



Chapter 2

jQuery

CT313H – WEB TECHNOLOGIES

Objectives

To introduce **jQuery** and
use jQuery in user and web application interaction

Content

- jQuery introduction
- DOM selection
- DOM element manipulation
- Event processing
- Effects
- AJAX with jQuery
- jQuery plugins

jQuery Introduction

What is jQuery?

- Fast, small, feature-rich JS library
- It makes HTML document traversal, manipulation, event-handling, animation and Ajax much simpler ever



Lightweight Footprint

Only 32kB minified and gzipped.
Can also be included as an AMD module



CSS3 Compliant

Supports CSS3 selectors to find elements as well as in style property manipulation



Cross-Browser

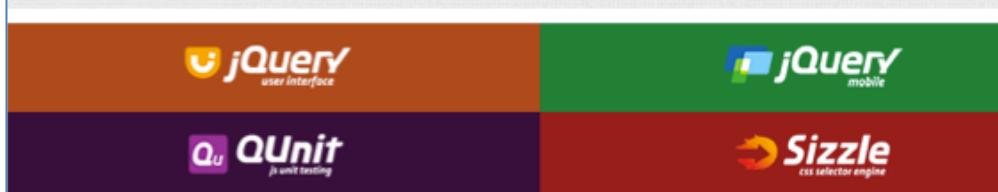
Chrome, Edge, Firefox, IE, Safari, Android, iOS, and more

Why jQuery?

- Simple and easy-to-use
- Free and open source
- Cross-browser
- Versatility
- Extensibility (through plugins)
- Big community ⇒ good support



Other jQuery Foundation Projects



PLATINUM MEMBERS



Who are using jQuery?



How to use jQuery?

1. Offline: download jQuery library

- Access: <https://jquery.com>
- Download the newest version of jQuery
- Unzip the jQUery library
 - *.min.js: used in deployment phase
 - *.js: used in development phase
- Create reference to jQuery library:

```
<head>
  <title>My jQuery page</title>
  <script type="text/javascript"
         src="<path/>jquery-1.x.x-min.js"></script>
  <!-- other scripts -->
</head>
```

How to use jQuery?

2. Online: use CDN (Content Delivery Network)

```
<head>
  <title>My jQuery page</title>
  <script type="text/javascript"
    src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js">
  </script>
  <!-- other scripts -->
</head>
```

- Microsoft CDN:

<https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.2.1.min.js>

Basic syntax

- A jQuery statement:
 1. Selects HTML elements in the webpage
 2. Operate on the selected elements
- Syntax: **`$(selector).action()`**
 - **\$**: indicates using jQuery functions (or `jQuery()` function)
 - **selector**: selects HTML elements to operate (CSS)
 - **action()**: action that will be perform on the selected elements
- Example:
 - `$("p").hide()`: hides all `<p>` elements

Check for document readiness

- HTML elements must be created before jQuery selects and accesses them
- jQuery allows to register a callback function that will be called after the document is ready:

```
$(document).ready(function() {  
    // jQuery methods go here...  
});
```

Shorthand syntax:

```
$(function() {  
    // jQuery methods go here...  
});
```

jQuery Selectors

jQuery Selectors

- Used to select HTML elements to process
- Support all CSS selectors
- Syntax: **`$(selector)`**
- Returned result: a jQuery object
 - Represents for 0 to many elements selected in the page
 - The **length** property is used to identify number of elements
 - Example:

```
$( 'div' ).length
```

returns the number of <div> tags in the webpage

jQuery Selectors

1. Element selector
2. ID selector
3. Class selector
4. Attribute selector
5. Descendent selector
6. Combine selector
7. Form related selectors
8. Selector based on position
9. Others

Element Selector

- Syntax: `$('<HTML tag name>')`
- Select all HTML elements identified by the selector
- Example:
 - `$('p')`: selects all `<p>` tags in the webpage
 - `$('div')`: selects all `<div>` tag in the webpage
 - `$('*')`: selects all elements in the webpage
 - `$(this)`: selects `<html>` tag

