Client/Server Programming for Internet Applications

TCSS460

Summer 2020

Lecture 7
jQuery (Part II)

TACOMA

Common Element Manipulations in jQuery
→ HTML Attributes *(review)*

- in jQuery, we can both set and get an **attribute** value by using the `attr()` method on any element returned from a selector
  - two parameters
    - **first**: attribute name [required]
    - **second**: value for modifying the attribute [optional]
  - if no second parameter is passed, method returns current value of attribute

```
<script>
    // link is assigned the href attribute of the first <a> tag
    var link = $("a").attr("href");
    // change all links in the page to http://uw.edu
    $("a").attr("href", "http://uw.edu");
    // change the class for all images on the page to fancy
    $("img").attr("class ", "fancy");
</script>
```

2

Common Element Manipulations in jQuery → HTML Properties *(review)*

- many HTML tags include **properties** as well as **attributes**
  - e.g. checked property of a radio button or checkbox
  - early versions of jQuery, HTML properties could be set using attr() method
  - the `prop()` method is now the **preferred** way to **retrieve** and **set** the **value** of a **property** although, `attr()` may return some (less useful) values

```
<input class="meh" type="checkbox" checked="checked">
...
<script>
    var theBox = $(".meh");
    theBox.prop("checked"); // evaluates to TRUE
    theBox.attr("checked"); // evaluates to "checked"
</script>
```

3

Common Element Manipulations in jQuery → Changing CSS *(review)*

- jQuery provides the extremely intuitive `css()` method
- two versions of this method (two different signatures):
  - **first**: retrieve value (single parameter containing CSS attribute)

```
// get the color of HTML tag with id element
var color = $("#element").css("background-color");
```

  - **second**: modify existing value

```
// set color to red
$("#element").css("background-color", "red");
```

if you test this in a browser and then inspect it, you will notice that jQuery modifies the element's style attribute

- e.g., `<div id="element" style="background-color: red">)`

4

Common Element Manipulations in jQuery
→ Changing CSS (manipulating classes)

- you can programmatically set CSS classes instead of overriding a particular CSS attribute individually
- this can be achieved using the jQuery methods
  - `addClass(className)` → add a CSS class
  - `removeClass(className)` → removes a CSS class
    - `className` can contain a **space-separated list of class** names to be added or removed
  - `hasClass(className)` is a method that returns true if the element has the `className` currently assigned
  - `toggleClass(className)` will add or remove a class, depending on whether it is currently present in the list of classes

5

Common Element Manipulations in jQuery

- `html()` method is an easy way to retrieve and manipulate the HTML contents

```
<script src="http://code.jquery.com/jquery-3.5.1.min.js"></script>

<script>
    // retrieve the content
    var content = $("#sample").html();
    // modify the content of an element
    $("#sample").html("brand new content");
    // modify the content of ALL <p> elements
    $("p").html("jQuery is fun");
</script>
```

- `html()` method should be used with **caution** since the `innerHTML` of a DOM element can itself contain nested HTML elements

6

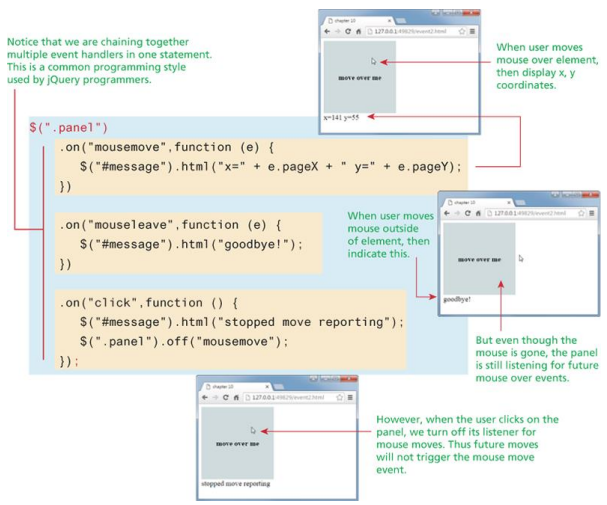
Event Handling in jQuery

- just like JavaScript, jQuery supports creation and management of listeners/handlers for JavaScript events
  - **conceptually** the same but some minor **syntactic** differences
- setting up **listeners** for particular events is done in much the same way as JavaScript
- while pure JavaScript uses the `addEventListener()` method, jQuery has
  - `on()`
  - `off()`
  - shortcut methods to *attach* events

