COMP 4021
Internet Computing

jQuery Introduction

Dik Lun Lee

jQuery JavaScript Library

- You all know how to use JavaScript to search particular elements in DOM and manipulate the DOM
- Is it an easy task? Most people say, NO …
- JQuery is a JavaScript library (http://jquery.com), also called a web tool kit, for DOM manipulation, event handling, client-server interaction
- Most popular JavaScript library in use today
  - Other toolkits: Yahoo UI Library (YUI), Google Web Toolkit, etc.

What jQuery Does

- Select DOM elements using CSS-like selectors
- Set properties of selected DOM elements
- Create, delete, show, hide DOM elements
- Defines event behavior (click, mouse movement, dynamic styles, animations, dynamic content)
- AJAX calls

While CSS separates style from structure, JQuery separates behavior from structure

jQuery Ready Function

- Execute a function as soon as a page is fully loaded

```
$(document).ready(function() {
  // binds a click event to all <A> elements
    $("a").click(function() {
    alert("Click on Link!");
  });
 });
```

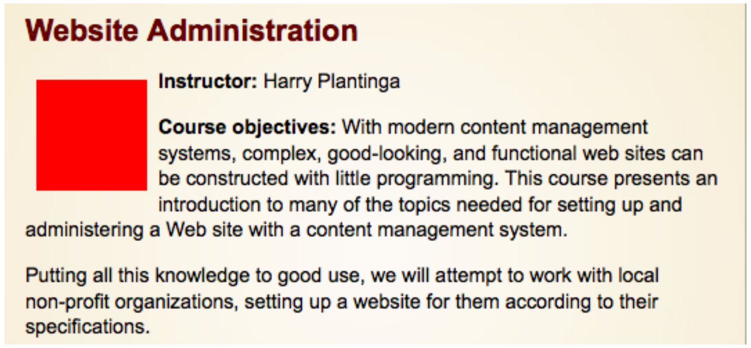
What is the problem of executing a function before a page is fully loaded?)

jQuery Example

Try to read this code even if do not know jQuery

http://cs.calvin.edu/curriculum/is/337/hplantin/examples/jquery3.html

```
$(document).ready(function() {
  $("#greenbox").click(function() {
    $("#greenbox").hide();
    $("#redbox").show();
  });
  $("#redbox").click(function() {
    $("#redbox").hide();
    $("#greenbox").show();
  });
});
```

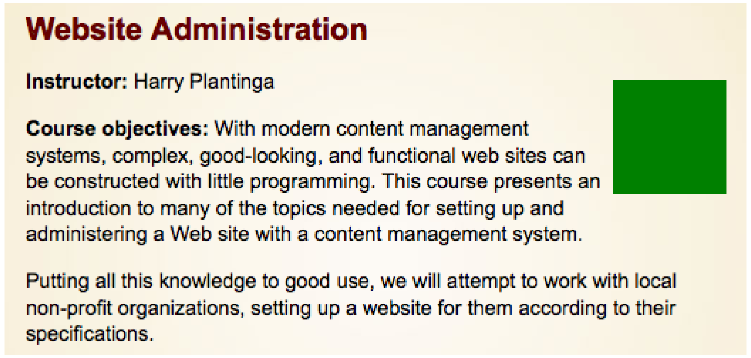


Website Administration

Instructor: Harry Plantinga

Course objectives: With modern content management systems, complex, good-looking, and functional web sites can be constructed with little programming. This course presents an introduction to many of the topics needed for setting up and administering a Web site with a content management system.

Putting all this knowledge to good use, we will attempt to work with local non-profit organizations, setting up a website for them according to their specifications.



