



# JQUERY (JavaScript Library)



# Nội dung

---

- **What is jQuery?**
- **jQuery Fundamentals**
  - Selectors
  - DOM Manipulation
  - jQuery DOM elements
- **AJAX**
  - jQuery AJAX Methods
  - Executing AJAX Requests

# Giới thiệu jQuery

---

- **Jquery** là một thư viện được viết từ **Javascript** giúp xây dựng các chức năng bằng Javascript dễ dàng, nhanh và giàu tính năng hơn.



# Giới thiệu jQuery

---

- **jQuery** có thể
  - Giúp định dạng lại và thay đổi thông tin trong trang web.
  - Tạo ra nhiều hiệu ứng trong trang như: các hiệu ứng tương tác user, các hiệu ứng đồ họa.v..
  - Tạo ra các plugin mới.
  - JQuery cũng được bootstrap sử dụng để chạy các code js của nó.
- Microsoft đã sử dụng jQuery trong Visual Studio
  - Sử dụng trong Microsoft's ASP.NET AJAX Framework and ASP.NET MVC Framework

# Giới thiệu jQuery

---

- **Các module JQuery phổ biến:**
  - **Ajax:** xử lý Ajax
  - **Attributes:** Xử lý các thuộc tính của đối tượng HTML
  - **Effect:** xử lý hiệu ứng
  - **Event:** xử lý sự kiện
  - **Form:** xử lý sự kiện liên quan tới form
  - **DOM:** xử lý Data Object Model
  - **Selector:** xử lý luồng lách giữa các đối tượng HTML

# Giới thiệu jQuery

---

## ▪ Các tính năng quan trọng của jQuery

- **Gọn nhẹ:** jQuery là một thư viện khá gọn nhẹ, khoảng 19KB (gzipped).
- **Tương thích đa nền tảng:** Nó tự động sửa lỗi và chạy được trên các trình duyệt phổ biến như Chrome, Firefox, Safari, MS Edge, IE, Android và iOS.
- **Dễ dàng tạo Ajax:** Nhờ thư viện jQuery, code được viết bởi Ajax có thể dễ dàng tương tác với server và cập nhật nội dung tự động mà không cần phải tải lại trang.

# Giới thiệu jQuery

---

## ▪ Các tính năng quan trọng của jQuery

- **Xử lý nhanh thao tác DOM:** jQuery giúp lựa chọn các phần tử DOM một cách dễ dàng, và chỉnh sửa nội dung của chúng bằng cách sử dụng Selector mã nguồn mở gọi là Sizzle.
- **Đơn giản hóa việc tạo hiệu ứng:** Giống với code snippet có hiệu ứng animation
- **Xử lý sự kiện:** jQuery xử lý các sự kiện đa dạng

# Cài đặt - tích hợp jQuery vào Website

---

## ▪ Cách 1:

- Tải file **js** tại <http://www.jquery.com>, chọn bản đã nén **.min.js**,
- Tích hợp vào trang bằng thẻ `<script>`.
- **Cú pháp:** `<script src=jquery-3.6.0.min.js "></script>`



# Cài đặt - tích hợp jQuery vào Website

---

- **Cách 2:** dùng CDN (Content Delivery Network) đưa thư viện jQuery vào trong code HTML trực tiếp từ CDN
  - Google CDN
  - Microsoft CDN
  - CDNJS CDN
  - jsDelivr CDN
- Tích hợp vào trang bằng thẻ <script>

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```



# **SELECTORS AND DOM MANIPULATION**

# Selectors - Chọn phần tử trong jQuery

---

- **Selector jQuery:** để chọn các phần tử khác nhau trong trang HTML theo id, tên thẻ, class

- Cú pháp: `$(selector)`

- Ví dụ:

```
//by tag
$("div") //document.querySelectorAll("div");

//by class
$(".menu-item")
//document.querySelectorAll(".menu-item");

//by id
$("#navigation")

//by combination of selectors
$("ul.menu li")
```

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector cơ bản

Selector	Ví dụ	Diễn giải
*	<code>\$("*")</code>	Chọn tất cả các phần tử
#id	<code>\$("#lastname")</code>	Chọn phần tử có id="lastname"
.class	<code>\$(".intro")</code>	Chọn phần tử có class="intro"
.class,.class	<code>\$(".intro,.demo")</code>	chọn phần tử có class là "intro" hoặc "demo"
element	<code>\$("p")</code>	Chọn các phần tử thẻ <p>
el1,el2,el3	<code>\$("h1,div,p")</code>	Chọn tất cả các phần tử thẻ <h1> <div> và<p>

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector nâng cao

Selector	Ví dụ	Diễn giải
:first	<code>\$("p:first")</code>	Chọn phần tử <code>&lt;p&gt;</code> đầu tiên trong luồng HTML
:last	<code>\$("p:last")</code>	Chọn phần tử <code>&lt;p&gt;</code> cuối cùng
:even	<code>\$("tr:even")</code>	Chọn các phần tử <code>&lt;tr&gt;</code> ở vị trí chẵn
:odd	<code>\$("tr:odd")</code>	Chọn các phần tử <code>&lt;tr&gt;</code> ở vị trí lẻ
:first-child	<code>\$("p:first-child")</code>	Chọn tất cả phần tử <code>&lt;p&gt;</code> là phần tử con đầu tiên trong phần tử cha chứa nó
:first-of-type	<code>\$("p:first-of-type")</code>	Chọn các phần tử <code>&lt;p&gt;</code> là phần tử <code>&lt;p&gt;</code> đầu tiên trong các phần tử con mà phần tử cha chứa

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector nâng cao

Selector	Ví dụ	Diễn giải
:last-child	<code>\$("p:last-child")</code>	Chọn các phần tử <code>&lt;p&gt;</code> là phần tử cuối cùng trong phần tử cha chứa nó.
:last-of-type	<code>\$("p:last-of-type")</code>	Chọn các phần tử <code>&lt;p&gt;</code> là phần tử <code>&lt;p&gt;</code> sau cùng thấy trong phần tử cha
:nth-child(n)	<code>\$("p:nth-child(2)")</code>	Tất cả các phần tử <code>&lt;p&gt;</code> là con thứ 2
:nth-last-child(n)	<code>\$("p:nth-last-child(2)")</code>	Tất cả phần tử <code>&lt;p&gt;</code> là con thứ 2 đếm từ dưới lên.
:nth-of-type(n)	<code>\$("p:nth-of-type(2)")</code>	Tất cả phần <code>&lt;p&gt;</code> là phần tử thứ 2 dạng <code>&lt;p&gt;</code> trong các phần tử con

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector nâng cao

Selector	Ví dụ	Diễn giải
:nth-last-of-type(n)	<code>\$("p:nth-last-of-type(2)")</code>	Tất cả các phần tử <code>&lt;p&gt;</code> thứ 2 đếm từ dưới lên.
parent < child	<code>\$("div &gt; p")</code>	Tất cả phần tử <code>&lt;p&gt;</code> là con trực tiếp của phần tử <code>&lt;div&gt;</code>
parent descendant	<code>\$("div p")</code>	Tất cả phần tử <code>&lt;p&gt;</code> là con, cháu ... của <code>&lt;div&gt;</code>
element + next	<code>\$("div + p")</code>	Chọn phần tử <code>&lt;p&gt;</code> là phần tử tiếp theo của phần tử <code>&lt;div&gt;</code>
element ~ siblings	<code>\$("div ~ p")</code>	Các phần tử <code>&lt;p&gt;</code> có cấp ngang hàng với một phần tử <code>&lt;div&gt;</code>

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector jQuery chọn phần tử danh sách

Selector	Ví dụ	Diễn giải
:eq(index)	<code>\$("ul li:eq(3)")</code>	Phần tử thứ 4 trong một danh sách
:gt(no)	<code>\$("ul li:gt(3)")</code>	Các phần tử có chỉ số lớn hơn 3
:lt(no)	<code>\$("ul li:lt(3)")</code>	Các phần tử trong danh sách có chỉ số nhỏ hơn 3
:not(selector)	<code>\$("input:not(:empty)")</code>	Các phần tử <input> không rỗng



# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector jQuery theo trạng thái

Selector	Ví dụ	Diễn giải
:header	<code>\$(":header")</code>	Tất cả các phần tử <code>&lt;h1&gt;</code> , <code>&lt;h2&gt;</code> ...
:animated	<code>\$(":animated")</code>	Các phần tử động
:focus	<code>\$(":focus")</code>	Phần tử đang giữ focus
:contains(text)	<code>\$(":contains('Hello')")</code>	Các phần tử có chứa chữ "Hello"
:has(selector)	<code>\$("div:has(p)")</code>	Các phần tử <code>&lt;div&gt;</code> trong nó có chứa một phần tử <code>&lt;p&gt;</code>

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector jQuery theo trạng thái

Selector	Ví dụ	Diễn giải
:empty	\$(":empty")	Tất cả các phần tử rỗng
:parent	\$(":parent")	Các phần tử là cha của 1 phần tử khác
:hidden	\$("p:hidden")	Tất cả các phần tử <p> đang ẩn
:visible	\$("table:visible")	Tất cả các <table> đang hiện thị

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector jQuery theo thuộc tính phần tử

Selector	Ví dụ	Diễn giải
[attribute]	<code>\$("[href]")</code>	Các phần tử có thuộc tính href
[attribute=value]	<code>\$("[href='default.htm']")</code>	Các phần tử có thuộc tính href và giá trị là "default.htm"
[attribute!=value]	<code>\$("[href!='default.htm']")</code>	Các phần tử có thuộc tính href với giá trị khác "default.htm"
[attribute\$=value]	<code>\$("[href\$='.jpg']")</code>	Các phần tử có thuộc tính href với giá trị là file ".jpg"
[attribute*=value]	<code>\$("[title*='hello']")</code>	Các phần tử có thuộc tính title và giá trị chứa "hello"

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector jQuery theo thuộc tính phần tử

Selector	Ví dụ	Diễn giải
[attribute =value]	<code>\$("[title ='Tomorrow']")</code>	Các phần tử có title bằng 'Tomorrow' hoặc bắt đầu bởi 'Tomorrow'
[attribute^=value]	<code>\$("[title^='Tom']")</code>	Các phần tử có title với giá trị bắt đầu bằng "Tom"
[attribute~=value]	<code>\$("[title~='hello']")</code>	Các phần tử có title, và giá trị có chứa "hello"
[attribute*=value]	<code>\$("[title*='hello']")</code>	Các phần tử có title và giá trị chứa "hello"

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector jQuery trong FORM

Selector	Ví dụ	Diễn giải
:input	<code>\$(":input")</code>	Tất cả các phần tử input
:text	<code>\$(":text")</code>	Tất cả các phần tử có type="text"
:password	<code>\$(":password")</code>	Tất cả phần tử có type="password"
:radio	<code>\$(":radio")</code>	Tất cả phần tử có type="radio"
:checkbox	<code>\$(":checkbox")</code>	Tất cả phần tử có type="checkbox"
:submit	<code>\$(":submit")</code>	Tất cả phần tử có type="submit"
:reset	<code>\$(":reset")</code>	Tất cả phần tử có type="reset"

# Selectors - Chọn phần tử trong jQuery

---

## ▪ Selector jQuery trong FORM

Selector	Ví dụ	Diễn giải
:button	\$(":button")	Tất cả phần tử có type="button"
:image	\$(":image")	Tất cả phần tử có type="image"
:file	\$(":file")	Tất cả phần tử có type="file"
:enabled	\$(":enabled")	Tất cả các phần tử input là enable
:disabled	\$(":disabled")	Các phần tử input bị vô hiệu
:selected	\$(":selected")	Các phần tử input là selected
:checked	\$(":checked")	Các phần tử input là checked

# jQuery Syntax

---

- **Cú pháp jQuery:** được thiết kế để chọn các phần tử HTML và thực hiện một số hành động trên các phần tử.