7

Event Handling in jQuery
→ Binding and Unbinding Events

- **jQuery** is much less verbose than **JavaScript**
- **jQuery** simplifies many **common tasks** when working with **events**



Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

8

Event Handling in jQuery
→ Binding and Unbinding Events (cont'd)

ex1

```
<button id="example">Click me</button>
<span id="message"></span>
<script>
    // javascript version
    document.getElementById("example").addEventListener("click", function () {
        document.getElementById("message").innerHTML = "you clicked";
    });

    // jquery version
    $("#example").on("click", function () {
        $("#message").html("you clicked");
    });

    // alternate jquery version using defined function instead of anonymous one
    $("#example").on("click", clicker);

    function clicker() {
    $("#message").html("you clicked");
    }

    // alternate jquery version using click() shortcut method
    $("#example").click(function () {
        $("#message").html("you clicked");
    });
</script>
```

9

Event Handling in jQuery
→ Page Loading

- it is a good practice to have your listeners setup inside the `window.addEventListener("load", …)` event
  - ensures the entire page and all DOM elements are **loaded before** trying to **attach listeners** to them
- **jQuery**: we do the same but use `$(document).ready()` event
- **example**: setup event listener after HTML document has been loaded and parsed into its DOM representation

```
<script>
    $(document).ready(function () {
        // set up listeners knowing page loads before this runs
        $("#example").click(function () {
            $("#message").html("you clicked");
        });
    });
</script>
```

10

Event Handling in jQuery
→ Page Loading

- `onload()` event in JavaScript (*lecture 5 examples*)
  - this event occurs when an object is loaded (in browser)
  - mainly associated with the body tag
    - you can run a script once the HTML page loads all content
  - can use onload to check browser type/version
    - may be load proper HTML page version based on browser information
  - can be used to deal with cookies

```
<p id="element"></p>
<script>
    window.onload=initPage;
    function initPage() {
       $("#element").html("Now redirecting to U of Washington website...  please wait");
    }
</script>
```

11

Event Handling in jQuery
→ Page Loading (cont'd)

ex2

- `onload()` example: redirect when document is ready

```
<!DOCTYPE html>
<script src="http://code.jquery.com/jquery-3.5.1.min.js"></script>
<html>
<body>
    <p id="element"></p>
    <script>
        window.onload=initPage;
        function initPage() {
           $("#element").html("Now redirecting to U of Washington website... please wait");
        }
        $(document).ready(function () {
           // when document is ready... call this function
           window.setTimeout(function () {
              window.location.href = "http://www.uw.edu";
           }, 3000);
           });
    </script>
</body>
</html>
```

12

DOM Manipulation
→ Creating Nodes

- **jQuery** provides many useful methods to manipulate DOM elements
  - we already introduced **`html()`, `attr()`, `prop()`, `css()`**
- **creating nodes**
- **jQuery** can convert strings containing valid DOM syntax into DOM objects **automatically**

```
// using JavaScript
var jsLink = document.createElement("a");
jsLink.href = "http://www.uw.edu";
jsLink.innerHTML = "UWashington";
jsLink.title = "new element using JS";
```

```
// using jQuery (latest version)
$('<a>', {
    href: 'http://uw.edu',
    title: 'new element using jQuery',
    text: 'UWashington'
}
);
```

13

DOM Manipulation
→ Adding DOM Elements

- after creating nodes, we must then **add** them to existing DOM tree:
- **`append()`** method is used to insert node(s) after the last child to the element(s) being selected
  - **parameters**: accepts HTML string, a DOM object or a jQuery object
- **`prepend()`** method is used to insert at the beginning of each element
  - **parameters**: accepts HTML string, a DOM object or a jQuery object

