

## Contents

Chương 1 - Giới thiệu về jQuery.....	1
Chương 2 – jQuery Selectors .....	7
Chương 3 - Attributes .....	20
Chương 4 – Sự kiện (Events).....	23
Chương 5 – Hiệu ứng (Effects).....	42
Chương 6 – Sửa đổi DOM.....	60
Chương 7: AJAX – Phần 1 .....	77
Chương 7 – AJAX – Phần 2 .....	91

# Chương 1 - Giới thiệu về jQuery

Với sự phát triển rất mau lẹ của Internet, người dùng ngày càng quan tâm hơn đến hình thức của một trang web. Trước đây một trang web chỉ cần có banner, nội dung và ít footer hời hợt là đã được cho là một trang web hoàn chỉnh. Nhưng bây giờ trang web đó phải có banner bắt mắt, nội dung hay và còn nhiều hiệu ứng lạ mắt khác nữa thì mới có thể thu hút được người đọc.

Chính vì thế những web designer bắt đầu chú ý đến các thư viện JavaScript mở như jQuery để tạo ra các hiệu ứng có thể tương tác trực tiếp với người đọc một cách nhanh chóng và dễ dàng hơn rất nhiều là sử dụng thuần JavaScript.

Nhưng nếu bạn là người mới làm quen với jQuery bạn sẽ thấy không biết phải bắt đầu từ đâu vì jQuery cũng giống như bất cứ thư viện nào khác cũng có rất nhiều functions. Cho dù bạn có đọc phần tài liệu hướng dẫn sử dụng của jQuery thì bạn vẫn thấy rất phức tạp và khó hiểu. Nhưng bạn yên tâm một điều là jQuery có cấu trúc rất mạch lạc và theo hệ thống. Cách viết code của jQuery được vay mượn từ các nguồn mà các web designer đa phần đã biết như HTML và CSS. Nếu từ trước đến nay bạn chỉ là Designer chứ không phải coder, bạn cũng có thể dễ dàng học jQuery vì kiến thức về CSS giúp bạn rất nhiều khi bắt đầu với jQuery.

Nhận thấy jQuery còn khá mới mẻ với nhiều bạn và nó cũng là thư viện được đông đảo người sử dụng. Izwebz giới thiệu đến các bạn loạt bài về jQuery. Trong loạt bài này chúng ta sẽ tìm hiểu về jQuery và các tính năng của nó. Trước hết bạn nên biết jQuery có thể làm được những gì.

Những gì JQuery có thể làm

**Hướng tới các thành phần trong tài liệu HTML.** Nếu không sử dụng thư viện JavaScript này, bạn phải viết rất nhiều dòng code mới có thể đạt được mục tiêu là di chuyển trong cấu

trúc cây (hay còn gọi là DOM = Document Object Model) của một tài liệu HTML và chọn ra các thành phần liên quan. JQuery cho phép bạn chọn bất cứ thành phần nào của tài liệu để “vọc” một cách dễ dàng như sử dụng CSS.

**Thay đổi giao diện của một trang web.** CSS là công cụ rất mạnh để định dạng một trang web nhưng nó có một nhược điểm là không phải tất cả các trình duyệt đều hiển thị giống nhau. Cho nên JQuery ra đời để lấp chỗ trống này, vì vậy các bạn có thể sử dụng nó để giúp trang web có thể hiển thị tốt trên hầu hết các trình duyệt. Hơn nữa JQuery cũng có thể thay đổi class hoặc những định dạng CSS đã được áp dụng lên bất cứ thành phần nào của tài liệu HTML ngay cả khi trang web đó đã được trình duyệt load thành công. Thay đổi nội dung của tài liệu. JQuery không phải chỉ có thể thay đổi bề ngoài của trang web, nó cũng có thể thay đổi nội dung của chính tài liệu đó chỉ với vài dòng code. Nó có thể thêm hoặc bớt nội dung trên trang, hình ảnh có thể được thêm vào hoặc đổi sang hình khác, danh sách có thể được sắp xếp lại hoặc thậm chí cả cấu trúc HTML của một trang web cũng có thể được viết lại và mở rộng. Tất cả những điều này bạn hoàn toàn có thể làm được nhờ sự giúp đỡ của API (Application Programming Interface = Giao diện lập trình ứng dụng).

**Tương tác với người dùng.** Cho dù công cụ bạn dùng có mạnh mẽ đến mấy, nhưng nếu bạn không có quyền quyết định khi nào nó được sử dụng thì công cụ đó cũng coi như bỏ. Với thư viện JavaScript như JQuery, nó cho bạn nhiều cách để tương tác với người dùng ví dụ như khi người dùng nhấp chuột vào đường link thì sẽ có gì xảy ra. Nhưng cái hay của nó là không làm cho code HTML của bạn rối tung lên chính là nhờ các Event Handlers. Hơn nữa Event Handler API sẽ bảo đảm rằng trang web của bạn tương thích hầu hết với các trình duyệt, điều này đã và đang làm đau đầu rất nhiều các web designer.

**Tạo hiệu ứng động cho những thay đổi của tài liệu.** Để tương tác tốt với người dùng, các web designer phải cho người dùng thấy được hiệu ứng gì sẽ xảy ra khi họ làm một tác vụ nào đó. JQuery cho phép bạn sử dụng rất nhiều hiệu ứng động như mờ dần, chạy dọc chạy ngang v.v.. và nếu vẫn chưa đủ, nó còn cho phép bạn tự tạo ra các hiệu ứng của riêng mình.

**Lấy thông tin từ server mà không cần tải lại trang web.** Đây chính là công nghệ ngày càng trở nên phổ biến Asynchronous JavaScript And XML (AJAX), nó giúp người thiết kế web tạo ra những trang web tương tác cực tốt và nhiều tính năng. Thư viện JQuery loại bỏ sự phức tạp của trình duyệt trong quá trình này và cho phép người phát triển web có thể tập trung vào các tính năng đầu cuối. Đơn giản hoá các tác vụ JavaScript. Ngoài những tính năng như đã nêu ở trên, JQuery còn cho phép bạn viết code JavaScript đơn giản hơn nhiều so với cách truyền thống như là các vòng lặp và điều khiển mảng.

Tại sao JQuery làm việc tốt

Người dùng ngày càng quan tâm hơn đến Dynamic HTML, đó cũng là nền móng cho sự ra đời của những JavaScript Frameworks. Có frameworks thì chỉ tập trung vào một vài tính năng vừa nêu ở trên, có cái thì rườm rà bao gồm tất cả những hiệu ứng, tập tính và nhồi nhét vào một package. Để đảm bảo là một thư viện “nhẹ nhàng” nhưng vẫn “ngon bổ rẻ” với các tính năng đã nêu ở trên, JQuery sử dụng những chiến lược sau:

**Tận dụng kiến thức về CSS.** Các JQuery Selector hoạt động y chang như CSS Selector với cùng cấu trúc và cú pháp. Chính vì thế thư viện JQuery là cửa ngõ cho các web designer muốn thêm nhiều tính năng hơn nữa cho trang web của mình. Bởi vì điều kiện tiên quyết để

trở thành một web designer chuyên nghiệp là khả năng sử dụng CSS thuần thục. Với kiến thức có sẵn về CSS, bạn sẽ có sự khởi đầu thuận lợi với jQuery.

**Hỗ trợ Plugin.** Để tránh bị rơi vào trạng thái quá tải tính năng, jQuery cho phép người dùng tạo và sử dụng Plugin nếu cần. Cách tạo một plugin mới cũng khá đơn giản và được hướng dẫn cụ thể, chính vì thế cộng đồng sử dụng jQuery đã tạo ra một loạt những plugin đầy tính sáng tạo và hữu dụng.

**Xoá nhòa sự khác biệt giữa trình duyệt.** Một thực tế tồn tại là mỗi một hệ thống trình duyệt lại có một kiểu riêng để đọc trang web. Dẫn đến một điều làm đau đầu các web designer là làm thế nào để cho trang web có thể hiển thị tốt trên mọi trình duyệt. Cho nên đôi khi người ta phải làm hẳn một phần code phức tạp để đảm bảo rằng trang web của họ được hiển thị gần như tương đồng ở các trình duyệt phổ biến. JQuery giúp bạn thêm một lớp bảo vệ cho sự khác biệt của trình duyệt và giúp quá trình này diễn ra dễ dàng hơn rất nhiều.

**Luôn làm việc với Set.** Ví dụ khi chúng ta yêu cầu jQuery tìm tất cả các thành phần có class là delete và ẩn chúng đi. Chúng ta không cần phải loop qua từng thành phần được trả về. Thay vào đó, những phương pháp như là hide() được thiết kế ra để làm việc với set thay vì từng thành phần đơn lẻ. Kỹ thuật này được gọi là vòng lặp ẩn, điều đó có nghĩa là chúng ta không phải tự viết code để loop nữa mà nó vẫn được thực thi, chính vì thế code của chúng ta sẽ ngắn hơn rất nhiều.

**Cho phép nhiều tác vụ diễn ra trên cùng một dòng.** Để tránh phải sử dụng những biến tạm hoặc các tác vụ lặp tốn thời gian, jQuery cho phép bạn sử dụng kiểu lập trình được gọi là Chaining cho hầu hết các method của nó. Điều đó có nghĩa là kết quả của các tác vụ được tiến hành trên một thành phần chính là thành phần đó, nó sẵn sàng cho tác vụ tiếp theo được áp dụng lên nó. Những chiến lược được nêu ở trên giúp kích thước của jQuery rất nhỏ bé chỉ khoảng trên dưới 20Kb dạng nén. Nhưng vẫn đảm bảo cung cấp cho chúng ta những kỹ thuật để giúp code trên trang nhỏ gọn và mạch lạc.

Jquery sở dĩ trở nên phổ biến là do cách sử dụng đơn giản và bên cạnh đó còn có một cộng đồng sử dụng mạnh mẽ vẫn ngày ngày phát triển thêm Plugin và hoàn thiện những tính năng trọng tâm của jQuery. Cho dù thực tế là vậy, nhưng jQuery lại là thư viện JavaScript hoàn toàn miễn phí cho mọi người sử dụng. Tất nhiên nó được bảo vệ bởi luật GNU Public License và MIT License, nhưng bạn cứ yên tâm là bạn có thể sử dụng nó trong hầu hết các trường hợp kể cả thương mại lẫn cá nhân.

Tạo trang web đầu tiên với sự hỗ trợ của jQuery

Bởi vì jQuery là một thư viện JavaScript do vậy để sử dụng nó bạn phải chèn nó vào trang web thì mới có thể sử dụng được. Có hai cách để chèn jQuery vào một trang web.

## 1. Tự host jQuery

Vào trang chủ của [jQuery](https://jquery.com/) và download phiên bản mới nhất. Thường thì có 2 phiên bản của jQuery cho bạn download. Phiên bản chưa nén dành cho những người phát triển và đang học như bạn. Còn phiên bản nén kia dành cho phần sử dụng trực tiếp trên trang vì nó có dung lượng nhỏ hơn rất nhiều so với phiên bản chưa nén. Bạn không cần phải cài đặt jQuery, bạn chỉ cần đặt đường link tới thư viện đó là được. Bất cứ khi nào bạn cần sử dụng jQuery, bạn chỉ cần gọi nó trong tài liệu HTML đến nơi lưu trữ nó trên host của bạn.

## 2. Dùng phiên bản có sẵn trên server của Google

Ngoài cách trên ra bạn cũng có thể sử dụng phiên bản nén của jQuery có sẵn trên server của Google. Sử dụng cách này có 2 điều lợi là a) tiết kiệm băng thông cho trang web của bạn và b) jQuery sẽ được load nhanh hơn nếu máy của người dùng đã cache jQuery.

Tuy nhiên trong phần sắp tới chúng ta sẽ sử dụng phiên bản có sẵn trên server của Google mà không cần phải download về máy. Cú pháp để chèn jQuery sử dụng file có sẵn trên server của Google như sau:

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
type="text/javascript"></script>
```

### Chuẩn bị tài liệu HTML

Trong hầu hết các ví dụ được sử dụng trong loạt bài này thì có 3 thành phần được sử dụng nhiều nhất đó chính là tài liệu HTML, Stylesheet CSS và một tài liệu JavaScript để thực hiện lệnh trên đó. Trong ví dụ đầu tiên chúng ta sẽ sử dụng một tài liệu HTML đơn giản với một header, sidebar, content và footer. Trong phần content sẽ có 3 đoạn văn bản và một số class có sẵn. Tất nhiên bạn phải sử dụng CSS để định dạng cho tài liệu HTML này. Bởi vì đây là tutorial về jQuery cho nên tôi sẽ không giải thích về các thuộc tính cũng như chức năng của CSS. Nếu có điểm nào không rõ bạn có thể tham khảo phần CSS ngay trên izwebz.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>jQuery Introduction</title>
<link rel="stylesheet" href="stylesheet.css" type="text/css" media="screen"
/>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
type="text/javascript"></script>
<script src="first-jquery.js" type="text/javascript"></script>
</head>
<body>
<div id="wrapper">
<div id="header">
<h1 id="logo">jQuery Introduction</h1>
</div>
<div id="mainContent">
<div id="sidebar">
<ul id="nav">
<li><a href="#">Home Page</a></li>
<li><a href="#">About Me</a></li>
<li><a href="#">Forum</a></li>
<li><a href="ebook.pdf">Ebooks</a></li>
<li><a href="http://www.jquery.com">Tutorials</a></li>
<li><a href="#">Photoshop</a></li>
<li><a href="mailto:email@yahoo.com">Email</a></li>
</ul>
</div><!--end #sidebar-->
<div id="primary">
<h3>Lorem ipsum dolor sit amet</h3>
<p class="text">
some text here
```

```

    </p>
    <div>
      <h3>Lorem ipsum dolor sit amet</h3>
      <p>
        some text here
      </p>
    </div>
    <h3>Lorem ipsum dolor sit amet</h3>
    <p class="text">
      some text here
    </p>
  </div><!--end #primary-->
</div><!--end #mainContent-->
<div id="footer">
  <p>&copy;2010 Izwebz - Demon Warlock</p>
</div><!--end #footer-->
</div><!--end #wrapper-->
</body>
</html>

```

Ở đoạn code trên bạn dễ dàng nhận thấy rằng thư viện jQuery được đặt ở dưới Stylesheet. Đây là một điểm rất quan trọng mà bạn cần lưu ý là thứ tự của các file khi gọi. Ban đầu phải là CSS load trước, khi trang web đã load xong phần CSS thì chúng ta mới thêm vào phần thư viện jQuery cuối cùng mới là code jQuery chúng ta tự viết ra. Nếu không khi code jQuery của bạn sẽ không làm việc đúng như mong đợi nếu thư viện jQuery chưa được load.

### Bắt đầu code jQuery

Bây giờ bạn mở trình soạn thảo code lên và tạo một file tên là first-jquery.js và file này đã được chúng ta chèn vào trong dòng code:

```
<script src="first-jquery.js" type="text/javascript"></script>
```

Gõ vào file vừa tạo 3 dòng code như sau:

```
$(document).ready(function() {
  $('.text').addClass('important');
});
```

Thao tác cơ bản nhất của jQuery là chọn một phần nào đó của tài liệu HTML. Bạn tiến hành nó bằng cách sử dụng hàm \$(). Thường thì nằm giữa dấu ngoặc () là một chuỗi dưới dạng tham số, nó có thể là những CSS Selectors. Trong ví dụ này chúng ta muốn tìm tất cả những thành phần nào có class="text", cú pháp giống như khi bạn viết code CSS vậy. Tất nhiên ở những bài sau chúng ta sẽ tham khảo thêm nhiều những lựa chọn khác hay hơn nữa. Trong chương 2 chúng ta sẽ nghiên cứu một vài cách khác để lựa chọn các thành phần trong tài liệu HTML.