- **Cú pháp:** `$(selector).action()`

Ví dụ:

- `$(this).hide()` // *hides current element.*
- `$("p").hide()` // *hides all paragraphs.*
- `$("p.test").hide()` // *hides all paragraphs with class="test".*
- `$("#test").hide()` // *hides the element with id="test"*

# jQuery Syntax

---

- **\$( document ).ready()**

- Các câu lệnh **jQuery** bắt đầu với sự kiện **document.ready**. Code bên trong **\$( document ).ready ()** sẽ chỉ thực thi khi trang DOM đã sẵn sàng.

- **Cú pháp:**

```
<script type="text/javascript">
  $(document).ready(function()
  {
    // your code
    $(selector).action();
  } );
</script>
```



# jQuery Syntax

---

- **`$( document ).ready()`**

- `$(document).ready(function(){}):` đại diện cho sự kiện sẵn sàng của tài liệu và thực thi mã **jQuery** sau khi tài liệu được tải đầy đủ và sẵn sàng.

Ví dụ:

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
    $(document).ready(function(){
        $("p").text("Hello World!");
    });
</script>
```

# DOM Traversal

---

- **jQuery traversing:** là "move through - di chuyển qua" được sử dụng để "tìm" hoặc chọn các phần tử HTML dựa trên mối quan hệ của chúng với các phần tử khác.
- Các phương thức **DOM traversal** giúp duyệt xung quanh cây DOM rất dễ dàng.
- Để duyệt qua DOM cần xác định mối quan hệ giữa các phần tử trong cây DOM.

# DOM Traversal

---

- **Di chuyển trong DOM với jQuery:** Sử dụng các hàm jQuery như `parent()`, `children()`, `next()`, `pre()`, `find()` để lựa chọn phần tử trong DOM dựa trên phần tử đang chọn

Hàm	Giải thích
<code>.parent()</code>	lấy phần tử cha trực tiếp của phần tử.
<code>.parents()</code>	lấy phần tử các phần tử cha (kể cả không trực tiếp)
<code>.children()</code>	lấy các phần tử con
<code>.siblings()</code>	các phần tử ngang hàng (anh em)
<code>.next()</code>	phần tử ngang hàng tiếp theo
<code>.nextAll()</code>	tất cả các phần tử ngang hàng tiếp theo

# DOM Traversal

---

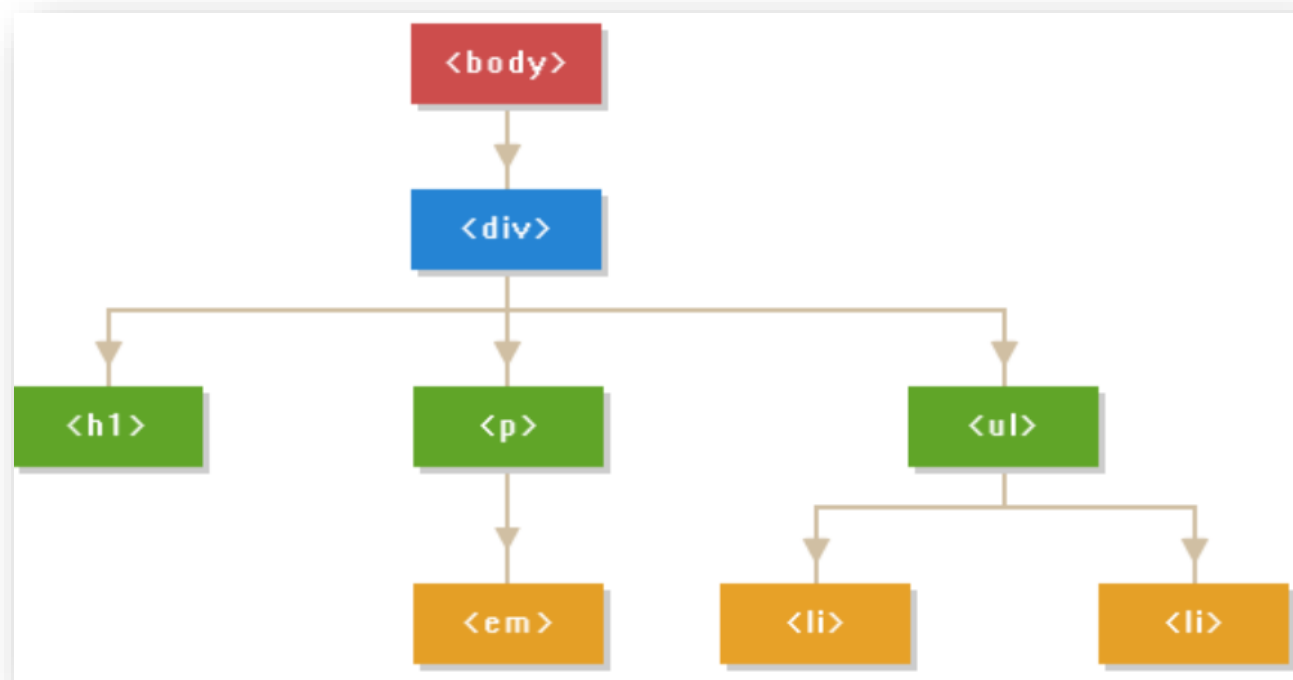
## ▪ Di chuyển trong DOM với jQuery:

Hàm	Giải thích
<code>.prev()</code>	phần tử ngang hàng trước
<code>.prevAll()</code>	tất cả các phần tử ngang hàng phía trước
<code>.eq(index)</code>	phần tử có thứ tự index trong tập hợp chọn được
<code>.find()</code>	tìm phần tử trong các phần tử con, cháu ...
<code>.each()</code>	Lặp lại các phần tử được chỉ định và thực hiện hàm gọi lại cho từng phần tử.
<code>.first()</code>	Lấy phần tử được chỉ định xuất hiện đầu tiên
<code>.next()</code>	Lấy phần tử ngay sau của phần tử được chỉ định

# DOM Traversal

---

- Di chuyển trong DOM với jQuery:
- Ví dụ: cây DOM



# jQuery Traversing - Ancestors

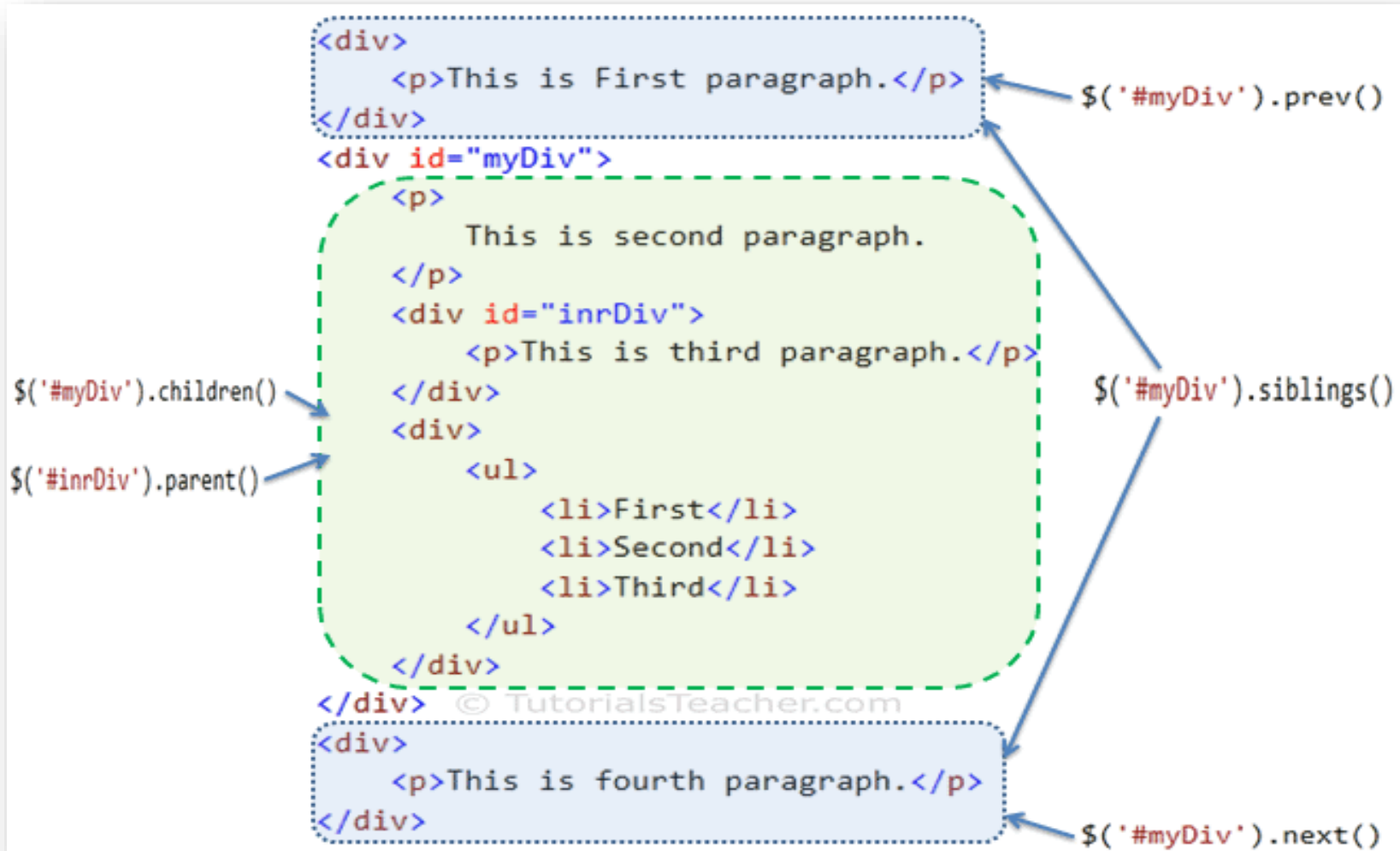
---

- **Duyệt phần tử parent, grandparent, great-grandparent**
- **Phương thức:**
  - parent()
  - parents()
  - parentsUntil()

# jQuery Traversing - Ancestors

## ■ jQuery parent()

Ví dụ:



# jQuery Traversing - Ancestors

---

## ▪ jQuery parent()

Ví dụ:

```
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script>
    $(document).ready(function(){
        $("li").parent().addClass("highlight");
    });
</script>
```



# jQuery Traversing - Ancestors

---

## ▪ jQuery parent()

Ví dụ:

```
<body>
  <div class="container">
    <h1>Hello World</h1>
    <p>This is a <em>simple paragraph</em>.</p>
    <ul>
      <li>Item One</li>
      <li>Item Two</li>
    </ul>
  </div>
</body>
```

This is a *simple paragraph*.

- Item One
- Item Two

# jQuery Traversing - Ancestors

---

- **jQuery parents():** lấy các phần tử cha (**ancestors**) của phần tử đã chọn

Ví dụ:

```
<style>
  *      { margin: 10px;}
  .frame{ border: 2px solid green;}
</style>
```

