



Chương 2

jQuery

CT275 – CÔNG NGHỆ WEB

Mục tiêu

Giới thiệu **jQuery** và
ứng dụng jQuery vào việc thực hiện
các **tương tác** giữa người dùng và ứng dụng web

Nội dung

- Giới thiệu jQuery
- Lựa chọn các phần tử DOM
- Thao tác trên DOM
- Xử lý sự kiện
- Tạo các hiệu ứng
- Giao tiếp bất đồng bộ (Ajax)
- Sử dụng jQuery plugins

Giới thiệu jQuery

Introduction

jQuery là gì?

- 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

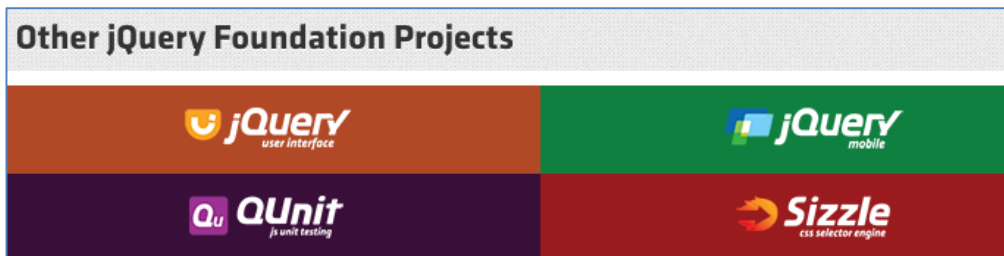


Cross-Browser

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

Tại sao sử dụng jQuery?

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



Ai đang sử dụng jQuery?

Google



ESPN



NETFLIX



Bank of America



DELL

amazon

Làm sao để sử dụng jQuery?

1. Offline: tải thư viện jQuery

- Truy cập trang website: <https://jquery.com>
- Tải phiên bản jQuery mới nhất
- Giải nén thư viện jQuery
 - *.min.js: dùng khi triển khai ứng dụng
 - *.js: dùng khi phát triển ứng dụng
- Tham chiếu thư viện jQuery vào ứng dụng:

```
<head>  
  <title>My jQuery page</title>  
  <script type="text/javascript"  
    src="<đường dẫn/>jquery-1.x.x-min.js"></script>  
  <!-- các script khác -->  
</head>
```


Làm sao để sử dụng jQuery?

2. Online: sử dụng CDN (Content Delivery Nertwork)

```
<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>
  <!-- các script khác -->
</head>
```

- Microsoft CDN:

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

Cú pháp cơ bản

- Một câu lệnh jQuery:
 1. Chọn thành phần của trang web (HTML elements)
 2. Thực hiện thao tác/xử lý trên thành phần được chọn
- Cú pháp: **`$(selector).action()`**
 - **`$`**: ký hiệu sử dụng chức năng jQuery (jQuery() function)
 - **`selector`**: truy vấn (tìm) phần tử HTML cần xử lý (CSS)
 - **`action()`**: thao tác cần thực hiện trên các phần tử HTML được lựa chọn bởi các selector
- Ví dụ:
 - **`$("p").hide()`**: ẩn tất cả các phần tử `<p>` trong trang web

Tài liệu đã sẵn sàng chưa?

- Các phần tử HTML **phải được tạo ra trước** khi jQuery chọn và thao tác trên chúng
- jQuery cho phép đăng ký **hàm callback** sẽ được thực thi khi tài liệu được tải xong:

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

Cú pháp **rút gọn**:

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

Bộ chọn JQuery

jQuery Selector

Bộ chọn jQuery

- Cho phép chọn các phần tử HTML để thao tác
- Hỗ trợ hầu hết các bộ chọn CSS (CSS selector)
- Cú pháp: **`$(selector)`**
- Kết quả trả về là 1 **đối tượng jQuery**
 - Đại diện cho **0 hoặc nhiều** phần tử trong trang web
 - Sử dụng thuộc tính **`length`** để xác định số phần tử
 - Ví dụ:
`$('div').length`
trả về số thẻ `<div>` trong trang web

Các loại bộ chọn trong jQuery

- Element selector
- ID selector
- Class selector
- Attribute selector
- Descendent selector
- Kết hợp các selector
- Bộ chọn liên quan đến form
- Bộ chọn dựa trên vị trí
- Các bộ chọn khác