* Symbol represents all tag
`this`: `<html>` element

ID Selector and Class Selector

- ID selector:
 - Selects 1 element by id
 - Syntax: `$('#<id>')`
- Class selector:
 - Select all elements belong to the given class
 - Syntax: `$('.<class>')`
- Example:
 - `$('#header')`: select the element with the id is header
 - `$('.test')`: selects all elements belong to class test

Attribute Selector

- Select HTML elements with a desired attributes

Syntax	Selected elements
[attribute]	Have the attribute
[attribute="avalue"]	Have the attribute with the desired avalue
[attribute~= "avalue"]	Have the attribute containing the word avalue
[attribute*= "avalue"]	Have the attribute containing avalue
[attribute^= "avalue"]	Have the attribute beginning with avalue
[attribute\$= "avalue"]	Have the attribute ending with avalue
HTML-tag[attribute-selector]	HTML elements HTML-tag with the attribute selected by the attribute-selector

More ref.: https://www.w3schools.com/cssref/css_selectors.asp

Attribute Selector

- Example:
 - `$(["href"])`: selects all elements that have href attribute
 - `$(["href*='xxx'"])`: selects all element whose href attribute contains 'xxx'
 - `("a[href]")`: selects all `<a>` elements that have href attribute
 - `("input[type='text'])`: selectes all `<input>` elements whose href attribute value is "text"

Descendent selector

space

- Syntax: ances-selector  desc-selector
- Selects all elements that are descendants of a given ancestor
- Note: use symbol **>** instead of space to select **direct** descendants
- Example:
 - `$("ul em")`: selects all `` elements inside `` elements
 - `$("ul>b")`: selects all `` elements that are direct chidlrens of `` elements

Combine selectors and Nested selectors

- Selectors can be combined, for example:
 - `$("p.intro")`: selects `<p>` elements that belong to class intro
 - `$("a[href]")`: selects all `<a>` elements that have href attribute
- Selectors can also be nested:
 - `$("div#header p em.required")`: selects an `` element that belongs to class required and is a descendant of a `<div>` element with the id is “header”
- Selector exercise: <https://testmozusercontent.com/38107/student>

Selectors for Form

- `$(":button")`
- `$(":checkbox")`
- `$(":file")`
- `$(":image")`
- `$(":password")`
- `$(":submit")`
- ...

⇒ Selects all `<input>` with the expected **type**

Position-based selectors

- Select elements based on their position inside the DOM model:
 - `$(".article:eq(2)")`: selects the 3rd element in class article
 - `$(".article.gt(1)")`: selects the elements of class article whose index is greater than 1
 - `$(".article.lt(3)")`: selects 3 first elements of class article
 - `$(".article:first")`: selects the 1st element of class article
 - Some other position-based selectors:
`::after`, `::before`, `:first-child`, `::first-letter`,
`::first-line`, `:last-child`, `:nth-child(n)`, ...

Other selectors

Example	Selected elements
<code>\$('p:contains("Bootstrap")')</code>	<p> tags that contain “Bootstrap”
<code>\$('div:has("h2")')</code>	<div> tags that contain atleast 1 <h2> element
<code>\$('option:not(:selected)')</code>	<option> tags that don't have selected attribute
<code>\$('p:hidden')</code>	<p> tags that are hidden

- `:active, :checked, :disabled, empty, :enabled, :focus, :hover, :invalid, :not(selector), :has(element), :visited, :valid, :required, :`

JQuery traversing methods

- Used to iterate DOM elements in returned by a jQuery selector
- Some related methods:
 - `$('span').parent()`: selects the **direct** parent of
 - `$('.article').children('p')`: selects all <p> elements that are the **direct** children of elements whose class is article
 - `$('.article').find('p')`: selects all <p> elements that are the children of <p> tags belonging to class article
 - `$('.article').siblings('p')`: selects all <p> tags that are **siblings** of the elements belonging to class article

Filter methods

- Used to **filter** elements returned by a jQuery selector:
 - `$('#content h2').first()`: select the 1st child of every `<h2>`, which is the child of an element whose id is content
 - `$('#menu > li').eq(1)`: select all `` elements that are the 2nd immeditely child of an element whose id is menu
 - `$('div').not('.article')`: select `<div>` elements that do not belongs to class article
 - `$('li').has('ul')`: select `` elements that contain some `` elements
 - ...