Hàm \$() chính là một jQuery Object, đây là nền móng cho tất cả những gì chúng ta sẽ học từ bây giờ. JQuery Object bao gồm không hoặc nhiều thành phần DOM và cho phép chúng ta tương tác với chúng bằng nhiều cách. Trong trường hợp này chúng ta muốn thay đổi cách hiển thị của những phần này trong trang, chúng ta thực hiện nó bằng cách thay đổi class của nó.

Thêm vào một class mới

Phương pháp `.addClass()`, cũng giống như hầu hết các phương pháp jQuery khác, được đặt tên theo chức năng của nó. Khi được gọi, nó sẽ thêm một class vào thành phần chúng ta đã chọn. Tham số duy nhất của nó là tên class sẽ được thêm vào. Phương pháp này và đối ngược với nó là `.removeClass()`, sẽ cho phép chúng ta quan sát jQuery hoạt động như thế nào khi chúng ta khám phá những phương pháp lựa chọn có sẵn của jQuery. Còn bây giờ, code jQuery của chúng ta chỉ đơn giản thêm một `class="important"`, và class này đã được khai báo trong stylesheet với các thuộc tính như viền đỏ và nền hồng nhạt.

```
border: 1px solid red; background: pink;
```

## Lorem ipsum dolor sit amet

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.

## Lorem ipsum dolor sit amet

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.

## Lorem ipsum dolor sit amet

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.

Bạn cũng nhận thấy rằng chúng ta không phải chạy một vòng lặp nào để thêm class vào các đoạn văn bản có cùng chung class. Đây chính là vòng lặp ẩn của các phương pháp jQuery, như trong ví dụ này là `.addClass()`, cho nên bạn chỉ phải gọi đúng một lần và chỉ có vậy để thay đổi những thành phần bạn muốn trong tài liệu. Bây giờ nếu bạn chạy thử trang web vừa tạo trên trình duyệt bạn sẽ thấy 2 đoạn văn có cùng class là `.text` sẽ bị tô hồng và có viền màu đỏ.

Đến đây chúng ta đã kết thúc phần một của loạt bài về jQuery. Trong bài này bạn đã biết được jQuery có thể làm những gì? Bạn cũng đã học được cách để sử dụng jQuery trên một tài liệu HTML và cuối cùng là dùng thử một phương pháp của jQuery là `.addClass()`.

## Chương 2 – jQuery Selectors

Thư viện jQuery tận dụng kiến thức và thế mạnh của CSS Selector để cho phép bạn nhanh chóng và dễ dàng truy cập nhiều phần tử hoặc nhóm các phần tử trong DOM (Document Object Model). Trong chương 2 này chúng ta sẽ khám phá một vài những Selector này và cả những Selector của jQuery. Chúng ta cũng sẽ tìm hiểu thêm về cách di chuyển trong cây thư mục và nó cho chúng ta thêm linh động để đạt được những gì mình muốn.

### Document Object Model (Mô hình đối tượng tài liệu)

Một trong những tính năng mạnh mẽ nhất của jQuery là khả năng chọn các thành phần trong DOM một cách dễ dàng. Nói nôm na thì DOM là một dạng phả hệ của các thành phần HTML. Các thành phần này có mối tương quan với nhau như một “gia đình” HTML hạnh phúc. Khi chúng ta nói đến các mối quan hệ này bạn hãy liên tưởng đến mối quan hệ trong gia đình như ông bà, bố mẹ, anh chị em v.v.. Bạn có thể xem bài Hướng đối tượng dựa vào cấp bậc XHTML để biết rõ hơn về mối quan hệ của các thành phần HTML.

### Hàm \$()

Cho dù bạn sử dụng Selector nào đi chăng nữa trong jQuery, bạn luôn bắt đầu bằng một dấu dollar (\$) và một đôi ngoặc đơn như: \$(). Tất cả những gì có thể được sử dụng trong CSS cũng có thể được lồng vào dấu ngoặc kép (") và đặt vào trong hai dấu ngoặc đơn, cho phép chúng ta áp dụng các phương pháp jQuery cho tập hợp các phần tử phù hợp.

Ba thành phần quan trọng nhất của jQuery Selector là tên thẻ HTML, ID và Class. Bạn có thể chỉ sử dụng nó hoặc kết hợp với những Selector khác để chọn. Dưới đây là một ví dụ về mỗi Selector khi sử dụng một mình.

Selector	CSS	jQuery	Diễn giải
<u>Thẻ HTML</u>	P	<u>\$( 'p' )</u>	<u>Chọn tất cả các thẻ p trong tài liệu HTML</u>
ID	<u>#vi-du</u>	<u>\$( '#vi-du' )</u>	<u>Chỉ chọn thành phần nào có ID là #vi-du</u>
Class	<u>.featured</u>	<u>\$( '.featured' )</u>	<u>Chọn tất cả các thành phần có class là .featured</u>

Như đã nói ở chương 1, khi chúng ta thêm các phương pháp vào hàm \$(), thì các phần tử nằm trong đối tượng jQuery sẽ được tự động loop và diễn ra ở “hậu trường”. Cho nên chúng ta không cần phải sử dụng bất cứ một vòng lặp nào cả, như vòng lặp for chẳng hạn, điều này thường phải làm trong khi viết code về DOM. Sau khi bạn đã nắm bắt được những khái niệm cơ bản, bây giờ chúng ta sẽ khám phá thêm những tính năng mạnh mẽ khác của jQuery.

### CSS Selector

Thư viện jQuery hỗ trợ gần như toàn bộ các CSS Selector chuẩn từ CSS1 cho đến CSS3. Chính việc này đã cho phép những người làm web không phải lo lắng về liệu trình duyệt đó có hỗ trợ những Selector mới hay không (đặc biệt là trình duyệt IE) miễn là trình duyệt đó có bật JavaScript.



**Lưu ý:** những người làm web có kinh nghiệm và trách nhiệm luôn nên áp dụng nguyên tắc nâng cao lũy tiến và giảng cấp hài hoà cho code của họ. Họ phải luôn chắc chắn rằng trang web luôn được hiển thị chính xác, cho dù không được đẹp như khi JavaScript bị tắt hoặc khi nó được bật. Chúng ta sẽ bàn thêm về nguyên tắc này trong suốt chiều dài của loạt bài này.

Để tìm hiểu jQuery sử dụng CSS Selector như thế nào thì cách tốt nhất là làm bằng ví dụ. Trong ví dụ dưới đây chúng ta sẽ sử dụng một dạng danh sách thường được dùng để làm thanh di chuyển trên web. Code HTML sẽ như sau.

```
<ul id="nav">
  <li><a href="#">Home Page</a></li>
  <li><a href="#">About Me</a></li>
  <li><a href="#">Forum</a></li>
  <li><a href="ebook.pdf">Ebooks</a>
    <ul>
      <li><a href="#">jQuery</a></li>
      <li><a href="#">CSS</a></li>
      <li><a href="#">HTML</a></li>
    </ul>
  </li>
  <li><a href="http://www.jquery.com">Tutorials</a></li>
  <li><a href="#">Photoshop</a>
    <ul>
      <li><a href="#">Action</a></li>
      <li><a href="#">Effect</a></li>
      <li><a href="#">Plugins</a></li>
    </ul>
  </li>
  <li><a href="mailto:email@yahoo.com">Email</a></li>
</ul>
```

Trong đoạn code HTML ở trên chúng ta đơn giản chỉ có một unordered list với id="nav" đóng vai trò là menu chính. Khi chưa có style gì áp dụng vào nó thì kết quả khi xem ở trình duyệt sẽ như hình dưới đây. Đây chính là định dạng mặc định của trình duyệt cho các Unorder List.

- [Home Page](#)
- [About Me](#)
- [Forum](#)
- [Ebooks](#)
  - [jQuery](#)
  - [CSS](#)
  - [HTML](#)
- [Tutorials](#)
- [Photoshop](#)
  - [Action](#)
  - [Effect](#)
  - [Plugins](#)
- [Email](#)



## Định dạng list-item

Tất nhiên trong ví dụ này bạn hoàn toàn có thể sử dụng CSS để định dạng menu này, nhưng vì chúng ta muốn khám phá jQuery nên chúng ta tạm thời coi như CSS không tồn tại. Giả sử trong ví dụ này bạn muốn những list-item chính có gạch chân mà những ul phụ của nó sẽ không có gạch chân.

```
.highlight {  
    border-bottom: 1px solid #e6db55;  
    padding: 5px;  
}
```

Thay vì chúng ta sẽ thêm class trực tiếp vào tài liệu HTML, chúng ta sẽ sử dụng jQuery để thêm class vào những list-item tầng 1 như: Homepage, About Me, Forum, Ebooks, Tutorials, Photoshop và Email.

```
$(document).ready(function() {  
    $('#nav > li').addClass('highlight');  
});
```

Như đã bàn ở chương 1, chúng ta bắt đầu đoạn code jQuery với `$(document).ready()`, nó sẽ chạy ngay khi DOM đã được load. Dòng thứ 2 sử dụng CSS Child selector (`>`) để thêm `class='highlight'` chỉ cho list item tầng 1. Nói theo ngôn ngữ của chúng ta thì đoạn code trên có nghĩa như sau: jQuery hãy tìm mỗi một list item (`li`) là con trực tiếp (`>`) của thành phần có ID là `nav` (`#nav`). Với `class='highlight'` được thêm vào, menu của chúng ta được như sau.

[Home Page](#)

---

[About Me](#)

---

[Forum](#)

---

[Ebooks](#)

- [jQuery](#)
  - [CSS](#)
  - [HTML](#)
- 

[Tutorials](#)

---

[Photoshop](#)

- [Action](#)
  - [Effect](#)
  - [Plugins](#)
- 

[Email](#)

---

Để định dạng cho những list item tầng 2 có rất nhiều cách. Nhưng một trong những cách chúng ta sẽ sử dụng trong phần này là pseudo-class phủ định. Bằng cách này chúng ta sẽ đi xác định tất cả những item nào mà không có `class='highlight'`. Chúng ta sẽ viết code như sau:

```
$(document).ready(function() {
```

```

$('#nav > li').addClass('highlight');
$('#nav li:not(.highlight)').addClass('background');
});

```

Đoạn code trên có nghĩa như sau:

1.Chọn tất cả những danh sách là con trực tiếp của #nav

2.Những danh sách này phải không có class='highlight' (:not(.highlight))

Và chúng ta sẽ được hình như hình dưới, tất nhiên bạn phải khai báo class='background' trong file CSS của mình.



## Attribute Selectors

Attribute Selectors là bộ Selector phụ của CSS cũng rất hữu dụng. Nó cho phép chúng ta chọn một thành phần nào đó dựa vào đặc tính HTML của nó như: thuộc tính Title của link hoặc thuộc tính Alt của image. Ví dụ để chọn tất cả các tấm hình có thuộc tính Alt chúng ta làm như sau:

```

$('img[alt]')

```

## Định dạng cho đường liên kết

Nếu bạn biết sơ qua về Regular Expressions trong ngôn ngữ lập trình như PHP thì Attribute Selector trong jQuery chịu ảnh hưởng bởi phương pháp này. Ví dụ dấu (^) dùng để xác định giá trị tại điểm bắt đầu hoặc (\$) kết thúc của một chuỗi. Nó cũng có thể sử dụng dấu (\*) để chỉ một giá trị tại một vị trí bất kỳ trong một chuỗi hoặc sử dụng dấu chấm than (!) để biểu thị một giá trị phủ định. Trong phần CSS này chúng ta sẽ định dạng các đường liên kết như sau:

```

a {

```

```

color: #00c;
}

.email {
    padding-right: 20px;
    background: url(images/mail.png) no-repeat right center;
}

.ebook {
    padding-right: 20px;
    background: url(images/pdf.png) no-repeat right center;
}

.hyperlink {
    padding-right: 20px;
    background: url(images/external.png) no-repeat right center;
}

```

Sau đó chúng ta thêm 3 class là email, ebook và hyperlink vào những đường liên kết thích hợp bằng cách sử dụng jQuery. Để thêm một class vào tất cả những đường liên kết email, chúng ta sẽ tạo một selector và nó sẽ tìm tất cả những thành phần anchor (a) với thuộc tính href bắt đầu bằng chuỗi mailto như sau:

```

$(document).ready(function() {
    $('a[href^=mailto:]').addClass('email');
});

```

Để thêm một class vào tất cả các đường liên kết đến những tệp tin .pdf, chúng ta sử dụng dấu \$ thay vì dấu ^ như ở trên. Bởi vì lần này chúng ta chỉ chọn những đường liên kết nào có thuộc tính href kết thúc bằng cụm .pdf.

```

$(document).ready(function() {
    $('a[href^=mailto:]').addClass('mailto');
    $('a[href$=.pdf]').addClass('ebook');
});

```




Attribute Selector cũng có thể được kết hợp với nhau. Ví dụ chúng ta cũng có thể thêm một class là hyperlink cho tất cả các đường liên kết với giá trị href bắt đầu bằng http và chứa cụm từ hyper trong nó.

```

$(document).ready(function() {
    $('a[href^=mailto:]').addClass('mailto');
    $('a[href$=.pdf]').addClass('pdflink');
    $('a[href^=http][href*=jquery]').addClass('hyperlink');
});

```

Với 3 class được áp dụng như trên cho các đường liên kết, chúng ta sẽ được kết quả như hình dưới đây. Bạn sẽ thấy cạnh mỗi đường link sẽ có thêm một hình icon chỉ cho người dùng biết một cách rất trực quan là đường liên kết đó là về cái gì.

Home Page
About Me
Forum
Ebooks 
◦ jQuery
◦ CSS
◦ HTML
Tutorials 
Photoshop
◦ Action
◦ Effect
◦ Plugins
Email 

## Selector riêng của jQuery

Dường như còn chưa vừa ý với những CSS Selector, jQuery có một hệ thống những Selector của riêng nó. Hầu hết những Selector này đều cho phép chúng ta chọn bất cứ thành phần nào trong tài liệu HTML. Cú pháp cho những Selector này tương đồng với cú pháp của CSS pseudo-class, nơi mà các selector bắt đầu bằng dấu hai chấm (:). Ví dụ, để chọn thẻ div thứ 2 của tập hợp những thẻ div có class='horizontal', chúng ta có cách viết code như sau:

```
$('div.horizontal:eq(1)');
```

Bạn nên lưu ý ở phần này là :eq(1) chọn thẻ div thứ hai từ tập hợp trả về bởi vì JavaScript đánh số array bắt đầu từ số 0. Ngược lại, CSS lại là bắt đầu từ số 1. Cho nên khi bạn sử dụng nth-child CSS Selector như là \$('div:nth-child(1)') sẽ chọn tất cả các thẻ div là con đầu tiên của thành phần cha mẹ. Tất nhiên đây là diễn giả là vậy, nhưng nếu trong thực tế thì bạn nên dùng \$('div:first-child') thì hợp lý hơn.