Element Selector

- Cú pháp: `$('<tên phần tử HTML>')`
- Chọn tất cả các **phần tử** (thẻ) trong trang web được chỉ định bởi selector
- Ví dụ:
 - `$('p')`: chọn tất cả các thẻ `<p>` trong trang web
 - `$('div')`: chọn tất cả các thẻ `<div>` trong trang web
 - `$('*')`: chọn tất cả các phần tử trong trang web
 - `$(this)`: chọn phần tử `<html>` của trang web

Ký hiệu ***** thay thế cho tất cả các thẻ
this: phần tử `<html>` của trang web

ID Selector và Class Selector

- ID selector:

- Chọn **1 phần tử** trong trang web **có id** được chỉ định bởi selector
- Cú pháp: `$('#<id>')`

- Class selector:

- Chọn các phần tử **thuộc lớp** được chỉ định bởi selector
- Cú pháp: `$('<class>')`

- Ví dụ:

- `$('#header')`: chọn phần tử có id là **header** trong trang web
- `$('.test')`: chọn các phần tử thuộc lớp **test** trong tài liệu

Attribute Selector

- Chọn các phần tử HTML có thuộc tính được chỉ định bởi selector

Cú pháp	Phần tử HTML được chọn
[attribute]	Có thuộc tính attribute
[attribute="avalue"]	Có thuộc tính attribute có giá trị là avalue
[attribute~="avalue"]	Có thuộc tính attribute có chứa từ avalue
[attribute*="avalue"]	Có thuộc tính attribute có chứa avalue
[attribute^="avalue"]	Có thuộc tính attribute bắt đầu bằng avalue
[attribute\$="avalue"]	Có thuộc tính attribute kết thúc bằng avalue
HTML-tag[attribute-selector]	Là phần tử là HTML-tag, và có thuộc tính được chỉ định bởi attribute-selector


Xem thêm: https://www.w3schools.com/cssref/css_selectors.asp

Attribute Selector

- Ví dụ:

- `$([href])`: chọn tất cả các phần tử có thuộc tính href
- `$([href*="xxx"])`: chọn tất cả các phần tử có thuộc tính href có chứa giá trị 'xxx'
- `$(a[href])`: chọn tất cả các phần tử <a> có thuộc tính href
- `$(input[type="text"])`: chọn tất cả các phần tử <input> và có thuộc tính type có giá trị là text

Descendent selector

- Cú pháp: `ances-selector`  `desc-selector`
- Chọn tất cả các thành phần **desc-selector** là “con cháu” (descendant) của **ances-selector**
- Lưu ý: dùng ký hiệu **>** thay cho khoảng trắng để chọn con trực tiếp
- Ví dụ:
 - `$(ul em)`: chọn tất cả các thành phần `` nằm trong các danh sách ``
 - `$(ul>b)`: chọn các phần tử `` là con trực tiếp của ``

Kết hợp nhiều selector và Selector nhiều cấp

- Có thể kết hợp nhiều selector
 - `$(p.intro)`: chọn các phần tử `<p>` thuộc lớp `intro`
 - `$(a[href])`: chọn các phần tử `<a>` có thuộc tính `href`
- Các selector có thể lồng nhau nhiều cấp
 - `$(div#header p em.required)`: chọn các phần tử `` thuộc lớp `required` và là hậu duệ của một phần tử `<p>` là con cháu của phần tử `<div>` có `id` là `header`
- Bài tập về selector: <https://testmozusercontent.com/38107/student>

Các bộ chọn liên quan đến Form

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

⇒ Chọn tất cả các **<input>** có **type** tương ứng

Các bộ chọn dựa trên vị trí

- jQuery cho phép lựa chọn các phần tử dựa trên vị trí của chúng trong tập hợp:
 - `$(' .article:eq(2) ')`: chọn các phần tử thứ 3 thuộc lớp article (có chỉ số trong tập các phần tử .article là 2)
 - `$(' .article.gt(1) ')`: chọn tất cả các phần tử thứ 2 trở đi thuộc lớp article (chỉ số > 1)
 - `$(' .article.lt(3) ')`: chọn 3 phần tử đầu tiên thuộc lớp article
 - `$(' .article:first ')`: chọn p/tử đầu tiên có class là article
 - Một số selector khác:
`::after`, `::before`, `:first-child`, `::first-letter`,
`::first-line`, `:last-child`, `:nth-child(n)`,...