14

DOM Manipulation
→ Adding DOM Elements (cont'd)

```
<div class="dest">
existing content
</div>
```

```
var link = $('<a href="http://funwebdev.com">Fun</a>');
```

```
$(".dest").append(link);
```

```
<div class="dest">
existing content
<a href="http://funwebdev.com">Fun</a>
</div>
```

```
$(".dest").prepend(link);
```

```
<div class="dest">
<a href="http://funwebdev.com">Fun</a>
existing content
</div>
```

```
link.appendTo($(".dest"));
```

```
<div class="dest">
existing content
<a href="http://funwebdev.com">Fun</a>
</div>
```

```
link.prependTo($(".dest"));
```

```
<div class="dest">
<a href="http://funwebdev.com">Fun</a>
existing content
</div>
```

```
$(".dest").before(link);
```

```
<a href="http://funwebdev.com">Fun</a>
<div class="dest">
existing content
</div>
```

```
$(".dest").after(link);
```

```
<div class="dest">
existing content
</div>
<a href="http://funwebdev.com">Fun</a>
```

```
link.insertBefore($(".dest"));
```

```
<a href="http://funwebdev.com">Fun</a>
<div class="dest">
existing content
</div>
```

```
link.insertAfter($(".dest"));
```

```
<div class="dest">
existing content
</div>
<a href="http://funwebdev.com">Fun</a>
```

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

15

DOM Manipulation
→ Wrapping Existing DOM in New Tags

- one of the most common ways you can enhance a website that supports JavaScript is to add new HTML tags as needed to support some jQuery functions
- **example**

```
<div class="external-links">
    <div class="gallery">Uffuzi Museum</div>
    <div class="gallery">National Gallery</div>
    <div class="link-out">funwebdev.com</div>
</div>
```

- assume we wanted to wrap all the gallery items in the whole page inside, another →

```
$(".gallery").wrap('<div class="galleryLink"</div>');
```

```
<div class="external-links">
    <div class="galleryLink">
        <div class="gallery">Uffuzi Museum</div>
    </div>
    <div class="galleryLink">
        <div class="gallery">National Gallery</div>
    </div>
    <div class="link-out">funwebdev.com</div>
</div>
```

resulting HTML

16

DOM Manipulation
→ Wrapping Existing DOM in New Tags (cont'd)

- let's extend the previous example such that it will create a unique div for each element
- **example**
- and we then apply this jQuery…

```
<div class="external-links">
    <div class="gallery">Uffuzi Museum</div>
    <div class="gallery">National Gallery</div>
    <div class="link-out">funwebdev.com</div>
</div>
```

```
$(".gallery").wrap(function () {
    return "<div class='galleryLink' title='Visit " + $(this).html() + "'></div>";
});
```

```
<div class="external-links">
    <div class="galleryLink" title="Visit Uffuzi Museum">
        <div class="gallery">Uffuzi Museum</div>
    </div>
    <div class="galleryLink" title="Visit National Gallery">
        <div class="gallery">National Gallery</div>
    </div>
    <div class="link-out">funwebdev.com</div>
</div>
```

resulting HTML

17

Effects and Animation

- **jQuery** provides easy-to-use animation and effects
- **animation and effects shortcuts**
  - one of the common features performed within a dynamic web page is to show and hide elements
    - can be done using `css()` (can cause an element to change **instantaneously**)
  - `hide()` and `show()` methods allow developers to easily hide elements **gradually** rather than through an immediate change