Traversal and filter methods ~~are slower~~
than the CSS selectors

DOM Element Manipulation

Create DOM element

- Syntax:

```
var <var-name> = $(<element definition>)
```

- Example:

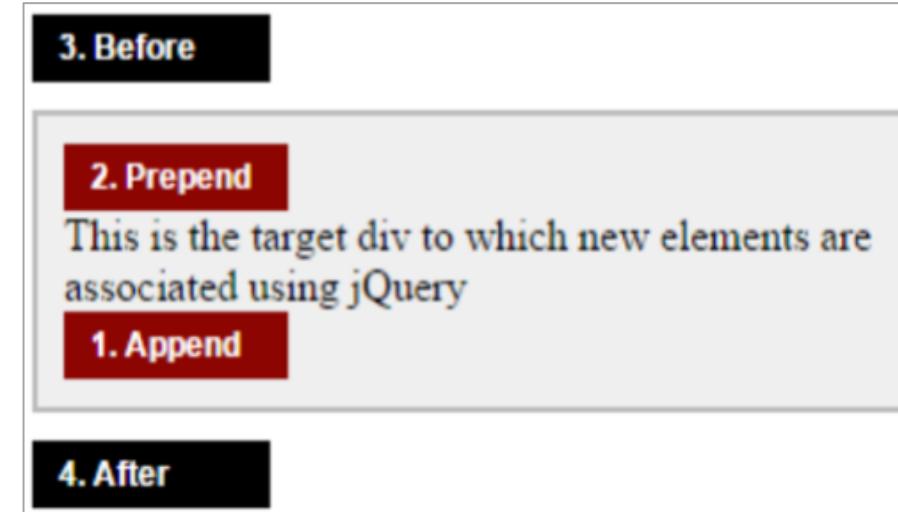
```
var menuli1 = $('<li class="menu-li1">Italian Food</li>');
```

or

```
var menuli1 = $('<li/>', {  
    'class': 'menu-li1',  
    'text': 'Italian Food'  
});
```

Insert elements

- An element can be inserted in the following orders (**sibling**):
 - **Before** an existing element: `insertBefore()`, `before()`
 - **After** an existing element: `insertAfter()`, `after()`
- An element can also be inserted **inside** an existing element (**children**):
 - `appendTo()`, `append()`
Append at the **end**
 - `prependTo()`, `prepend()`
Append at the **beginning**



Insert elements

```
<ul>
  <li class="menu-li1">French Food</li>
  <ul class="menu-ul1">
    <li class="menu-li2"><a href="#">Link1</a></li>
    <li class="menu-li2"><a href="#">Link2</a></li>
    <li class="menu-li2"><a href="#">Link3</a></li>
  </ul>
  <li class="menu-li1">American Food</li>
  <ul class="menu-ul1">
    <li class="menu-li2"><a href="#">Link1</a></li>
    <li class="menu-li2"><a href="#">Link2</a></li>
    <li class="menu-li2"><a href="#">Link3</a></li>
  </ul>
</ul>
```



Insert before (sibling)

- Insert an item **before** the item “French Food”:

- `insertBefore()`:

```
var menuli1 = $('- Italian Food</li>');
menuli1.insertBefore('.menu-li1:first');

```

- `before()`:

```
var menuli1 = $('- Italian Food</li>');
$('.menu-li1:first').before(menuli1);

```

or:

```
$('.menu-li1:first').before(
    '<li class="menu-li1">Italian Food</li>');
```

Insert after (sibling)

- Insert an item after the last item in the menu:

- **insertAfter()**:

```
var menuli1 = $('- Italian Food</li>');
menuli1.insertAfter('.menu-ul1:last');

```

- **after()**:

```
var menuli1 = $('- Italian Food</li>');
$('.menu-ul1:last').after(menuli1);

```

or:

```
$('.menu-ul1:last').after(
    '<li class="menu-li1">Italian Food</li>');
```

Append at the end (children)