Các bộ chọn khác

Ví dụ	Phần tử HTML được chọn
<code>\$('p:contains("Bootstrap")')</code>	Thẻ <p> chứa chuỗi “Bootstrap”
<code>\$('div:has("h2")')</code>	Thẻ <div> có chứa ít nhất một phần tử <h2>
<code>\$('option:not(:selected)')</code>	Thẻ <option> không có thuộc tính selected
<code>\$('p:hidden')</code>	Các phần tử <p> đang ẩn

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

Duyệt DOM

- Ta có thể duyệt qua các thành phần trong kết quả trả về của một bộ chọn jQuery
- Các phương thức jQuery để duyệt qua các phần tử:
 - `$('span').parent()`: chọn phần tử cha trực tiếp của từng thẻ ``
 - `$('.article').children('p')`: chọn các phần tử `<p>` là con trực tiếp của các phần tử có class là `.article`
 - `$('.article').find('p')`: chọn các phần tử `<p>` là hậu duệ của các phần tử có class là `.article`
 - `$('.article').children('p')`: chọn các phần tử `<p>` theo ngay phía sau các phần tử `<h2>`

Các phương thức lọc

- Ta cũng có thể **lọc** ra một số các phần tử trong kết quả của một câu truy vấn jQuery để thao tác:
 - `$('#content h2').first()`: chọn phần tử đầu tiên trong các phần tử `<h2>` là con/cháu của phần tử có id là content
 - `$('#menu > li').eq(1)`: chọn phần tử `` thứ 2 là con trực tiếp của phần tử có id là menu
 - `$('div').not('.article')`: chọn các phần tử `<div>` không thuộc class `.article`
 - `$('li').has('ul')`: chọn các phần tử `` có chứa các phần tử ``
 - ...

Thường các hàm duyệt và lọc DOM có hiệu suất thấp hơn các bộ lọc CSS

Thao tác các phần tử DOM

DOM Element Manipulation

Tạo phần tử

- Cú pháp:

```
var <tên biến> = $(<định nghĩa phần tử>)
```

- Ví dụ:

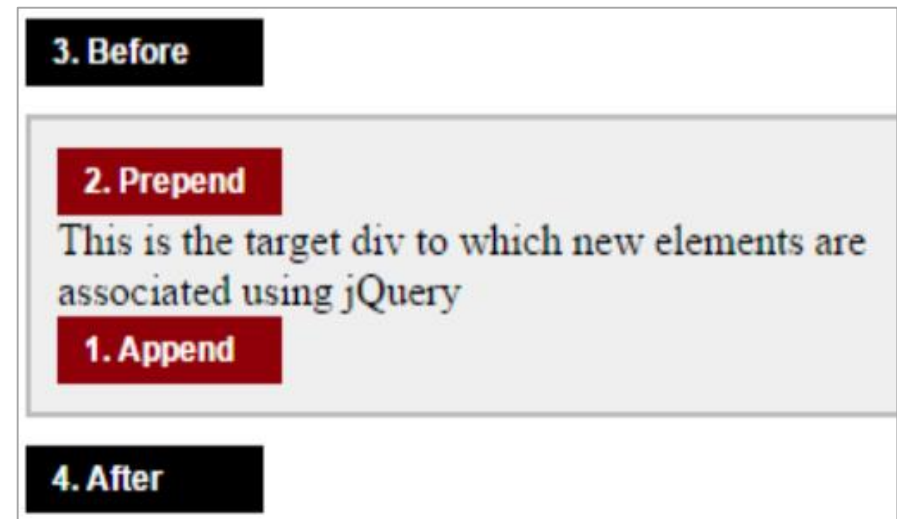
```
var menuLi1 = $( ' <li class="menu-li1">Italian Food</li> ' );
```

hoặc

```
var menuLi1 = $( '<li/>', {  
    'class': 'menu-li1',  
    'text': 'Italian Food'  
});
```

Chèn các phần tử

- Một phần tử có thể được chèn vào theo các thứ tự:
 - **Trước** một phần tử đã tồn tại: `insertBefore()`, `before()`
 - **Sau** một phần tử đã tồn tại: `insertAfter()`, `after()`
- Trong một phần tử nào đó:
 - `appendTo()`, `append()`
Chèn tại vị trí **cuối**
 - `prependTo()`, `prepend()`
Chèn tại vị trí **đầu**