## Định dạng bảng kiểu kẻ sọc

Hai trong số những jQuery selector rất hữu dụng là :odd và :even. Trong ví dụ dưới đây chúng ta sẽ sử dụng một trong hai selector này để định dạng cho bảng kiểu kẻ sọc với code HTML như sau:

```
<table>
    <tr>
        <td>Movies</td>
        <td>Actors/ Actresses</td>
        <td>Year Make</td>
    </tr>
</table>
```

```

        <td>Terminator</td>
        <td>arnold schwarzenegger</td>
        <td>1991</td>
    </tr>
    <tr>
        <td>Die Hard</td>
        <td>Bruce Willis</td>
        <td>2000</td>
    </tr>
    <tr>
        <td>Speed</td>
        <td>Sandra Bullock</td>
        <td>1997</td>
    </tr>
    <tr>
        <td>Independence Day</td>
        <td>Will Smith</td>
        <td>1999</td>
    </tr>
    <tr>
        <td>Armageddon</td>
        <td>Bruce Willis</td>
        <td>1997</td>
    </tr>
    <tr>
        <td>Under Siege</td>
        <td>Steven Seagal</td>
        <td>1996</td>
    </tr>
    <tr>
        <td>Avatar</td>
        <td>Unknown</td>
        <td>2010</td>
    </tr>
</table>

```

Bây giờ bạn có thể thêm style vào stylesheet cho tất cả các dòng của bảng và sử dụng một `class='alt'` cho những dòng chẵn.

```

.alt {
    background: #dda;
}

td {
    padding: 10px;
}

```

Cuối cùng chúng ta sẽ viết code jQuery để gắn class vào cho những dòng chẵn của bảng ( ).

```

$(document).ready(function() {
    $('tr:odd').addClass('alt');
});

```

Bạn có thấy code ở trên có điều gì lạ không? Odd tiếng Việt là lẻ và Even là chẵn. Chúng ta nói sẽ tô màu cho dòng chẵn nhưng lại sử dụng `:odd`? Thực ra vấn đề ở đây cũng tương tự như `:eq()` ở trên, bởi vì `:odd` và `:even` sử dụng dạng đánh số từ số 0 như trong JavaScript. Cho

nên dòng thứ nhất đếm là số 0 (số chẵn) và dòng thứ hai đếm là 1 (số lẻ). Do đó với dòng code jQuery như trên dưới đây là kết quả chúng ta có được.

Movies	Actors/ Actresses	Year Make
Terminator	arnold schwarzenegger	1991
Die Hard	Bruce Willis	2000
Speed	Sandra Bullock	1997
Independence Day	Will Smith	1999
Armageddon	Bruce Willis	1997
Under Siege	Steven Seagal	1996
Avatar	Unknown	2010

Vấn đề có vẻ như đã được giải quyết ở đây, nhưng nếu bạn có một bảng thứ hai trên cùng một trang thì kết quả lại không như bạn muốn. Ví dụ, dòng cuối cùng của bảng trên có màu cỏ úa thì dòng đầu tiên của bảng kế tiếp sẽ có màu trắng. Có cách để tránh tình trạng này là sử dụng `:nth-child()` Selector. Selector này có thể lấy tham số là odd, even hoặc chữ số. Nhưng cũng lưu ý bạn là `:nth-child()` là selector duy nhất của jQuery đánh số theo thứ tự từ 1. Cho nên để đạt được kết quả như mong muốn và nhất quán với nhiều bảng trên trang, chúng ta có đoạn code mới như sau:

```
$(document).ready(function() {  
  $('tr:nth-child(even)').addClass('alt');  
});
```

Bây giờ giả sử chúng ta muốn tô đậm đỏ cho cột nào đó có chứa tên chú Bruce Willis thì trước hết bạn phải thêm một `class='red'` vào phần stylesheet và sau đó thì viết code jQuery như sau sử dụng `:contains()` Selector.

```
$(document).ready(function() {  
  $('tr:nth-child(even)').addClass('alt');  
  $('td:contains(Bruce Willis)').addClass('red');  
});
```

Bây giờ thì bảng của chúng ta đã tô đậm và in đỏ chú Bruce Willis.

Movies	Actors/ Actresses	Year Make
Terminator	arnold schwarzenegger	1991
Die Hard	<b>Bruce Willis</b>	2000
Speed	Sandra Bullock	1997
Independence Day	Will Smith	1999
Armageddon	<b>Bruce Willis</b>	1997
Under Siege	Steven Seagal	1996
Avatar	Unknown	2010

Tôi cũng phải lưu ý với bạn là :contains() Selector có phân biệt giữa IN HOA và in thường. Cho nên nếu bạn chỉ gõ \$('td:contains(bruce willis)') mà không viết hoa thì sẽ không có cột nào được chọn cả.

Phải thừa nhận rằng với ví dụ đơn giản như trên, bạn không cần phải sử dụng jQuery cũng đạt được kết quả như mong muốn. Tuy nhiên, jQuery kết vọi với CSS, là một lựa chọn phù hợp cho kiểu định dạng khi mà nội dung được tạo ra tự động từ CSDL và chúng ta không có khả năng chi phối code HTML cũng như code được xuất ra từ PHP chẳng hạn.

### Form Selector

Khi làm việc với form, những selector của jQuery giúp bạn tiết kiệm thời gian để chọn chỉ những thành phần nào mình muốn. Bảng biểu sau là những selector của jQuery để làm việc với form.

<u>Seleotor</u>	<u>Thành phần được chọn</u>
:text, :checkbox, :radio, :image, :submit, :reset, :password, :file	<u>Phần tử Input với thuộc tính có cùng tên với Selector. Ví dụ :text sẽ chọn &lt;input type='text' /&gt;</u>
:input	<u>Input, textarea, select và các button</u>
:button	<u>Các phần tử nút và input với thuộc tính type bằng các nút</u>
:enable	<u>Phần tử form được bật (enable)</u>
:disable	<u>Phần tử form được tắt (disable)</u>
:checked	<u>Nút radio hoặc hộp kiểm được đánh dấu</u>

Cũng giống như những Selector khác, form Selector cũng có thể được kết hợp để cho đối tượng chọn được cụ thể hơn. Ví dụ chúng ta có thể chọn tất cả các nút radio được đánh dấu (chứ không phải hộp kiểm) với \$(':radio:checked') hoặc chọn tất cả các trường nhập mật



khẩu và trường nhập dữ liệu bị tắt với `$(':password, :text:disabled')`. Cho dù với jQuery Selector, nhưng chúng ta vẫn sử dụng nguyên tắc của CSS để chọn các phần tử cần chọn.

## Phương pháp di chuyển trong DOM

Những jQuery selector vừa được giới thiệu ở trên cho phép chúng ta chọn một tập hợp các phần tử khi chúng ta di chuyển ngang qua hoặc dọc xuống cây DOM và chọn lọc kết quả. Nếu đây là cách duy nhất để chọn các phần tử thì những lựa chọn của chúng ta cũng bị hạn chế khá nhiều (mặc dù trong thực tế những selector đã rất mạnh mẽ đặc biệt là khi mang ra so sánh với cách di chuyển trong DOM truyền thống). Có nhiều trường hợp khi bạn cần phải chọn cha mẹ hoặc ông bà của các phần tử trở nên quan trọng, chính vì vậy phương pháp di chuyển trong DOM được giới thiệu. Với những phương pháp này chúng ta có thể đi lên, đi xuống, ngang dọc hoặc xung quanh cây DOM rất dễ dàng.

Một vài phương pháp có chức năng gần như tương đồng với những người ‘anh em’ Selector ở trên. Như trong ví dụ về định dạng bảng kiểu kẻ sọc ở trên chúng ta thêm `class='alt'` với `$('tr:odd').addClass('alt')`; cũng có thể được viết lại với phương pháp `.filter()` như sau:

```
$('tr').filter(':odd').addClass('alt');
```

Trong đa số các trường hợp thì hai cách trên hỗ trợ cho nhau. Hơn nữa, đặc biệt là phương pháp `.filter()` cực kỳ mạnh mẽ ở chỗ nó có thể lấy một hàm làm tham số của nó. Hàm đó cho phép chúng ta tạo ra những phép kiểm tra phức tạp để xác định xem một thành phần nào đó có nên được giữ lại trong tập hợp kết quả trả về. Nói ví dụ chúng ta muốn thêm một class cho tất cả những đường liên kết ngoài. jQuery không có selector nào có thể tiến hành tác vụ này. Nếu không có hàm trong phương pháp `.filter()`, chúng ta bắt buộc phải sử dụng vòng lặp để nhảy qua từng thành phần và kiểm tra nó riêng rẽ. Tuy nhiên với những hàm trong phương pháp `.filter()` sau, chúng ta vẫn có thể dựa vào vòng lặp ẩn của jQuery và giữ cho code của chúng ta gọn gàng.

```
$('a').filter(function() {  
    return this.hostname && this.hostname != location.hostname;  
}).addClass('external');
```

Dòng code thứ 2 lọc tập hợp các phần tử `<a>` với hai tiêu chí sau:

1. Nó phải có thuộc tính href với tên miền (`this.hostname`). Chúng ta sử dụng phép kiểm tra này để loại bỏ những liên kết dạng `mailto` và những thứ tương tự.
2. Tên miền mà nó liên kết tới (`this.hostname`) không được giống (`!=`) với tên miền của trang hiện tại (`location.hostname`).

Nói chính xác hơn thì phương pháp `.filter()` lặp qua tập hợp những phần tử phù hợp, kiểm tra từng giá trị trả về bằng hàm đã tạo. Nếu hàm trả về là `false`, thì phần tử đó sẽ bị loại khỏi tập hợp. Còn nếu giá trị trả về là `true`, thì phần tử đó được giữ lại. Bây giờ chúng ta sẽ xem lại bảng kiểu kẻ sọc và xem xem có thể làm gì với phương pháp di chuyển này.

## Định dạng từng ô cụ thể

Ở ví dụ trên chúng ta đã thêm `class='red'` cho những ô có chứa chữ Bruce Willis. Nếu bây giờ chúng ta cũng muốn định dạng cho ô bên cạnh ô chứa Bruce Willis, chúng ta có thể bắt đầu với Selector mà chúng ta đã tạo, và sau đó chỉ đơn giản nối nó với phương pháp `.next()`.

```
$(document).ready(function() {
    $('td:contains(Bruce Wiliss)').next().addClass('red');
});
```

Bảng của bạn sẽ được như sau

Movies	Actors/ Actresses	Year Make
Terminator	arnold schwarzenegger	1991
Die Hard	Bruce Willis	2000
Speed	Sandra Bullock	1997
Independence Day	Will Smith	1999
Armageddon	Bruce Willis	1997
Under Siege	Steven Seagal	1996
Avatar	Unknown	2010

Phương pháp `.next()` chỉ lựa chọn các phần tử ngay sát cạnh nó. Để tô đỏ đậm cho tất cả các ô đằng sau ô có chứa Bruce Willis, chúng ta có thể sử dụng phương pháp `.nextAll()`.

```
$(document).ready(function() {
    $('td:contains(Bruce Wiliss)').nextAll().addClass('red');
});
```

Bên cạnh phương pháp `.next()` và `.nextAll()` chúng ta còn có `.prev()` và `.prevAll()`. Thêm nữa, `.siblings()` chọn tất cả các phần tử có cùng chung một cấp bậc trên DOM, mà không cần quan tâm đến nó xuất hiện trước hoặc sau phần tử được chọn.

Để bao gồm cả ô ban đầu (là ô có chứa Bruce Willis) và những ô theo sau nó, chúng ta có thể thêm phương pháp `.andSelf()`:

```
$(document).ready(function() {
    $('td:contains(Bruce Wiliss)').nextAll().andSelf().addClass('red');
});
```

Movies	Actors/ Actresses	Year Make
Terminator	arnold schwarzenegger	1991
Die Hard	<b>Bruce Willis</b>	<b>2000</b>
Speed	Sandra Bullock	1997
Independence Day	Will Smith	1999
Armageddon	<b>Bruce Willis</b>	<b>1997</b>
Under Siege	Steven Seagal	1996
Avatar	Unknown	2010

Bạn cũng nên biết rằng có vô số những kết hợp của selector và phương pháp di chuyển mà dựa vào đó chúng ta có thể chọn cùng một tập hợp các phần tử. Ví dụ này sẽ cho bạn thấy một cách khác để chọn mỗi một ô trong một dòng mà ô đó có chứa chữ Bruce Willis:

```
$(document).ready(function() {
    $('td:contains(Bruce Willis)').parent().children().addClass('red');
});
```

Ở đây thay vì chúng ta di chuyển theo kiểu ngang hàng, chúng ta di chuyển lên trên một bậc của cây DOM ( ) với phương pháp .parent() và sau đó chọn tất cả các ô của dòng bằng phương pháp .children().

Movies	Actors/ Actresses	Year Make
Terminator	arnold schwarzenegger	1991
<b>Die Hard</b>	<b>Bruce Willis</b>	<b>2000</b>
Speed	Sandra Bullock	1997
Independence Day	Will Smith	1999
<b>Armageddon</b>	<b>Bruce Willis</b>	<b>1997</b>
Under Siege	Steven Seagal	1996
Avatar	Unknown	2010

### Kết hợp (chaining)

Phương pháp di chuyển kết hợp như chúng ta vừa khám phá ở trên thể hiện khả năng kết hợp của jQuery. Với jQuery bạn có thể chọn tập hợp các phần tử và thao tác nhiều tác vụ lên

chúng, tất cả trên cùng một dòng code. Kiểu kết hợp này không những giữ cho code jQuery được súc tích mà còn tăng khả năng hoạt động của mã. Nhưng để cho dễ đọc hơn, bạn cũng có thể tách ra thành nhiều hàng. Ví dụ một dãy kết hợp các phương pháp có thể được viết trên một dòng như sau:

```
$('#td:contains(Bruce Willis)').parent().find('td:eq(1)').addClass('red').end().find('td:eq(2)').addClass('red');
```

... hoặc cắt nhỏ ra từng dòng

```
$('#td:contains(Bruce Willis)') // Tìm tất cả các dòng có chứa Bruce Willis
.parent() // Di chuyển lên một tầng
.find('td:eq(1)') // Tìm td với thứ tự là 1 (dòng thứ 2)
.addClass('red') // Thêm class='red'
.end() // quay về với bố mẹ của ô chứa Henry
.find('td:eq(2)') // Tìm tiếp td với thứ tự là 2 (dòng 3)
.addClass('red') // thêm class='red'
```

Tất nhiên cách di chuyển kiểu như trên là lòng vòng đến mức thừa thãi và không có trong thực tế. Bởi vì có nhiều cách khác đơn giản hơn, trực tiếp hơn. Tuy nhiên nó cũng cho bạn thấy được sự linh hoạt tuyệt vời mà kiểu kết hợp cho phép chúng ta.

Viết code kiểu kết hợp thế này như là nói một tràng trong một hơi không nghỉ. Nó giúp bạn đạt mục tiêu nhanh chóng, nhưng lại khó cho người khác hiểu được. Cho nên tách nó ra và thêm comments có thể giúp bạn tiết kiệm thời gian và công sức sau này nếu phải chỉnh sửa code.

## Hướng tới các thành phần DOM

Mỗi một Selector và hầu hết các phương pháp của jQuery đều trả về một đối tượng jQuery. Đây chính là điều chúng ta luôn mong đợi, bởi vì khả năng tiến hành vòng lặp ẩn và kết hợp nó có thể làm. Nhưng cũng có lúc chúng ta muốn hướng tới một phần tử DOM một cách trực tiếp. Ví dụ, chúng ta muốn sử dụng một tập hợp các phần tử cho một thư viện JavaScript khác. Hoặc chúng ta muốn hướng tới tên thẻ của một phần tử, mà nó lại có sẵn như là một thuộc tính của phần tử DOM. Tuy trường hợp này hiếm khi xảy ra, jQuery có phương pháp `.get()`. Để hướng tới thành phần DOM đầu tiên chỉ đến bởi một đối tượng jQuery, chúng ta sẽ sử dụng `.get(0)`. Nếu phần tử DOM cần phải nằm trong một vòng lặp, chúng ta sẽ sử dụng `.get(index)`. Như vậy, nếu chúng ta muốn biết tên thẻ của một thành phần với id='my-element', chúng ta sẽ viết code như sau:

```
var myTag = $('#my-element').get(0).tagName;
```

Để tiện dụng hơn nữa, jQuery cung cấp cách viết tắt cho phương pháp `.get()`. Thay vì viết như dòng code ở trên, chúng ta có thể sử dụng cặp ngoặc vuông `[]` ngay đằng sau selector:

```
var myTag = $('#my-element')[0].tagName;
```

Không phải là ngẫu nhiên mà cú pháp này nhìn giống như là một array của các phần tử DOM, sử dụng cặp ngoặc vuông như là xé đi lớp vỏ để tới danh sách các nốt, có bao gồm luôn cả index (trong trường hợp này là 0) cũng giống như lôi từng thành phần DOM ra vậy.