- Append an item after the last menu item (Link 3 - American):

```
var menuLi2 = $('<li class="menu-li2">  
    <a href="#">Link4</a></li>');
```

```
menuLi2.appendTo('.menu-ul1:last');
```

Or:

```
$('.menu-ul1:last').append(menuLi2);
```

Shorthand:

```
$('.menu-ul1:last').append(  
    '<li class="menu-li2"><a href="#">Link4</a></li>');
```

Append at the beginning (children)

- Add an item before the first existing item (Link 1 - French):

```
var menuLi2 = $('<li class="menu-li2">  
    <a href="#">Link0</a></li>');
```

```
menuLi2.prependTo('.menu-ul1:first');
```

or:

```
$('.menu-ul1:first').prepend(menuLi2);
```

or:

```
$('.menu-ul1:first').prepend(  
    '<li class="menu-li2"><a href="#">Link0</a></li>');
```

Move elements

- Use the insert/append methods
- Moved element is the method's parameter
- Selectors are used to identify the moved element
- Example: move the 1st item of French food to the end of the list:

```
$('.menu-li1:last').append($('.menu-li1:first'));
```

or:

```
$('.menu-li1:first').appendTo($('.menu-li1:last'));
```



Copy elements

- Copying an element includes 2 steps:
 - 1) **Clone** (copy) the element
 - 2) **Insert** the new element to the desired position

- Example: copy the 1st element of “French Food” to the end of the list:

```
① var f_item=$('#.menu-li2:first').clone();  
② $('#.menu-li2:last').append(f_item);
```

Or:

```
$('#.menu-li2:last').append(                           ①  
                                                               $('.menu-li2:first').clone());  
                                                                 ②
```



Delete and replace elements

- Delete an element:
 - **detach()**: the deleted element can be reused (added back)
 - **remove()**: delete and dispose an element (cannot reuse)
- Replace an element: **replaceWith()**, **replaceAll()**:

```
$('.menu-li1').each(function() {  
    $(this).replaceWith('<h3>' + $(this).text() + '</h3>');  
});
```

```
$('.menu-li1').each(function() {  
    $('<h3>'+$(this).text()+'</h3>').replaceAll($(this));  
});
```

Get element content: HTML vs. text

- Get element content, including the html code: `html()`
- Get element text, , excluding the html code: `text()`
- Example:
 - `$('p').html()`: get the content of the 1st `<p>` element in the page, including the html code (This is a `paragraph.`)
 - `$('p:last').text()`: get the text of the last element in the page, excluding html code (This a another paragraph.)
`<body> <p>This is a paragraph.</p>`
 `<p>This is another paragraph.</p>`
`</body>`

Note: don't use these functions to get value of the `<input>` elements

Set element content: HTML vs. text

- `html()` and `text()` can also be used to set elements' value
- Example:
 - `$('p').html('This is the 1st paragraph')`: the 1st paragraph in the page will display
“This is the 1st paragraph”
 - `$('p').text('This is the 1st paragraph')`: the 1st paragraph in the page will display
“This is the 1 st paragraph”

Get element attributes

- Get the **HTML** attribute:

- **.attr(attrName)**: get the value of the attribute **attrName** of the 1st element in the set of selected elements
- **.attr(attrName, attrVal)**: set the value of the attribute **attrName** of the 1st element in the set of selected elements

- Get the **DOM** attribute:

- **.prop(propName)**: get the value of the attribute **propName** of the 1st element in the selected elements
- **.prop(propName, propVal)**: set the value of the attribute **propName** of the 1st element in the selected elements

DOM properties: selectedIndex, tagName, nodeName, nodeType, ownerDocument, defaultChecked, or defaultSelected

Access element CSS

- Get/Set the **CSS** value:
 - `.css(propName)`: get the value of the CSS property `propName` of an element
 - `.css(propName, propVal)`: set the value for the CSS property `propName` of an element

//Get background value of the 1st <div>

```
$('div').css('backgroundColor');
```

//Set CSS style for the 1st element of the class article

```
$('.article').css({
  'backgroundColor': '#C2F5BF',
  'borderColor': 'yellow',
});
```