Chèn các phần tử

```

<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>

```

French Food
Link1
Link2
Link3
American Food
Link1
Link2
Link3

Chèn vào trước một phần tử có sẵn

- Chèn 1 menu item vào **trước** “French Food”:
 - `insertBefore()`:

```
var menuLi1 = $('<li class="menu-li1">Italian Food</li>');  
menuLi1.insertBefore('.menu-li1:first');
```
 - `before()`:

```
var menuLi1 = $('<li class="menu-li1">Italian Food</li>');  
$('.menu-li1:first').before(menuLi1);
```
- hoặc:

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

Chèn vào sau một phần tử có sẵn

- Chèn 1 menu item vào **sau** menu item cuối cùng:

- **insertAfter():**

```
var menuLi1 = $('<li class="menu-li1">Italian Food</li>');  
menuLi1.insertAfter('.menu-ul1:last');
```

- **after():**

```
var menuLi1 = $('<li class="menu-li1">Italian Food</li>');  
$('.menu-ul1:last').after(menuLi1);
```

hoặc:

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

Chèn vào như phần tử con cuối cùng

- Chèn 1 menu item vào **sau** menu item cuối cùng:

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

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

hoặc:

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

hoặc:

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


Chèn vào như phần tử con đầu tiên

- Chèn 1 menu item vào **sau** menu item cuối cùng:

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

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

hoặc:

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

hoặc:

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

Di chuyển các phần tử

- Dùng **các hàm chèn** phần tử
- Phần tử được di chuyển là **tham số** của hàm chèn phần tử
- Phần tử này là kết quả của một sự lựa chọn phần tử
- Ví dụ: chuyển phần tử đầu tiên của “French Food” xuống cuối cùng:

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

Hoặc:

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

French Food	
Link1	
Link2	
Link3	
American Food	
Link1	
Link2	
Link3	

Sao chép các phần tử

- Gồm 2 bước:
 - 1) Tạo bản sao của phần tử
 - 2) Chèn bản sao vào vị trí thích hợp
- Ví dụ: sao chép phần tử đầu tiên của “French Food” xuống cuối cùng:

French Food	
	Link1
	Link2
	Link3
American Food	
	Link1
	Link2
	Link3

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

Hoặc:

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

①
②

Loại bỏ, thay thế các phần tử

- Loại bỏ phần tử:
 - **detach()**: Phần tử có thể được thêm ngược trở lại trang
 - **remove()**: loại bỏ hẳn, không thể sử dụng lại
- Thay thế phần tử: **replaceWith()**, **replaceAll()**:

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

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

Lấy nội dung phần tử: HTML vs. text

- Truy xuất nội dung của phần tử, kể cả mã html: **html()**
- Truy xuất text của phần tử, không kể mã html: **text()**
- Ví dụ:
 - `$('p').html()`: trả về nội dung của phần tử `<p>` đầu tiên trong trang web, kể cả mã html (This is a ``paragraph.``)
 - `$('p:last').text()`: trả về nội dung của phần tử `<p>` cuối cùng trong trang web, không bao gồm các thẻ html (This a another paragraph.)

```
<b>body</b> <p>This is a <b>paragraph.</b></p>  
           <p>This is <b>another paragraph.</b></p>  
</b>
```

Lưu ý: không dùng 2 hàm này để lấy giá trị các `<input>`

Gán nội dung phần tử: HTML vs. text

- Các hàm: `html()` và `text()` cũng có thể được sử dụng để đặt nội dung cho các phần tử
- Ví dụ:
 - `$('p').html('This is the 1st paragraph')`: nội dung của phần tử `<p>` đầu tiên trong trang web khi hiển thị ra sẽ là “This is the 1st paragraph”
 - `$('p').text('This is the 1st paragraph')`: nội dung của phần tử `<p>` đầu tiên trong trang web khi hiển thị ra sẽ là “This is the 1 `st` paragraph”

Truy xuất thuộc tính của một phần tử

- Lấy thuộc tính **HTML**:
 - **.attr(attrName)**: lấy giá trị thuộc tính **attrName** của phần tử đầu tiên trong các phần tử được chọn
 - **.attr(attrName, attrVal)**: gán giá trị **attrVal** cho thuộc tính **attrName** của p/tử đầu tiên trong các p/tử được chọn
- Lấy thuộc tính **DOM**:
 - **.prop(propName)**: lấy giá trị của thuộc tính **propName** của phần tử đầu tiên trong các phần tử được chọn
 - **.prop(propName, propVal)**: đặt giá trị **propVal** cho thuộc tính **propName** của p/tử đầu tiên trong các p/tử được chọn

