

- Before jQuery 1.5, it was handled by 3 more options.
 - { success: function() {}, error: function() {}, complete: function() {} }
- Can not register multiple callbacks

```
$.ajax({  
  url: '/url/to/serverResource',  
  success: function(response, status, xhr) {  
    // do something after successful request  
  },  
  error: function(xhr, status, errorThrown) {  
    // handle failed request  
  },  
  complete: function(xhr, status) {  
    // do something whether success or error  
  }  
});
```

\$.ajax responses (cont)

- \$.ajax implements the Promise interface
 - returns a jqXHR promise object (superset of xhr)
 - Promise objects are derived from the Deferred object
 - Read api.jquery.com/category/deferred-object/
- jQuery 1.5+:
 - .done() and .fail() methods
- jQuery 1.6+:
 - .always() method



\$.deferred

- 3 important methods
 - .resolve()
 - .reject()
 - .promise()
- 3 important events to attach a callback
 - .done()
 - .fail()
 - .always()



\$.deferred (cont)

- Once `resolve()` is called, the `.done` callback attached will be executed.
- Once `reject()` is called, the `.fail` callback attached will be executed.
- The `.always()` is executed whether the deferred is resolved or rejected.

```
var deferred = $.Deferred();

deferred.resolve("hello world");

deferred.done(function(value) {
    alert(value);
});
```



.promise() method

- Another important method of deferred object.
- Returns an object with almost the same interface than the Deferred, but
 - it only has then method to attach callbacks
 - it does not have the methods to resolve and reject.
- The \$.ajax method in JQuery returns a Promise
- Can add as many callbacks as you want



\$.ajax responses (1.5+)

- Methods can be called multiple times to add more than one handler.
- Can store result of Ajax request in variable and attach handlers later for more manageable code structure.
- Handlers will be invoked immediately if the Ajax operation is already completed.



\$.ajax responses (1.5+)

```
var myOptions = {  
  url: 'http://api.jquery.com/jsonp/',  
  dataType: 'jsonp',  
  data: {  
    title: search  
  }  
};
```

```
$.ajax( myOptions )  
  .done( successFn )  
  .fail( errorFn )  
  .always( completeFn );
```



\$.ajax responses (1.5+)

- Multiple function arguments
- Array of functions
- Multiple, chained .done() methods

```
request.done (successFnA, successFnB, successFnC) ;  
request.done ([successFnD, successFnE, successFnF]) ;  
request.done (successFnG) .done (successFnH) .done (successFnJ) ;
```



\$.ajax convenience methods

- \$.get()
- \$.post()
- \$.getJSON()
- \$.getScript()
- My recommendation:
Use **\$.ajax()**





Q&A



Thank you



References

Revision History

Date	Version	Description	Updated by	Reviewed and Approved By
15.12.01	1.0.0	Change styles follow new CSC theme	Tri Phan	



BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING