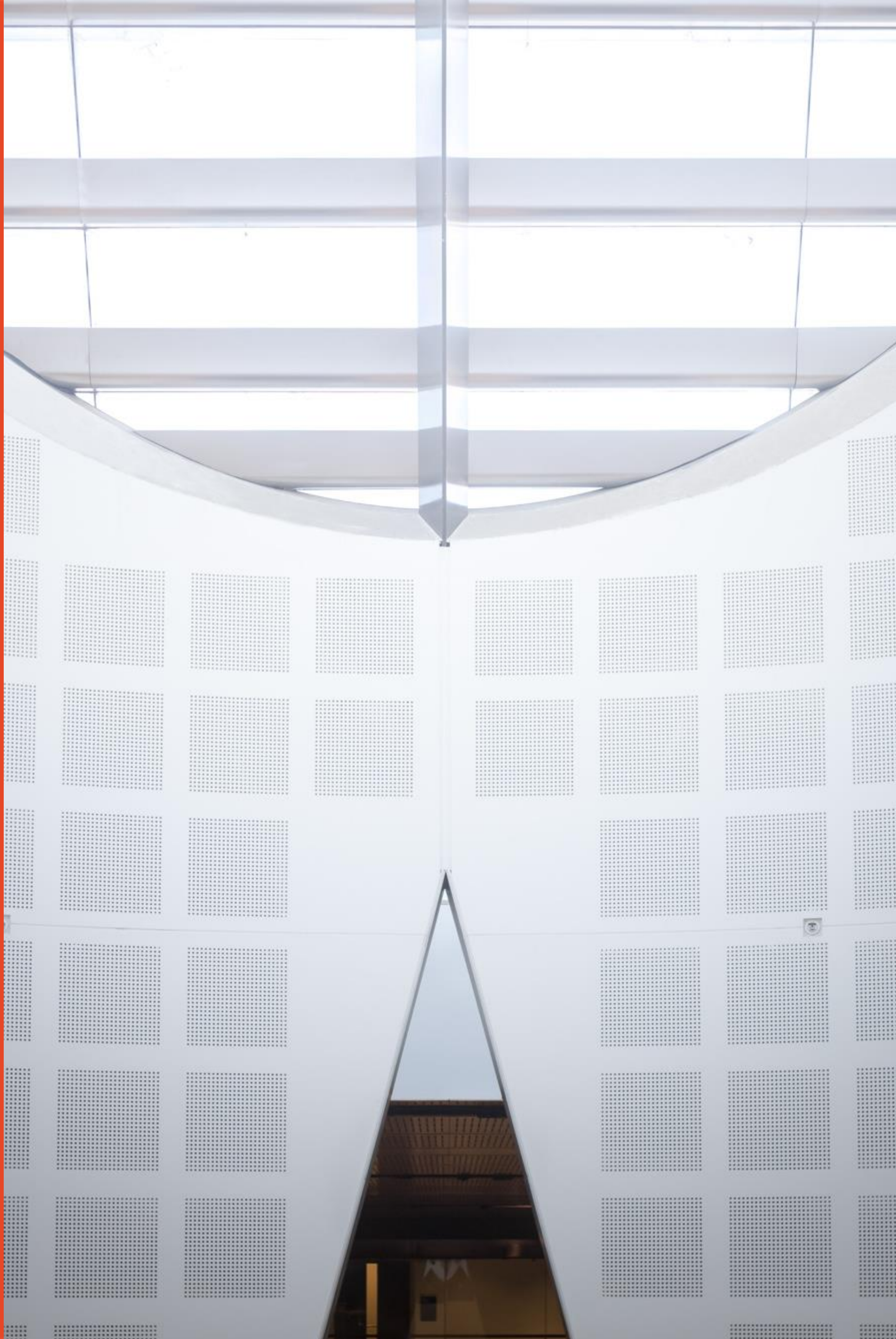
COMP5347: Web Application Development

Client-Side Libraries

Dr. Basem Suleiman

School of Computer Science



Outline

- JavaScript Frameworks
- Intro to jQuery
  - Selectors
  - Event handler and DOM Manipulation
  - Ajax requests
- Integrate jQuery with Expressjs Application

Revisits Client Side Technologies

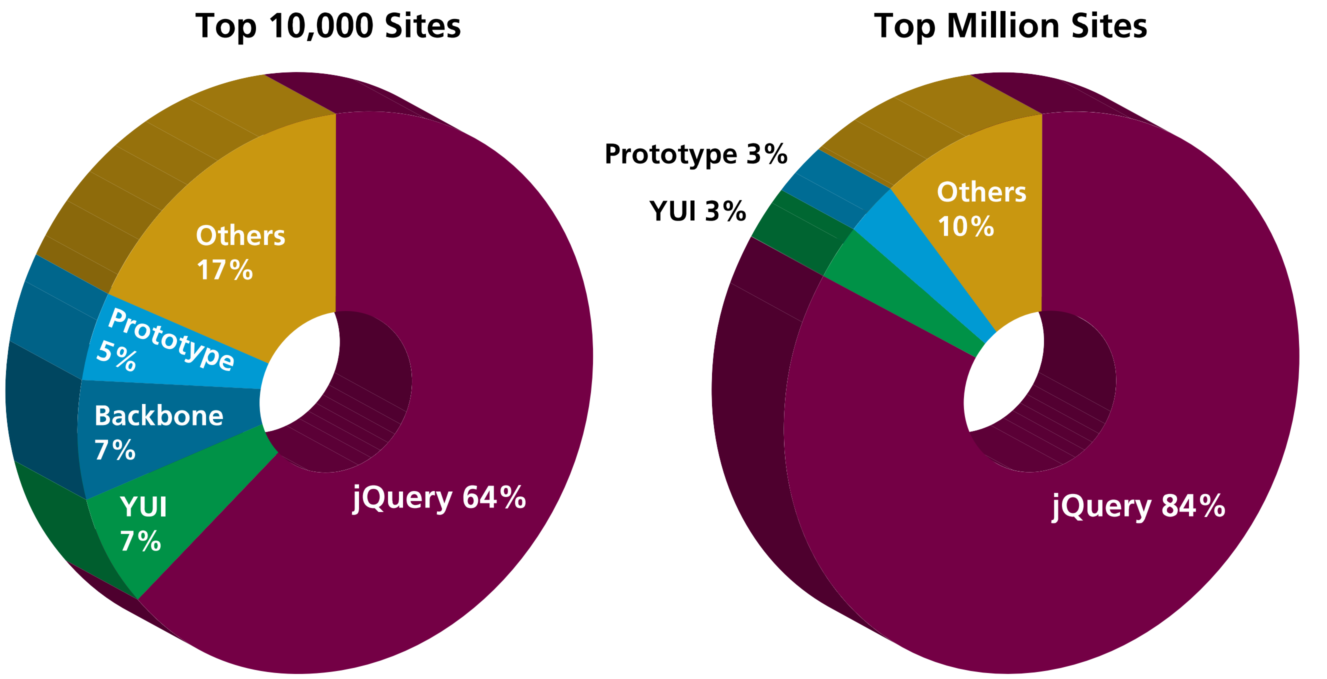
- Web client is not a pure passive receiver of data sent from the server

- Modern client has lots of interactive features to make it like desktop GUI
  - HTML5
  - CSS3
  - JavaScript

- Many client side JavaScript libraries
  - jQuery
  - Specialized libraries, e.g. D3.js, various google libraries

- Client side “scripting” becomes real application development with its own model, view and controller
  - AngularJS framework
  - Backbone MVC framework

JavaScript Frameworks



Comparison of the most popular JavaScript frameworks

Randy Connolly, Ricardo Hoar (2017). Fundamentals of Web Development, Global Edition, Pearson

Outline

- JavaScript Frameworks
- Intro to jQuery
  - Selectors
  - Event handler and DOM Manipulation
  - Ajax requests
- Integrate jQuery with Expressjs Application

jQuery

- jQuery is a lightweight JavaScript library
  - Provides methods to wrap common JavaScript tasks
    - HTML/DOM and CSS manipulation (e.g., selecting elements)
    - HTML event methods (e.g., register element's event handler)
    - AJAX (managing asynchronized request)
    - Effects and animation

- jQuery will run consistently across all major browsers
  - Cross-browser knowledge and issues are considered

- Adopted by major companies including Google, Microsoft and IBM

Using jQuery

- The library is released as a single JavaScript file
  - Can be downloaded then installed locally
  - Include it from a CDN like Google, Microsoft or jQuery itself
    - Reference it in the HTML <script> tag

- Using a Content Delivery Network (CDN):

```
<script src="http://code.jquery.com/jquery-3.1.0.min.js"> </script>
```

- Use a failsafe in case the CDN is down

Using jQuery – Failsafe Loading

```
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script type="text/javascript">
window.jQuery ||
document.write('<script src="/jquery-1.9.1.min.js"><\/script>');
</script>
```

- Pros of CDN host:
  - The bandwidth is offloaded to reduce the demand on your servers
  - The user may already have cached the third-party file; reducing the total loading time

- Cons of CDN host:
  - jQuery will fail if the third-party host fails (unlikely but possible)

jQuery Function

- The jQuery syntax is customized for **selecting** HTML elements and performing some **action** on the element(s)
  - Remember getElementById() …
- The **jQuery()** or **$()** function
- **$(selector).action()**
  - $ sign to define/access jQuery
  - **(selector)** to "query (or find)" HTML elements
  - jQuery **action()** to be performed on the element(s)
- The **$()** function always returns a set of results

jQuery – Selectors

- Selecting using regular JavaScript

```
var node = document.getElementById("here");
var link = document.querySelectorAll("ul li");
```

- equivalent selection using jQuery

```
var node = $("#here");
var link = $("ul li");
```

- Example with action
  - Hide all elements with class="test“
  - $(".test").hide()

jQuery – Main Selectors

- The selectors are very similar to CSS selectors

- The four basic selectors are:
  - $("*") **Universal selector** matches all elements (slow)
  - $("tag") **Element selector** matches all elements with the given element name
  - $(".class") **Class selector** matches all elements with the given CSS class
  - $("#id") **Id selector** matches all elements with a given HTML id attribute.

- Other selectors defined in CSS can be used

jQuery – Basic Selector Examples

- Select the single <div> element with id="grab"
  - **var singleElement = $("#grab");**
- Get a set of all the <a> elements the selector
  - **var allAs = $("a");**
- Select all odd <tr> elements
  - $(“tr:odd”)
- These selectors replace the use of **`getElementById()`** and similar functions entirely

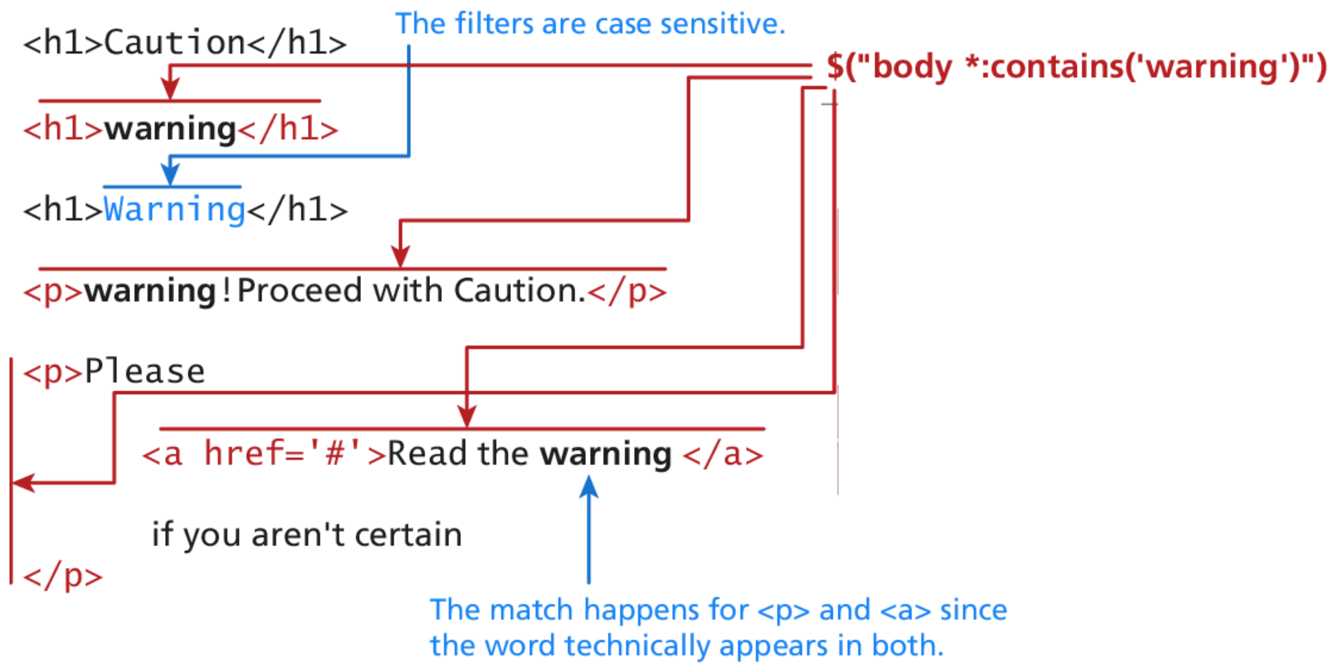
jQuery – Advanced Selectors

- Pseudo class selector
  - E.g. Selecting all links that have been visited
  - **var visitedLinks = $("a:visited");**

- Beyond CSS selectors
  - Content Filters
    - Select elements based on criteria
  - Form Selectors
    - Shorthand version to select form elements

jQuery – Content Filters Selector

– $("body *:contains('warning')”)

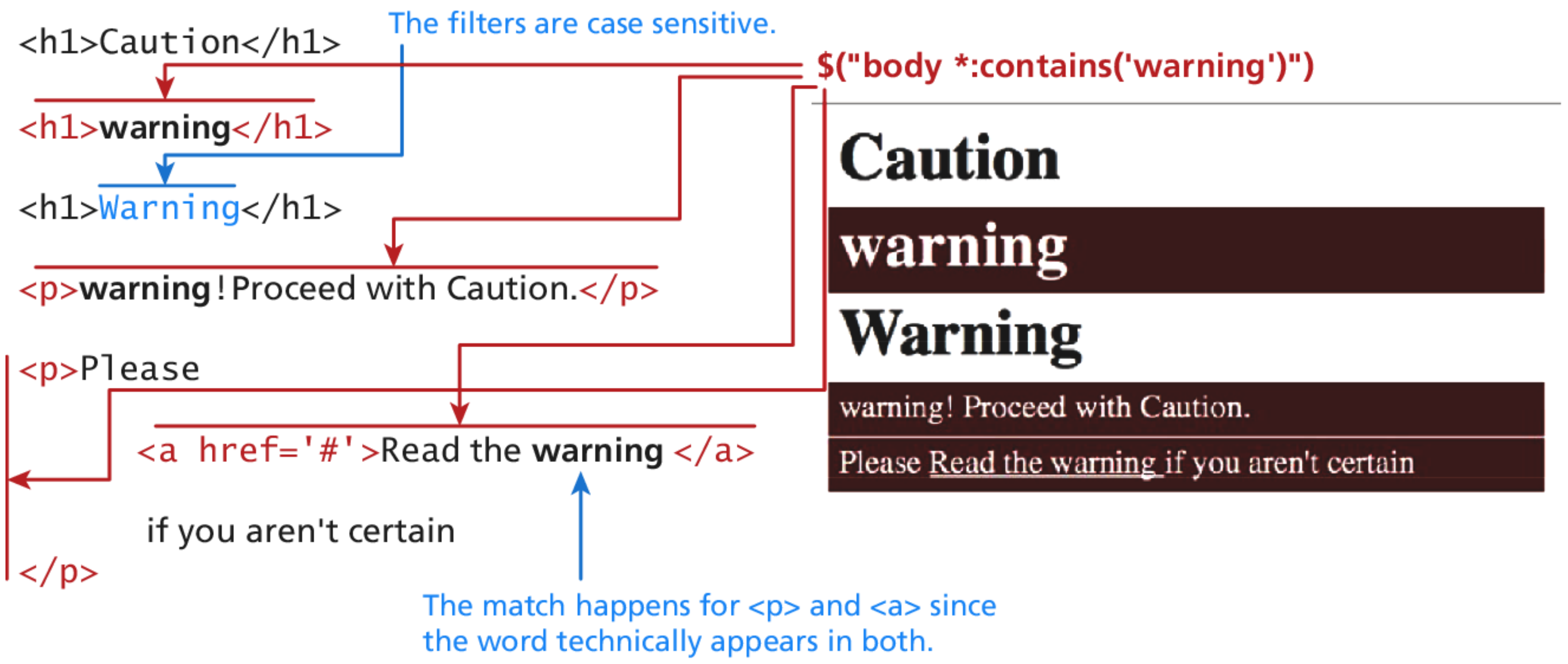


```
<h1>Caution</h1>
<h1>warning</h1>
<h1>Warning</h1>
<p>warning!Proceed with Caution.</p>
<p>Please
   <a href='#'>Read the warning </a>
   if you aren't certain
</p>
```

The filters are case sensitive.
$("body *:contains('warning')")
The match happens for <p> and <a> since the word technically appears in both.

jQuery – Content Filters Selector

– $("body *:contains('warning')”)



The filters are case sensitive.
$("body *:contains('warning')")
The match happens for <p> and <a> since the word technically appears in both.

```
<h1>Caution</h1>
<h1>warning</h1>
<h1>Warning</h1>
<p>warning!Proceed with Caution.</p>
<p>Please
  <a href='#'>Read the warning </a>
  if you aren't certain
</p>
```

Caution

warning

Warning

warning! Proceed with Caution.

Please Read the warning if you aren't certain

jQuery – HTML Attributes and Properties

- We can both set and get an attribute value by using the **attr()** method

```
var link = $("a").attr("href");
$("a").attr("href","http://funwebdev.com");
$("img").attr("class","fancy");
```

jQuery – HTML Attributes and Properties

- The **prop()** method is the preferred way to retrieve and set the value of a property.

```
<input class="meh" type="checkbox" checked="checked">
```

```
    var theBox = $(".meh");

    theBox.prop("checked"); // evaluates to TRUE
```

Form Selectors

Selector	CSS Equivalent	Description
$(:button)	$("button, input[type='button']")	Selects all buttons
$(:checkbox)	$('[type=checkbox]‘)	Selects all checkboxes
$(:checked)	No Equivalent	Selects elements that are checked. This includes radio buttons and checkboxes.
$(:disabled)	No Equivalent	Selects form elements that are disabled.
$(:enabled)	No Equivalent	Opposite of :disabled
$(:file)	$('[type=file]')	Selects all elements of type file
$(:focus)	$(document.activeElement)	The element with focus
$(:image)	$('[type=image]')	Selects all elements of type image
$(:input)	No Equivalent	Selects all <input>, <textarea>, <select>, and <button>
$(:password)	$('[type=password]')	Selects all password fields
$(:radio)	$('[type=radio]')	Selects all radio elements
$(:reset)	$('[type=reset]')	Selects all the reset buttons
$(:selected)	No Equivalent	Selects all the elements that are currently selected of type <option>. It does not include checkboxes or radio buttons.
$(:submit)	$('[type=submit]')	Selects all submit input elements
$(:text)	No Equivalent	Selects all input elements of type text. $('[type=text]') is almost the same, except that $(:text) includes <input> fields with no type specified.

Outline

- JavaScript Frameworks
- Intro to jQuery
  - Selectors
  - Event handler and DOM manipulation
  - Ajax requests
- Integrate jQuery with Expressjs Application

jQuery – Event Handling

- jQuery supports creation and management of handlers for JavaScript events
- jQuery has **on()** and **off()** methods and shortcut methods to attach events
    - Pure JavaScript uses the **addEventListener()** method

jQuery – Registering Event Handler

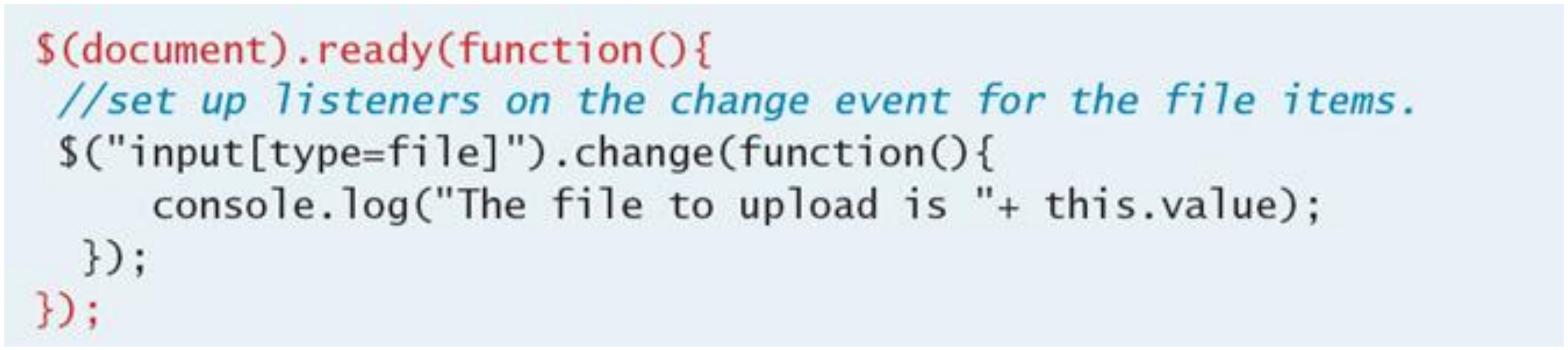
- Standard event handling syntax

  -
    ```
    $("p").click(function(){
      // action goes here!!
    });
    ```

- Common DOM events are
  - `click, dbclick, mouseenter, mouseleave`

jQuery – Registering Event Handler

- The Document Ready Event
  - Best practice to put jQuery code inside a document ready event
  - The ready event is defined by jQuery, fired after the DOM is completed



```
$(document).ready(function(){
 //set up listeners on the change event for the file items.
 $("input[type=file]").change(function(){
     console.log("The file to upload is "+ this.value);
  });
});
```

JQuery – DOM Manipulation

– Create DOM element/node JavaScript)

```
  // pure JavaScript way
var jsLink = document.createElement("a");
jsLink.href = "http://www.funwebdev.com";
jsLink.innerHTML = "Visit Us";
jsLink.title = "JS";
```

jQuery – DOM Manipulation

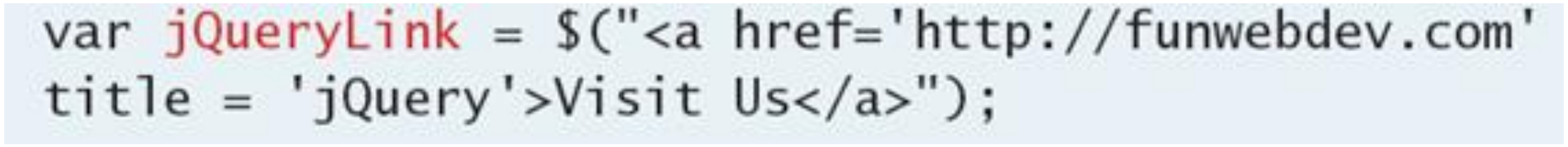
– Create DOM element/node (jQuery)

```
// jQuery way
var jQueryLink = $("<a href='http://funwebdev.com'
title = 'jQuery'>Visit Us</a>");

// jQuery long-form way
var jQueryVerboseLink = $("<a></a>");
jQueryVerboseLink.attr("href",'http://funwebdev.com');
jQueryVerboseLink.attr("title","jQuery verbose");
jQueryVerboseLink.html("Visit Us");
```

DOM Manipulation – Appending Elements

- Appending DOM Elements
  - The **append() method** takes as a parameter an HTML string, a DOM object, or a jQuery object. That object is then added as the last child to the element(s) being selected



```
var jQueryLink = $("<a href='http://funwebdev.com'
title = 'jQuery'>Visit Us</a>");
```

DOM Manipulation – Appending Elements

- Appending DOM Elements

```
var jQueryLink = $("<a href='http://funwebdev.com'
title = 'jQuery'>Visit Us</a>");
```

HTML Before

```
<div class="external-links">
     <div class="linkOut">
        funwebdev.com
    </div>
     <div class="linkIn">
         /localpage.html
    </div>
     <div class="linkOut">
        pearson.com
    </div>
<div>
```

jQuery append

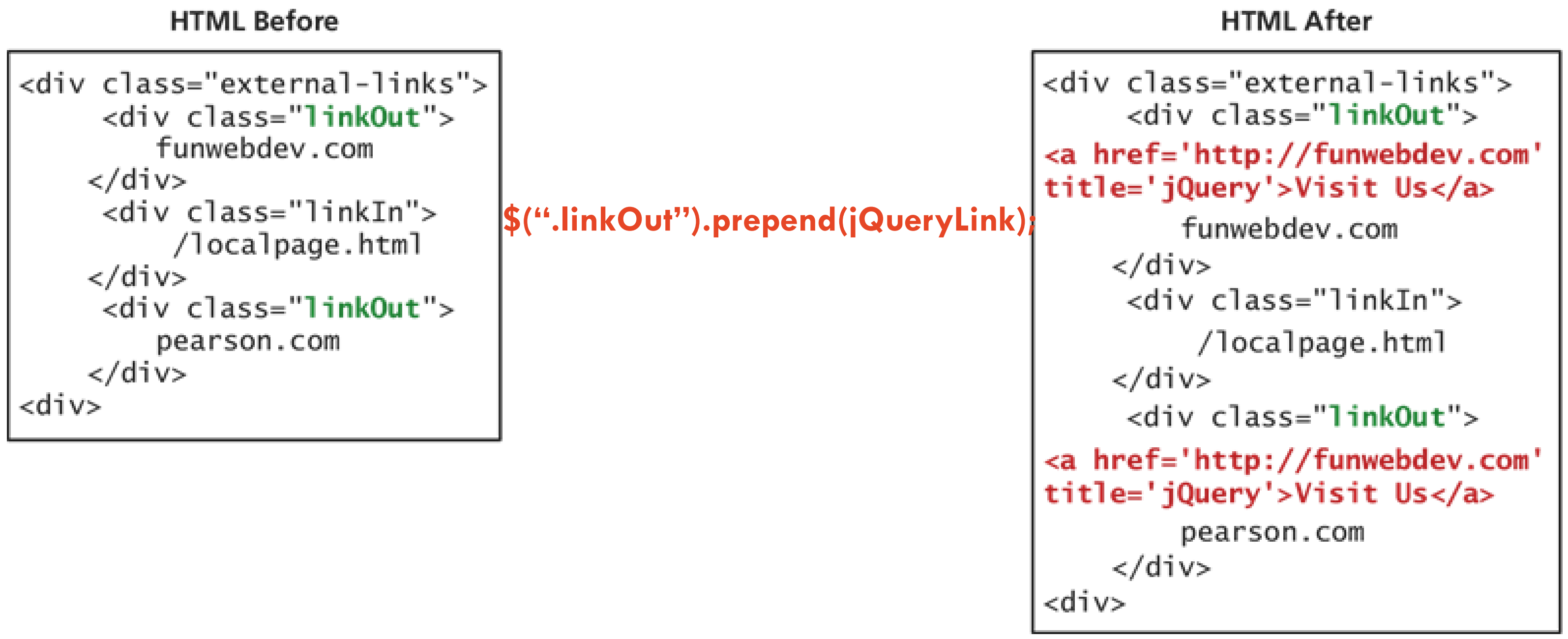
$(".linkOut").append(jQueryLink);

HTML After

```
<div class="external-links">
     <div class="linkOut">
        funwebdev.com
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
    </div>
     <div class="linkIn">
         /localpage.html
    </div>
     <div class="linkOut">
        pearson.com
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
    </div>
<div>
```

Normal DOM manipulation

- Prepending DOM Elements
  - The **prepend() method** adds the new element as the first child rather than the last



HTML Before

```
<div class="external-links">
      <div class="linkOut">
          funwebdev.com
     </div>
      <div class="linkIn">
           /localpage.html
     </div>
      <div class="linkOut">
          pearson.com
     </div>
<div>
```

$(“.linkOut”).prepend(jQueryLink);

HTML After

```
<div class="external-links">
      <div class="linkOut">
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
          funwebdev.com
     </div>
      <div class="linkIn">
           /localpage.html
     </div>
      <div class="linkOut">
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
          pearson.com
     </div>
<div>
```

jQuery – DOM Manipulation

```
<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>
```

jQuery – Useful Methods

- **attr()** – set/get attribute value on any element from a selector
  - var link = $("a").attr("href");
  - $("img").attr("class","fancy");

- **css()** - to set/get CSS properties on any element from a selector
  - var color = $("#element")**.css(**"background-color"**)**;
  - $("#colourBox")**.css**("**background-color**", "**#FF0000**")

JQuery – Useful Methods

- The **html()** - get the HTML contents of an element. If passed with a parameter, it updates the HTML of that element

- The **val()** returns the value of the element. It is mainly used to get the value of form element.

Outline

- Intro to jQuery
  - Selectors
  - Event handler and DOM man
  - Ajax requests
- Integrate jQuery with Expressjs Application

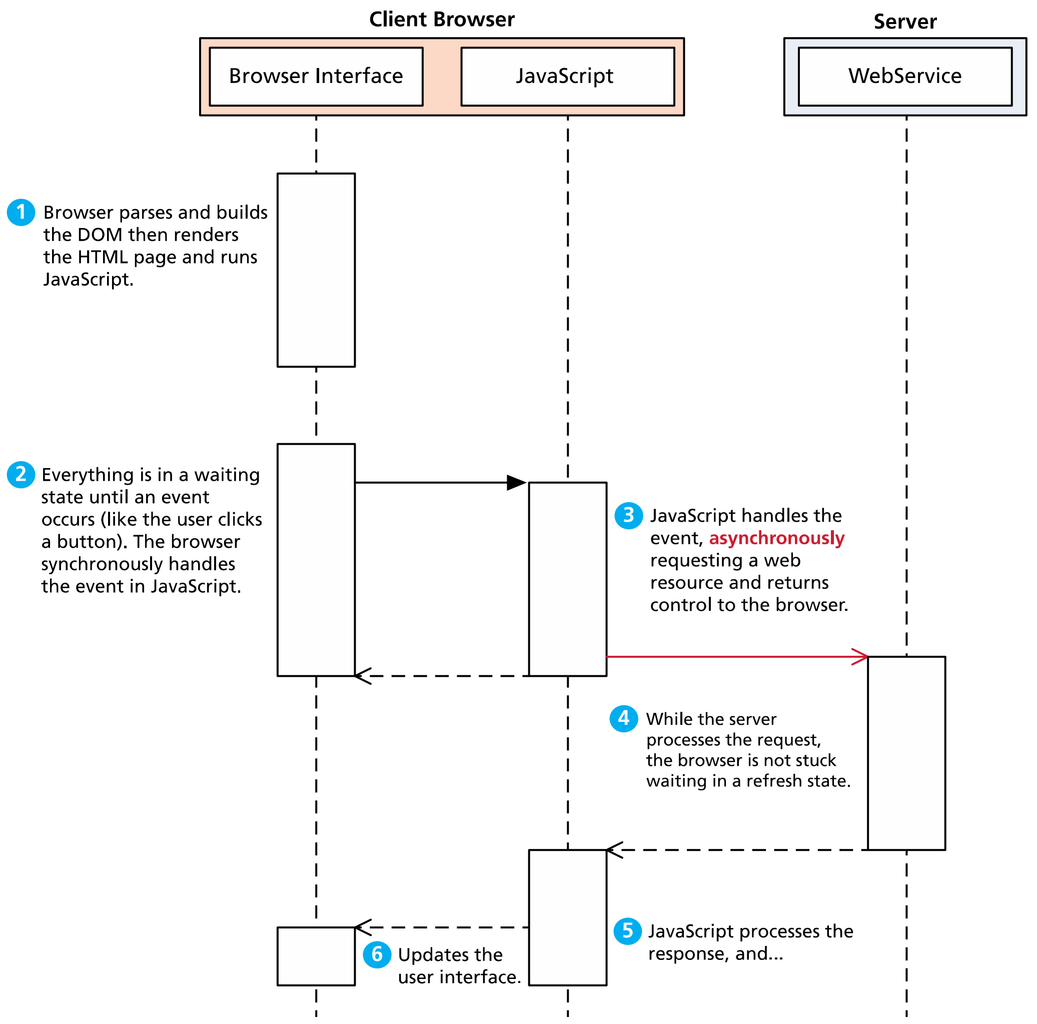
Synchronous and Asynchronous Requests

- **Synchronous** request
  - Browser sends a request then **WAIT** for the response and then render
  - Non-responsive: the user cannot interact with the client while the server is processing the request
  - Originally designed for a web of hypertext documents

- **Asynchronous** request
  - user can interact with the application while the server processes the request concurrently
  - Client-side script creating an ***XMLHttpRequest object*** to manage a request and implicit/explicit callback function to handle the response

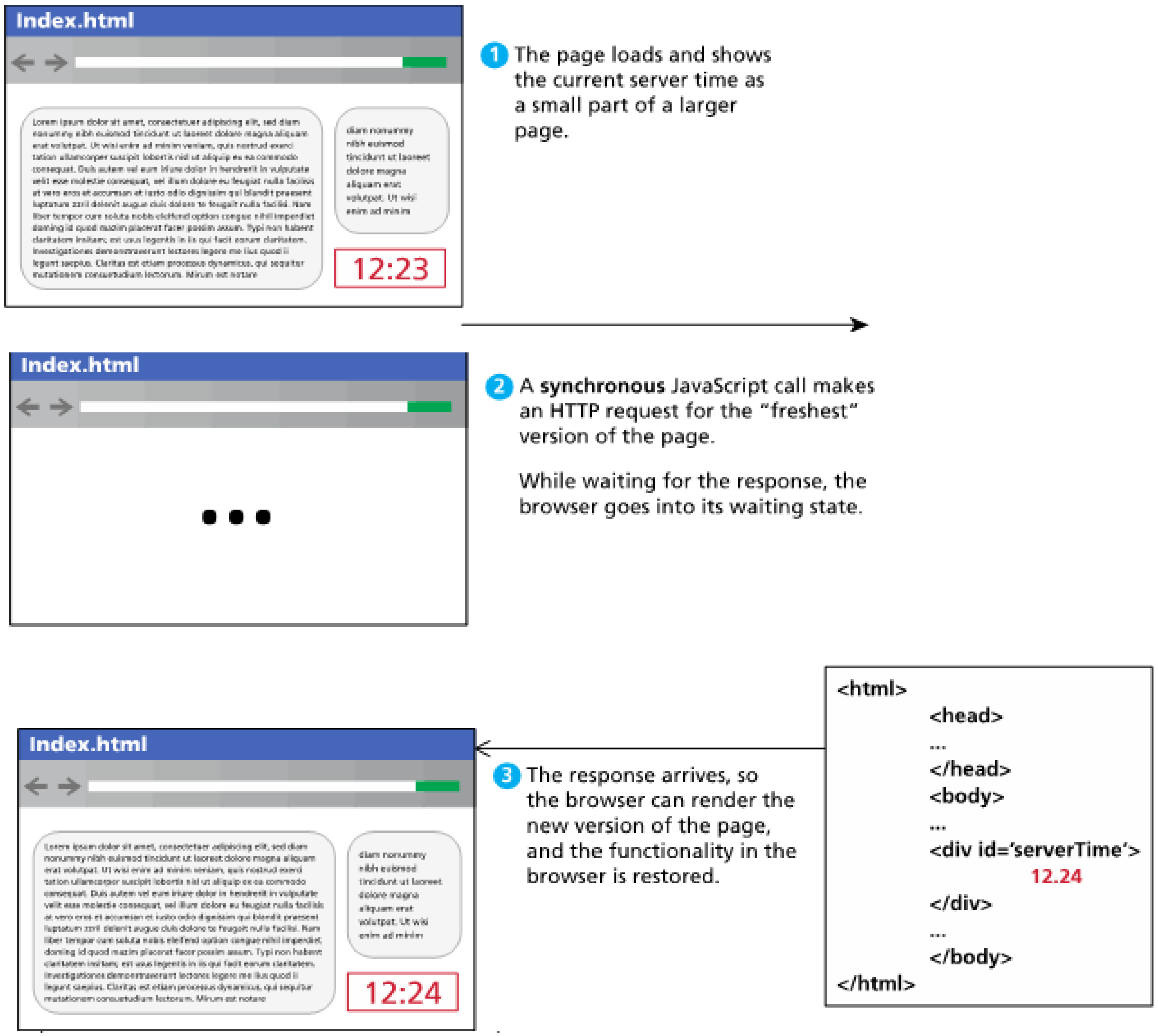
Asynchronous JavaScript with XML (AJAX)

– AJAX is a paradigm that allows a browser to send messages to the server without interrupting the flow of what’s shown in the browser



Client Browser
Server
Browser Interface
JavaScript
WebService
1 Browser parses and builds the DOM then renders the HTML page and runs JavaScript.
2 Everything is in a waiting state until an event occurs (like the user clicks a button). The browser synchronously handles the event in JavaScript.
3 JavaScript handles the event, asynchronously requesting a web resource and returns control to the browser.
4 While the server processes the request, the browser is not stuck waiting in a refresh state.
5 JavaScript processes the response, and...
6 Updates the user interface.

AJAX – Synchronous Request



1 The page loads and shows the current server time as a small part of a larger page.

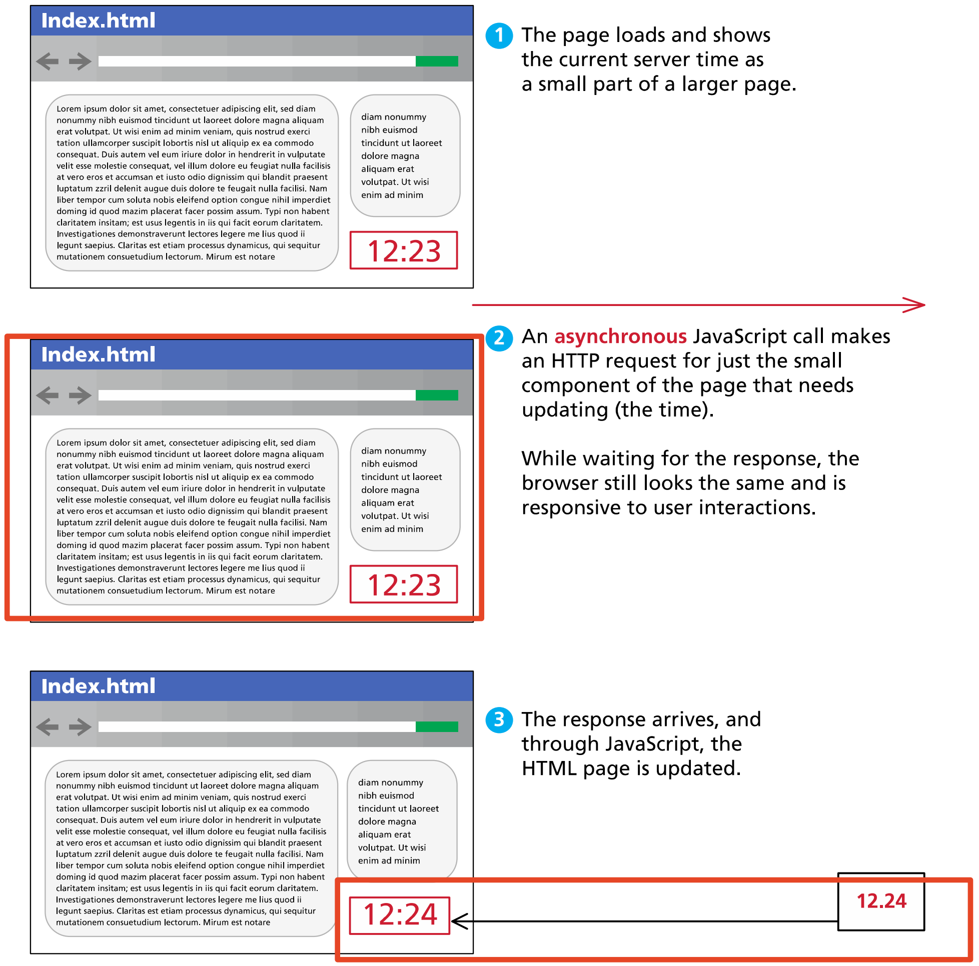
2 A **synchronous** JavaScript call makes an HTTP request for the "freshest" version of the page.

While waiting for the response, the browser goes into its waiting state.

3 The response arrives, so the browser can render the new version of the page, and the functionality in the browser is restored.

```
<html>
        <head>
        ...
        </head>
        <body>
        ...
        <div id='serverTime'>
                12.24
        </div>
        ...
        </body>
</html>
```

AJAX – Asynchronous Request



1 The page loads and shows the current server time as a small part of a larger page.

2 An asynchronous JavaScript call makes an HTTP request for just the small component of the page that needs updating (the time).

While waiting for the response, the browser still looks the same and is responsive to user interactions.

3 The response arrives, and through JavaScript, the HTML page is updated.

To load the time value asynchronously into <div id="timeDiv)

```
$("timeDiv").load("current-time.js");
```

jQuery – AJAX Support

- `load()`
  - `$(selector).load(URL,data,callback);`
  - Load URL's response into the selected element, optional `data` can be sent along with the request; optional callback can be executed after load() finishes
  - A GET request is sent if no data is present, otherwise a POST request is sent

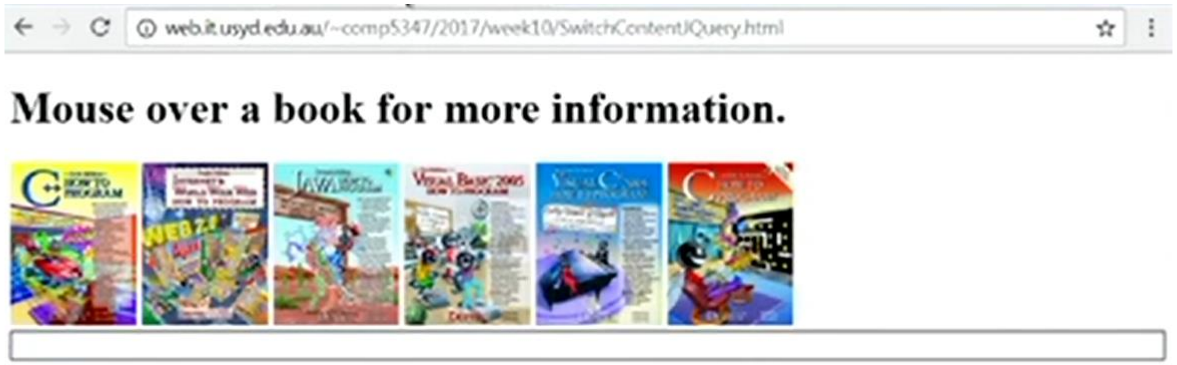
- `get()`
  - `$.get(URL, data, callback);`
  - Request data using HTTP GET method; the optional callback parameter is the name of a function to be executed after the response arrives

- `post()`
  - `$.post(URL, data, callback);`
  - Request data using HTTP POST methods; optional `data` can be sent along with the request; optional callback can be executed after response arrives

More about jQuery APIs - http://api.jquery.com/

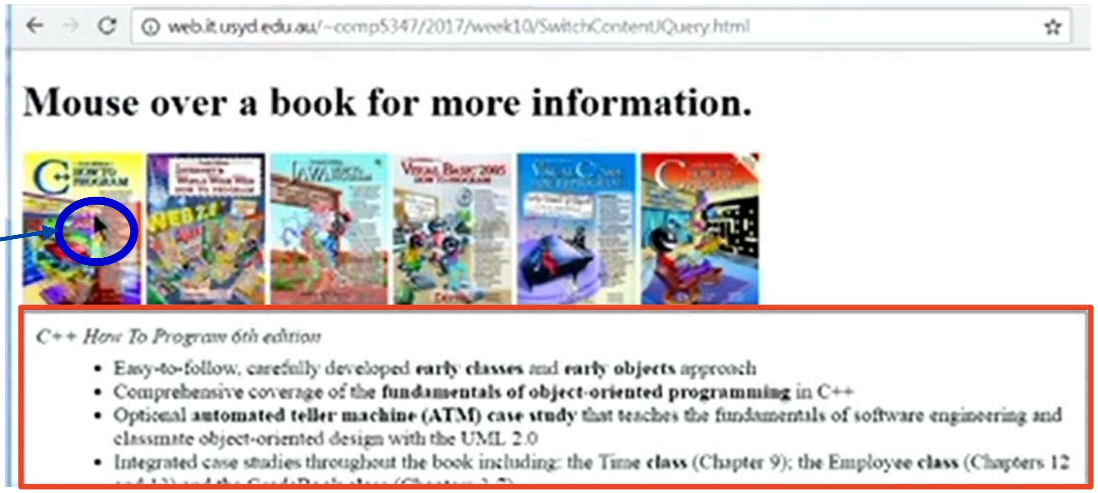
Asynchronous Request – Source Code Example

– The *XMLHttpRequest* object will fetch a static file from the server, the JavaScript running on the client browser dynamically insert the content into the current DOM tree.


web.it.usyd.edu.au/~comp5347/2017/week10/SwitchContentJQuery.html

Mouse over a book for more information.

When the mouse is moved on any of the picture, a description of the corresponding book is shown

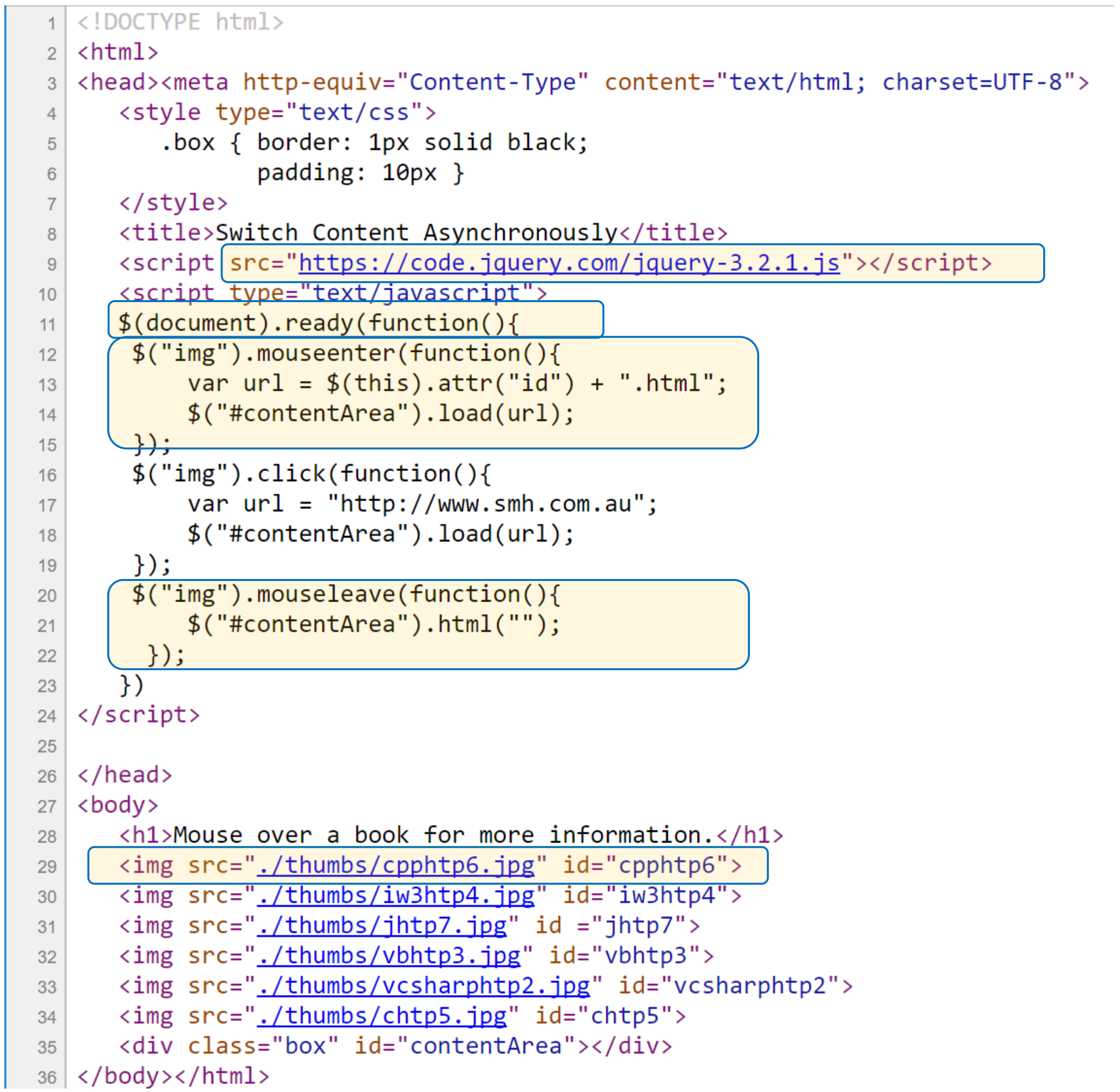

web.it.usyd.edu.au/~comp5347/2017/week10/SwitchContentJQuery.html

Mouse over a book for more information.

C++ How To Program 6th edition

- Easy-to-follow, carefully developed **early classes** and **early objects** approach
- Comprehensive coverage of the **fundamentals of object-oriented programming** in C++
- Optional **automated teller machine (ATM) case study** that teaches the fundamentals of software engineering and classmate object-oriented design with the UML 2.0
- Integrated case studies throughout the book including: the Time **class** (Chapter 9); the Employee **class** (Chapters 12

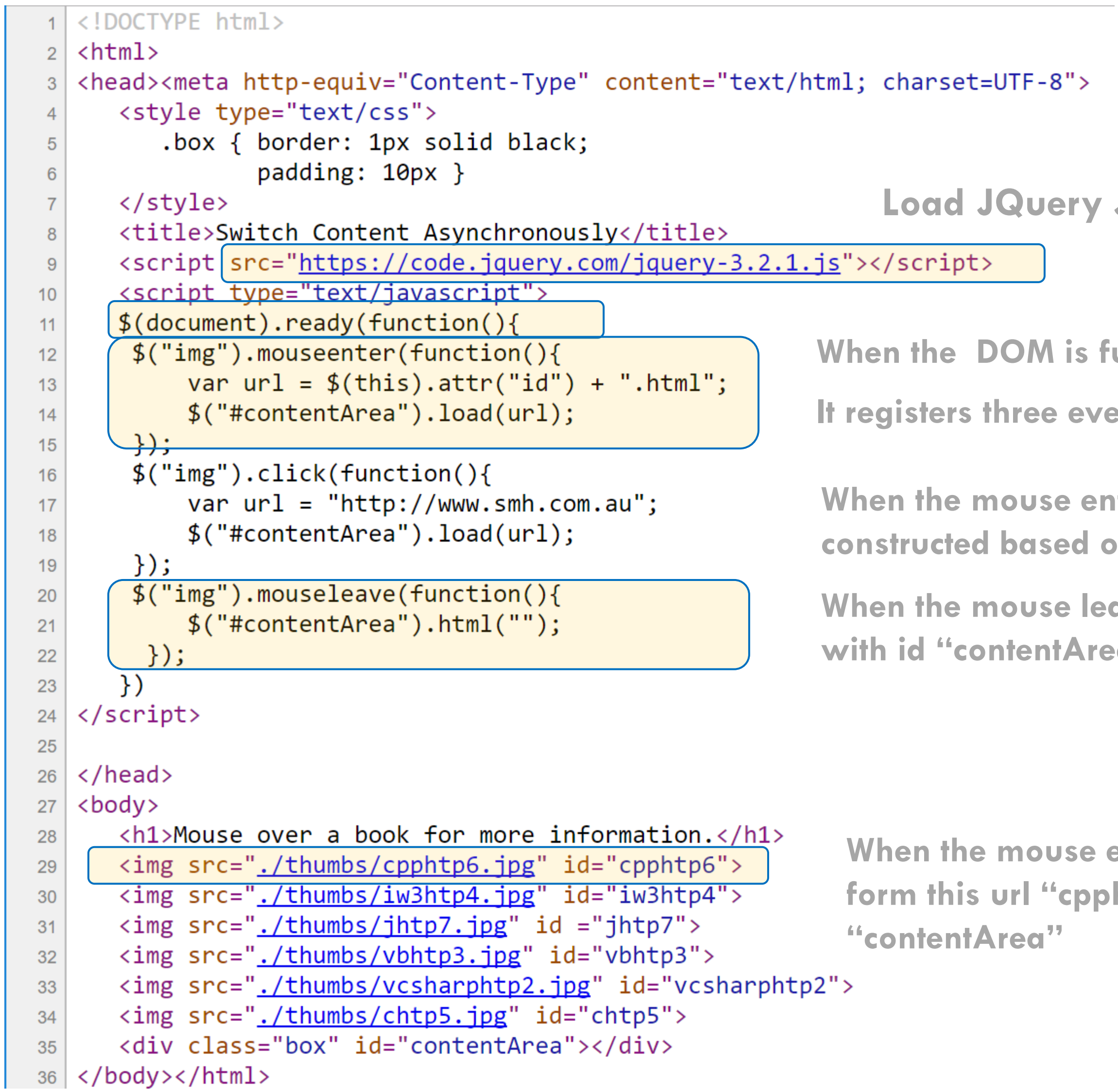
How this Works – Source Code



```html
<!DOCTYPE html>
<html>
<head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <style type="text/css">
      .box { border: 1px solid black;
             padding: 10px }
   </style>
   <title>Switch Content Asynchronously</title>
   <script src="https://code.jquery.com/jquery-3.2.1.js"></script>
   <script type="text/javascript">
   $(document).ready(function(){
    $("img").mouseenter(function(){
        var url = $(this).attr("id") + ".html";
        $("#contentArea").load(url);
    });
    $("img").click(function(){
        var url = "http://www.smh.com.au";
        $("#contentArea").load(url);
    });
    $("img").mouseleave(function(){
        $("#contentArea").html("");
     });
   })
</script>

</head>
<body>
   <h1>Mouse over a book for more information.</h1>
   <img src="./thumbs/cpphtp6.jpg" id="cpphtp6">
   <img src="./thumbs/iw3htp4.jpg" id="iw3htp4">
   <img src="./thumbs/jhtp7.jpg" id ="jhtp7">
   <img src="./thumbs/vbhtp3.jpg" id="vbhtp3">
   <img src="./thumbs/vcsharphtp2.jpg" id="vcsharphtp2">
   <img src="./thumbs/chtp5.jpg" id="chtp5">
   <div class="box" id="contentArea"></div>
</body></html>
```

How this Works – Source Code



```
<!DOCTYPE html>
<html>
<head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <style type="text/css">
      .box { border: 1px solid black;
             padding: 10px }
   </style>
   <title>Switch Content Asynchronously</title>
   <script src="https://code.jquery.com/jquery-3.2.1.js"></script>
   <script type="text/javascript">
   $(document).ready(function(){
    $("img").mouseenter(function(){
        var url = $(this).attr("id") + ".html";
        $("#contentArea").load(url);
    });
    $("img").click(function(){
        var url = "http://www.smh.com.au";
        $("#contentArea").load(url);
    });
    $("img").mouseleave(function(){
        $("#contentArea").html("");
     });
   })
</script>

</head>
<body>
   <h1>Mouse over a book for more information.</h1>
   <img src="./thumbs/cpphtp6.jpg" id="cpphtp6">
   <img src="./thumbs/iw3htp4.jpg" id="iw3htp4">
   <img src="./thumbs/jhtp7.jpg" id ="jhtp7">
   <img src="./thumbs/vbhtp3.jpg" id="vbhtp3">
   <img src="./thumbs/vcsharphtp2.jpg" id="vcsharphtp2">
   <img src="./thumbs/chtp5.jpg" id="chtp5">
   <div class="box" id="contentArea"></div>
</body></html>
```

Load JQuery JavaScript library

When the DOM is fully loaded, execute this function;

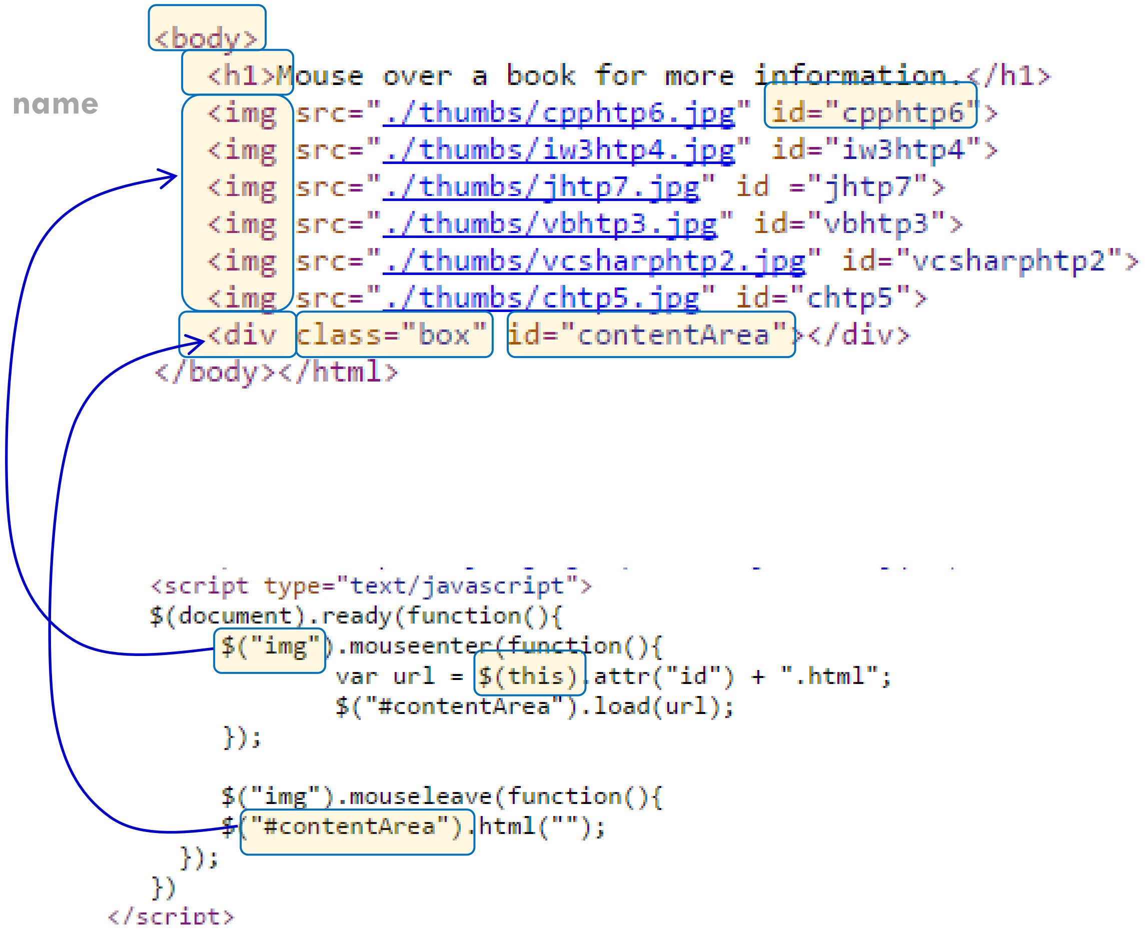
It registers three event handler functions to the <img> tag ;

When the mouse enters an imageThe url is constructed based on image tag's id value

When the mouse leaves an image, clear the tag with id "contentArea";

When the mouse enters this image, load the content form this url "cpphtp6.html" to the division with id "contentArea"

Selectors in the Example code

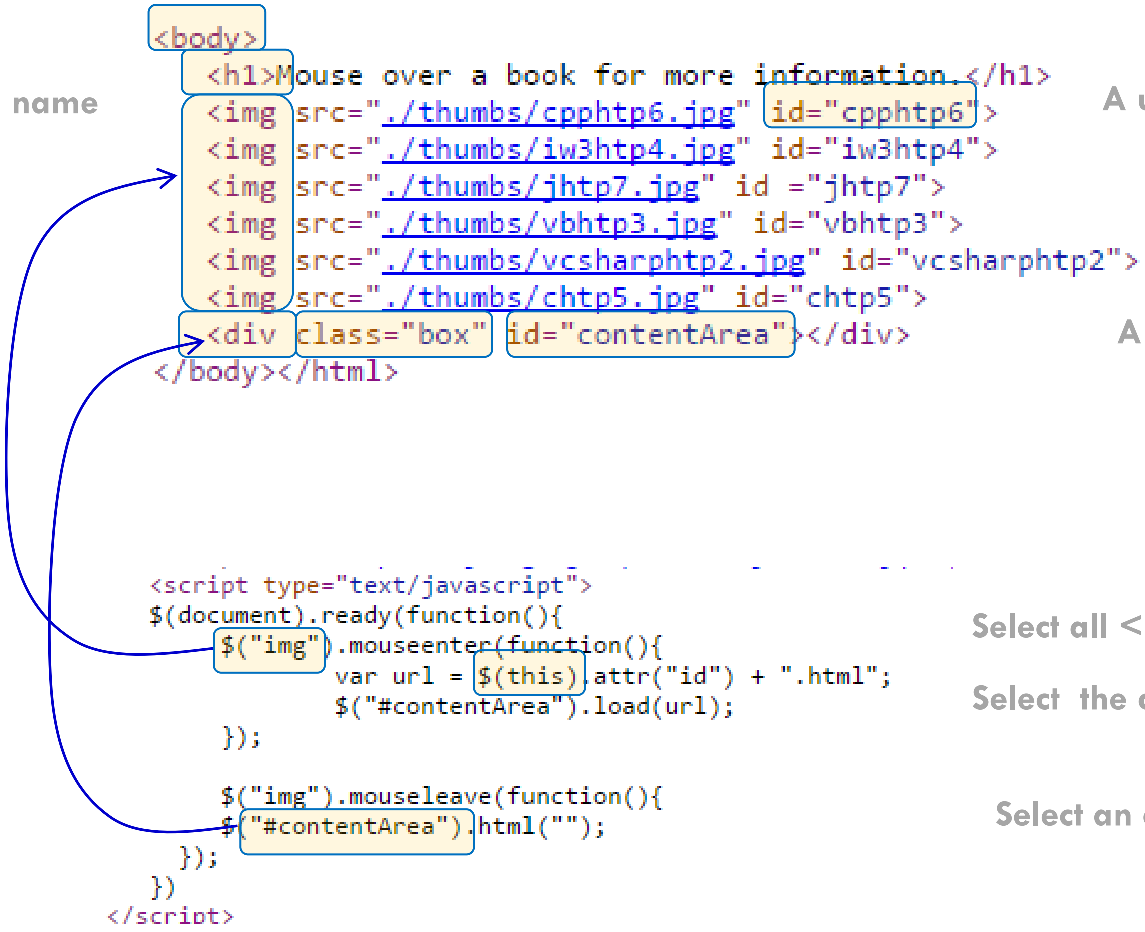


```
<body>
  <h1>Mouse over a book for more information.</h1>
  <img src="./thumbs/cpphtp6.jpg" id="cpphtp6">
  <img src="./thumbs/iw3htp4.jpg" id="iw3htp4">
  <img src="./thumbs/jhtp7.jpg" id ="jhtp7">
  <img src="./thumbs/vbhtp3.jpg" id="vbhtp3">
  <img src="./thumbs/vcsharphtp2.jpg" id="vcsharphtp2">
  <img src="./thumbs/chtp5.jpg" id="chtp5">
  <div class="box" id="contentArea"></div>
</body></html>
```

```
<script type="text/javascript">
$(document).ready(function(){
    $("img").mouseenter(function(){
            var url = $(this).attr("id") + ".html";
            $("#contentArea").load(url);
    });

    $("img").mouseleave(function(){
    $("#contentArea").html("");
  });
})
</script>
```

Selectors in the Example code



```
<body>
   <h1>Mouse over a book for more information.</h1>
   <img src="./thumbs/cpphtp6.jpg" id="cpphtp6">
   <img src="./thumbs/iw3htp4.jpg" id="iw3htp4">
   <img src="./thumbs/jhtp7.jpg" id ="jhtp7">
   <img src="./thumbs/vbhtp3.jpg" id="vbhtp3">
   <img src="./thumbs/vcsharphtp2.jpg" id="vcsharphtp2">
   <img src="./thumbs/chtp5.jpg" id="chtp5">
   <div class="box" id="contentArea"></div>
</body></html>
```

name

A unique id

A class

```
<script type="text/javascript">
$(document).ready(function(){
    $("img").mouseenter(function(){
            var url = $(this).attr("id") + ".html";
            $("#contentArea").load(url);
    });

    $("img").mouseleave(function(){
    $("#contentArea").html("");
  });
})
</script>
```

Select all <img> elements

Select the current element

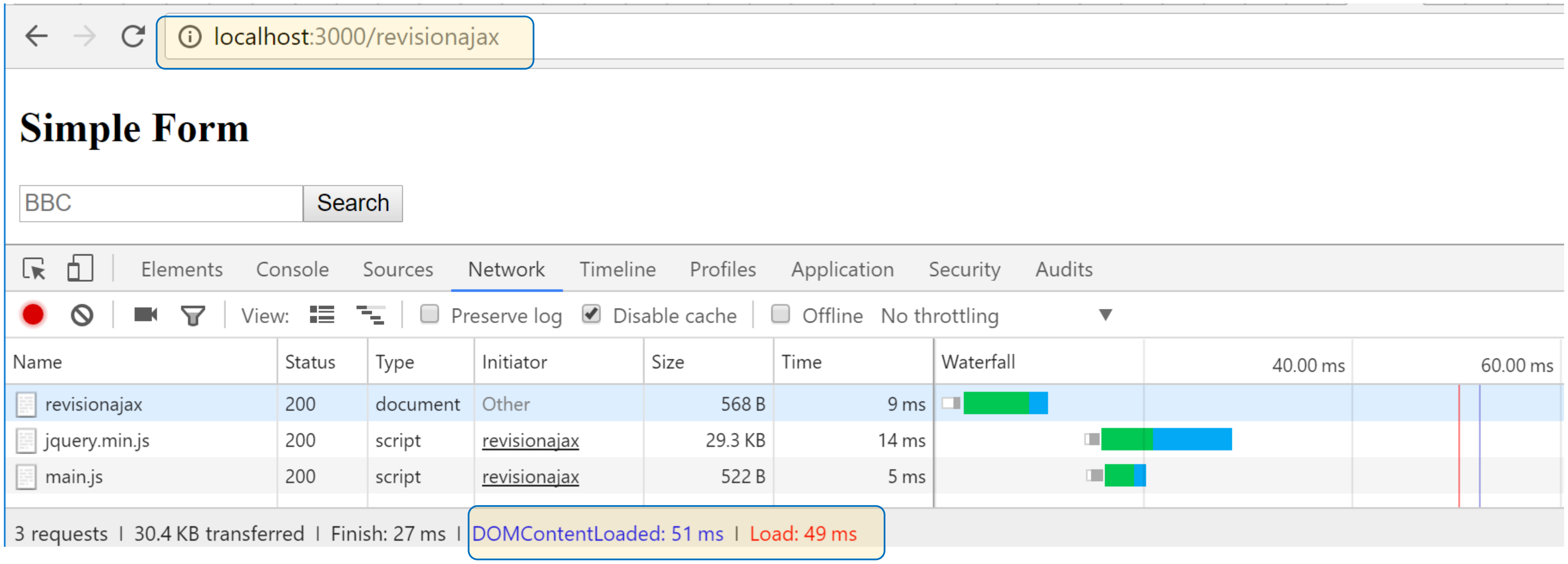
Select an element with id “contentArea”

Outline

- Intro to jQuery
  - Selectors
  - Event handler and DOM man
  - Ajax requests
- Integrate jQuery with Expressjs Application

Express with jQuery

– We can add AJAX support to the simple app (this week tutorial)



localhost:3000/revisionajax

Simple Form

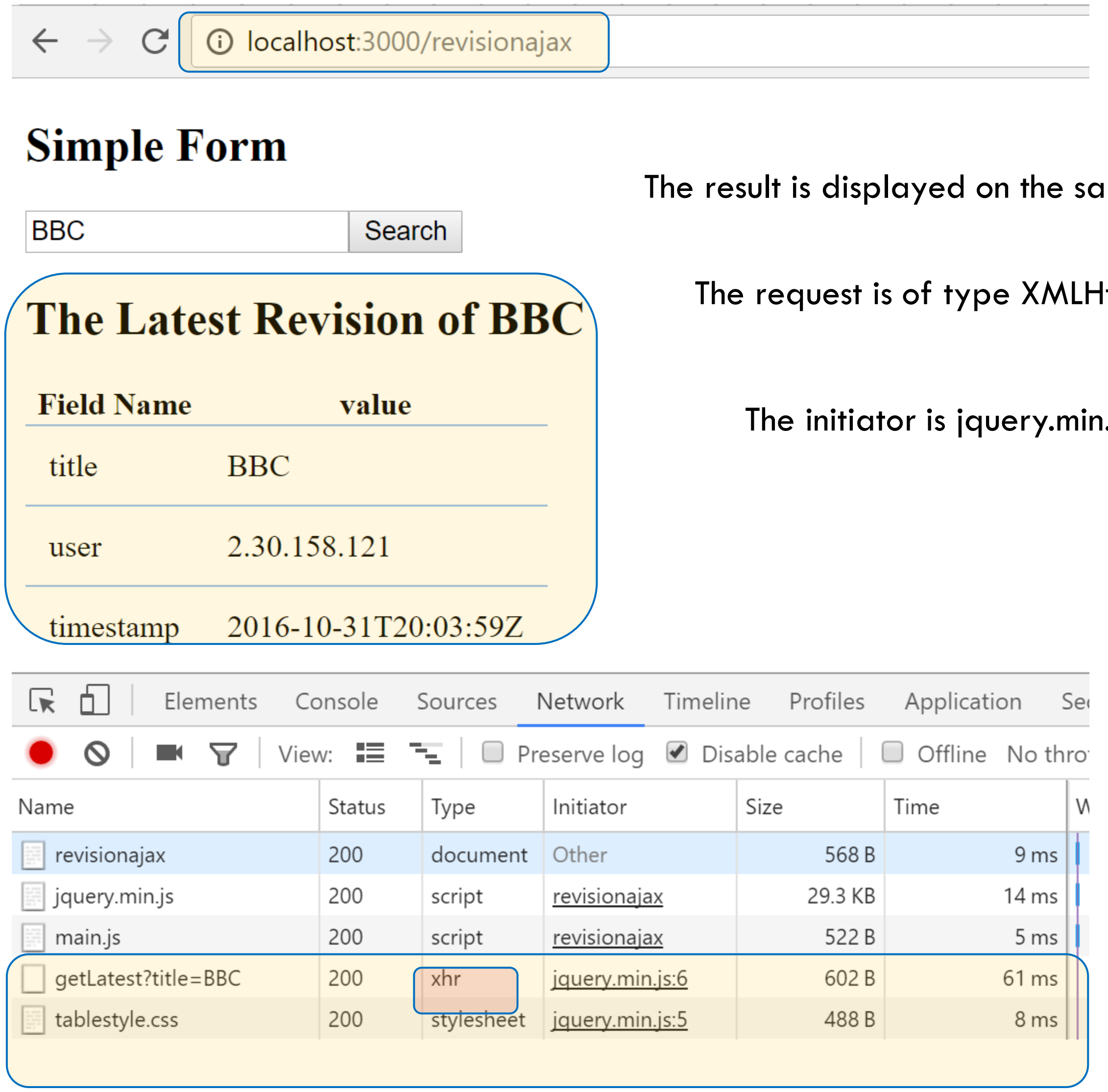
BBC | Search

Name	Status	Type	Initiator	Size	Time
revisionajax	200	document	Other	568 B	9 ms
jquery.min.js	200	script	revisionajax	29.3 KB	14 ms
main.js	200	script	revisionajax	522 B	5 ms

3 requests | 30.4 KB transferred | Finish: 27 ms | DOMContentLoaded: 51 ms | Load: 49 ms

Effect of render blocking JavaScript

AJAX Frontend Output



localhost:3000/revisionajax

Simple Form

BBC | Search

The Latest Revision of BBC

Field Name	value
title	BBC
user	2.30.158.121
timestamp	2016-10-31T20:03:59Z

Elements Console Sources Network Timeline Profiles Application Se
View: Preserve log Disable cache Offline No thro

Name	Status	Type	Initiator	Size	Time
revisionajax	200	document	Other	568 B	9 ms
jquery.min.js	200	script	revisionajax	29.3 KB	14 ms
main.js	200	script	revisionajax	522 B	5 ms
getLatest?title=BBC	200	xhr	jquery.min.js:6	602 B	61 ms
tablestyle.css	200	stylesheet	jquery.min.js:5	488 B	8 ms

The result is displayed on the same page

The request is of type XMLHttpRequest (XHR)

The initiator is jquery.min.js

Changes to the title form view

– Add a place holder for results
– Add reference to scripts
– Change the submit button behaviour

titleFormAjax.pug