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

Truy xuất thuộc tính của một phần tử

- Lấy thuộc tính **CSS**:

- **.css(propName)**: lấy giá trị của thuộc tính CSS **propName** của phần tử đầu tiên trong các phần tử được chọn
- **.css(propName, propVal)**: gán giá trị **propVal** cho thuộc tính **propName** của p/tử đầu tiên trong các p/tử được chọn

//Lấy background của phần tử <div> đầu tiên

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

//Thiết đặt style cho phần tử đầu tiên thuộc lớp article

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


Thao tác với các lớp CSS

- Một số hàm thao tác trên lớp CSS của các phần tử:
 - **.addClass(className)**: thêm phần tử vào lớp **className**
 - **.removeClass(className)**: xóa phần tử ra khỏi lớp **className**
 - **.toggleClass(className)**: “bật/tắt” **className** của một phần tử
 - **.hasClass(className)**: kiểm tra một phần tử có thuộc lớp **className** hay không (true/false)

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

Kích thước các phần tử

- **.height(), .width()**: truy xuất chiều dài, chiều rộng của phần tử
- **.innerHeight(), .innerWidth()**: truy xuất chiều dài, rộng + padding
- **.outerHeight(), .outerWidth()**: truy xuất chiều dài, rộng + padding, border + margin

Xử lý sự kiện

Event Handling

Gắn kết xử lý sự kiện

- Xử lý sự kiện:
 - JS cho phép kết hợp một đoạn mã lệnh (hàm) với 1 sự kiện
 - Khi sự kiện xảy ra, mã lệnh đó sẽ được gọi để xử lý sự kiện

- Cú pháp:

```
$( '<selector>' ).on( '<event>', [ 'child selector' ], function() {  
    //Đoạn mã Lệnh xử Lý sự kiện  
});
```

Hoặc:

```
$( '<selector>' ).<event>( [parameter], function() {  
    //Đoạn mã Lệnh xử Lý sự kiện  
});
```

Các loại sự kiện

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

- Ví dụ

/ Khi click vào phần tử thuộc lớp header thì màu nền của phần tử sẽ chuyển thành màu vàng */*

```
$('.header').click(function() {  
    $(this).css('background-color', 'yellow');  
});
```

Ví dụ

```
$('.header').on('click', function() {  
    //Xử lý sự kiện click của các phần tử thuộc lớp header  
});
```

```
$('.header').on('click mouseleave', function() {  
    //Xử lý sự kiện click và mouse leave của các phần tử thuộc lớp header  
});
```

```
$('.article').on('click', '.header', function() {  
    //Xử lý sự kiện click của các phần tử thuộc lớp header là con/cháu của lớp article  
});
```

```
$(document).on('click', '.btn-control', function() {  
    //Xử lý sự kiện click cho các phần tử thuộc lớp btn-control  
});
```

Gắn kết nhiều sự kiện cho cùng phần tử

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

Gỡ bỏ xử lý sự kiện

- Để **gỡ bỏ xử lý sự kiện** đã gán trước đó:

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

- Ví dụ:

`.off('click')`: loại bỏ xử lý sự kiện **click**

`.off('click', clickHandle)`: bỏ hàm **clickHandle** ra khỏi danh sách các hàm xử lý sự kiện **click**

`.off('click')`: loại bỏ tất cả các xử lý sự kiện

Gỡ bỏ xử lý sự kiện

```
<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

Đối tượng sự kiện (Event Object)

- Các hàm xử lý sự kiện sẽ được nhận vào 1 đối tượng chứa các thông tin liên quan đến sự kiện
- Một số thuộc tính, phương thức của đối tượng này:
 - **pageX, pageY**: tọa độ vị trí chuột khi xảy ra sự kiện
 - **target**: phần tử DOM phát sinh sự kiện
 - **type**: loại sự kiện (click, dblclick,...)
 - **data**: dữ liệu được truyền vào khi gắn kết sự kiện
 - **preventDefault()**: ngăn ngừa xử lý mặc định của sự kiện

Đối tượng sự kiện (Event Object)

//hiển thị tọa độ của con trỏ chuột khi chuột di chuyển

```
$(document).mousemove(function(event){
    $("span").text("X: " + event.pageX +
                    ", Y: " + event.pageY);
});
```

//disable chức năng của link, thay bằng chức năng khác