Website Administration

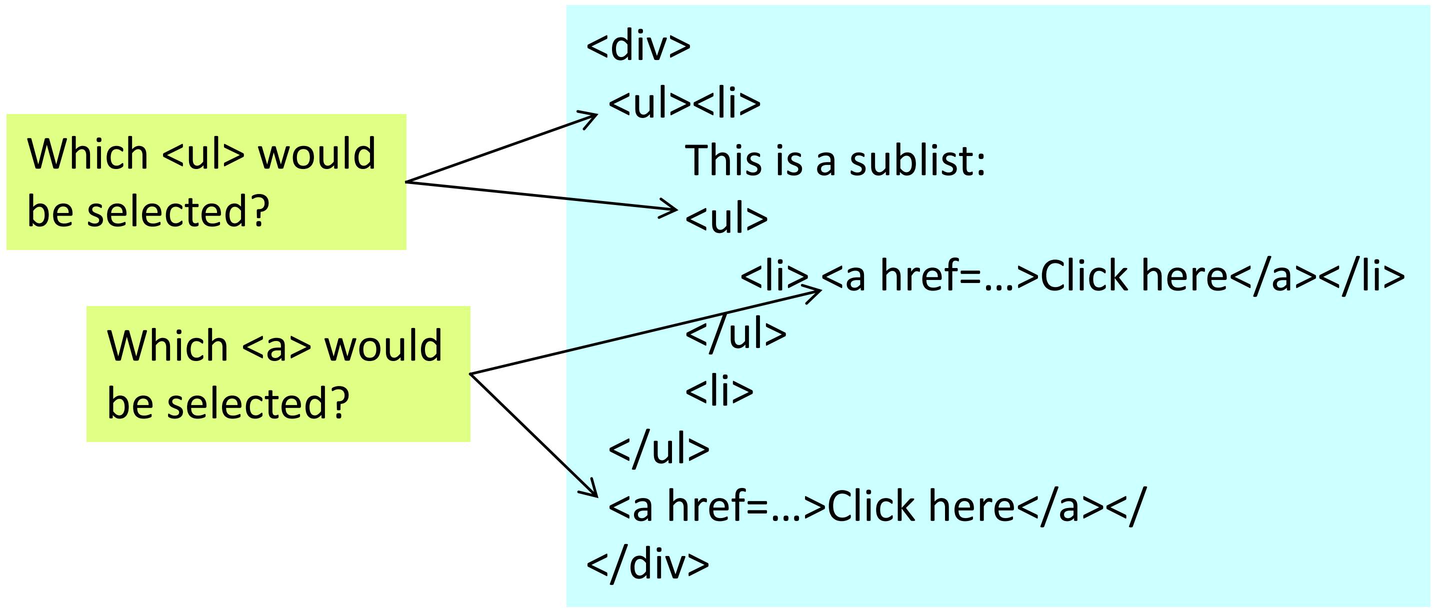
Instructor: Harry Plantinga

Course objectives: With modern content management systems, complex, good-looking, and functional web sites can be constructed with little programming. This course presents an introduction to many of the topics needed for setting up and administering a Web site with a content management system.

Putting all this knowledge to good use, we will attempt to work with local non-profit organizations, setting up a website for them according to their specifications.

Some Selectors

- **A > B** means B must be a direct child of A
- **A B** means B must be a descendant of A
- **$("div > ul a")** reads: All <a> elements which are descendants of <ul> elements which are direct children of <div> elements



```
<div>
  <ul><li>
        This is a sublist:
        <ul>
            <li><a href=…>Click here</a></li>
        </ul>
        <li>
  </ul>
  <a href=…>Click here</a></
</div>
```

Select and Action Example

- Select all <li> elements within element with ID=orderedlist, and add the class "blue" (defined in CSS)

```
$(document).ready(function() {
  $("#orderedlist > li").addClass("blue");
 });
```

```
<ol id="orderedlist">
  <li class="blue">… </li>
```

```
Likewise: $("#orderedlist > li").removeClass("blue");
```

```
<ol id="orderedlist">
  <li >… </li>
```

For another example see: https://jsfiddle.net/qb2n7h5L/2/

Some Useful Selectors

$('#id')	id of element
$('p')	tag name
$('.class')	CSS class
$('p.class')	<p> elements having the CSS class
$('p:first')	$('p:last') / $('p:odd') / $('p:even')
$('p')[1]	gets the 2nd <p> element (0 based)
$('p a')	<a> elements, descended from a <p>
$('p>a')	<a> elements, direct child of a <p>
$('p+a')	<a> elements, directly following a <p>
$('p, a')	<p> and <a> elements
$('li:has(ul)')	<li> elements that have at least one <ul> descendent
$(':not(p)')	all elements but <p> elements
$('p:hidden')	only <p> elements that are hidden
$('p:empty')	<p> elements that have no child elements

Some Useful jQuery Functions

.each()	iterate over the set
.size()	number of elements in set
.get(n)	get just the nth element (0 based)
.eq(n)	get just the nth element (0 based) also .lt(n) & .gt(n)
.not('p')	don't include 'p' elements in set
.add('p')	add <p> elements to set
.remove()	removes all the elements from the page DOM
.empty()	removes the contents of all the elements
.filter(fn/sel)	selects elements where the func returns true or sel
.find(selector)	selects elements meeting the selector criteria
.parent()	returns the parent of each element in set
.children()	returns all the children of each element in set
.next()	gets next element of each element in set
.prev()	gets previous element of each element in set
.siblings()	gets all the siblings of the current element

Add Page Elements

- $(‘#target’).before(‘<p>Inserted before #target</p>’);
- $(‘#target’).after(‘<p>This is added after #target</p>’);
- $(‘#target’).append(‘<p>Goes inside #target, at end</p>’);
- $(‘#target’).wrap(‘<div></div>’);

Adding Events

- For every JavaScript event, like onclick, onchange, onsubmit, there is a jQuery equivalent
  - Mouseover events – bind, hover, toggle
  - Button click events
  - Keystrokes