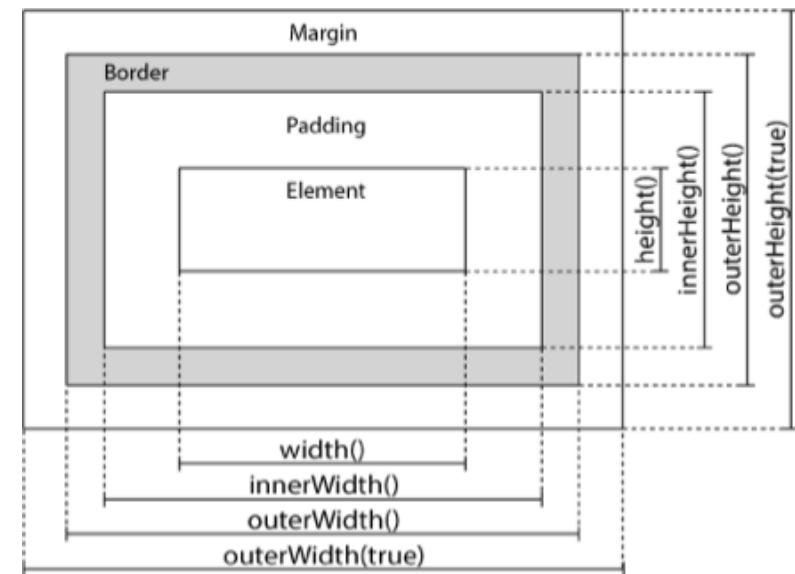
Access element CSS

- Other methods for accessing CSS of elements:
 - **.addClass(`clsName`)**: add element to the CSS class `clsName`
 - **.removeClass(`clsName`)**: remove element from the CSS class `clsName`
 - **.toggleClass(`clsName`)**: toggle the CSS class `clsName` for the element
 - **.hasClass(`clsName`)**: check the element is belonging to the CSS class `clsName` or not (returns true/false)

```
$("button").click(function(){
    $("h1, h2, p").addClass("blue");
    $("div").toggleClass("important");
});
```

Get/Set element size

- **.height()**, **.width()**: get/set element width and height
- **.innerHeight()**, **.innerWidth()**: get/set element width and height + padding
- **.outerHeight()**,
.outerWidth(): get/set element width and height + padding + border + margin



Event Handling

Event handling

- Event handle:
 - We can **bind** a segment of code (function) to an event
 - When the event occurs, the function will be call to **handle** the event
- Syntax to bind a function to an event:

```
$( '<selector>' ).on( '<event>', [ 'child selector' ], function() {  
    //Code to handle the event  
});
```

Hoặc:

```
$( '<selector>' ).<event>([parameter], function() {  
    //Code to handle the event  
});
```

HTML events

Mouse	Keyboard	Form	Window/Document
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

- Example:

```
/* When an element belonging to class header are clicked, its background  
will become yellow */  
$('.header').click(function() {  
    $(this).css('background-color', 'yellow');  
});
```

More example on event handling

```
$('.header').on('click', function() {  
    //Handle the click event for elements of the class header  
});  
  
$('.header').on('click mouseleave', function() {  
    // Handle the click and mouse leave events for elements of the class header  
});  
  
$('.article').on('click', '.header', function() {  
    //Handle the click event for elements of the class header that are the children of class article  
});  
  
$(document).on('click', '.btn-control', function() {  
    // Handle the click event for elements of the class btn-control  
});
```

Event handling for multiple elements

```
$( "p" ).on( {
    mouseenter: function(){
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function(){
        $(this).css("background-color", "lightblue");
    },
    click: function(){
        $(this).css("background-color", "yellow");
    }
});
```

Remove event handler

- Syntax:

```
$( 'selector' ).off( 'events' , 'child-selector' ,  
                      handle-Function)
```

- Example:

.off('click'): remove the `click` event handler

.off('click', `clickHandle`): remove the `clickHandle` function out of the `click` event handler

.off(): remove all event hadlers

Remove event handler

```
<script>
  function changeSize() {
    $(this).animate({fontSize: "+=10px"});
  }

  function changeSpacing() {
    $(this).animate({letterSpacing: "+=5px"});
  }

$(document).ready(function(){
  $("p").on("click", changeSize);
  $("p").on("click", changeSpacing);
  $("button").click(function(){
    $("p").off("click", changeSize);
  });
});
</script>
```

This is a paragraph.

Click any p element to increase size and letterspacing.

[Remove the changeSize\(\) event handler](#)

Event Objects

- Each event handler function receives an **event object** that contains information of the triggered event
- Some basic attributes and method of the event object:
 - **pageX, pageY**: mouse position where the event occur
 - **target**: the DOM element that triggers the event
 - **type**: event type (click, dblclick,...)
 - **data**: data passed when create the event binding
 - **preventDefault()**: prevent the default event handling

Event Objects – Example

```
//display the mouse position while the mouse is moving
$(document).mousemove(function(event){
    $("span").text("X: " + event.pageX +
                  ", Y: " + event.pageY);
});

//disable links' default behaviour and define a new behaviour
$(document).ready(function(){
    $("a").click(function(event){
        event.preventDefault();
        var newDiv = $("<div>default click prevented</div>");
        $("#log").append(newDiv);
    });
});
```

[Go to Google.com](https://google.com/)
default click prvented
default click prvented

```
<body>
  <a href="https://google.com/">Go to Google.com</a>
  <div id="log"></div>
</body>
```

Effects

jQuery Effects

- Popular effects:
 - Hide, show
 - Fade in, fade out
 - Slide up, slide down
 - Animation
 - Toggle
- Typically, an effect function has 2 optional parameters:
 - Effect speed (ms)
 - A callback function that will be called when the effect ends

jQuery Effect – Example

```
$('.menu-ul1').hide();

$('.menu-li1').toggle(function() {
    $(this).next('.menu-ul1').show();
},function(){
    $(this).next('.menu-ul1').hide();
});

$('.menu-li1').toggle(function() {
    $(this).next('.menu-ul1').fadeIn();
},function(){
    $(this).next('.menu-ul1').fadeOut();
});
```

jQuery Effect – Example

```
$('.menu-li1').toggle(function() {
    $(this).next('.menu-ul1').slideDown();
},function() {
    $(this).next('.menu-ul1').slideUp();
});
```

```
$('.menu-li2').click(function() {
    $(this).animate({
        borderLeftWidth:"0px",
        borderRightWidth:"12px"
    }, 600);
});
```

AJAX

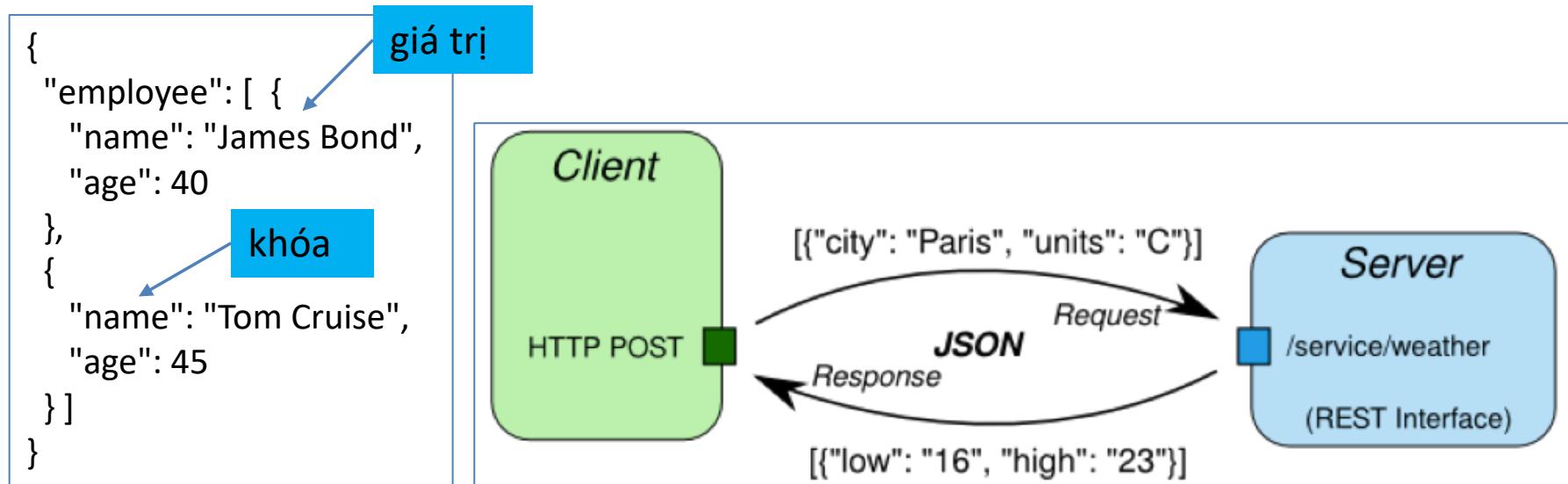
Asynchronous JavaScript And XML

What is AJAX?

- AJAX: Asynchronous JavaScript and XML
- Used to create **asynchronous call** to web servers from web browsers
 - **Asynchronous**: asynchronous **communication**, using the XMLHttpRequest object
 - **JavaScript**: a **programming language** used to create the asynchronous communication from the client side (browser)
 - **XML**: a **markup language** used as a data format in the communication between web browsers and servers (**JSON** is an alternative)

JSON

- JSON: JavaScript Object Notation
 - A light weight format used in storing and transporting data
 - Syntax is similar to JavaScript OOP syntax
 - Easy to transform JSON data to JS objects



jQuery AJAX

- jQuery provides several methods for AJAX functions:
 - `$(“selector”).load() :` load data from the server and put returned data into the selected element
 - `$get()`: requests data from the server using HTTP GET
 - `$post()`: requests data from the server using HTTP POST
 - `$ajax()`: performs an asynchronous AJAX request

jQuery AJAX – load()

- Syntax:

```
$(selector).load(URL, data, callback);
```

- **URL**: URL you wish to load
- **data**: (optional) data to send to the server
- **callback**: (optional) callback function to run when load method is completed
 - **response**: contains the result data from the request
 - **status**: contains the status of the response ("success", "notmodified", "error", "timeout", or "parsererror")
 - **xhr**: contains the XMLHttpRequest object

jQuery AJAX – load()

```
$("#div1").load("demo_test.txt",
    function(responseTxt, statusTxt, xhr) {
        if(statusTxt == "success")
            alert("External content loaded successfully!");
        if(statusTxt == "error")
            alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
});
```

https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_ajax_load_callback

https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_ajax_load_data

jQuery AJAX – get()

- Syntax:

```
$.get(url [, data] [, callback] [, dataType])
```

- **url**: URL you wish to request
- **data**: (optional) data sent along with the request
- **callback**: (optional) callback function to run when load method is completed
 - **response**: contains the result data from the request
 - **status**: contains the status of the response ("success", "notmodified", "error", "timeout", or "parsererror")
 - **xhr**: contains the XMLHttpRequest object

jQuery AJAX – get()

- Request "demo_test.php" and display the returned data

```
$("button").click(function() {  
    $.get("demo_test.php", function(data, status) {  
        alert("Data: " + data + "\nStatus: " + status);  
    });  
});
```

- More examples on get() function:

```
$.get("test.php"); //Request "test.php" and ignore return results  
  
//Request "test.php", pass data to the server (ignore return results):  
$.get("test.php", { name:"Donald", town:"Ducktown" }));  
  
//Request "test.php" and pass arrays of data to the server (ignore return results):  
$.get("test.php", { 'colors[]' : ["Red", "Green", "Blue"] }));
```

jQuery AJAX – ajax()

- Syntax:

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

- Popular parameters (names):
 - **method**: type of request (POST/GET)
 - **url**: URL to send the request to (default: current page)
 - **data**: data to be sent to the server
 - **success(result, status, xhr)**: a function to be run when the request succeeds
 - **error(xhr, status, error)**: a function to run if the request fails

More names: https://www.w3schools.com/jquery/ajax_ajax.asp

jQuery AJAX – ajax()

```
$(document).ready(function(){
    $("button").click(function(){
        $.ajax({url: "jquery_ajax.txt", async: false,
            success: function(result) {
                $("div").html(result);
            }});
    });
});
```

```
<body>
    <div><h2>Let AJAX change this text</h2></div>
    <button>Change Content</button>
</body>
```

jquery_
ajax.txt

AJAX is not a programming language.

It is just a technique for creating better and more interactive web applications.

Let AJAX change this text

Change Content



AJAX is not a programming language.
It is just a technique for creating better and more interactive web applications.

Change Content

jQuery Plugins

jQuery plugins

- A jQuery plugin is a set of jQuery code to provide a particular functionality
- jQuery allows users to create plugins to extend jQuery functionality:
 - effects, presentation, form validation, form auto fill, extend table interaction,...
- Common jQuery libraries:
<https://plugins.jquery.com/>

Common jQuery plugins

- Image:
 - ColorBox (A lightweight customizable lightbox plugin):
<http://www.jacklmoore.com/colorbox/>
 - Cycle (transition effects):
<http://malsup.com/jquery/cycle/download.html>
 - Jcrop (add image cropping functionality to your web app):
http://deepliquid.com/content/Jcrop_Download.html
- Form:
 - JQuery Validation (form validation): <https://jqueryvalidation.org/>
- Table:
 - DataTable (add advanced interaction controls to HTML table):
<https://www.datatables.net/>

jQuery plugin – Example

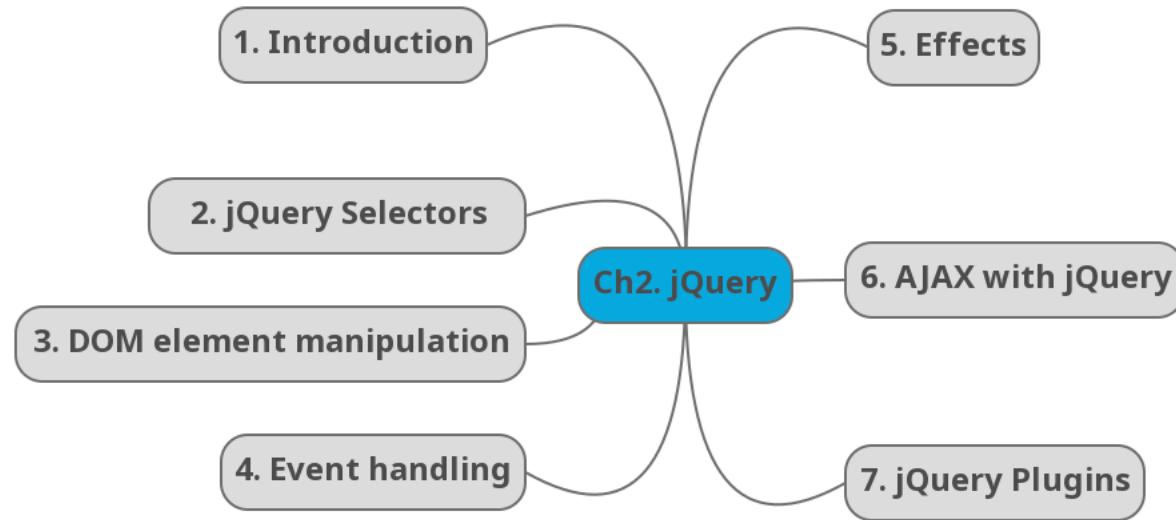
```
<html>
  <head>
    <link rel="stylesheet" href="colorbox.css">
    <script src="jquery.min.js"></script>
    <script src="jquery.colorbox-min.js"></script>
  </head>
  <body>
    <a class='gallery' href='image1.jpg'>Photo_1</a>
    <a class='gallery' href='image2.jpg'>Photo_2</a>
    <a class='gallery' href='image2.jpg'>Photo_2</a>
    <script>
      $('a.gallery').colorbox({rel:'group1'});
    </script>
  </body>
</html>
```

jQuery Plugins – Example



<http://www.jacklmoore.com/colorbox/example1/>

Recap





Question?

CT313H – WEB TECHNOLOGIES