```
$(document).ready(function(){
    $("a").click(function(event){
        event.preventDefault();
        var newDiv = $("<div>default click prevented</div>E");
        $("#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>
```

Hiệu ứng

Effects

Các hiệu ứng trong jQuery

- Một số hiệu ứng cơ bản trong jQuery:
 - Ẩn (hide), hiện (show)
 - Hiệu ứng làm mờ dần, hiện dần (fade in, fade out)
 - Hiệu ứng trượt (slide up/down)
 - Hiệu ứng thay đổi hình dạng (animation)
 - Bật/Tắt các hiệu ứng (toggle)
- Thông thường các hàm hiệu ứng của jQuery nhận vào 2 **tham số** tùy chọn:
 - **Tốc độ** hiệu ứng (ms)
 - Hàm **callback**, sẽ được gọi khi kết thúc hiệu ứng

Ví dụ

```
$('.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();  
});
```

Ví dụ

```
$('.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

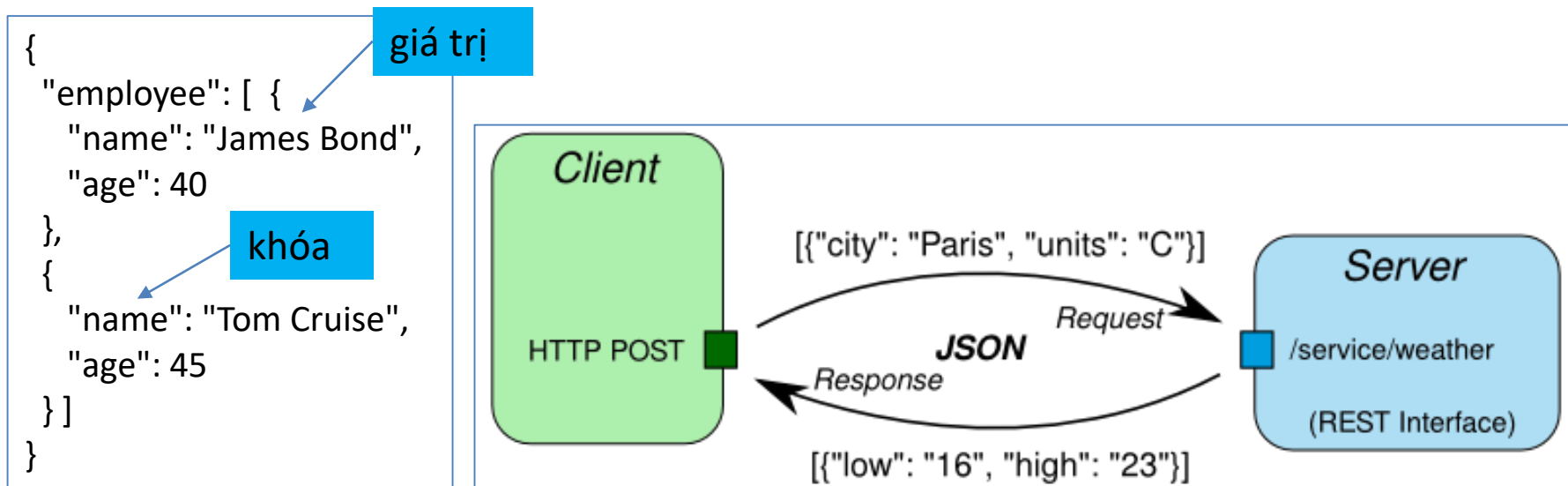
AJAX là gì?

- AJAX: Asynchronous JavaScript and XML
- Là sự kết hợp các công nghệ cho phép thực hiện các lời gọi bất đồng bộ đến web server từ các web browser
 - **Asynchronous**: giao tiếp bất đồng bộ, dùng đối tượng XMLHttpRequest
 - **JavaScript**: ngôn ngữ dùng để thực hiện lời gọi bất đồng bộ từ web browser
 - **XML**: định dạng dùng để trao đổi dữ liệu giữa browser và server (ngoài ra, có thể dùng JSON)

JSON là gì?

- JSON: JavaScript Object Notation

- Là một định dạng trao đổi dữ liệu dựa trên văn bản
- Có cú pháp tương đồng với cú pháp tạo các đối tượng trong JavaScript
- Dễ dàng chuyển đổi dữ liệu JSON với các đối tượng JavaScript



AJAX với jQuery

- Một số hàm jQuery hỗ trợ cho AJAX:
 - `$("#selector").load()`: tải dữ liệu từ máy chủ và thay đổi nội dung HTML của phần tử bằng dữ liệu trả về
 - `$get()`: tải dữ liệu từ máy chủ sử dụng HTTP GET
 - `$post()`: tải dữ liệu từ máy chủ sử dụng HTTP POST
 - `$ajax()`: gửi một yêu cầu HTTP bất đồng bộ đến máy chủ

AJAX với jQuery – load()

- Phương thức load():

`$(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

AJAX với jQuery – 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

AJAX với jQuery – get()

- Cú pháp:

`$.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

AJAX với jQuery – get()

- Yêu cầu "demo_test.asp" và hiển thị KQ trả về:

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

- Ví dụ khác:

```
$.get("test.php"); //Request "test.php" and ignore return results
```

```
//Request "test.php", pass data to the server and ignore return results:
```

```
$.get("test.php", { name:"Donald", town:"Ducktown" });
```

```
//Request "test.php" with data and alert the result of the request:
```

```
$.get("test.php", { 'colors[]' : ["Red", "Green", "Blue"] });
```

AJAX với jQuery – ajax()

- Cú pháp:

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

- Các đối số (name) thông dụng:

- **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

AJAX với jQuery – 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

<p>AJAX is not a programming language.</p>

<p>It is just a technique for creating better and more interactive web applications.</p>

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 Plugin

- Là một tập mã lệnh jQuery nhằm cung cấp 1 tính năng nào đó
- jQuery cho phép người dùng tạo các plugin để bổ sung thêm các chức năng
 - tạo hiệu ứng, trình chiếu, kiểm tra tính hợp lệ của dữ liệu form, tự động điền thông tin cho form, mở rộng khả năng tương tác với bảng,...
- Thư viện các jQuery plugin thông dụng:
<https://plugins.jquery.com/>

Các plugin phổ biến

- Hỗ trợ làm việc với hình ảnh:
 - 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

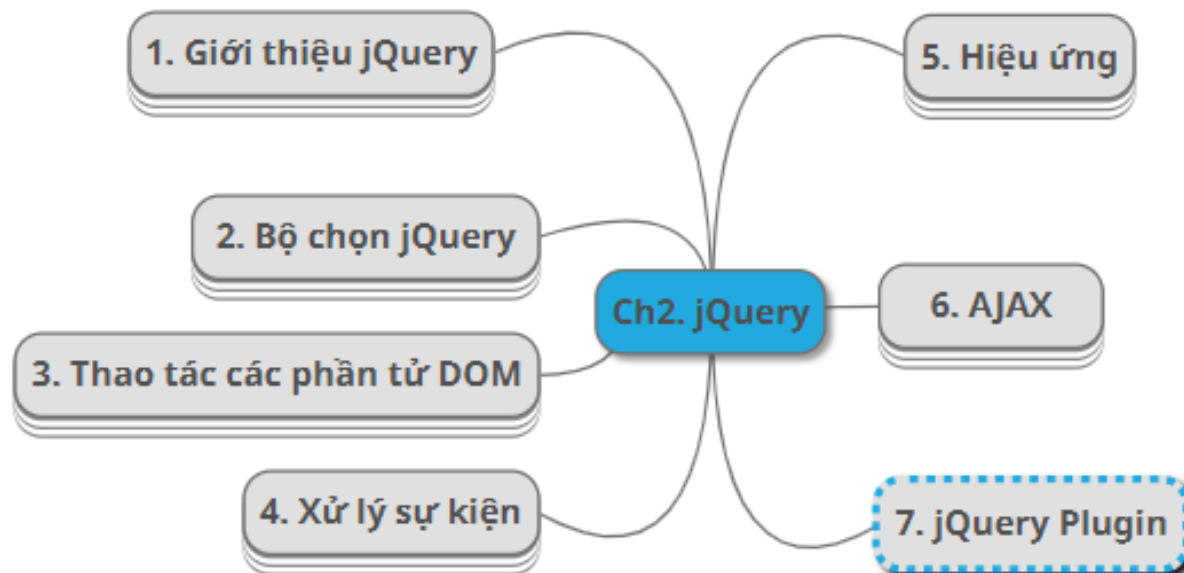
```
<html>
  <head>
    <link rel="stylesheet" href="colorbox.css">
    <script src="jquery.min.js"></script>
    <script src="jquery.colorbox-min.js"></script>
  </head>
  <body>
    <a class='gallery' href='image1.jpg'>Photo_1</a>
    <a class='gallery' href='image2.jpg'>Photo_2</a>
    <a class='gallery'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

jQuery Plugin

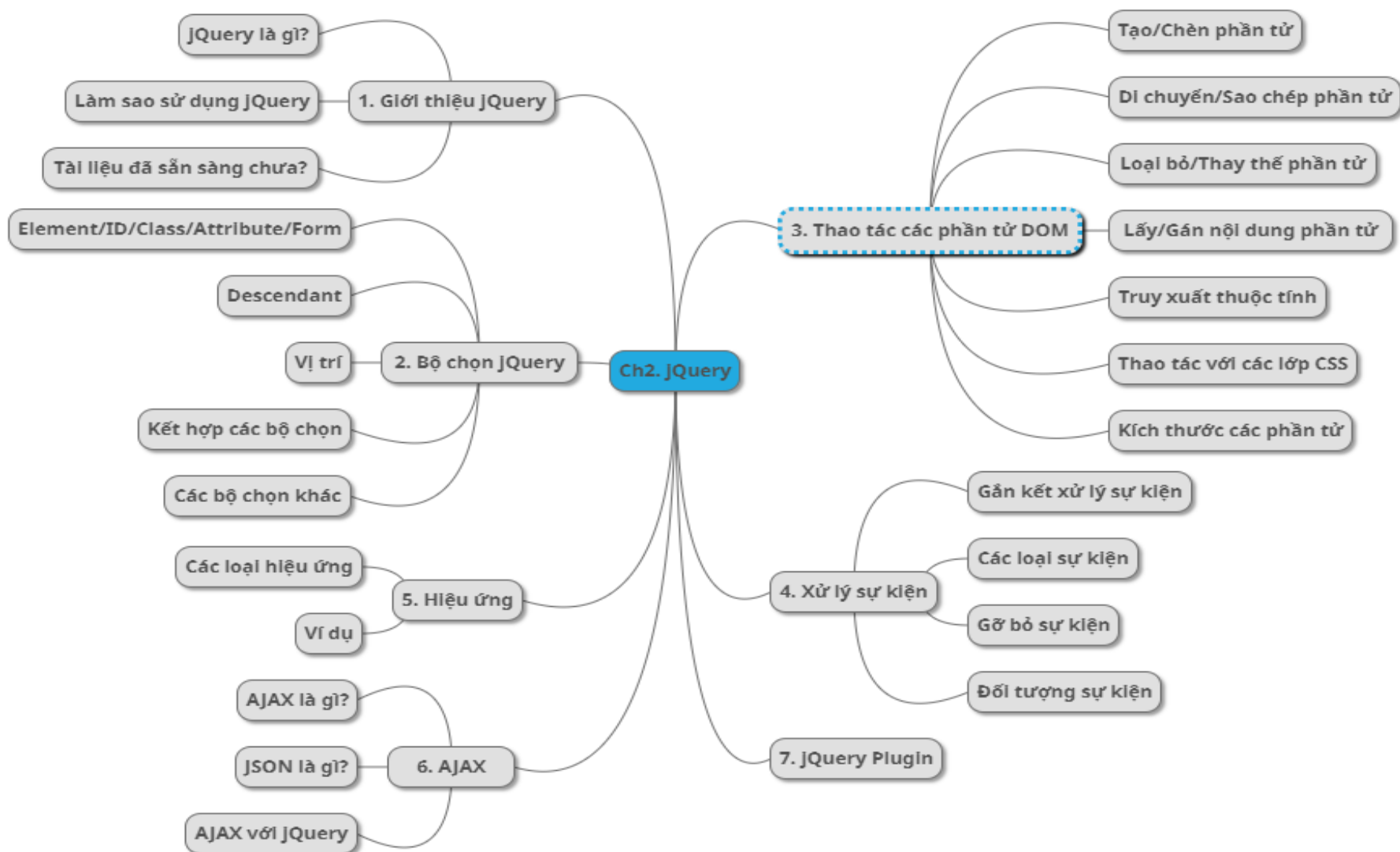


<http://www.jacklmoore.com/colorbox/example1/>

Tóm tắt



Tóm tắt





Question?

CT275 – CÔNG NGHỆ WEB