Event Binding

- $(‘img’).bind(‘click’,function(event){alert(‘Howdy’;});
- $(‘img’).bind(‘click’,imgclick(event));
- $(‘img’).unbind(‘click’,imgclick());
- $(‘img’).unbind(‘click’);

- $(‘img’).one(‘click’,imgclick(event)); // event handling function is run only once

- $(‘img’).click(imgclick);
- $(‘img’).toggle(click1, click2);
- $(‘img’).hover(mouseover, mouseout); // functions to call when mouse enters and leaves the image

‘Event’ properties

event.target	ref to element triggering event
Event.target.id	id of element triggering event
event.currentTarget	
event.type	type of event triggered
event.data	second parm in the bind() func
Various mouse coordinate properties	
Various keystroke related properties	

Shortcut Event Binding

- .click(func)
- .submit(func)
- .dblclick(func)
- .mouseover(func)
- .mouseout(func)
- .select(func)

Useful Event Functions

.hide()	display:true
.show()	display:none
.toggle(func1, func2)	first click calls func1, next click executes func2
.hover(over, out)	mouseover, mouseout

AJAX

- What is AJAX
- The basic AJAX function – XMLHttpRequest
- Initiating a request
- Getting the response

AJAX Call

- Sending GET Ajax request:

```
jQuery.get(
   url
   [, data]
   [, success(data, textStatus, jqXHR)]
   [, dataType] )
```

- **url:** a string containing the URL of called program
- **data:** a map or string sent to the server
- **success(data, textStatus, jqXHR):** callback function
- **dataType:** type of data expected from the server

Loading Content

- Load a file from server into a div:

```
$("div").load("content.htm");
```

- Invoke a server program to return data from server:

```
// passing parameters to server program
  $("#content").load("getcontent.php",
                 {d:0123} );
```

Parameter passed to server

Sending GET/POST requests

```
$.get("test.php", {id:1},
        function(data){alert(data);});

$.post("test.php", {id:1},
        function(data){alert(data);});
```

Parameter passed to server

- Similar to .load() function but with callback function

Introducing JSON Data

- JSON: JavaScript Object Notation
  - A standard to represent object data for passing between applications (e.g., client programs and server programs)
  - How to pass an array from a PHP program to a JS program on browser?

Name/Value	`"Id" : "0123"`
Object	`{ "Id " : "0123" ,`
	`  "Name" : "Lee"`
	`}`
Array (of	`[ "Id" : "0123",`
name/value	`  "Id" : "1123",`
pairs)	`  "Id" : "2123"`
	`]`

Array of	`[ { "Id" : "0123",`
objects	`      "Name" : "Lee", },`
	`  { "Id" : "1123",`
	`      "Name" : "Chan" }`
	`]`
Object	`{ "Id " : "0123" ,`
values	`  "Name" : {`
	`      "fname" : "Dik",`
	`      "lname" : "Lee" }`
	`}`

Retrieving JSON Data

-
  ```
  $.getJSON(URL, [data string sent to server],
                 [callback function(result){…}] )
  ```

  ```
  $.getJSON("users.php", {id:1},
            function(users) {
                  alert(users[0].Name);
            });
  ```

Suppose users is an array of objects

Processing Data with .each()

```
$("div").each(function(index, value) {
    $(value).append(" div " + index);
});
```

- .each() loops through all matching div elements and executes the callback function on each element
- This example appends a message to each div; see https://jsfiddle.net/jfkLq54t/

Using .each() in jQuery

```
var arr = [ "one", "two", "three", "four", "five" ];
jquery.each(arr, function(index, value) {
    alert(index + " : " + value);
 });
```

- .each() loops through all array element and executes the callback function on each element
- See https://jsfiddle.net/jLqnwLe3//

Processing JSON Data with .each()

-
  ```
  $.getJSON(“users.php", function(users){
    $.each(users, function(index, value){
        $("div").append(value.Name + “ “);
    });
  });
  ```

Returns users as an array of objects:
[{ "Id" : "0123",
"Name" : "Lee", },
{ "Id" : "1123",
"Name" : "Chan" }]

Returns users as an object:
{ "Id " : "0123" , "Name" : "Lee“ }

-
  ```
  $.getJSON(“users.php", function(users){
    $.each(users, function(key, value){
       $("div").append(“<tr><td>” + key + “</td>” +
                       “<td>” + value + “/td></tr>” );
    });
  });
  ```

Take Home Message

- JQuery is a popular JavaScript library
- jQuery implements many popular interaction functions (e.g., autocomplete, which will be discussed next)
- Provide a CSS-like selectors to specific elements to which actions are applied (compared to navigating DOM using JavaScript only)
- Convenient event handling and Ajax functions