Với những kỹ năng mà chúng ta đã học trong chương này, chúng ta đã có thể định dạng cho từng một và từng phụ của một danh sách sử dụng những CSS Selector cơ bản, áp dụng những style khác nhau cho các loại đường liên kết khác nhau sử dụng Attribute Selector, tô màu khác nhau cho bảng kẻ sọc bằng cách sử dụng jQuery selector như :odd và :even hoặc Selector mới của CSS là :nth-child(), và cuối cùng là tô đậm đỏ cho từng ô trong bảng bằng cách kết hợp các phương pháp jQuery. Cho đến bây giờ chúng ta sử dụng sự kiện `$(document).ready()` để thêm class vào tập hợp các phần tử. Trong chương tới, chúng ta sẽ khám phá những cách để thêm class vào những sự kiện người dùng tự tạo.

## Chương 3 - Attributes

### A. Class

**addClass( class )** Kiểu trả về: jQuery

Thêm các class đã xác định vào mỗi tập phần tử phù hợp. Nếu có thêm nhiều class thì các class được

các nhau bởi khoảng trắng.

Ví dụ: Thêm class “Maudu” vào các thẻ p.

```
$("#p").addClass("Maudu");
```

**removeClass( class )** Kiểu trả về: jQuery

Loại bỏ tất cả hoặc các class đã xác định khỏi tập phần tử phù hợp.

Ví dụ: Loại bỏ class “Maudu” khỏi các thẻ p.

```
$("#p").removeClass("Maudu");
```

**toggleClass( class )** Kiểu trả về: jQuery

Thêm class nếu class chưa tồn tại hoặc loại bỏ nếu class đã tồn tại.

Ví dụ: Thêm class “Maudu” vào thẻ p nếu class “Maudu” chưa tồn tại trong thẻ p hoặc loại bỏ

class “Maudu” khỏi thẻ p nếu nó tồn tại.

```
$("#p").toggleClass("Maudu");
```

Ví dụ: Vi\_du\_9\_6.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Vi_du_9_6.aspx.cs"
Inherits="Vi_du_9_6" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title>Vi_du_9_6</title>

    <script src="jquery-1.3.2-vsdoc.js" type="text/javascript"></script>

    <script src="jquery-1.3.2.js" type="text/javascript"></script>

    <script type="text/javascript" language="javascript">

        $(document).ready(function() {

            $("p").addClass("under");

            $("p:last").removeClass("highlight");

            $("p").click(function() {

                $("p").removeClass("highlight");

                $(this).toggleClass("highlight");

            });

        });

    </script>

    <style type="text/css">

        p { margin: 4px; font-size:16px; font-weight:bolder; }

        .blue { color:blue; }

        .under { text-decoration:underline; }

        .highlight { background:yellow; }

    </style>
```

```
</head>

<body>

  <p class="blue">Visual Studio 2000</p>

  <p class="blue">ASP.NET 3.5</p>

  <p class="blue highlight">Chào mừng bạn đến với jQuery</p>

</body>

</html>
```

## B.HTML, Text

**html()** Kiểu trả về: String

Lấy nội dung html (innerHTML) của phần tử.

Ví dụ: Mỗi khi click vào thẻ p lấy nội dung html của thẻ p đó và thông báo nội dung lấy được.

```
$("p").click(function() {alert($(this).html());});
```

**html( val )** Kiểu trả về: jQuery

Thiết lập nội dung html (innerHTML) cho phần tử.

Ví dụ: Thiết lập nội dung html cho thẻ div.

```
$("div").html("<b>Chào các bạn!<i> Chúc buổi học hôm nay thú vị.</i></b>");
```

**text()** Kiểu trả về: String

Lấy nội dung text (innerText) của phần tử.

Ví dụ: Mỗi khi click vào thẻ p lấy nội dung text của thẻ p đó và thông báo nội dung lấy được.

```
$("p").click(function() {alert($(this).text());});
```

**text( val )** Kiểu trả về: jQuery

Thiết lập nội dung text (innerText) cho phần tử.



Ví dụ: Thiết lập nội dung text cho thẻ div.

```
$("#div").text("Chào các bạn! Chúc buổi học hôm nay thú vị");
```

## Chương 4 – Sự kiện (Events)

JavaScript có một số cách được lập sẵn để phản ứng với những tương tác của người dùng và những sự kiện khác. Để làm cho trang web năng động và tương tác tốt, chúng ta cần phải tận dụng chức năng này, để vào những thời điểm phù hợp, chúng ta có thể sử dụng những kỹ thuật jQuery đã học và sắp học. Bạn cũng có thể làm những việc sau với anh bạn thân JavaScript, nhưng jQuery nâng cao và mở rộng những cơ chế quản lý sự kiện cơ bản để giúp nó có cú pháp đẹp hơn, tiết kiệm thời gian hơn và tất nhiên cũng mạnh mẽ hơn.

Thực hiện tác vụ khi trang được load

Chúng ta đã biết cách làm cho jQuery phản ứng như thế nào khi trang web được load. Bộ quản lý sự kiện `$(document).ready()` có thể được dùng để kích hoạt một hàm nào đó, nhưng chúng ta có thể bàn thêm một chút về nó.

Định thời gian thực thi code

Trong chương 1, chúng ta đã biết rằng `$(document).ready()` là cách của jQuery thực hiện các tác vụ tương đương với cách mà JavaScript thực hiện tác vụ với onload event được lập sẵn. Thực tế thì hai cách này đều có tác dụng giống nhau, nhưng chúng lại kích hoạt tác vụ ở những thời điểm hơi khác nhau.

Sự kiện `window.onload` được kích hoạt khi mà trình duyệt đã hoàn toàn load xong tài liệu. Điều này có nghĩa rằng mọi phần tử trên trang đã sẵn sàng để được thao tác bởi JavaScript. Đây chính là một điểm thuận lợi để chúng ta viết code mà không phải lo lắng về trật tự load.

Mặt khác, bộ quản lý đăng ký sử dụng `$(document).ready()` được kích hoạt khi DOM hoàn toàn sẵn sàng để sử dụng. Điều này cũng có nghĩa rằng mọi thành phần có thể được truy cập bởi code của chúng ta, nhưng không nhất thiết là tài liệu liên quan đã được download. Ngay sau khi HTML được download và chuyển qua cây DOM, code có thể được thực thi.

**Lưu ý:** Để đảm bảo rằng trang web vẫn có định dạng trước khi code JavaScript được thực hiện, người ta thường đặt `<link rel="stylesheet">` đứng trước thẻ `<script>` trong phần `<head>` của tài liệu.

Ví dụ chúng ta có một trang thư viện hình ảnh, trang đó bao gồm nhiều hình có dung lượng lớn mà chúng ta có thể ẩn, hiện, di chuyển hoặc thao tác với jQuery. Nếu bây giờ chúng ta thiết lập giao diện sử dụng sự kiện onload, thì người dùng sẽ phải đợi cho đến khi mọi tấm hình đã được download trước khi họ có thể sử dụng trang web. Hoặc tệ hơn, nếu những cách xử lý chưa được gán cho các phần tử có cách xử lý mặc định riêng như là các đường liên kết, thì việc tương tác với người dùng sẽ tạo ra những điều không mong đợi. Tuy nhiên khi chúng ta sử dụng `$(document).ready()`, thì giao diện sẽ sẵn sàng để sử dụng sớm hơn rất nhiều với những cách xử lý mong muốn.

**Lưu ý:** Cách sử dụng `$(document).ready()` luôn được ưa chuộng hơn là sử dụng bộ quản lý onload, nhưng chúng ta cũng nên nhớ rằng bởi vì những tệp tin hỗ trợ có thể chưa được load, cho nên những thuộc tính như độ cao và chiều rộng của tấm hình có thể chưa có sẵn trong lúc này. Nếu thực sự cần thiết, chúng ta có thể sử dụng bộ quản lý onload (hoặc hay hơn có thể sử dụng jQuery để thiết lập bộ quản lý cho load event). Hai cách này hoàn toàn tương thích với nhau.

Nhiều đoạn mã trên cùng một trang

Cách thường dùng để đăng ký bộ quản lý sự kiện thông qua JavaScript là gán một hàm cho thuộc tính tương ứng của phần tử DOM. Giả sử như chúng ta đã định nghĩa một hàm:

```
function doStuff() {  
  //làm một cái gì đó  
}
```

Sau đó chúng ta có thể gán nó trong phần code HTML như sau:

```
<body onload="doStuff();>
```

Hoặc chúng ta cũng có thể gán nó trong code JavaScript:

```
window.onload = doStuff;
```

Hai cách này đều thực thi hàm khi trang được load. Nhưng điểm mạnh của cách thứ hai nằm ở chỗ những cách xử lý được tách rời khỏi mã HTML.

**Lưu ý:** Bạn nên chú ý là khi chúng ta gán một hàm làm bộ quản lý, chúng ta sử dụng tên hàm nhưng bỏ hai dấu ngoặc đơn. Nếu có hai dấu ngoặc, hàm đó sẽ được gọi ngay lập tức. Còn nếu không có dấu ngoặc, tên hàm chỉ đơn giản được định danh, và có thể được dùng để gọi sau này.

Nếu chỉ với một hàm thì cách này cũng sử dụng được. Nhưng nếu chúng ta có thêm một hàm nữa:

```
function doOtherStuff() {  
  //làm một tác vụ khác  
}
```

Sau đó chúng ta cũng thử chạy hàm này khi trang được load `window.load = doOtherStuff;`

Bạn sẽ thấy hàm thứ hai sẽ thắng hàm đầu tiên. Thuộc tính `.onload` chỉ có thể một lúc chứa một hàm tham chiếu, cho nên chúng ta không thể thêm vào cách xử lý hiện tại. Cơ chế `$(document).ready()` giải quyết trường hợp này rất êm xuôi. Mỗi một lần phương thức được gọi, nó sẽ thêm một hàm mới vào danh sách cách xử lý nội bộ, nên khi trang được load, tất cả các hàm sẽ được thực hiện. Các hàm sẽ thực hiện theo thứ tự mà chúng được đăng ký.

Cách viết tắt cho code ngắn gọn

Kết cấu `$(document).ready()` thực chất là gọi phương thức `.ready()` cho một đối tượng jQuery mà chúng ta đã tạo ra từ phần tử DOM. Hàm `$()` cung cấp cách viết tắt cho chúng ta bởi vì nó là một tác vụ phổ biến. Khi được gọi mà không có tham số, thì hàm này sẽ hoạt động như là khi tài liệu đã được thông qua. Cho nên thay vì chúng ta viết:

```
$(document).ready(function() {  
  //code ở đây  
});
```

Chúng ta có thể viết

```
$.ready(function() {  
  //code ở đây  
});
```

Hơn nữa, hàm `$()` có thể lấy một hàm khác làm tham số cho nó. Cho nên khi chúng ta làm như thế, jQuery sẽ tiến hành một lệnh gọi ẩn đến `.ready()`, do vậy cách viết như sau cũng cho kết quả tương tự

```
$(function()  
  //code ở đây{  
});
```

Tất nhiên cách viết trên ngắn gọn hơn, nhưng tôi khuyên bạn nên sử dụng kiểu viết đầy đủ để cho rõ ràng là đoạn code này có tác dụng gì.

Cùng làm việc với những thư viện khác

Trong một vài trường hợp chúng ta cần phải sử dụng nhiều hơn một thư viện JavaScript trên cùng một trang. Bởi vì nhiều thư viện cùng sử dụng ký hiệu nhận dạng `$` do nó ngắn và thuận tiện, cho nên chúng ta phải có cách nào đó để tránh xảy ra xung đột giữa những tên này.

Thật may mắn khi mà jQuery cung cấp một phương thức gọi là `.noConflict()` để trả ký hiệu nhận dạng `$` về cho các thư viện khác. Cách sử dụng phương thức `.noConflict()` thường thì như sau:

```
<script src="prototype.js" type="text/javascript"></script>  
<script src="jquery.js" type="text/javascript"></script>  
<script type="text/javascript">  
  jQuery.noConflict();  
</script>  
<script src="myscript.js" type="text/javascript"></script>
```

Đầu tiên thư viện Prototype được gọi, đây cũng là một thư viện JavaScript. Sau đó là bản thân jQuery được gọi và nó sẽ sử dụng `$` cho nó. Tiếp theo phương pháp `.noConflict()` được gọi để giải phóng `$`, quyền điều khiển bây giờ lại quay trở về với thư viện được gọi đầu tiên, ở đây là Prototype. Bây giờ code của chúng ta có thể sử dụng cả hai thư viện, nhưng bất cứ khi nào chúng ta muốn sử dụng một phương thức jQuery, chúng ta cần phải sử dụng jQuery thay vì dấu `$` làm ký hiệu nhận dạng.

Phương thức `.ready()` còn có một điểm nữa có thể giúp chúng ta trong trường hợp này. Hàm gọi ngược mà chúng ta đã chuyển cho nó có thể nhận một tham số đơn: chính là bản thân đối tượng jQuery. Điều này cho phép chúng ta đặt lại tên cho nó mà không sợ bị xung đột.

```
jQuery(document).ready(function($) {  
  //trong đây, chúng ta có thể sử dụng $ bình thường.  
});
```

Hoặc sử dụng kiểu viết tắt chúng ta đã học ở trên

```
jQuery(function($) {  
  //code sử dụng $  
});
```

### Sự kiện cơ bản

Có nhiều lúc chúng ta muốn thực hiện một tác vụ nào đó vào những thời điểm mà không chỉ là lúc trang được load. Cũng như với JavaScript cho phép chúng ta đón chặn sự kiện load trang với `<body onload=`”> hoặc `window.onload`. Nó cung cấp điểm neo cho những sự kiện được người dùng khởi xướng như: nhấp chuột (onclick), trường nhập liệu bị thay đổi (onchange) và cửa sổ thay đổi kích thước (onresize). Khi được gán trực tiếp vào các phần tử trong DOM, những cách này cũng có mặt hạn chế giống như những điều chúng ta đã nói về onload. Cho nên, jQuery cho chúng ta những cách cải tiến hơn để xử lý những sự kiện này.

### Bộ nút thay đổi màu chữ

Để minh họa cho những cách quản lý sự kiện, giả sử chúng ta muốn có một trang web có thể thay đổi màu sắc các đoạn văn tùy theo ý của người dùng. Chúng ta sẽ cho phép người dùng nhấp chuột vào 3 nút để thay đổi màu sắc theo kiểu Mặc định, Màu đỏ và Màu Xanh. Khi nhấn vào nút Màu Đỏ, thì nội dung sẽ chuyển thành màu đỏ, khi nhấn vào nút Màu Xanh thì nội dung sẽ thành màu xanh và cuối cùng khi nhấn vào nút Mặc định thì nội dung quay về trạng thái ban đầu.

Trong thực tế, người làm web có kinh nghiệm luôn áp dụng nguyên tắc [nâng cao lũy tiến](#). Nếu JavaScript không được bật thì nút thay đổi màu sắc phải bị ẩn đi, còn không thì vẫn phải hoạt động bằng các đường liên kết để cho ra những phiên bản khác nhau của trang. Trong tutorial này, chúng ta giả sử người dùng có bật JavaScript. Dưới đây là code HTML của nút thay đổi màu sắc của chúng ta:

```
<div id="switcher">  
  <h3>Đổi định dạng</h3>  
  <div class="button selected" id="switcher-default">Mặc  
Định</div>  
  <div class="button" id="switcher-red">Màu đỏ</div>  
  <div class="button" id="switcher-green">Màu Xanh</div>  
</div>  
<div class="text">  
  <h1>This is the first para</h1>  
  <p class="chapter">Pellentesque habitant morbi tristique  
senectus et netus et malesuada fames ac turpis egestas.  
Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor  
sit amet, ante.  
Donec eu libero sit amet quam egestas semper.  
</p>  
  <h1>This is the second para</h1>  
  <p>Pellentesque habitant morbi tristique senectus et netus et  
malesuada fames ac turpis egestas.  
Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor  
sit amet, ante.  
Donec eu libero sit amet quam egestas semper.  
</p>  
</div>
```

Với một chút CSS trang web mẫu của chúng ta sẽ có dạng như hình dưới đây.



Để bắt đầu, chúng ta thử với nút Màu Xanh. Bạn cần phải viết một chút code CSS để chỉ cho nó biết sẽ thay đổi như thế nào:

```
body.green .text{
    color: green;
}
```

Mục đích của chúng ta sẽ là thêm một class='green' vào thẻ <body>. Điều này cho phép stylesheet tái định dạng trang web sao cho phù hợp. Với kiến thức bạn học được trong chương 2, chúng ta biết sẽ phải viết code như thế nào:

```
$('body').addClass('green');
```

Tuy nhiên, lần này chúng ta muốn khi người dùng nhấp chuột vào nút thì class='green' mới được thêm vào chứ không phải như trước là khi trang được load. Để làm được việc này, chúng ta sẽ cần đến phương thức .bind(). Phương thức này cho phép chúng ta cụ thể hoá bất cứ một sự kiện JavaScript nào và gán một cách xử lý cho nó. Trong trường hợp này sự kiện được gọi là click và cách xử lý là một hàm bao gồm một dòng code như ở trên:

```
$(document).ready(function() {
$('#switcher-green').bind('click', function() {
$('body').addClass('green');
});
});
```

Nếu bạn lưu lại và xem thử trên trình duyệt, khi bạn nhấp chuột vào nút màu xanh, nội dung của nó sẽ biến thành màu xanh lá cây.



Đó là tất cả những gì bạn cần phải làm để gán một sự kiện. Thế mạnh của phương thức `.ready()` cũng được sử dụng ở đây. Phương thức `.bind()` được gọi nhiều lần nhưng vẫn hoạt động tốt và nhiều cách xử lý được thêm vào cùng một sự kiện khi cần thiết. Tất nhiên đây không phải là cách hay nhất hoặc hiệu quả nhất để hoàn hành tác vụ này. Ở phần tiếp theo của tutorial, chúng ta sẽ mở rộng và cải tiến mã của chúng ta để chúng ta có thể tự hào về nó.

Làm các nút khác hoạt động

Bây giờ chúng ta đã có nút màu xanh hoạt động như ý, việc tiếp theo chúng ta phải làm là cho hai nút còn lại là Mặc định và Màu Đỏ cũng có thể hoạt động được. Vấn đề khá là đơn giản, chúng ta sẽ sử dụng phương thức `.bind()` để thêm vào một bộ xử lý click cho mỗi một nút, thêm hoặc bỏ class khi cần. Chúng ta sẽ viết mã như sau:

```
$(document).ready(function() {
    $('#switcher-default').bind('click', function() {
        $('body').removeClass('red').removeClass('green');
    });

    $('#switcher-red').bind('click', function(){
        $('body').addClass('red').removeClass('green');
    });
    $('#switcher-green').bind('click', function() {
        $('body').addClass('green').removeClass('red');
    });
});
```

Trong file stylesheet bạn phải có luật sau

```
body.red .text {  
color: red;  
}
```

Với đoạn mã ở trên chúng ta đã tiến hành những việc như sau:

1. Nếu người dùng nhấn vào thẻ div với ID #switcher-default thì sẽ bỏ cả hai class là red và green đi.
2. Nếu người dùng nhấn vào thẻ div với ID #switcher-red thì chúng ta sẽ thêm class='red' và bỏ class='green' đi.
3. Tương tự như bước 2 nhưng bỏ class='red' và thêm class='green'.

Và bây giờ khi nhấn vào nút Màu Đỏ thì chữ sẽ được tô thành màu đỏ

#### Bộ xử lý sự kiện ngữ cảnh

Nút thay đổi màu sắc của chúng ta đã làm việc như mong muốn, nhưng người dùng không biết được là nút nào đang bị bấm. Cách chúng ta sẽ làm là thêm class='selected' cho nút nào đang được chọn và bỏ class đó đi ở những nút không được nhấp. Chúng ta chỉ đơn giản làm cho nút đang được chọn được tô đậm hơn một chút.

```
.selected {  
font-weight: bold;  
}
```

Chúng ta cũng có thể làm tương tự như cách đã làm ở trên bằng cách gọi mỗi nút bằng ID riêng và thêm và bỏ class nếu cần. Nhưng thay vào đó, chúng ta có thể tìm hiểu thêm một giải pháp khác hay hơn và linh động hơn. Nó sẽ sử dụng ngữ cảnh mà bộ xử lý sự kiện đang hoạt động.

Bất cứ khi nào một bộ xử lý sự kiện được kích hoạt, thì từ khoá this đại diện cho phần tử DOM được gán một kiểu xử lý. Ở những phần trước bạn cũng đã biết rằng hàm \$() có thể lấy một phần tử DOM làm tham số cho nó. Bằng cách viết \$(this) trong bộ xử lý sự kiện, chúng ta tạo ra một đối tượng jQuery tương ứng với phần tử DOM, và chúng ta có thể thao tác với nó như là chúng ta đã chọn nó bằng bộ chọn CSS. Do đó chúng ta có thể viết

```
$(this).addClass('selected');
```

Chèn dòng code này vào cả 3 bộ xử lý nó sẽ thêm class='selected' mỗi khi nút được nhấn. Để loại bỏ class ở những nút khác, chúng ta có thể tận dụng chức năng vòng lặp ẩn của jQuery để có:

```
$('#switcher .button').removeClass('selected');
```

Khi bạn chèn dòng code trên vào code jQuery của bạn, nó sẽ loại bỏ hết các class ở tất cả các thẻ div có class='button'.

```
$(document).ready(function() {  
    $('#switcher-default').bind('click', function() {  
        $('body').removeClass('red').removeClass('green');    });  
});
```



```

        $('#switcher .button').removeClass('selected');
        $(this).addClass('selected')
    });
    $('#switcher-red').bind('click', function(){
        $('body').addClass('red').removeClass('green');
        $('#switcher .button').removeClass('selected');
        $(this).addClass('selected');
    });
    $('#switcher-green').bind('click', function() {
        $('body').addClass('green').removeClass('red');
        $('#switcher .button').removeClass('selected');
        $(this).addClass('selected')
    });
});

```

## Đổi định dạng

Mặc Định

Màu đỏ

Màu Xanh

## This is the first para

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.

Khái quát chung các mệnh đề bằng cách sử dụng bộ xử lý ngữ cảnh cho phép chúng ta làm việc hiệu quả hơn. Chúng ta có thể tách phần highlight nút được chọn vào một bộ xử lý riêng, bởi vì nó giống nhau ở cả 3 nút. Cho nên chúng ta có thể làm như sau:

```

$(document).ready(function() {
    $('#switcher-default').bind('click', function() {
        $('body').removeClass('red').removeClass('green');
    });
    $('#switcher-red').bind('click', function(){
        $('body').addClass('red').removeClass('green');
    });
    $('#switcher-green').bind('click', function() {
        $('body').addClass('green').removeClass('red');
    });
    $('#switcher .button').bind('click', function() {
        $('#switcher .button').removeClass('selected');
        $(this).addClass('selected');
    });
});

```

Đoạn tối ưu hoá mã này sử dụng 3 chức năng của jQuery mà chúng ta đã học. Đầu tiên là vòng lặp ẩn được sử dụng để gán cùng một bộ xử lý click cho mỗi nút bấm với một lần gọi

.bind(). Thứ hai, thứ tự cách xử lý cho phép chúng ta gán hai hàm cho cùng một sự kiện click mà hàm thứ 2 không đè lên hàm thứ nhất. Cuối cùng, chúng ta sử dụng khả năng kết hợp của jQuery để ghép hai đoạn thêm và bỏ class trên cùng một dòng.

### Tiếp tục tối giản mã

Công đoạn tối ưu hoá mã chúng ta vừa làm ở trên là một ví dụ của tái cơ cấu. Tái cơ cấu mã có nghĩa là chúng ta phải chỉnh sửa mã đang có sao cho nó có thể hoạt động hiệu quả hơn và gọn gàng hơn. Để tìm ra thêm những yếu tố khác có thể tái cơ cấu mã, chúng ta hãy xem xét những cách xử lý mà chúng ta vừa gán cho mỗi nút. Tham số của phương thức .removeClass() là không bắt buộc, khi được bỏ qua, nó loại bỏ tất cả các class của phần tử. Dựa vào điều này, chúng ta có thể rút ngắn mã xuống chút xíu nữa.

```
$(document).ready(function() {
    $('#switcher-default').bind('click', function() {
        $('body').removeClass();
    });
    $('#switcher-red').bind('click', function() {
        $('body').removeClass().addClass('red');
    });
    $('#switcher-green').bind('click', function() {
        $('body').removeClass().addClass('green');
    });
    $('#switcher .button').bind('click', function() {
        $('#switcher .button').removeClass('selected');
        $(this).addClass('selected');
    });
});
```

Ở đoạn mã trên thứ tự của các tác vụ bị thay đổi một chút để phù hợp với việc loại bỏ class không sử dụng tham số. Chúng ta cần phải gọi phương thức .removeClass() trước, như thế nó sẽ không loại bỏ mất class khi chúng ta gọi .addClass() trên cùng một dòng.

**Lưu ý:** Trong ví dụ này chúng ta có toàn quyền quyết định với mã HTML, cho nên chúng ta có thể loại bỏ toàn bộ class. Tuy nhiên khi chúng ta viết mã để sử dụng lại (như là viết một Plugin), thì chúng ta nên tôn trọng những class khác và giữ nguyên chúng.

Hiện tại chúng ta vẫn chạy cùng một đoạn mã cho mỗi bộ xử lý nút bấm. Phần này có thể tái cơ cấu rất dễ dàng bởi một bộ xử lý nút chung:

```
$(document).ready(function() {
    $('#switcher .button').bind('click', function() {
        $('body').removeClass();
        $('#switcher .button').removeClass('selected');
        $(this).addClass('selected');
    });
    $('#switcher-red').bind('click', function() {
        $('body').removeClass().addClass('red');
    });
    $('#switcher-green').bind('click', function() {
        $('body').removeClass().addClass('green');
    });
});
```

Chúng ta đã di chuyển bộ xử lý nút chung lên trên những nút khác. Vậy nên phương thức `.removeClass()` cần phải xảy ra trước khi phương thức `.addClass()` được gọi. Việc này có thể làm được bởi vì jQuery luôn kích hoạt bộ xử lý sự kiện theo thứ tự mà chúng được đăng ký.

Cuối cùng chúng ta cũng có thể loại bỏ luôn cả các bộ xử lý cho từng nút bằng cách sử dụng sự kiện ngữ cảnh. Bởi vì từ khoá `this` cho chúng ta một phần tử DOM chứ không phải là một đối tượng jQuery, chúng ta có thể sử dụng những tính năng của DOM để xác định ID của mỗi phần tử khi nó được click. Do đó chúng ta có thể gán cùng một bộ xử lý cho tất cả các nút, và những bộ xử lý này tiến hành những tác vụ khác nhau cho mỗi nút bấm.

```
$(document).ready(function() {
    $('#switcher .button').bind('click', function() {
        $('body').removeClass();
        if (this.id == 'switcher-red') {
            $('body').addClass('red');
        } else if (this.id == 'switcher-green') {
            $('body').addClass('green');
        }
        $('#switcher .button').removeClass('selected');
        $(this).addClass('selected');
    });
});
```

Trên đây chúng ta chỉ sử dụng một mệnh đề `if ... else` bình thường để kiểm tra xem nút đang được click có ID là gì rồi đưa ra cách xử lý tương ứng cho từng trường hợp.

### Những sự kiện viết tắt

Gán một bộ xử lý cho một sự kiện (ví dụ như là một sự kiện click) rất thường xảy ra, cho nên jQuery cung cấp một cách ngắn gọn hơn. Đó chính là những phương pháp viết tắt sự kiện, nó hoạt động giống như khi chúng ta sử dụng `.bind()` nhưng tiết kiệm được vài chữ.

Ví dụ, đoạn mã của chúng ta có thể sử dụng `.click()` thay vì `.bind()` như sau:

```
$(document).ready(function() {
    $('#switcher .button').click(function() {
        $('body').removeClass();
        if (this.id == 'switcher-red') {
            $('body').addClass('red');
        } else if (this.id == 'switcher-green') {
            $('body').addClass('green');
        }
        $('#switcher .button').removeClass('selected');
        $(this).addClass('selected');
    });
});
```

Những phương pháp viết tắt sự kiện như thế này tồn tại với tất cả những sự kiện DOM tiêu chuẩn:

- blur
- change
- click
- dblclick
- error

- focus
- keydown
- keypress
- keyup
- load
- mousedown
- mousemove
- mouseout
- mouseover
- mouseup resize
- scroll
- select
- submit
- unload

Mỗi phương pháp viết tắt sẽ gán một bộ xử lý vào một sự kiện với tên tương ứng

### Sự kiện phức hợp

Mỗi một phương thức xử lý sự kiện của jQuery đều tương ứng trực tiếp với một sự kiện thuần JavaScript. Tuy nhiên, jQuery có một vài bộ xử lý riêng được thêm vào cho dễ sử dụng và tối ưu hoá việc tương thích các trình duyệt. Một trong những phương thức này chính là `.ready()`, mà chúng ta đã bàn ở trên. Hai phương thức `.toggle()` và `.hover()` là hai bộ xử lý sự kiện tùy chỉnh nữa của jQuery. Chúng đều được gọi là bộ xử lý sự kiện phức hợp bởi vì chúng tương tác với người dùng và phản ứng lại với họ sử dụng nhiều hơn một hàm.

### Ẩn và hiện những tính năng tiên tiến

Giả sử chúng ta muốn ẩn bộ thay đổi màu sắc khi không cần thiết bằng cách làm cho ba nút biến mất. Chúng ta sẽ cho phép người dùng nhấp chuột vào phần tiêu đề “**đổi kiểu dáng**” để ẩn nút ẩn đi nhưng vẫn giữ nguyên tiêu đề. Nếu người dùng nhấp chuột thêm lần nữa sẽ hiện lại các nút bấm. Chúng ta cần thêm một class nữa để xử lý những nút được ẩn.

```
.hidden {
    display: none;
}
```

Chúng ta có thể thêm chức năng vừa nói ở trên vào bằng cách lưu trạng thái hiện tại của nút vào một biến, sau đó kiểm tra giá trị của nó mỗi khi tiêu đề được nhấp để xác định nên hay không nên loại bỏ `class='hidden'` trên các nút bấm. Chúng ta cũng có thể kiểm tra trực tiếp sự hiện diện của class trên một nút, và sử dụng thông tin này để quyết định sẽ phải làm gì. Nhưng thay vào đó, jQuery cho chúng ta một phương thức gọi là `.toggle()`, sẽ làm tất cả những việc phức tạp trên cho mình.

Phương thức `.toggle()` có thể lấy hai hoặc nhiều tham số, mỗi một tham số là một hàm. Khi nhấp chuột lần đầu sẽ chạy hàm thứ nhất, nhấp chuột lần thứ hai sẽ kích hoạt hàm thứ hai v.v.. Khi mỗi hàm đã được kích hoạt, vòng lặp lại bắt đầu từ hàm như trước. Với `.toggle()`, chúng ta có thể tiến hành ẩn hiện nút thay đổi kiểu dáng khá đơn giản:

```
$(document).ready(function() {
    $('#switcher h3').toggle(function() {
        $('#switcher .button').addClass('hidden');
    });
});
```

```
    }, function() {  
        $('#switcher .button').removeClass('hidden');  
    });  
  
    $('#switcher .button').click(function() {  
        $('body').removeClass();  
        if (this.id == 'switcher-red') {  
            $('body').addClass('red');  
        } else if (this.id == 'switcher-green') {  
            $('body').addClass('green');  
        }  
        $('#switcher .button').removeClass('selected');  
        $(this).addClass('selected');  
    });  
});
```

Sau khi nhấp chuột một lần nút sẽ bị ẩn đi

### **Đổi định dạng**

## **This is the first para**

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.

Nhấp chuột thêm lần nữa sẽ hiện lại

## Đổi định dạng

Mặc Định

Màu đỏ

Màu Xanh

## This is the first para

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.

Một lần nữa chúng ta lại sử dụng vòng lặp ẩn để ẩn các nút đi mà không cần phải sử dụng đến một phần tử bao quanh. Với trường hợp cụ thể này, jQuery cung cấp một cách nữa dùng để ẩn hiện bộ nút của chúng ta. Chúng ta có thể sử dụng phương thức `.toggleClass()` để tự động kiểm tra sự hiện diện của class trước khi thêm hoặc loại bỏ nó

```
$(document).ready(function() {  
    $('#switcher h3').click(function() {  
        $('#switcher .button').toggleClass('hidden')  
    });  
});
```

Trong trường hợp này, `.toggleClass()` là giải pháp hay hơn, nhưng `.toggle()` là cách linh hoạt hơn khi tiến hành hai hoặc nhiều tác vụ khác nhau.

### Highlight nút bấm

Để chứng minh cho khả năng của sự kiện click có thể làm việc với những thành phần không nhấp chuột được, chúng ta đã tạo ra một giao diện với những nút mà thực chất chỉ là những thẻ div trở thành một phần sống động của trang, trực chờ người dùng sử dụng nó. Bây giờ chúng ta có thể làm cho nút bấm có thể thay đổi trạng thái khi di chuột qua, cho người dùng biết rằng những nút này sẽ làm một việc gì đó nếu được bấm.

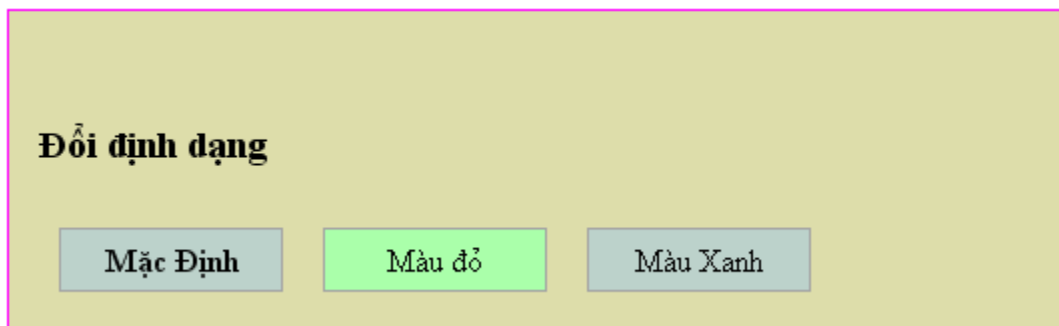
```
#switcher .hover {  
    cursor: pointer;  
    background-color: #afa;  
}
```

CSS cung cấp một pseudo-class gọi là `:hover`, cho phép stylesheet chi phối một thành phần khi người dùng di chuột qua nó. Nhưng trong IE6, chức năng này bị giới hạn chỉ với những đường liên kết, cho nên chúng ta không sử dụng nó cho tất cả các trình duyệt được. Thay vào đó, jQuery cho phép chúng ta sử dụng JavaScript để thay đổi kiểu dáng của một phần tử và có thể tiến hành bất cứ tác vụ nào lên nó. Kể cả khi di chuột lên phần tử và di chuột ra khỏi phần tử đó.

Phương thức `.hover()` lấy hai tham số, giống như ví dụ về `.toggle()` ở trên. Trong trường hợp này, hàm đầu tiên sẽ được thực hiện khi chuột di qua nó và hàm thứ hai sẽ được kích hoạt khi chuột ra khỏi nó. Chúng ta có thể thay đổi class cho các nút tại thời điểm này để tạo ra hiệu ứng rollover:

```
$(document).ready(function() {
    $('#switcher h3').click(function() {
        $('#switcher .button').toggleClass('hidden')
    });
    $('#switcher .button').hover(function() {
        $(this).addClass('hover');
    }, function() {
        $(this).removeClass('hover');
    });
});
```

Chúng ta lại một lần nữa sử dụng vòng lặp ẩn và ngữ cảnh sự kiện để có đoạn mã ngắn hơn và đơn giản hơn. Nếu bây giờ khi bạn di chuột qua các nút bấm, bạn sẽ thấy được hiệu ứng Rollover như hình



Sử dụng `.hover()` cũng giúp chúng ta tránh được những rắc rối tạo ra bởi lan truyền sự kiện (**event propagation**) trong JavaScript. Để hiểu được lan truyền sự kiện là gì, chúng ta hãy xem xét JavaScript quyết định phần tử nào sẽ được xử lý kiện.

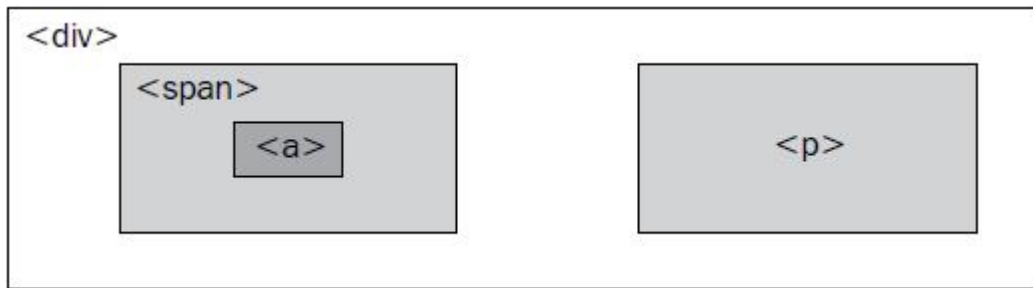
Đường đi của một sự kiện

Khi một sự kiện xảy ra trên một trang, toàn bộ cấu trúc bậc thang của các phần tử DOM đều có cơ hội để xử lý sự kiện. Thử tưởng tượng một mô hình như sau:

```
<div class='mainContent'>
<span class='date'><a href='http://www.izwebz.com'>jQuery tutorial from
izwebz</a></span>
<p> Khi một sự kiện xảy ra trên một trang, toàn bộ cấu trúc bậc thang của
các phần tử DOM đều có cơ hội để xử lý sự kiện.
</p>
```

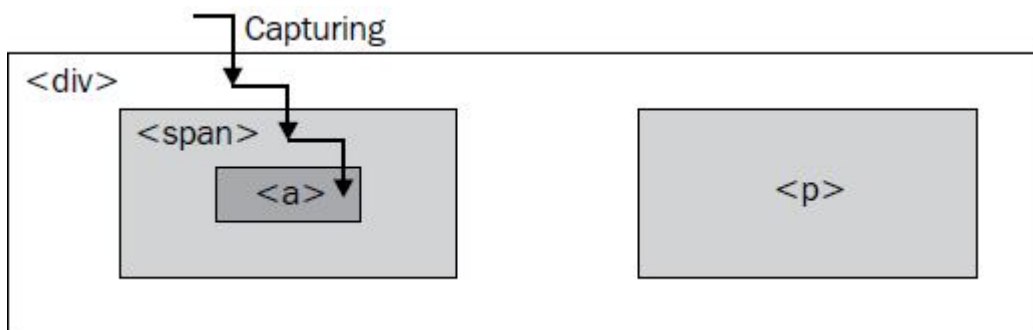
Chúng ta có thể hình tượng hoá đoạn code trên với hình minh họa sau





Với mỗi một sự kiện, sẽ có nhiều phần tử có thể chịu trách nhiệm xử lý. Ví dụ khi đường link ở ví dụ trên được bấm thì ba thành phần như `<div>`, `<span>` và `<a>` đều có cơ hội để phản ứng lại click đó. Bởi vì bạn thấy cả 3 thành phần trên đều nằm dưới con trỏ chuột của người dùng. Nhưng phần tử `<p>` lại không nằm trong mối tương tác này.

Có một cách cho phép nhiều phần tử phản ứng lại với một click được gọi là **Event Capturing**. Với Event Capturing, thì sự kiện được gửi tới phần tử chung nhất sau đó nó đi dần vào những phần tử cụ thể hơn. Ở ví dụ của chúng ta, thì sự kiện sẽ chạy qua thẻ `div`, sau đó đến `span` và cuối cùng là thẻ `a`.



Cách đối lập với cách trên được gọi là **Event Bubbling** (sự kiện bong bóng :-s). Cái này bạn tưởng tượng như trong bể cá trong nhà khi sủi nước vậy. Nước sủi ở dưới, nó bong bóng từ dưới đáy hồ lên trên mặt nước. Sự kiện được gửi tới thành phần cụ thể nhất, và sau khi phần tử này đã có cơ hội để phản ứng, sự kiện sẽ “thổi bong bóng” lên những thành phần chung hơn. Trong ví dụ của chúng ta thì thẻ `a` sẽ xử lý sự kiện trước, sau đó là thẻ `span` và `div` là cuối cùng.

Chẳng có gì là ngạc nhiên khi những người phát triển trình duyệt quyết định những mô hình khác nhau cho sự lan truyền sự kiện. Hệ thống DOM tiêu chuẩn sau này mới dần được phát triển thì định nghĩa rằng cả hai cách trên nên là như sau: đầu tiên sự kiện bị “bắt” bởi những thành phần chung nhất rồi mới đến những phần tử cụ thể hơn, sau đó sự kiện sẽ được “nổi bong bóng” lên trên đỉnh của cây DOM. Bộ xử lý sự kiện có thể được đăng ký ở một trong hai quá trình trên.

Tiếc là không phải toàn bộ các trình duyệt đều áp dụng tiêu chuẩn mới này, và ở những trình duyệt có hỗ trợ Capture thì người ta phải tự tay bật tính năng đó. Để tương thích với mọi trình duyệt, jQuery luôn đăng ký bộ xử lý sự kiện trong quá trình “bong bóng” của mô hình. Chúng ta có thể hiểu rằng, phần tử cụ thể nhất sẽ có cơ hội đầu tiên để phản ứng lại với một sự kiện.

## Phản ứng phụ của sự kiện bong bóng

Event Bubbling có thể tạo ra những biểu hiện không mong đợi, đặc biệt là một thành phần không chủ ý nào đó phản ứng lại với chuột của người dùng. Ví dụ bộ xử lý sự kiện MouseOut được gán vào thẻ div trong ví dụ của chúng ta. Khi chuột của người dùng di chuyển ra ngoài vùng div, thì bộ xử lý MouseOut sẽ được chạy như mong đợi. Bởi vì đây là tầng cao nhất của nấc thang cho nên không thành phần nào khác được “nhìn thấy” sự kiện. Mặt khác, khi trở chuột di chuyển ra ngoài phần tử a, một sự kiện mouseout sẽ được gửi đến nó. Sự kiện này sẽ “nổi bong bóng” lên đến thẻ span và sau đó là thẻ div, và kích hoạt cùng một bộ xử lý sự kiện. Dây sự kiện bong bóng này có thể không được mong đợi trong ví dụ về nút thay đổi kiểu dáng của chúng ta và hiệu ứng tô màu cho đường link có thể bị tắt quá sớm.

Phương thức .hover() hiểu rõ những vấn đề liên quan đến sự kiện bong bóng, và khi chúng ta sử dụng phương thức này để gán sự kiện. Chúng ta có thể bỏ qua những vấn đề tạo ra bởi những thành phần không mong đợi sẽ phản ứng với sự kiện Mouse over và Mouse Out. Cho nên điều này làm cho phương pháp .hover() là một lựa chọn thích hợp để gán cho mỗi sự kiện liên quan đến chuột.

**Lưu ý:** Nếu bạn chỉ quan tâm đến khi người dùng di chuột qua hoặc thoát ra khỏi phần tử, mà không phải cả hai, bạn có thể gán sự kiện mouseenter và mouseleave của jQuery. Cách này cũng tránh được sự kiện bong bóng. Nhưng bởi vì sự kiện liên quan đến chuột thường đi với nhau, cho nên phương pháp .hover() thường là lựa chọn đúng đắn.

Kịch bản Mouse out vừa được nêu ra ở trên cho thấy chúng ta cần phải giới hạn phạm vi của một sự kiện. Trong khi .hover() xử lý tốt trường hợp này, nhưng cũng có trường hợp chúng ta cần phải giới hạn một sự kiện không để nó được gửi tới một thành phần cụ thể khác hoặc tạm thời giới hạn một sự kiện không bị gửi trong bao nhiêu lần.

## Thay đổi đường đi: đối tượng sự kiện

Chúng ta đã thấy một trường hợp mà Event Bubbling có thể tạo ra rắc rối. Chúng ta hãy cùng khám phá một trường hợp mà ở đó .hover() không giúp ích được gì. Chúng ta sẽ sửa lại đoạn code làm sụp nút thay đổi trạng thái ở trên. Giả sử chúng ta muốn mở rộng phạm vi vùng có thể nhấp chuột để kích hoạt hiệu ứng thu nhỏ hoặc mở rộng bộ nút thay đổi định dạng. Cách để thực hiện là di chuyển bộ xử lý sự kiện từ phần label h3 sang phần tử chứa nó là div

```
$('#switcher').click(function() {  
    $('#switcher .button').toggleClass('hidden')  
});
```

Sau khi bạn bỏ thẻ h3 ở phần Selector đi thì bây giờ nếu bạn click vào bất cứ chỗ nào trong phạm vi của bộ nút cũng sẽ làm ẩn hiện nó. Nhưng vấn đề ở đây là kể cả khi bạn nhấp chuột vào bất cứ nút nào, tuy nó vẫn thay đổi màu sắc như mong muốn, nhưng nó lại đóng lại. Đây chính là hiệu ứng bong bóng, sự kiện ban đầu được xử lý bởi các nút. Sau đó nó được truyền lên cây DOM cho đến khi nó chạm tới <div id='switcher'>, ở đó bộ xử lý mới được kích hoạt và ẩn hết các nút đi.

Để giải quyết vấn đề này chúng ta phải truy cập đến đối tượng sự kiện. Đây là cấu trúc của JavaScript được truyền qua mỗi bộ xử lý sự kiện của từng phần tử mỗi khi nó được kích hoạt. Nó cung cấp thông tin về sự kiện đó như là vị trí của con trỏ chuột tại thời điểm của sự kiện.

Nó cũng cung cấp một số phương pháp có thể được sử dụng để tạo ảnh hưởng đến quá trình của một sự kiện thông qua DOM.

Để sử dụng đối tượng sự kiện trong bộ xử lý, chúng ta chỉ cần thêm một tham số vào hàm:

```
$('#switcher').click(function(event) {  
    $('#switcher .button').toggleClass('hidden')  
});
```

### Đích sự kiện (Event Target)

Bây giờ chúng ta đã có đối tượng sự kiện như là một biến sự kiện trong bộ xử lý của chúng ta. Tính năng của đích sự kiện rất hữu dụng trong việc quản lý một sự kiện sẽ được xảy ra ở đâu. Tính năng này là một phần của DOM API (giao diện lập trình ứng dụng), nhưng không được cài đặt ở mọi trình duyệt. Với `.target`, chúng ta có thể xác lập phần tử nào sẽ nhận sự kiện đầu tiên nhất. Trong trường hợp này là một Click Event, đối tượng chính được nhấp chuột. Hãy nhớ rằng nó cho chúng ta một phần tử DOM để xử lý sự kiện, cho nên chúng ta có thể viết như sau:

```
$('#switcher').click(function(event) {  
    if(event.target == this) {  
        $('#switcher .button').toggleClass('hidden')  
    }  
});
```

Đoạn code này đảm bảo rằng đối tượng được click vào chỉ là `<div id='switcher'>`, chứ không phải là các phần tử phụ của nó. Bây giờ khi bạn nhấp chuột vào các nút sẽ không làm ẩn bộ chuyển đi mà nhấn vào vùng nền xung quanh sẽ làm ẩn nó đi. Nhưng nếu bạn nhấn vào nhãn của bộ chuyển là thẻ `h3` thì lại không có gì xảy ra bởi vì nó cũng là phần tử con. Chúng ta có thể thay đổi cách xử lý của các nút để đạt được mục tiêu.

### Ngăn chặn sự lan truyền sự kiện

Đối tượng sự kiện cung cấp phương pháp `.stopPropagation()`, có thể được sử dụng để ngăn chặn hoàn toàn quá trình bong bóng cho sự kiện. Giống như `.target`, phương pháp này là một tính năng thuần JavaScript, nhưng không tương thích với mọi trình duyệt. Miễn là khi chúng ta đã đăng ký tất cả những bộ xử lý sự kiện sử dụng jQuery, chúng ta có thể sử dụng nó mà không sợ bị lỗi.

Chúng ta sẽ loại bỏ `event.target == this` ở trên đi và thay vào đó là một ít mã cho các nút:

```
$('#switcher .button').click(function(event) {  
    $('#body').removeClass();  
    if (this.id == 'switcher-red') {  
        $('#body').addClass('red');  
    } else if (this.id == 'switcher-green') {  
        $('#body').addClass('green');  
    }  
    $('#switcher .button').removeClass('selected');  
    $(this).addClass('selected');  
    event.stopPropagation();  
});
```

Như ở trên chúng ta cũng cần phải thêm một tham số vào hàm mà chúng ta đang sử dụng để xử lý click, để chúng ta có được quyền vào đối tượng sự kiện. Sau đó chúng ta chỉ việc gọi

`event.stopPropagation()` để ngăn chặn các phần tử DOM khác không phản ứng lại sự kiện. Bây giờ khi bạn nhấp chuột, thì chỉ có các nút là xử lý sự kiện đó, và chỉ có các nút thôi, nếu nhấp chuột ra các vùng xung quanh nó sẽ ẩn hoặc hiện bộ nút.

### Tác dụng mặc định

Nếu bộ xử lý sự kiện click của chúng ta được đăng ký cho một phần tử `<a>` thay vì một thẻ `<div>`, chúng ta có thể gặp rắc rối. Khi người dùng nhấp chuột vào đường link, trình duyệt sẽ load một trang web mới. Đây không phải là hiệu ứng bong bóng mà chúng ta đã thảo luận ở trên, mà đây chính là tác dụng mặc định cho lần nhấp chuột vào đường liên kết. Cũng giống như khi phím Enter được nhấn khi người dùng điền form, sự kiện submit sẽ được kích hoạt và form sẽ được submit.

Nếu những tác dụng mặc định này không phải điều bạn muốn và bạn gọi `.stopPropagation()` nó cũng không có tác dụng gì. Những tác dụng này chẳng xảy ra ở chỗ nào của sự lan truyền sự kiện. Thay vào đó, phương pháp `.preventDefault()` sẽ ngăn chặn sự kiện ngay tại thời điểm trước khi tác dụng mặc định được kích hoạt.

**Lưu ý:** Gọi `.preventDefault()` thường chỉ hữu dụng khi chúng ta đã kiểm tra môi trường của sự kiện. Ví dụ trong khi điền form, chúng ta muốn kiểm tra tất cả các trường bắt buộc phải được điền đầy đủ, và chỉ ngăn chặn tác dụng mặc định nếu nó chưa được điền. Chúng ta sẽ học thêm về phần này ở các bài sau

Sự lan truyền sự kiện và tác dụng mặc định là hai chế tài độc lập, một trong hai có thể được ngăn chặn trong khi cái khác vẫn xảy ra. Nếu chúng ta muốn ngăn chặn cả hai, chúng ta có thể return false ngay trong bộ xử lý sự kiện của chúng ta, đó cũng chính là đường tắt để gọi cả hai `.stopPropagation()` và `.preventDefault()` cho sự kiện.

### Ủy thác sự kiện (Event Delegation)

Event Bubbling không phải lúc nào cũng gây ra trở ngại, chúng ta cũng có thể sử dụng nó để làm những việc có ích. Một kỹ thuật rất hay tận dụng bong bóng sự kiện được gọi là ủy thác sự kiện. Khi dùng chúng ta có thể sử dụng một bộ xử lý sự kiện trên một phần tử đơn và áp dụng lên nhiều phần tử khác.

**Lưu ý:** trong jQuery 1.3 một cặp phương pháp được giới thiệu là `.live()` và `.die()`. Hai phương pháp này làm việc giống như `.bind()` và `.unbind()`. Nhưng đằng sau nó, người ta sử dụng ủy thác sự kiện để làm những việc có ích mà chúng ta sẽ bàn thêm ở phần này.

Trong ví dụ của chúng ta, chúng ta chỉ có ba thẻ `<div class='button'>` được gán bộ xử lý nhấp chuột. Nhưng giả sử chúng ta có rất nhiều thẻ div thì sao? Việc này xảy ra nhiều hơn bạn tưởng. Hãy tưởng tượng nếu chúng ta có một bảng và rất nhiều dòng, mỗi dòng có một phần cần có bộ xử lý nhấp chuột. Vòng lặp ẩn có thể gán bộ xử lý sự kiện nhấp chuột dễ dàng, nhưng nó có thể làm mã của bạn hoạt động kém hiệu quả bởi vì vòng lặp được tạo ra bên trong jQuery để thao tác với các bộ xử lý kia.

Thay vào đó, chúng ta có thể gán một bộ xử lý nhấp chuột cho một phần tử gốc trong DOM. Một sự kiện nhấp chuột không gián đoạn sẽ dần dần được gửi tới các thành phần con do hiệu ứng bong bóng sự kiện và chúng ta có thể làm việc ở đó.

Để minh họa, chúng ta hãy áp dụng kỹ thuật này vào bộ thay đổi định dạng ở trên, cho dù số lượng các phần tử không nhiều để mà làm cách này, nhưng cũng đủ để chứng minh được cho bạn. Như đã thấy ở trên, chúng ta có thể sử dụng tính năng `event.target` để xác định thành phần nào đang nằm dưới con trỏ chuột khi người dùng nhấp chuột.

```
$('#switcher .button').click(function(event) {
    if($(event.target).is('.button')) {
        $('body').removeClass();
        if (this.id == 'switcher-red') {
            $('body').addClass('red');
        } else if (this.id == 'switcher-green') {
            $('body').addClass('green');
        }
    }
    $('#switcher .button').removeClass('selected');
    $(this).addClass('selected');
    event.stopPropagation();
})
```

Chúng ta vừa sử dụng một phương pháp mới được gọi là `.is()`. Phương pháp này chấp nhận những Selector chúng ta vừa học ở phần trước và kiểm tra đối tượng jQuery hiện tại với Selector. Nếu có ít nhất một phần tử phù hợp với Selector thì `.is()` sẽ trả về giá trị `True`. Trong ví dụ này, `$(event.target).is('.button')` hỏi xem thành phần được nhấp chuột có class nào gán cho nó không. Nếu có, thì hãy tiếp tục mã từ đó, với một thay đổi rất lớn: từ khóa `this` bây giờ nói đến `<div id='switcher'>`, cho nên mỗi khi chúng ta quan tâm đến nút được nhấp chuột, chúng ta phải gọi nó với `event.target`.

**Lưu ý:** Chúng ta cũng có thể kiểm tra sự tồn tại của một class trên một phần tử với phương pháp ngắn hơn là `.hasClass()`. Nhưng phương pháp `.is()` thì linh động hơn và có thể kiểm tra bất cứ Selector nào.

Tuy nhiên chúng ta vẫn có một hiệu ứng phụ không mong đợi trong đoạn mã trên. Khi chúng ta nhấp chuột vào một nút, cả nhóm sẽ bị đóng lại như trước khi chúng ta gọi `.stopPropagation()`. Bộ xử lý cho việc ẩn hiện bộ nút bây giờ được gán cho thành phần giống như các nút, cho nên nếu bạn chỉ ngăn chặn bong bóng sự kiện sẽ không chặn được việc thay đổi class bị kích hoạt. Để khắc phục vấn đề này, chúng ta sẽ loại bỏ `.stopPropagation()` và thay vào đó là một phương pháp `.is()` nữa.

```
$(document).ready(function() {
    $('#switcher').click(function(event) {
        if (!$ (event.target).is('.button')) {
            $('#switcher .button').toggleClass('hidden');
        }
    });
    $('#switcher .button').click(function(event) {
        if($(event.target).is('.button')) {
            $('body').removeClass();
            if (this.id == 'switcher-red') {
                $('body').addClass('red');
            } else if (this.id == 'switcher-green') {
                $('body').addClass('green');
            }
        }
        $('#switcher .button').removeClass('selected');
        $(this).addClass('selected');
    })
});
```

Thực tế thì ví dụ này hơi bị phức tạp hoá, nhưng khi số thành phần và bộ xử lý sự kiện tăng lên, thì uỷ thác sự kiện là cách mà bạn nên dùng.

**Lưu ý:** Uỷ thác sự kiện cũng hữu dụng trong các trường hợp khác mà chúng ta sẽ học sau này như là khi thêm vào các phần tử bởi phương pháp DOM Manipulation hoặc sử dụng AJAX.

### Loại bỏ một bộ xử lý sự kiện

Có những lúc chúng ta đã sử dụng xong một bộ xử lý sự kiện mà đã được đăng ký từ trước. Ví dụ như trạng thái của trang đã thay đổi mà sự kiện đó không còn phù hợp nữa. Bạn cũng có thể sử dụng những mệnh đề có điều kiện để xử lý tình huống này, nhưng cách hay hơn có thể là hoàn toàn gỡ bỏ (unbind) bộ xử lý đó.

Giả sử chúng ta muốn bộ nút thay đổi kiểu dáng vẫn được mở rộng bất cứ khi nào trang web không sử dụng trạng thái mặc định. Khi mà nút Màu Xanh và Màu Đỏ được nhấn, thì bộ nút sẽ không bị thay đổi gì khi người dùng nhấp chuột xung quanh nó. Chúng ta có thể làm được việc này bằng cách sử dụng phương pháp .unbind() để loại bỏ bộ xử lý đóng lại bộ nút khi mà một trong những nút bấm không phải là nút mặc định được nhấp chuột.

```
$(document).ready(function() {
    $('#switcher').click(function(event) {
        if (!$ (event.target).is('.button')) {
            $('#switcher .button').toggleClass('hidden');
        }
    });
    $('#switcher-red, #switcher-green').click(function() {
        $('#switcher').unbind('click');
    });
});
```

Bây giờ khi bạn nhấp chuột vào nút Màu Xanh hoặc Màu Đỏ, thì bộ xử lý nhấp chuột ở thẻ div mẹ sẽ bị loại bỏ, cho nên khi nhấp chuột vào vùng xung quanh của hộp sẽ không làm ẩn bộ nút đi. Nhưng hệ thống nút của chúng ta cũng không làm việc nữa. Bởi vì nó cũng được gán với bộ xử lý nhấp chuột của thẻ <div> mẹ do chúng ta viết lại để sử dụng uỷ thác sự kiện. Cho nên khi chúng ta gọi \$('#switcher').unbind('click'), cả hai cách xử lý đều bị loại bỏ.

## Chương 5 – Hiệu ứng (Effects)

Các hiệu ứng động của jQuery sẽ làm cho trang web của bạn thêm phần sinh động. JQuery cho phép bạn ẩn hiện, trượt lên trượt xuống các thành phần của trang web. Bạn cũng có thể cho nó xảy ra cùng một lúc hoặc theo thứ tự định trước. Trong phần này chúng ta sẽ tìm hiểu các hiệu ứng jQuery và kết hợp chúng để tạo ra những hiệu ứng hay.

### Thay đổi Inline CSS

Trước khi chúng ta học những hiệu ứng jQuery, chúng ta cần xem lại một chút về CSS. Trong những chương trước đây chúng ta thay đổi giao diện của các thành phần trên trang bằng cách khai báo thuộc tính của class trong một stylesheet riêng biệt. Sau đó chúng ta thêm hoặc loại bỏ những class đó bằng jQuery. Về cơ bản thì cách này nên được sử dụng để thêm CSS vào HTML bởi vì nó tôn trọng quy luật tách riêng phần trình bày và cấu trúc. Tuy nhiên,

đôi khi chúng ta muốn áp dụng style cho những thành phần chưa được, hoặc khó mà được, định dạng bằng stylesheet. Nhưng rất may mắn là jQuery có phương thức `.css()` để sử dụng cho những trường hợp này.

Phương thức này hoạt động bằng cả hai cách lấy và đặt. Để lấy giá trị của một thuộc tính, chúng ta chỉ cần chuyển tên của thuộc tính đó thành một chuỗi, dạng như `.css('backgroundColor')`. JQuery có thể hiểu được những thuộc tính kết hợp bởi nhiều từ và nối với nhau bằng dấu – như là trong CSS (`'background-color'`), hoặc dạng viết hoa chữ cái đầu như là (`'backgroundColor'`). Để định dạng thuộc tính style, phương thức `.css()` có hai cách sử dụng. Cách thứ nhất chỉ nhận một cặp thuộc tính – giá trị. Cách thứ hai là nhận một tập hợp các cặp thuộc tính – giá trị.

```
.css('property', 'value')
.css({property1: 'value1', 'property-2': 'value2'})
```

Những người đã quen với JavaScript sẽ nhận ra đây là dạng đối tượng trực tiếp JavaScript.

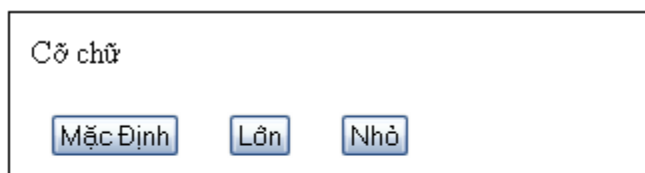
**Chú ý:** Các giá trị số không dùng dấu ngoặc kép trong khi giá trị chuỗi phải có dấu ngoặc kép. Tuy nhiên, khi sử dụng bản đồ ký hiệu, dấu ngoặc kép không bắt buộc cho những tên thuộc tính được viết dưới dạng in hoa chữ cái đầu.

Chúng ta sử dụng phương thức `.css()` cũng giống như cách chúng ta đã sử dụng `.addClass()`. Bằng cách gán nó cho một Selector sau đó thì Bind nó vào một sự kiện. Để minh họa, chúng ta sẽ sử dụng bộ nút thay đổi định dạng trong chương 3, nhưng với mã HTML khác.