```
<script>
  $(document).ready(function(){
    $("li").parents().addClass("frame");});
</script>
```

# jQuery Traversing - Ancestors

---

- **jQuery parents():** lấy các phần tử cha (**ancestors**) của phần tử đã chọn

Ví dụ:

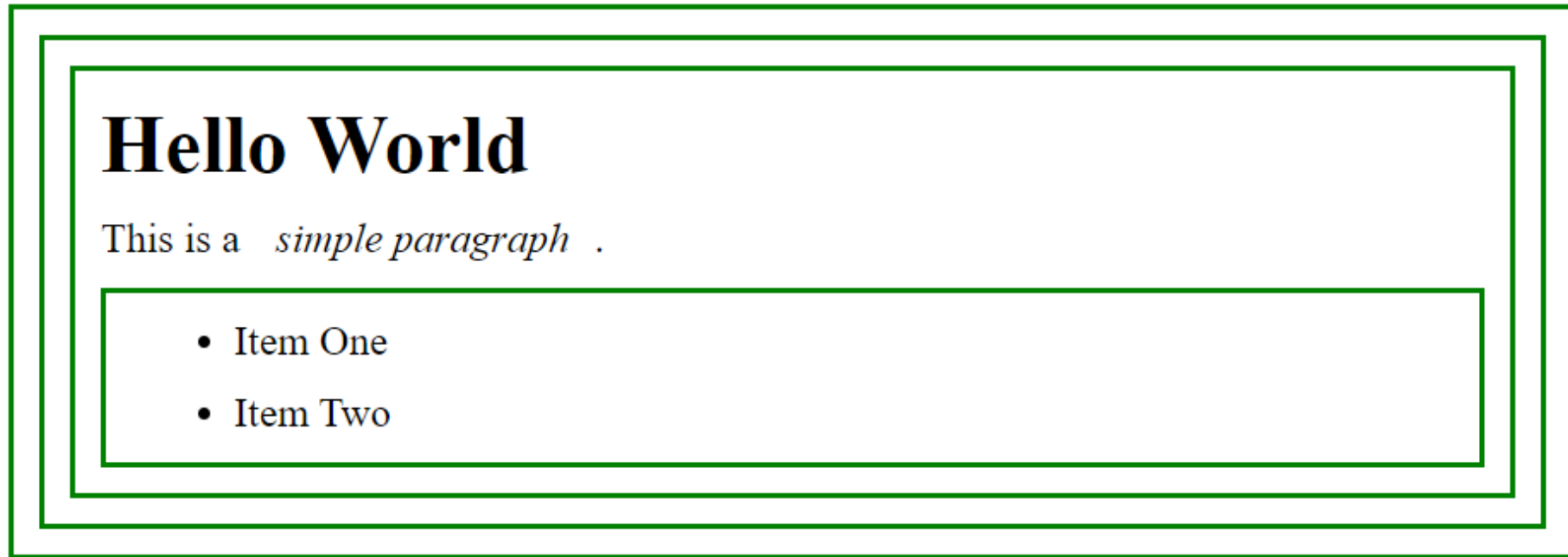
```
<div class="container">
  <h1>Hello World</h1>
  <p>This is a<em>simple paragraph</em>.</p>
  <ul>
    <li>Item One</li>
    <li>Item Two</li>
  </ul>
</div>
```

# jQuery Traversing - Ancestors

---

- **jQuery parents():** lấy các phần tử cha (**ancestors**) của phần tử đã chọn

Ví dụ:



# jQuery Traversing - Ancestors

---

- **jQuery parentsUntil():** lấy tất cả các phần tử **ancestors** giữa hai phần tử đã cho trong một hệ thống phân cấp DOM.
- Ví dụ:

```
<script>
    $(document).ready(function(){
        $("li").parentsUntil("html").addClass("frame");
    });
</script>
```

# jQuery Traversing - Ancestors

---

- **jQuery parentsUntil():** lấy tất cả các phần tử **ancestors** giữa hai phần tử đã cho trong một hệ thống phân cấp DOM.

- Ví dụ:

```
<script>
    $(document).ready(function(){
        $("li").parentsUntil("html").addClass("frame");
    });
</script>
```

## Hello World

This is a *simple paragraph* .

- Item One
- Item Two

# jQuery Traversing - Descendants

---

- **jQuery Descendants**: duyệt qua DOM để tìm con, cháu (child, grandchild, great-grandchild) của một phần tử được chọn.
- **Các phương thức:**
  - children()
  - find()

# jQuery Traversing - Descendants

---

- **jQuery children():** lấy các phần tử con trực tiếp của phần tử được chọn.

Ví dụ: highlight các phần tử con trực tiếp của <ul> là <li> bằng cách thêm lớp **.highlight** trên tài liệu.

```
<style>
    .highlight{ background: yellow; }
</style>
```

```
<script>
    $(document).ready(function(){
        $("ul").children().addClass("highlight"); });
</script>
```



# jQuery Traversing - Descendants

---

## ▪ jQuery children():

Ví dụ:

```
<div class="container">
  <h1>Hello World</h1>
  <p>This is a <em>simple paragraph</em>.</p>
  <ul>
    <li>Item One</li>
    <li>Item Two</li>
  </ul>
</div>
```

This is a *simple paragraph*.

- Item One
- Item Two

# jQuery Traversing - Descendants

---

- **jQuery find():** lấy các phần tử con của phần tử được chọn.
- Phương thức **find ()** và **children ()** tương tự nhau, nhưng
  - **find ()** tìm kiếm qua nhiều cấp của DOM đến con cuối cùng
  - **children ()** tìm kiếm ở một cấp duy nhất trên DOM

Ví dụ:

```
<script>
    $(document).ready(function(){
        $("div").find("li").addClass("frame");
    });
</script>
```

```
<style>
    *{ margin: 10px; }
    .frame{ border: 2px solid green; }
</style>
```

# jQuery Traversing - Descendants

---

## ▪ jQuery find():

Ví dụ:

```
<body>
  <div class="container">
    <h1>Hello World</h1>
    <p>This is a <em>simple paragraph</em>.</p>
    <ul>
      <li>Item One</li>
      <li>Item Two</li>
    </ul>
  </div>
</body>
```

# jQuery Traversing - Descendants

---

## ▪ jQuery find():

Ví dụ:

```
<body>
  <div class="container">
    <h1>Hello World</h1>
    <p>This is a <em>simple paragraph</em>.</p>
    <ul>
      <li>Item One</li>
      <li>Item Two</li>
    </ul>
  </div>
</body>
```

This is a *simple paragraph* .

- Item One
- Item Two

# jQuery Traversing Siblings

---

- **jQuery Traversing Siblings**: tìm các phần tử có cùng **parent**
- **Các phương thức**:
  - `siblings()`
  - `next()`
  - `nextAll()`
  - `nextUntil()`
  - `prev()`
  - `prevAll()`
  - `prevUntil()`

# jQuery Traversing Siblings

---

- **jQuery siblings():** lấy các phần tử anh em của phần tử đã chọn

Ví dụ:

```
<style>
    .highlight{ background: yellow; }
</style>
```

```
<script>
    $(document).ready(function(){
        $("p").siblings().addClass("highlight");});
</script>
```

# jQuery Traversing Siblings

---

- **jQuery siblings():** lấy các phần tử anh em của phần tử đã chọn

Ví dụ:

```
<div class="container">
  <h1>Hello World</h1>
  <p>This is a <em>simple paragraph</em>.</p>
  <ul>
    <li>Item One</li>
    <li>Item Two</li>
  </ul>
</div>
```

# jQuery Traversing Siblings

---

- **jQuery siblings():** lấy các phần tử anh em của phần tử đã chọn

Ví dụ:

**Hello World**

This is a *simple paragraph*.

- Item One
- Item Two



# jQuery Traversing Siblings

---

- **jQuery next():** lấy phần tử anh chị em tiếp theo của phần tử được chọn.

Ví dụ:

```
<style>
    * {
        display: block;
        border: 2px solid lightgrey;
        color: lightgrey;
        padding: 5px;
        margin: 15px;
    }
</style>
```

```
<script>
    $(document).ready(function(){
        $("h2").next().css({"color": "red", "border": "2px solid red"});});
</script>
```

# jQuery Traversing Siblings

---

- **jQuery next():** lấy phần tử anh chị em tiếp theo của phần tử được chọn.

Ví dụ:

```
<div>div (parent)
  <p>p</p>
  <span>span</span>
  <h2>h2</h2>
  <h3>h3</h3>
  <p>p</p>
</div>
```

div (parent)

p

span

h2

h3

p

# jQuery Traversing Siblings

---

- **jQuery nextAll():** lấy tất cả các phần tử anh chị em sau phần tử đã chọn.

Ví dụ:

```
<style>  
    .highlight{ background: yellow;}  
</style>
```

```
<script>  
    $(document).ready(function(){  
        $("p").nextAll().addClass("highlight"); });  
</script>
```

# jQuery Traversing Siblings

---

- **jQuery nextAll():** lấy tất cả các phần tử anh chị em kế tiếp sau phần tử được chọn.

Ví dụ:

```
<div class="container">
  <h1>Hello World</h1>
  <p>This is a <em>simple paragraph</em>.</p>
  <p>This is another paragraph.</p>
  <ul>
    <li>Item One</li>
    <li>Item Two</li>
  </ul>
</div>
```

# jQuery Traversing Siblings

---

- **jQuery nextAll():** lấy tất cả các phần tử anh chị em sau phần tử đã chọn.

Ví dụ:

This is a *simple paragraph*.

This is another paragraph.

- Item One
- Item Two

# jQuery Traversing Siblings

---

- **jQuery nextUntil():** lấy tất cả các phần tử cùng cha kế tiếp giữa hai phần tử đã cho trong một hệ thống phân cấp DOM.  
Ví dụ: highlight tất cả các phần tử anh em tiếp sau của phần tử <h1> ngoại trừ phần tử <ul>

```
<style>  
    .highlight{ background: yellow;}  
</style>
```

```
<script>  
    $(document).ready(function(){  
        $("h1").nextUntil("ul").addClass("highlight");});  
</script>
```

# jQuery Traversing Siblings

---

## ▪ jQuery nextUntil():

Ví dụ:

```
<div class="container">
  <h1>Hello World</h1>
  <p>This is a <em>simple paragraph</em>.</p>
  <p>This is another paragraph.</p>
  <ul>
    <li>Item One</li>
    <li>Item Two</li>
  </ul>
</div>
```

This is a *simple paragraph*.

This is another paragraph.

- Item One
- Item Two

# jQuery Traversing Siblings

---

- **jQuery prev():** lấy phần tử cùng cha ngay trước phần tử được chọn.

**Ví dụ** highlight phần tử trước của phần tử `<ul>` là `<p>`.

```
<style>
    .highlight{ background: yellow;}
</style>
```

```
<script>
    $(document).ready(function(){
        $("ul").prev().addClass("highlight");});
</script>
```



# jQuery Traversing Siblings

---

- **jQuery prev():** lấy phần tử cùng cha ngay trước phần tử được chọn.

Ví dụ highlight phần tử trước của phần tử `<ul>` là `<p>`.

```
<div class="container">
  <h1>Hello World</h1>
  <p>This is a <em>simple paragraph</em>.</p>
  <p>This is another paragraph.</p>
  <ul>
    <li>Item One</li>
    <li>Item Two</li>
  </ul>
</div>
```

This is a *simple paragraph*.

This is another paragraph.

- Item One
- Item Two

# jQuery Traversing Siblings

---

- **jQuery prevAll():** lấy tất cả các phần tử cùng cha, trước phần tử được chọn.

Ví dụ:

```
<style>
    .highlight{ background: yellow;}
</style>
```

```
<script>
    $(document).ready(function(){
        $("ul").prevAll().addClass("highlight"); });
</script>
```

# jQuery Traversing Siblings

---

- **jQuery prevAll():** lấy tất cả các phần tử cùng cha, trước phần tử được chọn.

Ví dụ:

```
<div class="container">
  <h1>Hello World</h1>
  <p>This is a <em>simple paragraph</em>.</p>
  <p>This is another paragraph.</p>
  <ul>
    <li>Item One</li>
    <li>Item Two</li>
  </ul>
</div>
```

Hello World

This is a *simple paragraph*.

This is another paragraph.

- Item One
- Item Two

# jQuery Traversing Siblings

---

- **jQuery prevUntil():** lấy tất cả các phần tử cùng cha trước đó giữa hai phần tử đã cho trong một hệ thống phân cấp DOM.

Ví dụ:

```
<style>
    .highlight{ background: yellow;}
</style>
```

```
<script>
    $(document).ready(function(){
        $("ul").prevUntil("h1").addClass("highlight");});
</script>
```

# jQuery Traversing Siblings

---

## ▪ jQuery prevUntil():

Ví dụ:

```
<div class="container">
  <h1>Hello World</h1>
  <p>This is a <em>simple paragraph</em>.</p>
  <p>This is another paragraph.</p>
  <ul>
    <li>Item One</li>
    <li>Item Two</li>
  </ul>
</div>
```

Hello World

This is a *simple paragraph*.

This is another paragraph.

- Item One
- Item Two

# jQuery Filtering

---

- **jQuery Filtering**: chọn một phần tử cụ thể dựa trên vị trí của nó trong một nhóm các phần tử.
- **Các phương thức:**
  - `filter()`
  - `first()`
  - `last()`
  - `eq()`
  - `slice()`
  - `has()`
  - `not()`

# jQuery Filtering

---

- **jQuery first():** lọc ra tập các phần tử phù hợp và trả về phần tử đầu tiên

Ví dụ