  - `fadeIn()` and `fadeOut()` shortcut methods control the opacity of an element

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

18

Effects and Animation (cont'd)
→ animation and effects shortcuts example

ex3

```
<!DOCTYPE html>
<script src="http://code.jquery.com/jquery-3.5.1.min.js"></script>
<div class="contact">
    <p>Eyhab Al-Masri</p>
    <div class="email">Show email</div>
</div>
<div class="contact">
    <p>Eyhab Al-Masri</p>
    <div class="email">Show email</div>
</div>
<script type='text/javascript'>
    $(".email").click(function () {
        // Build email from 1st letter of first name + lastname @uw.edu
        var fullName = $(this).prev().html();
        var firstName = fullName.split(" ")[0];
        var address = firstName.charAt(0) + fullName.split(" ")[1] + "@uw.edu";
        address=address.replace(/-/g,'');
        $(this).hide(); // hide the clicked icon
        $(this).html("<a href='mailto:" + address.toLowerCase() + "'>E-Mail Me</a>");
        $(this).show(1000); // take 1 second to show the email address
    });
</script>
```

19

Effects and Animation (cont'd)
→ sliding

ex4

```
<!DOCTYPE html>
<script src="http://code.jquery.com/jquery-3.5.1.min.js"></script>
<div id="menuBtn">Menu</div>
<ul id="menu">
    <li><a href="#">Menu item 1</a></li>
    <li><a href="#">Menu item 2</a></li>
    <li><a href="#">Menu item 3</a></li>
    <li><a href="#">Menu item 4</a></li>
</ul>
<script type='text/javascript'>
    $(function () {
        $("#menu").hide(); // hide menu when page loads
        $("#menuBtn").on("mouseenter", function () {
            // slide list down in 0.5 seconds when mouse hovers over it
            $("#menu").slideDown(500);
        });

        $("#menuBtn").on("mouseleave", function () {
            // slide list down in 0.5 seconds when mouse is no longer hovering over
            it
            $("#menu").slideUp(300);
        });
    });
</script>
```

jQuery Mouse Events
https://api.jquery.com/category/events/mouse-events/

20

Effects and Animation
→ Raw Animation

ex5

- animations introduced so far are all variations of the generic **`animate()`** method
- if you need to perform more advanced animation actions, you will need to make use of this method
  - **`animate()`** enables you to animate any numeric CSS property

```
<style>
      .notifyBox {
      width:400px;
      position: absolute;
      padding: 15px;
      border: 1px solid transparent;
      color: #3c763d;
      background-color: #dff0d8;
      text-align:center;
      }
</style>
```

```
<button id="notifyUp">Notify Up</button>
<button id="notifyDown">Notify Down</button>
<div class="notifyBox">Notification Box Activated. </div>
<script type='text/javascript'>
      $('#notifyUp').click(function () {
         $(".notifyBox").animate({right:'0px', top:"0"}, 1000)
      });
      $('#notifyDown').click(function () {
        $(".notifyBox").animate({right:'0px', bottom:"0"}, 1000)
      });
</script>
```

21

Asynchronous JavaScript with XML (AJAX)