```
doctype html

html(lang="en")
head
  title Ajax Search Example
    script(src="https://code.jquery.com/jquery-3.2.1.js")
    script(src="/js/main.js")
body
  h2 Simple Form
  input#title(type="search", placeholder="BBC")
  button#button(type='button') Search
  div#results
```

The Client-side Script

```
$(document).ready(function(){
  $('#button').on('click', function(e){
      var parameters = {title: $('#title').val() };
      $.get( 'revisionajax/getLatest',parameters, function(result) {
      $('#results').html(result);
     });
  });
});
```

```
$(document).ready(function(){
  $('#button').on('click', function(e){
    var data=$('#title').val();
    $('#results').load('revisionajax/getLatest?title='+data)
  });
});
```

The Client-side Script

```
$(document).ready(function(){
  $('#button').on('click', function(e){
      var parameters = {title: $('#title').val() };
      $.get( 'revisionajax/getLatest',parameters, function(result) {
      $('#results').html(result);
    });
  });
});
```

Another way of registering event handler

Get the value of this input item, and construct parameter data based on it

Put the result as the content of this element

Send a get request with data

These two are equivalent. The top one use **`$.get(url, data,callback)`**,
The bottom one uses **`element.load(url)`**

```
$(document).ready(function(){
  $('#button').on('click', function(e){
    var data=$('#title').val();
    $('#results').load('revisionajax/getLatest?title='+data)
  });
});
```

What else do we need to change?

- No major change except a few “wirings”
- The url to controller mapping
- Controller to new view

The jqXHR Object

- All jQuery Ajax requests return a jqXHR object to encapsulate the response from the server
  - jqXHR is a superset of the original **XMLHTTPRequest** object

- jqXHR can be used handle various server responses:
  - **jqXHR.done()** for success
  - **jqXHR.fail()** for error
  - **jqXHR.always()** is like the regular **try-catch-finally** block

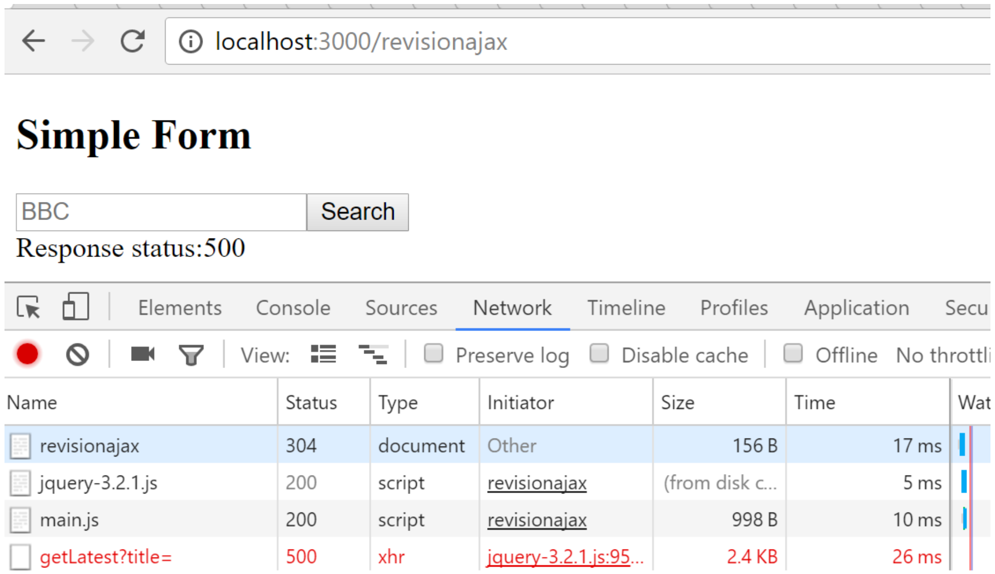
The jqXHR Object Example

```
$(document).ready(function(){
  $('#button').click(function(e){
var parameters = {title: $('#title').val() };
      var jqxhr = $.get( 'revisionajax/getLatest',parameters)
      jqxhr.done(function(result) {
         $('#results').html(result);
      });
      jqxhr.fail(function(jqXHR){
         $('#results').html("Response status:" + jqXHR.status)
      //console.log("Response status:" + jqXHR.status)
      })
  });
});
```

http://api.jquery.com/jQuery.ajax/#jqXHR

The jqXHR Object Example

```
$(document).ready(function(){
  $('#button').click(function(e){
var parameters = {title: $('#title').val() };
      var jqxhr = $.get( 'revisionajax/getLatest',parameters)
      jqxhr.done(function(result) {
         $('#results').html(result);
      });
      jqxhr.fail(function(jqXHR){
         $('#results').html("Response status:" + jqXHR.status)
      //console.log("Response status:" + jqXHR.status)
      })
  });
});
```



localhost:3000/revisionajax

Simple Form

BBC | Search

Response status:500

Name	Status	Type	Initiator	Size	Time	Wat
revisionajax	304	document	Other	156 B	17 ms	
jquery-3.2.1.js	200	script	revisionajax	(from disk c...	5 ms	
main.js	200	script	revisionajax	998 B	10 ms	
getLatest?title=	500	xhr	jquery-3.2.1.js:95...	2.4 KB	26 ms	

http://api.jquery.com/jQuery.ajax/#jqXHR

Same Origin Policy (SOP)

- Cross-origin scripting
  - malicious script (hosted on another domain) try to access the content of other pages on the user's browser
- Important security concept in modern browsers
  - Mostly, restrict what resources JavaScript (and other scripting language) can access inside a browser
    - DOM, Cookie, XMLHttpRequest, and so on
- An origin is defined by protocol, host name and port number
- If two pages are from same origin, the web browser permits scripts from one page to access data in a second page

AJAX – Same Origin Policy

- `XMLHttpRequest` object does not allow a web application to request resources from servers other than the one that served the web application (SOP on XHR)

- Sharing content lawfully between two domains become a challenge
  - E.g., www.funwebdev.com and images.funwebdev.com

AJAX – dealing with SOP

- Implement a server-side proxy— an application on the web application's web server—that can make requests to other servers on the web application's behalf

- **Cross-origin Resource Sharing (CORS)** uses new headers in the HTML5 standard to let site specify other domains that can share its content through JavaScript
  - E.g., Access-Control-Allow-Origin: www.funwebdev.com

Resources

- Randy Connolly, Ricardo Hoar, Fundamentals of Web Development, Global Edition, Pearson

- W3C school jQuery Tutorial
  - http://www.w3schools.com/jquery/default.asp

- jQuery API Documentation
  - http://api.jquery.com/

W8 Tutorial: Mongoose

W9 Lecture: Introduction to React/Angular

W9 Tutorial: jQuery/AJAX