```
<body>
  <ul>
    <li>First list item</li>
    <li>Second list item</li>
    <li>Third list item</li>
    <li>Last list item</li>
  </ul>
</body>
```

```
<script>
  $(document).ready(function(){
    $("ul li").first().addClass("highlight"); });
</script>
```

## Unordered List

- First list item
- Second list item
- Third list item
- Fourth list item

# jQuery Filtering

---

- **jQuery last():** lọc ra tập các phần tử phù hợp và trả về phần tử cuối cùng.

- Ví dụ

```
<body>
  <ul>
    <li>First list item</li>
    <li>Second list item</li>
    <li>Third list item</li>
    <li>Last list item</li>
  </ul>
</body>
```

```
<script>
  $(document).ready(function(){
    $("ul li").last().addClass("highlight"); });
</script>
```

## Another Unordered List

- First list item
- Second list item
- Third list item
- Fourth list item



# jQuery Filtering

---

- **jQuery eq():** lọc ra tập các phần tử so khớp và chỉ trả về một phần tử có số chỉ mục xác định.

Ví dụ

```
<body>
  <ul>
    <li>First list item</li>
    <li>Second list item</li>
    <li>Third list item</li>
    <li>Last list item</li>
  </ul>
</body>
```

```
<script>
  $(document).ready(function(){
    $("ul li").eq(1).addClass("highlight");});
</script>
```

## Unordered List

- First list item
- Second list item
- Third list item
- Fourth list item

# jQuery Filtering

---

- **jQuery filter():** lấy bộ chọn hoặc một hàm làm đối số để lọc tập hợp các phần tử đã so khớp dựa trên một tiêu chí cụ thể.

- Ví dụ

```
<body>
  <ul>
    <li>First list item</li>
    <li>Second list item</li>
    <li>Third list item</li>
    <li>Last list item</li>
  </ul>
</body>
```

```
<script>
  $(document).ready(function(){
    $("ul li").filter(":even").addClass("highlight");});
</script>
```

## Unordered List

- First list item
- Second list item
- Third list item
- Fourth list item

# jQuery Filtering

---

- **jQuery has():** lọc ra một tập các phần tử phù hợp và chỉ trả về những phần tử có phần tử con được chỉ định

Ví dụ

```
<script>
    $(document).ready(function(){
        $("ul li").has("ul").addClass("highlight");});
</script>
```

# jQuery Filtering

---

## ▪ jQuery has():

Ví dụ

- Section 1
- Section 2
- - Section 2.1
  - Section 2.2
  - Section 2.3
- Section 4

```
<ul>
  <li>Section 1</li>
  <li>Section 2</li>
  <li>
    <ul>
      <li>Section 2.1</li>
      <li>Section 2.2</li>
      <li>Section 2.3</li>
    </ul>
  </li>
  <li>Section 4</li>
</ul>
```

# jQuery Filtering

---

- **jQuery not():** lọc ra tập các phần tử phù hợp và trả về tất cả các phần tử không đáp ứng các điều kiện đã chỉ định.

Ví dụ

```
<ul>
  <li>First list item</li>
  <li>Second list item</li>
  <li>Third list item</li>
  <li>Fourth list item</li>
</ul>
```

## Unordered List

- First list item
- Second list item
- Third list item
- Fourth list item

```
<script>
  $(document).ready(function(){
    $("ul li").not(":even").addClass("highlight");});
</script>
```

# jQuery Filtering

---

- **jQuery slice()**: lọc ra tập các phần tử phù hợp được chỉ định bởi một loạt các chỉ số.
- Phương thức **slice()** nhận số chỉ mục bắt đầu và kết thúc làm đối số, trong đó chỉ mục bắt đầu chỉ định vị trí các phần tử bắt đầu được chọn và chỉ mục kết thúc chỉ định vị trí các phần tử ngừng được chọn.

# jQuery Filtering

---

## ▪ jQuery slice():

Ví dụ

```
<style>
    .highlight{ background: yellow; }
</style>
```

```
<script>
    $(document).ready(function(){
        $("ul li").slice(0, 2).addClass("highlight"); });
</script>
```

# jQuery Filtering

---

## ▪ jQuery slice():

Ví dụ

```
<ul>
  <li>First list item</li>
  <li>Second list item</li>
  <li>Third list item</li>
  <li>Fourth list item</li>
</ul>
```

### Unordered List

- First list item
- Second list item
- Third list item
- Fourth list item



# Adding DOM elements

---

- **appendTo()**: chèn các phần tử HTML vào cuối các phần tử đã chọn.
- Ví dụ:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("<span>Hello World!</span>").appendTo("p");
        });
    });
</script>
```

# Adding DOM elements

---

- **appendTo():**

Ví dụ:

```
<button>
```

```
    Insert span element at the end of each p  element
```

```
</button>
```

```
    <p>This is a paragraph.</p>
```

```
    <p>This is another paragraph.</p>
```

```
Insert span element at the end of each p element
```

This is a paragraph.Hello World!

This is another paragraph.Hello World!

# Adding DOM elements

---

- **prependTo()** chèn các phần tử HTML vào đầu các phần tử đã chọn.

- **Cú pháp:** `$(content).prependTo(selector)`

Ví dụ:

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("<span>Hello World!</span>").prependTo("p");});});
</script>
```

# Adding DOM elements

---

## ▪ prependTo()

Ví dụ (tt):

```
<button>
```

Insert span element at the end of each p element

```
</button>
```

```
<p>This is a paragraph.</p>
```

```
<p>This is another paragraph.</p>
```

Insert span element at the beginning of each p element

Hello World! This is a paragraph.

Hello World! This is another paragraph.

# Adding DOM elements

---

- **append():** chèn nội dung vào cuối các phần tử đã chọn.  
Ví dụ: click vào button thì thêm 1 phần tử trong danh sách

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("ol").append("<li>list item</li>");
        });
    });
</script>
```

# Adding DOM elements

---

## ▪ **append():**

Ví dụ: click vào button thì thêm 1 phần tử trong danh sách

```
<button>Add new list item</button>  
<ol>  
  <li>list item</li>  
  <li>list item</li>  
  <li>list item</li>  
</ol>
```

Add new list item

1. list item
2. list item
3. list item

# Adding DOM elements

---

- **prepend()**: chèn nội dung đã chỉ định vào đầu các phần tử đã chọn.

- **Cú pháp** `$(selector).prepend(content,function(index,html))`

Ví dụ:

```
<script>
$(document).ready(function(){
    $("#btn1").click(function(){
        $("p").prepend("<b>Prepended text</b>.");});
    $("#btn2").click(function(){
        $("ol").prepend("<li>Prepended item</li>");});
});
</script>
```

# Adding DOM elements

---

## ▪ **prepend():**

Ví dụ (tt):

```
<p>This is a paragraph.</p>  
<p>This is another paragraph.</p>  
<ol>  
  <li>List item 1</li>  
  <li>List item 2</li>  
  <li>List item 3</li>  
</ol>  
<button id="btn1">Prepend text</button>  
<button id="btn2">Prepend list item</button>
```

Prepended text. This is a paragraph.

Prepended text. This is another paragraph.

1. Prepended item
2. List item 1
3. List item 2
4. List item 3

Prepend text

Prepend list item



# Adding DOM elements

---

## ▪ Phương thức **after()** và **before()**

- **after()**: chèn nội dung vào SAU các phần tử được chọn.
- **before()**: chèn nội dung vào TRƯỚC các phần tử được chọn.

Ví dụ:

```
  
<br>  
<button id="btn1">Insert before</button>  
<button id="btn2">Insert after</button>
```

# Adding DOM elements

## ▪ Phương thức after() và before()

Ví dụ:

```
<script>
$(document).ready(function(){
    $("#btn1").click(function(){
        $("img").before("<b>Before</b>");
    });
    $("#btn2").click(function(){
        $("img").after("<i>After</i>");
    });
});
</script>
```



Insert before

Insert after



Before

After

Insert before

Insert after

# Adding DOM elements

---

## ▪ Phương thức **after()** và **before()**

- Có thể thêm nhiều phần tử vào trước hoặc sau phần tử được chọn

Ví dụ:

```
var txt1 = "<b>I </b>"; //Create element with HTML
var txt2 = $("<i></i>").text("love "); // Create with jQuery
var txt3 = document.createElement("b"); //Create with DOM
txt3.innerHTML = "jQuery!";
$("img").after(txt1, txt2, txt3);
```

# Removing Elements

---

- **jQuery remove ()** loại bỏ (các) phần tử đã chọn và các phần tử con của nó. Phương thức **remove ()** chấp nhận một tham số (tùy chọn) là phần tử cần loại bỏ.

- Cú pháp: `Element.remove()`

- Ví dụ:

```
<script>
  $(document).ready(function(){//Removes phần tử có class "hint"
    $("button").click(function(){
      $("p.hint").remove();});});
</script>
```

# Removing Elements

## ▪ jQuery remove ()

Ví dụ (tt):

```
<div class="container">
  <h1>Hello World!</h1>
  <p class="hint">
    <strong>Note:</strong> If you click the following
    button it will remove this paragraph.
  </p>
  <button type="button">Remove Hint Paragraph</button>
</div>
```

**Hello World!**

**Note:** If you click the following button it will remove this paragraph.

Remove Hint Paragraph

# Removing Elements

---

## ▪ jQuery remove ()

Ví dụ: Phương thức Remove() có một tham số

```
<script>
    $(document).ready(function(){//Removes phần tử có clas="hint"
        $("button").click(function(){
            $("p").remove(".hint"); }); });
</script>
```

# Removing Elements

---

- **empty()**: loại bỏ các phần tử con của phần tử đã chọn.

Ví dụ:

```
#div1{  
    height:100px;  
    width:300px;  
    border:1px solid black;  
    background-color:yellow;  
}
```

```
<div id="div1">  
    This is some text in the div.  
    <p>This is a paragraph in the div.</p>  
    <p>This is another paragraph in the div.</p>  
</div>  
<button>Empty the div element</button>
```

# Removing Elements

---

- **empty()**: loại bỏ các phần tử con của phần tử đã chọn.

Ví dụ:

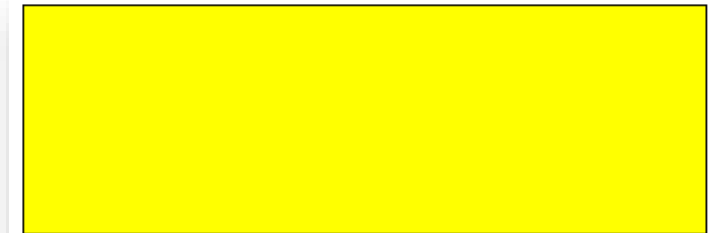
```
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("#div1").empty();
    });
  });
</script>
```

This is some text in the div.

This is a paragraph in the div.

This is another paragraph in the div.

Empty the div element



Empty the div element



# Removing Elements

---

- **removeAttr():** xóa một hoặc nhiều thuộc tính khỏi các phần tử đã chọn

Ví dụ:

```
<h1>This is a heading</h1>
<p style="font-size:120%;color:red">
    This is a paragraph.</p>
<p style="font-weight:bold;color:blue">
    This is another paragraph.</p>
<button>
    Remove the style attribute from all p elements
</button>
```

# Removing Elements

---

- **removeAttr():** xóa một hoặc nhiều thuộc tính khỏi các phần tử đã chọn

Ví dụ:

```
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("p").removeAttr("style");
    });
  });
</script>
```

**This is a heading**

This is a paragraph.

This is another paragraph.

Remove the style attribute from all p elements

**This is a heading**

This is a paragraph.

This is another paragraph.

Remove the style attribute from all p elements

# jQuery Objects

---

- **jQuery object:** là một tập các phần tử DOM và hoạt động giống như một mảng đặc biệt. Mọi thứ trong jQuery là một đối tượng.
- Khi tạo một phần tử mới hoặc chọn một phần tử hiện có, jQuery trả về các phần tử trong một tập hợp, được xem như một mảng các phần tử DOM.
- Cú pháp tạo một đối tượng

```
var obj = $("selector");
```

```
var obj = { property1:value1, property2:value2... propertyN:valueN } ;
```

# jQuery Objects

---

- **Các phương thức của jQuery object:**

- **addClass():** Thêm một hoặc nhiều lớp vào các phần tử đã chọn
- **removeClass():** Xóa một hoặc nhiều lớp khỏi các phần tử đã chọn
- **toggleClass():** Chuyển đổi giữa việc thêm / xóa các lớp khỏi các phần tử đã chọn
- **css():** Đặt hoặc trả về thuộc tính style

# jQuery Objects

---

- **addClass()** thêm một hoặc nhiều tên lớp vào các phần tử đã chọn. Để thêm nhiều lớp, tên lớp cách nhau bằng dấu cách.
- Cú pháp:

```
$(selector).addClass(classname,function(index,currentclass))
```

Ví dụ:

```
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Add a class name to the first p element </button>
```

```
.intro {
  font-size: 150%;
  color: red;
}
```

# jQuery Objects

---

## ▪ **addClass()**

Ví dụ:

```
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("p:first").addClass("intro");
    });
  });
</script>
```

**This is a heading**

This is a paragraph.

This is another paragraph.

Add a class name to the first p element

**This is a heading**

**This is a paragraph.**

This is another paragraph.

Add a class name to the first p element

# jQuery Objects

---

- **removeClass()**: xóa một thuộc tính lớp cụ thể khỏi các phần tử khác nhau

Ví dụ:

```
<h1 class="blue">Heading 1</h1>
<h2 class="blue">Heading 2</h2>
<p class="blue">This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Remove class from elements</button>
```

```
<style>
    .blue {
        color: blue;
    }
</style>
```

# jQuery Objects

---

## ▪ **removeClass()**:

Ví dụ:

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("h1, h2, p").removeClass("blue");
    });
});
</script>
```

## Heading 1

## Heading 2

This is a paragraph.

This is another paragraph.

Remove class from elements

## Heading 1

## Heading 2

This is a paragraph.

This is another paragraph.

Remove class from elements



# jQuery Objects

---

- **toggleClass():** chuyển đổi giữa adding và removing các lớp khỏi các phần tử đã chọn

Ví dụ:

```
<h1 class="blue">Heading 1</h1>
<h2 class="blue">Heading 2</h2>
<p class="blue">This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Remove class from elements</button>
```

```
<style>
    .blue {
        color: blue;
    }
</style>
```

# jQuery Objects

---

## ▪ toggleClass():

Ví dụ:

```
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("h1, h2, p").toggleClass("blue");
    });
  });
</script>
```

**Heading 1**

**Heading 2**

This is a paragraph.

This is another paragraph.

Toggle class

**Heading 1**

**Heading 2**

This is a paragraph.

This is another paragraph.

Toggle class

# Properties of jQuery Elements

---

- **css()**: thiết lập hoặc trả về một hoặc nhiều thuộc tính Style cho các phần tử đã chọn.

Ví dụ:

```
<h2>This is a heading</h2>  
<p style="background-color:#ff0000">This is a paragraph.</p>  
<p style="background-color:#00ff00">This is a paragraph.</p>  
<p style="background-color:#0000ff">This is a paragraph.</p>  
<button>Return background-color of p</button>
```

# Properties of jQuery Elements

---

- **html():** thiết lập hoặc trả về nội dung (innerHTML) của các phần tử đã chọn.
  - Khi dùng **html()** để trả về nội dung, thì nó sẽ **trả về nội dung** của phần tử được so khớp **đầu tiên**.
  - Khi dùng **html()** để thiết lập nội dung, thì nó sẽ **ghi đè lên nội dung** của **tất cả** các phần tử phù hợp.
- **Cú pháp:**

**Trả về nội dung:**    `$(selector).html()`

**Thiết lập nội dung:** `$(selector).html(content)`

**Thiết lập nội dung dùng function:**

`$(selector).html(function(index, currentcontent))`

# Properties of jQuery Elements

---

## ▪ **html():**

Ví dụ: trả về nội dung của phần tử HTML

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        alert($("#p").html());
    });
});
</script>
```

Return the content of the p element

This is a **paragraph.**

This page says

This is a <b>paragraph</b>.

OK

```
<button>Return the content of the p element</button>
<p>This is a <b>paragraph</b>.</p>
```

# Properties of jQuery Elements

---

## ▪ **html():**

Ví dụ: thiết lập nội dung cho phần tử HTML

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").html("Hello <b>world!</b>");
    });
});
</script>
```

# Properties of jQuery Elements

---

## ▪ **html():**

**Ví dụ:** thiết lập nội dung cho phần tử HTML dùng hàm

```
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("p").html(function(n){
        return "This p element has index: " + n;
      });
    });
  });
</script>
```

# Properties of jQuery Elements

- Các hàm thiết lập nội dung và thuộc tính cho phần tử **HTML** `text()`, `html()`, và `val()`:

Ví dụ:

```
<script>
$(document).ready(function(){
    $("#btn1").click(function(){
        $("#test1").text("Hello world!"); });
    $("#btn2").click(function(){
        $("#test2").html("<b>Hello world!</b>");});
    $("#btn3").click(function(){
        $("#test3").val("Dolly Duck");});
});
</script>
```

This is a paragraph.

This is another paragraph.

Input field:

Set Text

Set HTML

Set Value

Hello world!

**Hello world!**

Input field:

Set Text

Set HTML

Set Value



# Properties of jQuery Elements

---

- **jQuery Callback**: khi thực hiện các hiệu ứng, dòng lệnh tiếp theo có thể thực thi khi hiệu ứng chưa kết thúc. Điều này có thể tạo ra lỗi. Để ngăn chặn điều này, dùng **Callback**.
  - Hàm **Callback** được thực thi sau khi kết thúc hiệu ứng hiện tại.
  - Các phương thức: **text ()**, **html ()** và **val ()**, thường đi kèm với một hàm Callback.
- Cú pháp: `$(selector).hide(speed,callback);`

# Properties of jQuery Elements

---

## ▪ jQuery Callback:

Ví dụ:

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide("slow", function(){
            alert("The paragraph is now hidden");
        });
    });
});
</script>
```

<button>Hide</button>

<p>This is a paragraph with little content.</p>

Hide

This is a paragraph with little content.

Hide

This page says

The paragraph is now hidden

OK

# Properties of jQuery Elements

---

## ■ **css()**:

Ví dụ:

This page says

Background color = rgb(255, 0, 0)

OK

**This is a heading**

This is a paragraph.

This is a paragraph.

This is a paragraph.

Return background-color of p

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            alert("Background color = " + $("p").css("background-color"));
        });
    });
</script>
```

# jQuery Effects

---

- **show()** và **hide()**: hiển thị hoặc ẩn các phần tử HTML

- Cú pháp:

```
$(selector).hide(speed,callback);  
$(selector).show(speed,callback);
```

Ví dụ:

```
<div id="box">  
    
</div>  
<button type="button" class="hide-btn">Hide Image</button>  
<button type="button" class="show-btn">Show Image</button>
```

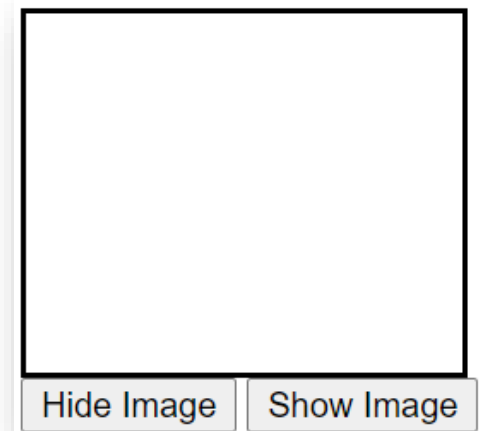
# jQuery Effects

---

## ▪ **show()** và **hide()**:

Ví dụ:

```
<script>
  $(document).ready(function(){
    $(".hide-btn").click(function(){
      $(".img").hide();
    });
    $(".show-btn").click(function(){
      $(".img").show();
    });
  });
</script>
```



# jQuery Effects

---

- **show() và hide()**: Tham số tốc độ (tùy chọn) chỉ định tốc độ ẩn hoặc hiển thị của đối tượng, có giá trị là "slow", "fast", hoặc milliseconds..

Ví dụ:

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("img").hide(1000);  
    });  
});
```

# jQuery Effects

---

- **fadeIn() và fadeOut():** hiển thị hoặc ẩn các phần tử HTML bằng cách tăng dần hoặc giảm độ mờ (opacity) của chúng

- Cú pháp:

```
$(selector).fadeIn(speed, callback);  
$(selector).fadeOut(speed, callback);
```

Ví dụ:

```
<button>Click to fade in boxes</button><br>  
<div id="div1" class="box" style="background-color:red;"></div>  
<div id="div2" class="box" style="background-color:green;"></div>  
<div id="div3" class="box" style="background-color:blue;"></div>
```

# jQuery Effects

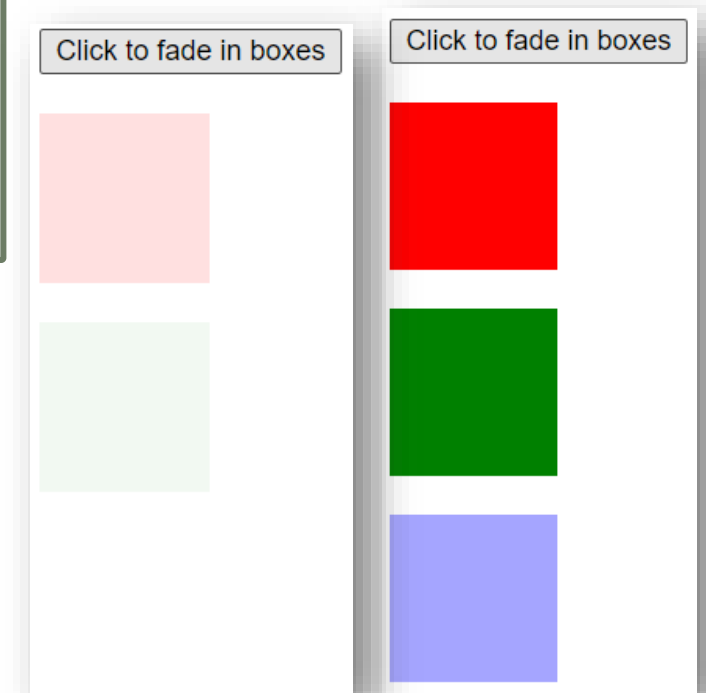
---

## ▪ fadeIn() và fadeOut():

Ví dụ:

```
.box{  
    width:80px;  
    height:80px;  
    display:none;  
}
```

```
<script>  
    $(document).ready(function(){  
        $("button").click(function(){  
            $("#div1").fadeIn();  
            $("#div2").fadeIn("slow");  
            $("#div3").fadeIn(3000);  
        });  
    });  
</script>
```





# jQuery Effects

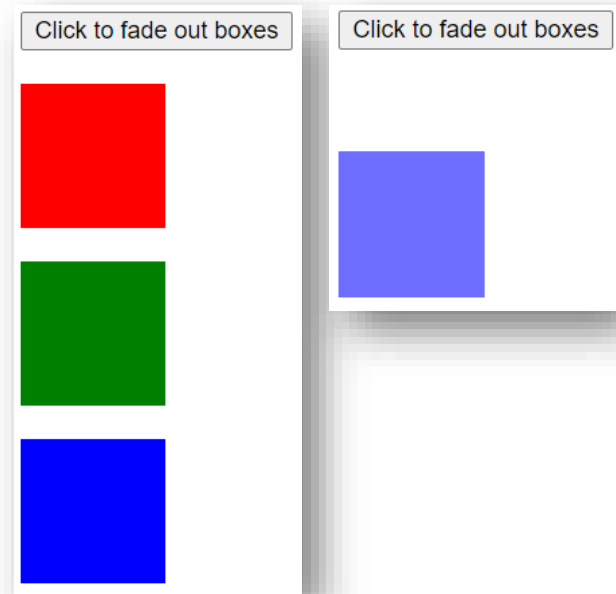
---

## ▪ fadeIn() và fadeOut():

Ví dụ:

```
.box{  
    width:80px;  
    height:80px;  
    display:none;  
}
```

```
<script>  
    $(document).ready(function(){  
        $("button").click(function(){  
            $("#div1").fadeOut();  
            $("#div2").fadeOut("slow");  
            $("#div3").fadeOut(3000);  
        });  
    });  
</script>
```



# jQuery Effects

---

- **fadeToggle()**: chuyển đổi giữa các phương thức **fadeIn ()** và **fadeOut ()**.

- Cú pháp: `$(selector).fadeToggle(speed,callback);`

Ví dụ:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("#div1").fadeToggle();
            $("#div2").fadeToggle("slow");
            $("#div3").fadeToggle(3000);
        });
    });
</script>
```

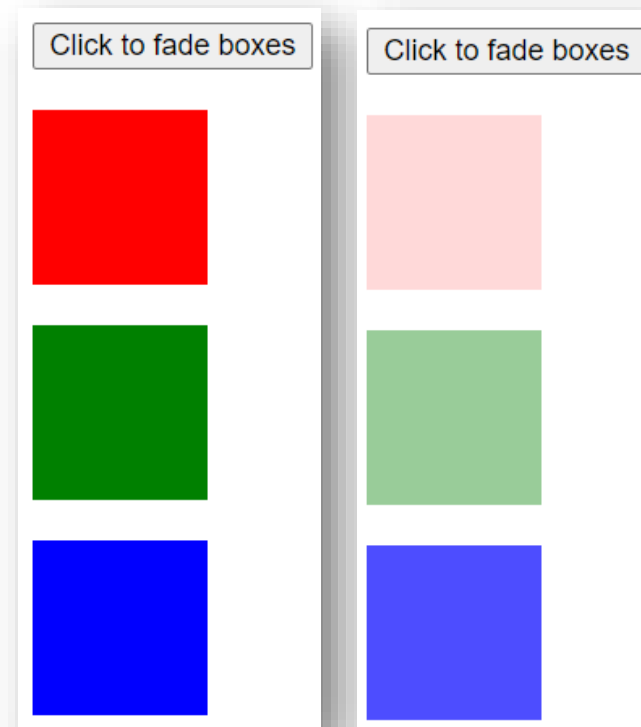
# jQuery Effects

---

- **fadeTo()**: làm cho đối tượng mờ dần đến một độ mờ nhất định (giá trị từ 0 đến 1).

Ví dụ:

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").fadeTo("slow", 0.15);
        $("#div2").fadeTo("slow", 0.4);
        $("#div3").fadeTo("slow", 0.7);
    });
});
</script>
```



# jQuery Effects

---

- **slideUp ()** và **slideDown ()** ẩn hoặc hiển thị các phần tử HTML bằng cách giảm dần hoặc tăng chiều cao của chúng (trượt lên hoặc xuống).

- **Cú pháp:**

```
$(selector).slideDown(speed,callback);  
$(selector).slideUp(speed,callback);
```

Ví dụ

```
#panel, #flip {  
    padding: 5px;  
    text-align: center;  
    background-color: #e5eccc;  
    border: solid 1px #c3c3c3;  
}
```

```
#panel {  
    padding: 50px;  
    display: none;  
}
```

# jQuery Effects

---

## ▪ `slideUp ()` và `slideDown ()`

Ví dụ

```
<div id="flip">Click to slide down panel</div>  
<div id="panel">Hello world!</div>
```

```
<script>  
  $(document).ready(function(){  
    $("#flip").click(function(){  
      $("#panel").slideDown("slow");  
    });  
  });  
</script>
```

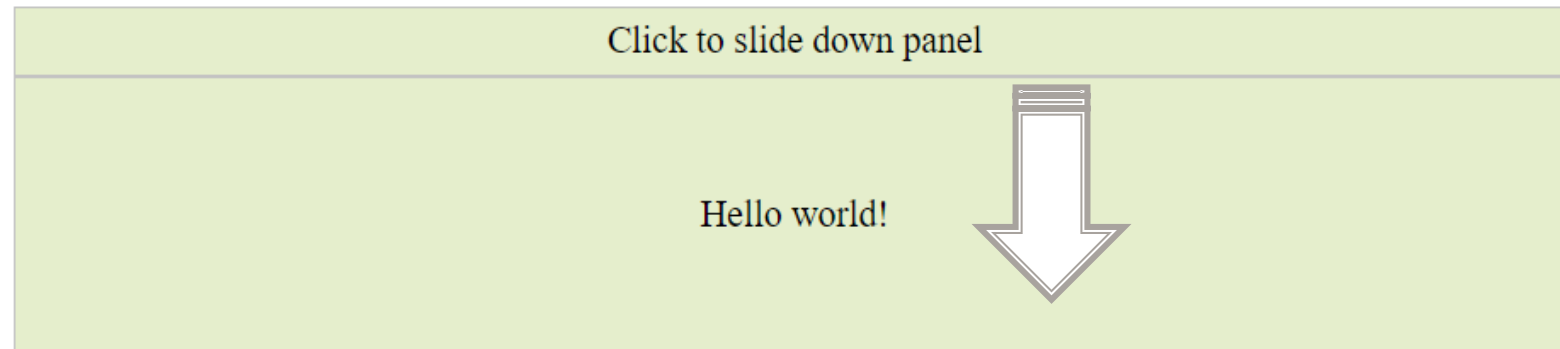
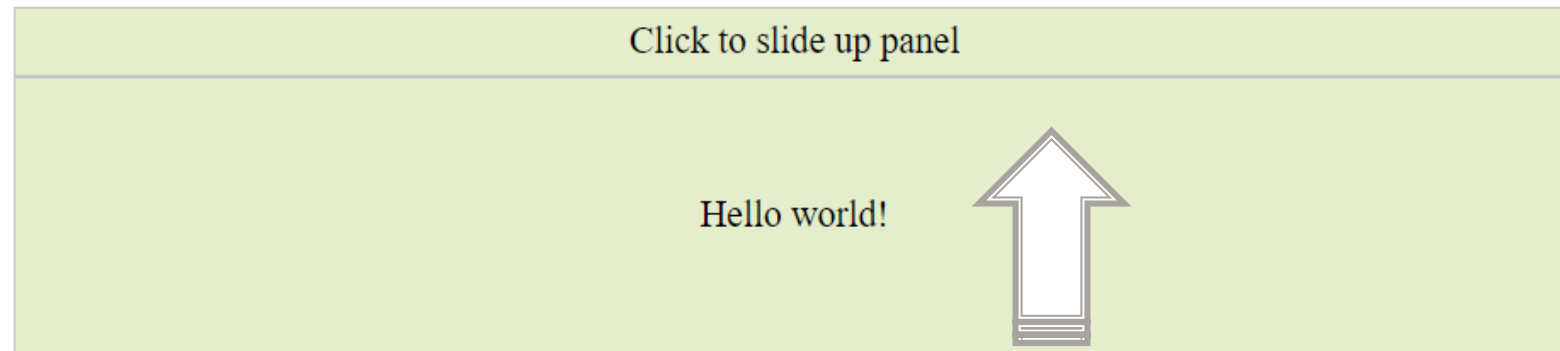
```
<script>  
  $(document).ready(function(){  
    $("#flip").click(function(){  
      $("#panel").slideUp("slow");  
    });  
  });  
</script>
```

# jQuery Effects

---

- **slideUp ()** và **slideDown ()**

Ví dụ



# jQuery Effects

---

- **slideToggle()**: chuyển đổi giữa các phương thức **slideDown()** và **slideUp()**.

Ví dụ:

```
<script>
  $(document).ready(function(){
    $("#flip").click(function(){
      $("#panel").slideToggle("slow");
    });
  });
</script>
```

# jQuery Effects

---

- **jQuery Chaining**: cho phép chạy nhiều lệnh jQuery, cái này đến lệnh khác, trên cùng phần tử. Để xâu chuỗi một hành động, chỉ cần nối hành động đó với hành động trước đó.

Ví dụ

```
<p id="p1">jQuery is fun!!</p>
```

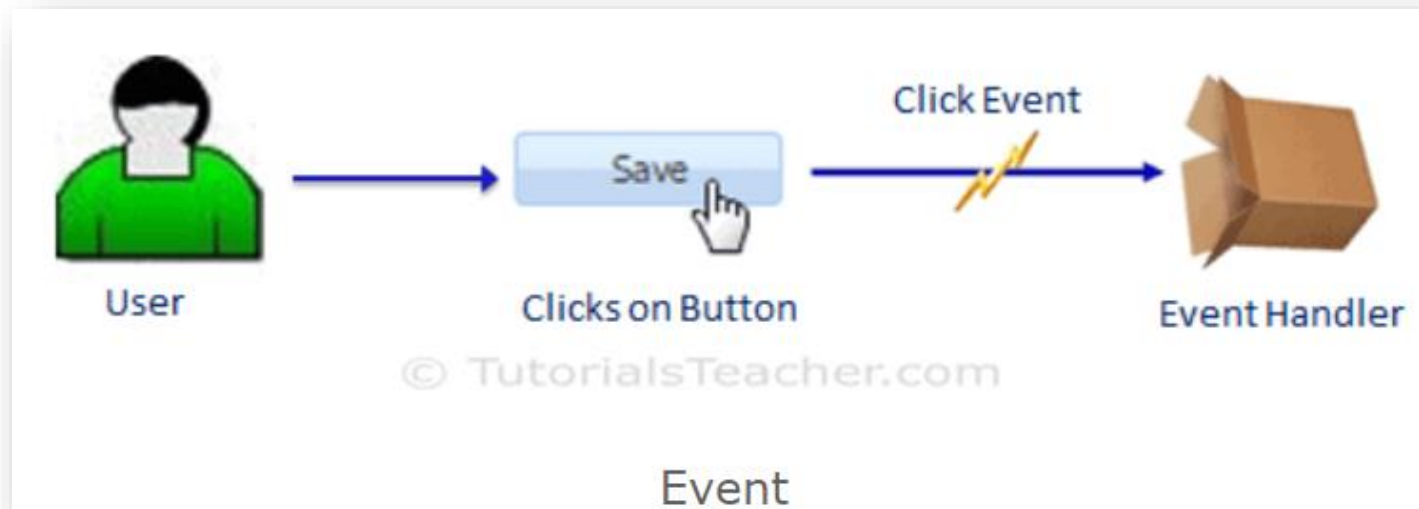
```
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("#p1").css("color","red").slideUp(2000).slideDown(2000);
    });
  });
</script>
```



# jQuery Events

---

- **jQuery Event:** là các hành động của người dùng tương tác với trang web và có thể nhận được phản hồi.
- Trong jQuery, hầu hết các **sự kiện DOM (DOM events)** đều có một **phương thức jQuery (jQuery method)** tương đương



# jQuery Events

---

- **jQuery Event:** Thư viện jQuery cung cấp các phương thức để xử lý các sự kiện DOM

Category	jQuery Method	DOM Event
Form events	blur	onblur
	change	onchange
	focus	onfocus
	focusin	onfocusin
	select	onselect
	submit	onsubmit

# jQuery Events

---

- **jQuery Event:** Thư viện jQuery cung cấp các phương thức để xử lý các sự kiện DOM

Category	jQuery Method	DOM Event
Keyboard events	keydown	onkeydown
	keypress	onkeypress
	keyup	onkeyup
	focusout	
Mouse events	Click	onclick
	dblclick	ondblclick
	focusout	
	hover	

# jQuery Events

---

- **jQuery Event:** Thư viện jQuery cung cấp các phương thức để xử lý các sự kiện DOM

Category	jQuery Method	DOM Event
Mouse events	mousedown	onmousedown
	mouseenter	onmouseenter
	mouseleave	onmouseleave
	mousemove	onmousemove
	mouseout	onmouseout
	mouseover	onmouseover
	mouseup	onmouseup
	Toggle	

# jQuery Events

---

- **jQuery Event:** Thư viện jQuery cung cấp các phương thức để xử lý các sự kiện DOM

Category	jQuery Method	DOM Event
<b>Browser events</b>	Error	onerror()
	Resize	onresize
	Scroll	onscroll
<b>Document loading</b>	Load	onload
	Ready	
	Unload	onunload

# jQuery Events

---

## ▪ Button Click Event

Ví dụ:

```
<script>
  $(document).ready(function () {
    $('#saveBtn').click(function () {
      alert('Save button clicked');
    });
  });
</script>
```

```
<h2>Demo: jQuery click() method</h2>
```

```
<input type="button" value="Save" id="saveBtn" />
```

# jQuery Events

---

## ▪ jQuery Event Object

Ví dụ:

```
<script>
  $(document).ready(function () {
    $('#saveBtn').click(function (eventObj) {
      alert('X =' + eventObj.pageX + ', Y =' + eventObj.pageY);
    });
  });
</script>
```

```
<h2>Demo: jQuery eventObject </h2>
<input type="button" value="Save" id="saveBtn" />
```

# jQuery Events

---

- *this* Keyword in Event Handler

Ví dụ:

```
<script>
$(document).ready(function () {
    $(':button').click(function (eventObj) {
        alert(this.value + ' ' + this.type + ' clicked');
    });
});
</script>
```

```
<h1>Demo: jQuery click() method </h1>
<input type="button" value="Save" id="saveBtn" />
<input type="button" value="Delete" id="delBtn" />
<input type="button" value="Clear" id="clearBtn" />
```



# Commonly Used jQuery Event Methods

---

- **dblclick()**: đính kèm một hàm xử lý sự kiện vào một phần tử HTML, Hàm thực thi khi người dùng double-clicks vào phần tử HTML

**Ví dụ:**

```
<script>
  $(document).ready(function(){
    $("p").dblclick(function(){
      $(this).hide();
    });
  });
</script>
```

```
<p>If you double-click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>
```

# Commonly Used jQuery Event Methods

---

- **mouseenter():** phương thức đính kèm một hàm xử lý sự kiện vào một phần tử HTML, hàm được thực thi khi trỏ chuột vào phần tử HTML

Ví dụ:

```
<script>
    $(document).ready(function(){
        $("#p1").mouseenter(function(){
            alert("You entered p1!");
        });
    });
</script>
```

```
<p id="p1">Enter this paragraph.</p>
```

# Commonly Used jQuery Event Methods

---

- **mouseleave():** phương thức đính kèm một hàm xử lý sự kiện vào một phần tử HTML. Hàm được thực thi khi con trỏ chuột rời khỏi phần tử HTML
- **Cú pháp:**

```
$(selector).mouseleave()
```

```
$(selector).mouseleave(function)
```

# Commonly Used jQuery Event Methods

---

## ▪ **mouseleave():**

Ví dụ:

```
<script>
    $(document).ready(function(){
        $("p").mouseenter(function(){
            $("p").css("background-color", "yellow");
        });
        $("p").mouseleave(function(){
            $("p").css("background-color", "lightgray");
        });
    });
</script>
```

# Commonly Used jQuery Event Methods

---

- **hover():** nhận hai hàm và là sự kết hợp của phương thức mouseenter () và mouseleave ().

Ví dụ:

```
<script>
    $(document).ready(function(){
        $("#p1").hover(function(){
            alert("You entered p1!");
        },
        function(){
            alert("Bye! You now leave p1!");
        });
    });
</script>
```

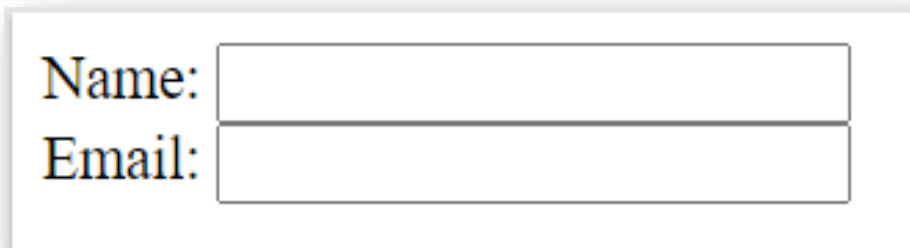
# Commonly Used jQuery Event Methods

---

- **focus()**: phương thức đính kèm một hàm xử lý sự kiện vào một trường biểu mẫu HTML. Hàm được thực thi khi field của form nhận focus

Ví dụ:

```
Name: <input type="text" name="fullname"><br>  
Email: <input type="text" name="email">
```



Name:

Email:

# Commonly Used jQuery Event Methods

---

## ▪ **focus():**

Ví dụ:

```
<script>
  $(document).ready(function(){
    $("input").focus(function(){
      $(this).css("background-color", "yellow");
    });
    $("input").blur(function(){
      $(this).css("background-color", "green");
    });
  });
</script>
```

Name:

Email:

# Commonly Used jQuery Event Methods

---

- **blur()**: hàm thực thi khi form field mất focus

Ví dụ:

```
<script>
  $(document).ready(function(){
    $("input").focus(function(){
      $(this).css("background-color", "yellow");
    });
    $("input").blur(function(){
      $(this).css("background-color", "green");
    });
  });
</script>
```



# Commonly Used jQuery Event Methods

---

- **on():** sử dụng để đính kèm một hoặc nhiều trình xử lý sự kiện cho các phần tử đã chọn và các phần tử con trong cây DOM.
- **Cú pháp**

```
$(selector).on(event, childSelector, data, function, map)
```

Ví dụ:

```
<script>
  $(document).ready(function(){
    $("p").on("click", function(){
      $(this).hide();
    });
  });
</script>
```

# Commonly Used jQuery Event Methods

---

## ▪ on():

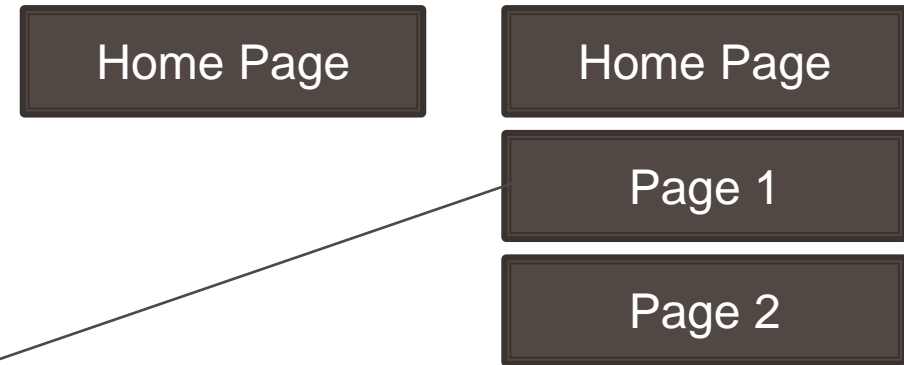
Ví dụ:

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

# Bài tập

---

- Thiết kế trang Index có menu:



- Thiết kế form gồm các phần tử

User Name

Password

Submit

Áp dụng phương thức **focus**  
**và blur**

- Thay đổi trạng thái của text
- Hiện thị thông báo lỗi nếu người dùng chưa nhập nội dung



---

---

# AJAX

---

---

# Giới thiệu AJAX

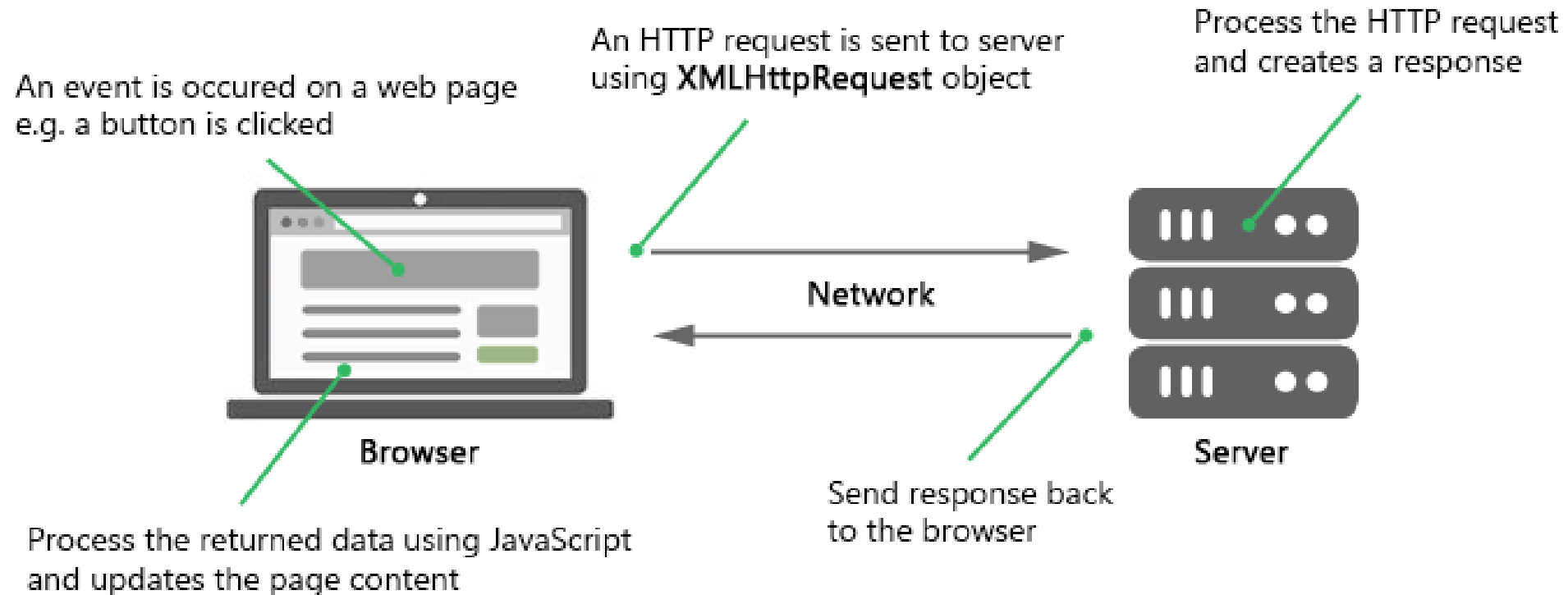
---

- **AJAX - Asynchronous JavaScript and XML:** là một bộ công cụ cho phép load dữ liệu từ server mà không yêu cầu tải lại trang.
- **AJAX** sử dụng chức năng sẵn có **XMLHttpRequest** (XHR) của trình duyệt để thực hiện một yêu cầu đến server và xử lý dữ liệu server trả về.



# Giới thiệu AJAX

---



# Sử dụng AJAX trong Javascript

---

- **Khởi tạo đối tượng:** Để sử dụng AJAX trong Javascript thì cần sử dụng đối tượng **XMLHttpRequest**.
- **Cú pháp:**

```
var variableName = new XMLHttpRequest();
```

  - **variableName** là tên biến gán cho object **XMLHttpRequest**.

# Sử dụng AJAX trong Javascript

---

- **Gửi yêu cầu đến server** là đối tượng yêu cầu mới được tạo bằng phương thức **open ()** của đối tượng **XMLHttpRequest**
  - **Phương thức open()** nhận 2 tham số: "GET", "POST", và URL

```
request.open("GET", "info.txt");  
request.open("POST", "add-user.php");
```



# Sử dụng AJAX trong Javascript

---

- **Gửi yêu cầu đến server** bằng phương thức **send()** của đối tượng XMLHttpRequest.

```
request.send();  
request.send(body);
```

# Sử dụng AJAX trong Javascript

---

■ Ví dụ:

```
<div id="demo">  
  <h2>Let AJAX change this text</h2>  
  <button type="button" onclick="loadDoc()">  
    Change Content  
  </button>  
</div>
```

**The XMLHttpRequest Object**

Change Content

# Sử dụng AJAX trong Javascript

---

■ Ví dụ:

```
function loadDoc() {  
  const xhttp = new XMLHttpRequest();  
  xhttp.onload = function() {  
    document.getElementById("demo").innerHTML= this.responseText;  
  }  
  xhttp.open("GET", "ajax_info.txt", true);  
  xhttp.send();  
}
```

# Sử dụng AJAX trong Javascript

---

- **Các thuộc tính của XMLHttpRequest:**

- **onreadystatechange:** Xác định một hàm được gọi khi thuộc tính **readyState** thay đổi.
- **readyState:** Trạng thái của **XMLHttpRequest**. Gồm các giá trị sau:
  - 0 - request chưa được khởi tạo.
  - 1 - kết nối đến server đang được thiết lập.
  - 2 - yêu cầu đã nhận được.
  - 3 - đang tiến hành xử lý.
  - 4 - request đã xong và dữ liệu trả về đã sẵn sàng để xử lý.