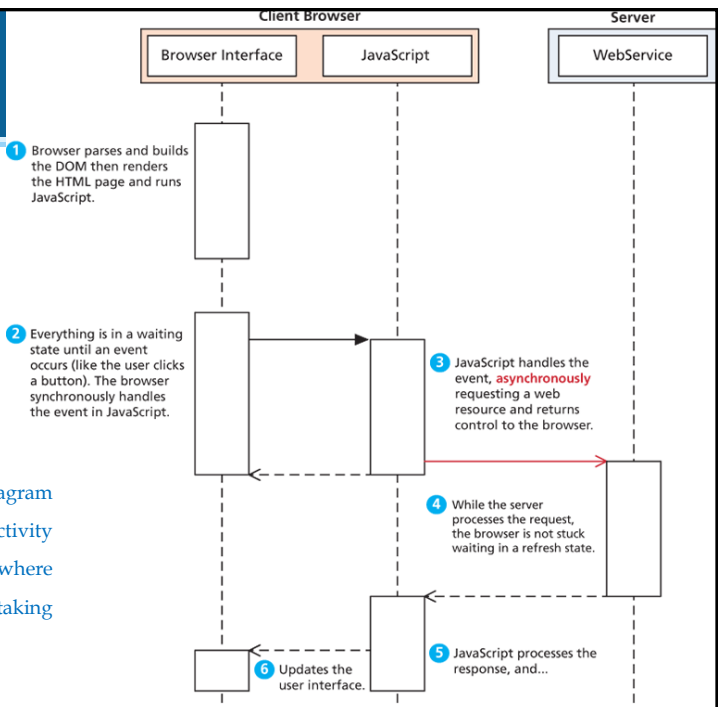
TCSS 460 - Summer 2020 22

AJAX

- **Asynchronous JavaScript with XML (AJAX)**
  - a paradigm that allows a web browser to send messages back to the server without interrupting the flow of what's being shown in the browser.
- this makes use of a browser's **multithreaded design**
  - one thread handle the browser and interactions while other threads wait for responses to asynchronous requests.
- **responses** to **asynchronous requests** are caught in JavaScript as **events**
  - events can subsequently trigger changes in the user interface or make additional requests
  - this differs from the typical **synchronous** requests we have seen thus far, which require the entire web page to refresh in response to a request.

23

AJAX (cont'd)



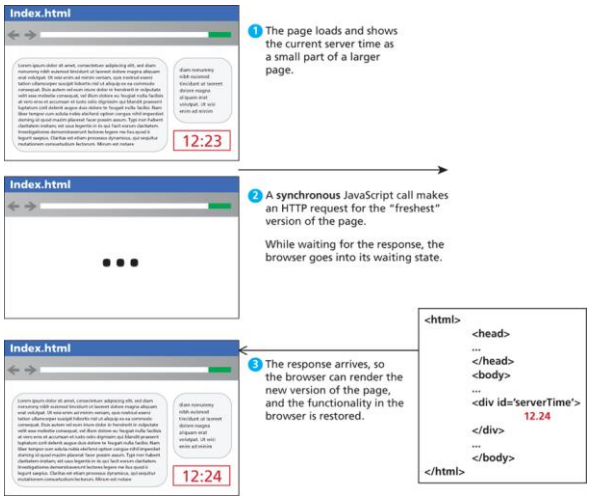
UML sequence diagram where the white activity bars illustrate where computation is taking place

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

24

AJAX (cont'd)

Illustration of a **synchronous** implementation of the server time web page

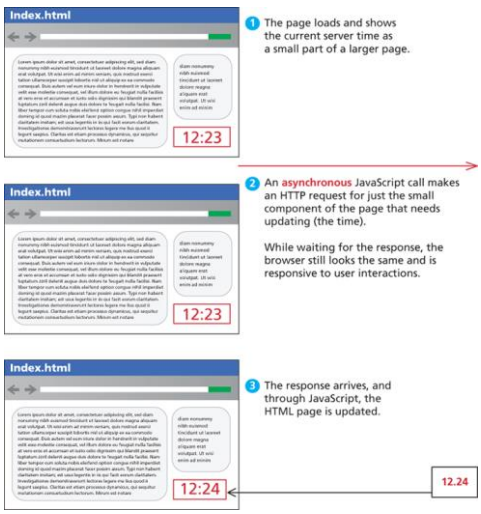


Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

25

AJAX (cont'd)

Illustration of an **AJAX** implementation of the server time web page



Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

26

AJAX
→ Making Asynchronous Requests (cont'd)

- jQuery provides a family of methods to make asynchronous requests
- example
  - assume we have a server-side script named curentTime.php that returns a single string and you would like to load that value **asynchronously** into the an HTML element
    - i.e. `<div id="timeDiv">`
- jQuery makes this extraordinarily simple:

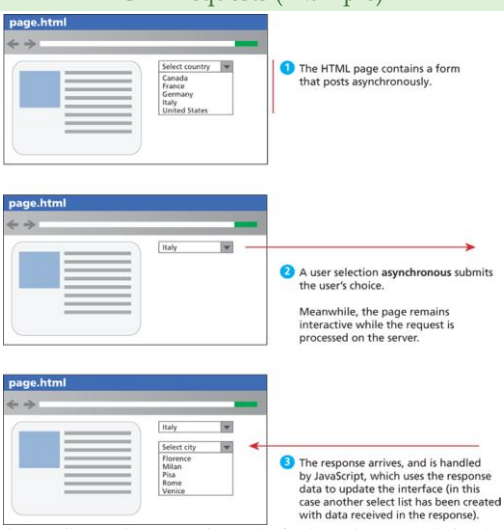
```
$("#timeDiv").load("currentTime.php");
```

- this code must be running on a web server in order for it to work
  - when this code is executed on the server, jQuery makes an HTTP GET request
  - browser receives response and sets HTML content of the element equal to the response

27

AJAX
→ Making Asynchronous Requests (cont'd)

GET Requests (Example)



Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

28

AJAX
→ Making Asynchronous Requests (cont'd)

Making Asynchronous requests – GET format

- **jQuery.get (** url [, data] [, success([data, textStatus, jqXHR])] [, dataType] **)**
- `url` is a string that holds the location to send the request
- `data` is an optional parameter (query string or JavaScript object literal)
- `success(data, textStatus, jqXHR)` is an optional callback function
  - `data` holding the body of the response as a string
  - `textStatus` holding the status of the request (i.e., "success")
  - `jqXHR` holding a jqXHR object
- `dataType` is an optional parameter to hold the type of data expected

29

AJAX
→ Making Asynchronous Requests (cont'd)

`jqXHR` object

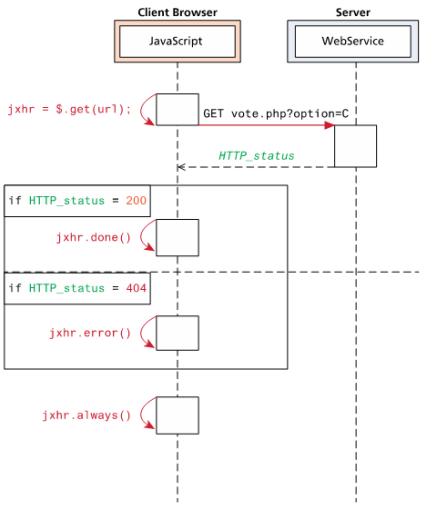
- `$.get()` requests made by jQuery return **a `jqXHR` object**
  - **jqXHR → jQuery XMLHttpRequest** (wraps browser native XMLHttpRequest)
- this object is a superset of the `XMLHttpRequest` object
  - `abort()` stops execution and prevents any callback or handlers from receiving the trigger to execute.
  - `getResponseHeader()` takes a parameter and gets the current value of that header.
  - `readyState` is an integer from 1 to 4 representing the state of the request.
    - the values include 2: request sent, 3: response being processed, and 4: completed.
  - `responseXML` and/or `responseText` the main response to the request.
  - `setRequestHeader(name, value)` allows headers to be changed for the request.
  - `status` is the HTTP request status codes
  - `statusText` is the associated description of the status code.

30

AJAX
→ Making Asynchronous Requests (cont'd)

jqXHR object

- **jqXHR** objects implement the methods
  - **done()**
  - **fail()**
  - **always()**



Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

31

AJAX
→ Making Asynchronous Requests (cont'd)

POST Method

- jQuery handles POST almost as easily as GET, with the need for an added field to hold our data.

```
$.get("serviceTravelCities.php", param)
```

to

```
$.post("serviceTravelCities.php", param)
```

- **POST → form serialization**
  - the **serialize()** method can be called on a DOM form element to encode it into a query string

```
var postData = $("#someForm").serialize();
$.post("formHandler.php", postData);
```

32

AJAX
→ Making Asynchronous Requests (cont'd)

POST Method (cont'd)

- here we add headers

```
$.ajax({ url: "vote.php",
    data: $("#voteForm").serialize(),
    async: true,
    type: post,
    headers: {"User-Agent" : "Homebrew Vote Engine",
         "Referrer": "http://uw.edu"
    }
});
```

https://api.jquery.com/jquery.ajax/

33

Module Topics

- ☑ jQuery (Part I)
- ☑ jQuery (Part II)
- JavaScript Frameworks

TCSS 460 - Summer 2020

34