# Sử dụng AJAX trong Javascript

---

- **Các thuộc tính của XMLHttpRequest:**

- **responseText:** Giá trị trả về dưới dạng string.
- **responseXML:** Giá trị trả về dưới dạng XML.
- **status:** Trả về trạng thái của request.
  - Ví dụ: 200: "OK"
  - 403: "Forbidden"
  - 404: "Not Found"
- **statusText:** Trả về trạng thái của request dưới dạng text.  
VD: Ok, Not Found.

# Sử dụng AJAX trong Javascript

---

## ▪ Phương thức của XMLHttpRequest

- **new XMLHttpRequest():** Tạo đối tượng XMLHttpRequest.
- **abort():** Hủy Request hiện tại.
- **getAllResponseHeaders():** Lấy ra thông tin header.
- **getResponseHeader():** Trả về cụ thể thông tin header.
- **open(method, url, async, user, psw):** Cấu hình cho một request mới.
- **send(string):** Gửi dữ liệu đến server đã được cấu hình ở phương thức open(). Trong đó string là data truyền theo nếu request là POST.
- **setRequestHeader():** Thiết lập các thông số header gửi lên.

# Cách hoạt động của AJAX

---

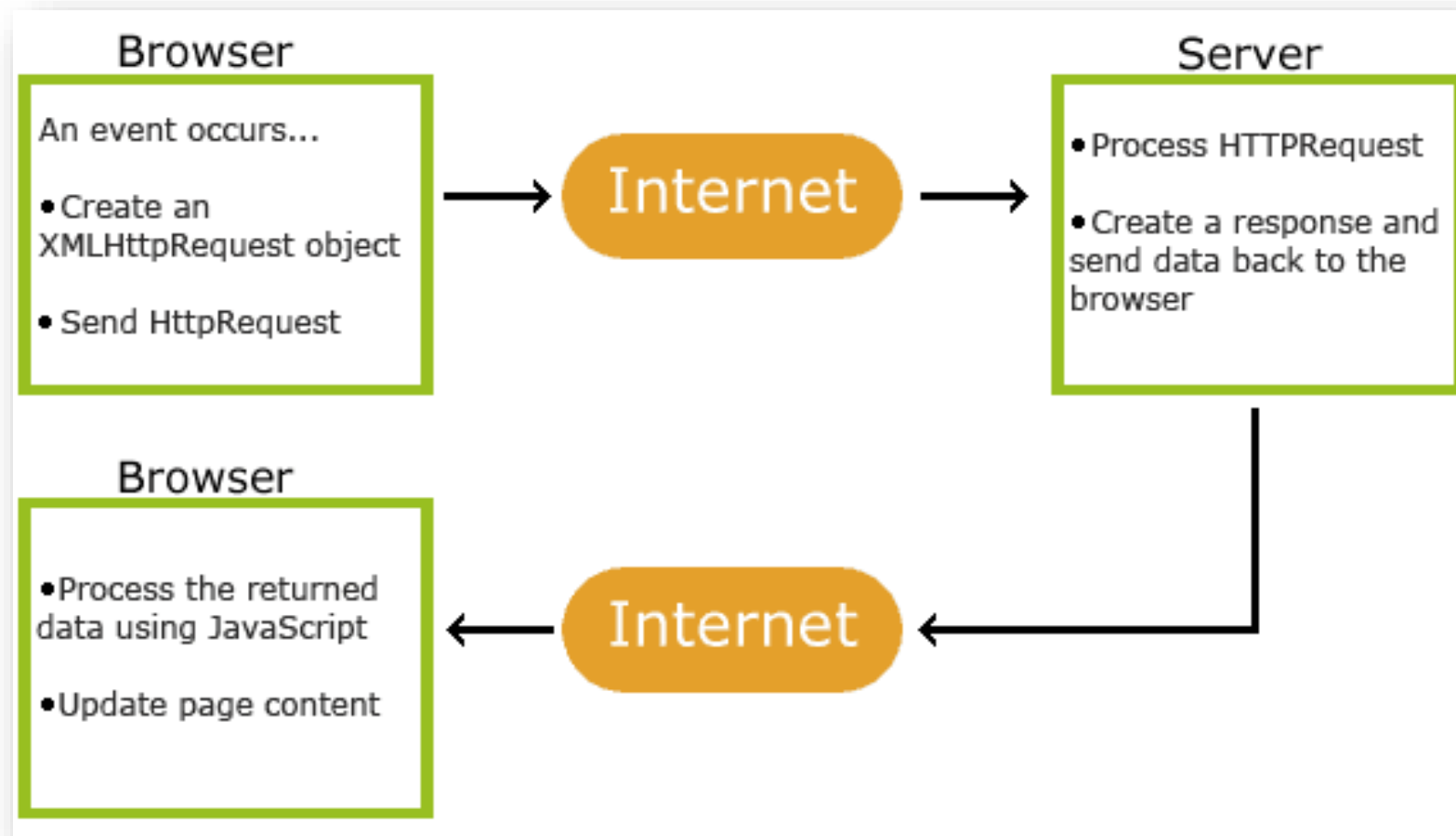
- **Các thành phần của AJAX:**

- **HTML/XHTML** làm ngôn ngữ chính và **CSS** để tạo styles.
- **The Document Object Model (DOM)** thị dữ liệu động và tạo tương tác.
- **XML** để trao đổi dữ liệu nội bộ và **XSLT** để xử lý dữ liệu hoặc dùng **JSON**.
- **XMLHttpRequest object** để giao tiếp bất đồng bộ.
- **JavaScript** làm ngôn ngữ lập trình để kết nối toàn bộ các công nghệ trên lại.

# Cách hoạt động của AJAX

---

## ▪ Hoạt động của AJAX





# Cách hoạt động của AJAX

---

## ▪ Hoạt động của AJAX

- **Trình duyệt** tạo một lệnh gọi JavaScript để kích hoạt **XMLHttpRequest** và tạo một yêu cầu **HttpRequest** gửi lên server.
- **Server** tiếp nhận, truy xuất và gửi lại dữ liệu cho trình duyệt.
- **Trình duyệt** nhận dữ liệu từ server và hiển thị lên trang. Không cần tải lại toàn bộ trang.

# POST & GET Request trong AJAX

---

## ▪ GET Request

- Phương thức GET thường được sử dụng để lấy hoặc truy xuất một số loại thông tin từ máy chủ mà không yêu cầu bất kì thao tác hoặc thay đổi nào trong cơ sở dữ liệu.

## ▪ POST Request

- Phương thức POST chủ yếu được sử dụng để gửi dữ liệu biểu mẫu (form) đến máy chủ web.

# AJAX and jQuery

---

- **jQuery** cung cấp các phương thức để thực hiện các chức năng ajax.
- Với các phương thức **jQuery AJAX**, người dùng có thể yêu cầu các TEXT, HTML, XML và JSON từ server sử dụng cả giao thức HTTP GET và HTTP POST
- Ngoài ra, các phương thức này cũng có thể tải dữ liệu từ xa trực tiếp vào các phần tử HTML được chọn trên trang HTML

# Các phương thức cơ bản của jQuery AJAX

---

- **jQuery Load:** Phương thức load() tải dữ liệu từ máy chủ và đưa dữ liệu trả về vào phần tử đã chọn.

- **Cú pháp:**

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

- Trong đó:
  - **URL:** địa chỉ của trang cần lấy dữ liệu
  - **data:** Cặp **key/value** gửi đi cùng với yêu cầu.
  - **callback:** Tên của một hàm sẽ được thực thi sau khi hoàn tất phương thức **load()**.

# Các phương thức cơ bản của jQuery AJAX

---

## ▪ jQuery Load:

Ví dụ:

```
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("#box").load("/examples/html/test-content.html");
    });
  });
</script>
```

# Các phương thức cơ bản của jQuery AJAX

- **jQuery Load:**

Ví dụ (tt):

Click button to load new content inside DIV box

Load Content



## Simple Ajax Demo

This is a simple example of Ajax loading.



```
<div id="box">  
  <h2>Click button to load new content inside DIV box</h2>  
</div>  
<button type="button">Load Content</button>
```

# Các phương thức cơ bản của jQuery AJAX

---

- **jQuery get ()**: gửi http GET request không đồng bộ đến máy chủ và truy xuất dữ liệu. GET được sử dụng để lấy một số dữ liệu từ máy chủ, có thể trả về dữ liệu đã lưu trong bộ nhớ cache.
- **Cú pháp**:

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

# Các phương thức cơ bản của jQuery AJAX

---

## ▪ jQuery get ():

Ví dụ:

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



# Các phương thức cơ bản của jQuery AJAX

---

## ▪ jQuery get ():

Ví dụ:

Send an HTTP GET request to a page and get the result back

click

This page says

Data: This is some text from an external ASP file.

Status: success

OK

Demo\_test.asp

```
<%  
    response.write("This is some text from an external ASP file.")  
%>
```

# Các phương thức cơ bản của jQuery AJAX

---

- **jQuery post ()**: phương thức yêu cầu dữ liệu từ máy chủ bằng cách sử dụng **HTTP POST request**

- **Cú pháp:** `$.post(URL,data,callback);`

Ví dụ

```
<button>
```

Send an HTTP POST request to a page and get the result back

```
</button>
```

# Các phương thức cơ bản của jQuery AJAX

---

## ▪ jQuery post ():

Ví dụ

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $.post("demo_test_post.asp",{
            name: "Donald Duck",
            city: "Duckburg"},
            function(data,status){
                alert("Data: " + data + "\nStatus: " + status);
            });
    });
});
</script>
```

# Các phương thức cơ bản của jQuery AJAX

---

- **jQuery ajax ()**: Phương thức **ajax()** cung cấp chức năng cốt lõi của Ajax trong jQuery. Nó gửi các HTTP requests đến máy chủ.

- **Cú pháp**:

```
$.ajax(url);  
$.ajax(url,[options]);
```

- **url**: địa chỉ của trang muốn gửi hoặc truy xuất dữ liệu tùy chọn:
- **options**: thẻ được chỉ định bằng cách sử dụng định dạng JSON. Tham số này là tùy chọn.

# Các phương thức cơ bản của jQuery AJAX

---

## ▪ jQuery ajax ():

Ví dụ:

```
<script>
$(document).ready(function () {
    $('#ajaxBtn').click(function(){
        $.ajax('/jquery/getdata',// request url{
            success: function (data, status, xhr) {// callback function
                $('#p').append(data);
            }
        });
    });
});
</script>
```

# Các phương thức cơ bản của jQuery AJAX

---

## ▪ jQuery ajax ():

Ví dụ (tt):

```
<h1> jQuery ajax() demo </h1>  
<input type="button" id="ajaxBtn" value="Send Ajax request" />  
<p>  
</p>
```

jQuery ajax() demo

Send Ajax request

Click

jQuery ajax() demo

Send Ajax request

This is dummy data

# JSON

---

- **JSON (JavaScript Object Notation):** là một định dạng dựa trên văn bản tiêu chuẩn để biểu diễn dữ liệu có cấu trúc dựa trên cú pháp đối tượng JavaScript.
- Một chương trình JavaScript có thể dễ dàng chuyển đổi dữ liệu JSON thành các đối tượng JavaScript.
- Vì định dạng chỉ là văn bản nên dữ liệu JSON có thể dễ dàng được gửi giữa các máy tính và được sử dụng bởi bất kỳ ngôn ngữ lập trình nào.
- JavaScript có một chức năng tích hợp để chuyển đổi các chuỗi JSON thành các đối tượng JavaScript:

# JSON

---

- **JSON (JavaScript Object Notation)**

Sinh viên tự tìm hiểu!!!