```
<div id="switcher">
  <div class="label">Cỡ chữ</div>
  <button id="switcher-default">Mặc Định</button>
  <button id="switcher-large">Lớn</button>
  <button id="switcher-small">Nhỏ</button>
</div><!--End #switcher-->

  <div class="speech">
    <p>
      Pellentesque habitant morbi tristique senectus et netus et
malesuada
      fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae,
ultricies eget,
      tempor sit amet, ante. Donec eu libero sit amet quam egestas
semper.
      Aenean ultricies mi vitae est.
    </p>
  </div><!--End .speech-->
```

Với một chút định dạng CSS cơ bản chúng ta có được hình dưới đây.



Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam vitae est.

Xem [Demo online](#) – Example 1

Khác với bộ nút ở chương trước, trong chương này chúng ta sẽ sử dụng phần tử `<button>`. Khi người dùng nhấp chuột vào nút Lớn hoặc Nhỏ sẽ tăng hoặc giảm cỡ chữ trong thẻ `<div class='speech'>`. Cuối cùng họ cũng có thể nhấn vào nút Mặc Định để trả cỡ chữ về giá trị ban đầu.

Nếu chúng ta chỉ muốn tăng cỡ chữ một lần với một giá trị đặt trước thì chúng ta vẫn có thể sử dụng phương thức `.addClass()`. Nhưng lần này chúng ta sẽ cho phép người dùng nhấp chuột nhiều lần vào nút, và mỗi lần nhấp cỡ chữ sẽ tăng hoặc giảm dần lên. Tất nhiên bạn cũng có thể tạo ra nhiều class và gán chúng cho mỗi lần nhấp chuột và sau đó chúng ta cho chạy vòng lặp qua từng class. Nhưng làm như thế nó thủ công quá và rất mất thời gian, cho nên cách lẹ hơn sẽ là lấy cỡ chữ của đoạn văn đó trước, sau đó thì tăng nó lên với giá trị mình muốn. Trong ví dụ này chúng ta sẽ cho tăng cỡ chữ lên 40% mỗi khi người dùng nhấp chuột.

Đoạn mã của chúng ta sẽ bắt đầu bằng `$(document).ready()` và bộ xử lý sự kiện `$('#switcher-large').click()`

```
$(document).ready(function() {  
  $('#switcher-large').click(function() {  
  });  
});
```

Để biết được cỡ chữ của đoạn văn đó là bao nhiêu rất đơn giản, jQuery có phương thức `.css()` cho phép bạn làm việc này. Tuy nhiên phương thức này lại trả về giá trị có thêm cái đuôi 'px', ví dụ là đoạn văn có kích thước chữ là 16px thì giá trị trả về sẽ là 16px. Cho nên chúng ta phải tìm cách cắt cái đuôi 'px' đó đi và chỉ giữ lại phần giá trị số là 16. Một điểm nữa là khi chúng ta định sử dụng một đối tượng jQuery nhiều lần, bạn nên nhớ lại selector bằng cách lưu đối tượng đó vào một biến.

```
$(document).ready(function() {  
  var $speech = $('#div.speech'); //lưu đối tượng jQuery vào biến $speech  
  $('#switcher-large').click(function() {  
    var num = parseFloat($speech.css('fontSize'), 10);  
  });  
});
```

Ở dòng code thứ 2 chúng ta đã tạo ra một biến là `$speech` và lưu đối tượng jQuery vào đó. Bạn cũng nên lưu ý cách tôi đặt tên biến bắt đầu bằng dấu \$, bởi vì trong JavaScript bạn hoàn toàn có thể sử dụng dấu \$ để đặt tên cho biến. Cho nên đây là cách để nhắc nhở chúng ta về sau là biến này đang chứa một đối tượng jQuery.

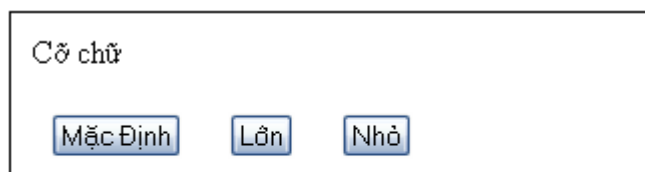


Trong bộ xử lý `.click()`, chúng ta sử dụng hàm `parseFloat()` để cắt đi phần đuôi 'px' và chỉ giữ lại phần giá trị số. Hàm `parseFloat()` sẽ kiểm tra một chuỗi theo thứ tự từ trái qua cho đến khi nó gặp một ký tự không phải là dạng số. Chuỗi số sẽ được biến thành dạng số thập phân. Trong ví dụ này nó sẽ biến chuỗi '16' thành dạng số 16 (quá giỏi >:<) và tất nhiên nó cũng sẽ cắt phần đuôi 'px' đi bởi vì nó không phải là số. Còn trong trường hợp chuỗi đó bắt đầu bằng chữ thay vì là số, hàm `.parseFloat()` sẽ trả về một giá trị là NaN, là chữ viết tắt của **Not A Number** (không phải số). Hàm `parseFloat()` có argument thứ hai để đảm bảo rằng giá trị số trả về dưới dạng hàng chục chứ không phải các dạng khác.

Cuối cùng chúng ta chỉ cần nhân biến `num` với 1.4 (tăng 40%) và sau đó đặt kích cỡ chữ bằng cách kết hợp `num` với 'px'

```
$(document).ready(function() {  
  var $speech = $('#div.speech');  
  $('#switcher-large').click(function() {  
    var num = parseFloat($speech.css('fontSize'), 10 );  
    num *= 1.4;  
    $speech.css('fontSize', num + 'px');  
  });  
});
```

**Lưu ý:** Phương trình `num *= 1.4` là dạng viết tắt phổ biến. Nó tương đương với `num = num * 1.4`. Nếu bạn thấy cách viết thường dễ hiểu hơn thì bạn cũng có thể sử dụng. Còn nếu không bạn cũng có thể dùng cách viết trên cho các phương trình khác như tính cộng `num += 1.4`, tính trừ `num -= 1.4`, tính chia `num /= 1.4` và chia với số dư `num %= 1.4`.



Pellentesque habitant morbi tristique senectus  
malesuada fames ac turpis egestas. Vestibulum  
quam, feugiat vitae, ultricies eget, tempor sit a  
Donec eu libero sit amet quam egestas semper  
ultricies mi vitae est.

Xem [Demo online](#) – Example 2

Bây giờ để nút Nhỏ Hơn có thể hoạt động, chúng ta sẽ chia `num /= 1.4`. Hơn nữa chúng ta sẽ kết hợp cả hai phép tính trên vào một bộ xử lý `.click()` cho tất cả các phần tử `<button>` nằm trong thẻ `<div id='switcher'>`. Sau khi đã tìm ra được giá trị số, và dựa vào ID của nút nào được nhấn, chúng ta sẽ sử dụng phép nhân hoặc chia. Dưới đây là đoạn mã để làm việc này.

```
$(document).ready(function() {  
  var $speech = $('#div.speech');
```

```

$('#switcher button').click(function() {
var num = parseFloat( $speech.css('fontSize'), 10 );
if (this.id == 'switcher-large') {
num *= 1.4;
} else if (this.id == 'switcher-small') {
num /= 1.4;
}
$speech.css('fontSize', num + 'px');
});
});

```

Nhớ lại ở chương 3 chúng ta có thể lấy thuộc tính id của một phần tử DOM bằng cách sử dụng từ khoá this, trong trường hợp này nó xuất hiện trong mệnh đề if ... else. Làm như vậy thì nó hiệu quả hơn là phải tạo ra một đối tượng jQuery chỉ để kiểm tra giá trị của một thuộc tính.

Tiếp theo chúng ta cũng phải làm cho nút Mặc Định hoạt động để người dùng có thể trả về giá trị mặc định lúc ban đầu. Việc chúng ta cần làm là lưu kích thước font chữ của đoạn văn vào một biến ngay khi DOM sẵn sàng. Sau đó chúng ta có thể gọi lại giá trị này mỗi khi nút Mặc Định được nhấp. Chúng ta cũng có thể sử dụng thêm một mệnh đề else ... if nữa, nhưng có lẽ mệnh đề Switch trong trường hợp này là hợp lý hơn.

```

$(document).ready(function() {
var $speech = $('div.speech');
var defaultSize = $speech.css('fontSize');
$('#switcher button').click(function() {
var num = parseFloat( $speech.css('fontSize'), 10 );
switch (this.id) {
case 'switcher-large':
num *= 1.4;
break;
case 'switcher-small':
num /= 1.4;
break;
default:
num = parseFloat(defaultSize, 10);
}
$speech.css('fontSize', num + 'px');
});
});

```

Ở đoạn code trên chúng ta vẫn kiểm tra giá trị của this.id và thay đổi kích thước chữ dựa vào nó, nhưng nếu giá trị của nó không phải là 'switcher-large' hoặc 'switcher-small' thì nó sẽ mặc định là kích cỡ ban đầu.

Xem [Demo online](#) – Example 3

#### Ẩn hiện cơ bản

Hai phương thức .hide() và .show(), khi không có tham số, có thể được coi là phương thức rút gọn của .css('display','string'), ở đó 'string' là một giá trị bất kỳ. Hiệu ứng đạt được của hai phương thức này thì cũng đơn giản như tên gọi, có nghĩa là nó sẽ ẩn hoặc hiện một thành phần nào đó.

Phương thức `.hide()` sẽ làm cho thuộc tính inline style cho các phần tử phù hợp trở thành `display:none`. Nhưng cái hay của phương thức này là ở chỗ nó ghi nhớ giá trị thuộc tính của `display` (thường là `inline` hoặc `block`) trước khi nó bị đổi thành `none`. Ngược lại, phương thức `.show()` lại trả về giá trị thuộc tính `display` ban đầu trước khi nó bị biến thành `display:none`.

Tính năng này của `.show()` và `.hide()` đặc biệt có ích khi bạn muốn ẩn một thành phần nào đó có thuộc tính `display` mặc định đã được khai báo trong stylesheet. Ví dụ, phần tử `<li>` có giá trị `display` mặc định là `display:block`, nhưng chúng ta lại muốn biến nó thành `display:inline` để sử dụng trong menu nằm ngang. May cho chúng ta là khi sử dụng phương thức `.show()` cho một thành phần bị ẩn như ở ví dụ này là những thẻ `<li>`. Nó sẽ không trả lại giá trị mặc định là `display:block`, bởi vì nếu như thế thì mỗi thẻ `<li>` lại xuất hiện trên một hàng thì hỏng hết. Thay vào đó, các phần tử sẽ được trả lại trạng thái trước là `display:inline`, như thế thì menu nằm ngang mới có thể hoạt động được.

Để minh họa cho tính năng trên, chúng ta sẽ thêm một đoạn văn bản thứ hai và một thẻ link “read more” vào sau đoạn văn thứ nhất.

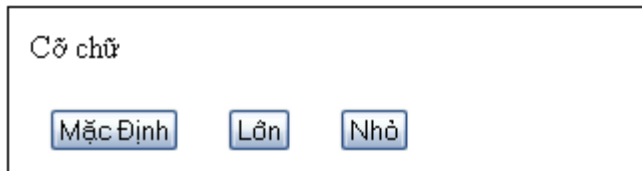
```
<div id="switcher">
  <div class="label">Cỡ chữ</div>
  <button id="switcher-default">Mặc Định</button>
  <button id="switcher-large">Lớn</button>
  <button id="switcher-small">Nhỏ</button>
</div><!--End #switcher-->

  <div class="speech">
    <p>
      Pellentesque habitant morbi tristique senectus et netus et
malesuada
      fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae,
ultricies eget,
      tempor sit amet, ante. Donec eu libero sit amet quam egestas
semper.
      Aenean ultricies mi vitae est.
    </p>
    <p>
      Pellentesque habitant morbi tristique senectus et netus et
malesuada
      fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae,
ultricies eget,
      tempor sit amet, ante. Donec eu libero sit amet quam egestas
semper.
      Aenean ultricies mi vitae est.
    </p>

    <a href="#" class="more">Read More</a>
```

Khi DOM sẵn sàng thì đoạn văn thứ hai bị ẩn đi

```
$(document).ready(function() {
  $('p:eq(1)').hide();
});
```



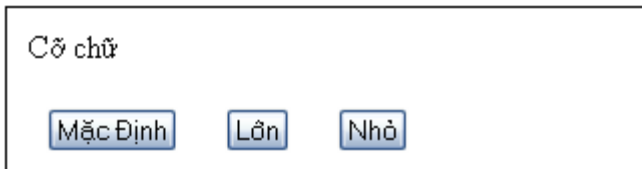
Pellentesque habitant morbi tristique senectus et netus et malesuada fames feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet vitae est.

[Read More](#)

Xem [Demo online](#) – Example 4

Khi đường link Read More được click thì đoạn văn thứ 2 sẽ xuất hiện và chữ Read More sẽ bị ẩn đi

```
$(document).ready(function() {
    $('p:eq(1)').hide();
    $('a.more').click(function() { //Khi thẻ <a class='more'> được
click
        $('p:eq(1)').show(); //cho hiển thị đoạn văn thứ 2
        $(this).hide(); //this ở đây là chỉ đối tượng jQuery a.more ẩn
đi
        return false; // ngăn không cho đường link hoạt động như mặc
định
    });
});
```



Pellentesque habitant morbi tristique senectus et netus et malesuada fames feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet vitae est.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet vitae est.

Dòng mã mà bạn cần lưu ý ở trên là đoạn `return false`. Bởi vì mặc định của đường liên kết mỗi khi được nhấp là sẽ liên kết đến trang khác hoặc phần nào đó. Nhưng khi ta thêm dòng `return false`, nó sẽ giúp ngăn chặn mặc định của đường liên kết.

Hai phương thức `.show()` và `.hide()` ở trên tuy ngắn gọn và dễ sử dụng nhưng nó lại không được “mượt” cho lắm, cho nên ở phần tiếp theo chúng ta sẽ làm cho nó mượt hơn.

Hiệu ứng và tốc độ

Khi ta thêm tốc độ, hay nói chính xác hơn là khoảng thời gian, vào phương thức `.show()` hoặc `.hide()`, nó sẽ trở thành hiệu ứng động xảy ra trong một khoảng thời gian định trước. Ví dụ

nghư phương thức `.hide()` làm giảm chiều cao, độ rộng và tính trong suốt của một phần tử cùng một lúc cho đến khi cả ba giá trị đều bằng không. Đến lúc đó thuộc tính của CSS là `display: none` sẽ được áp dụng. Mặt khác, phương thức `.show()` thì tăng chiều cao của một phần tử từ trên xuống dưới, chiều rộng từ trái qua phải và độ trong suốt từ 0 đến 1 cho đến khi phần tử đó hoàn toàn có thể được nhìn thấy.

## Tốc độ

Với hiệu ứng jQuery, chúng ta có thể sử dụng ba tốc độ có sẵn là: `'slow'`, `'normal'` và `'fast'`. Sử dụng `.show('slow')` sẽ làm cho hiệu ứng Show diễn ra trong 0.6 giây, `.show('normal')` trong vòng 0.4 giây và `.show('fast')` là 0.2 giây. Nếu bạn muốn chính xác hơn nữa, bạn có thể sử dụng `millisecond` như: `.show(850)`. Không giống như dạng chữ, khi dùng `millisecond`, số không cần phải bỏ trong dấu nháy.

Quay lại ví dụ ở trên, bây giờ chúng ta sẽ thêm tốc độ vào xem nó như thế nào.

```
$(document).ready(function() {
  $('p:eq(1)').hide();
  $('a.more').click(function() {
    $('p:eq(1)').show('slow');
    $(this).hide();
    return false;
  });
});
```

Xem [Demo online](#) – Example 5

## Fade in và Fade out

Hai phương thức `.show()` và `.hide()` cũng đã đủ ‘độ mượt’ rồi, nhưng có đôi khi bạn lại thấy ‘mượt’ quá cũng không tốt. Cho nên jQuery cho chúng ta một số những hiệu ứng động có sẵn khác để thêm phần linh hoạt. Ví dụ khi bạn muốn cả đoạn văn xuất hiện từ từ bằng cách tăng dần độ trong suốt của nó, chúng ta có thể sử dụng `.fadeIn('slow')`:

```
$(document).ready(function() {
  $('p:eq(1)').hide();
  $('a.more').click(function() {
    $('p:eq(1)').fadeIn('slow');
    $(this).hide();
    return false;
  });
});
```

Nếu muốn bạn cũng có thể thử với `.fadeOut('slow')` xem công dụng của nó như thế nào.

## Hiệu ứng đa hợp

Đôi khi chúng ta muốn đảo trạng thái ẩn hiện một thành phần nào đó thay vì chỉ hiển thị nó một lần như chúng ta đã làm ở các bước trên. Đảo trạng thái có thể làm được bằng cách kiểm tra độ nhìn thấy của một tập hợp các thành phần trước, sau đó thì gán một phương thức phù hợp cho nó. Sử dụng lại hiệu ứng `fade`, chúng ta có thể sửa lại mã như sau:

```
$(document).ready(function() {
```

```

var $firstPara = $('p:eq(1)');
$firstPara.hide();
$('a.more').click(function() {
  if ($firstPara.is(':hidden')) {
    $firstPara.fadeIn('slow');
    $(this).text('read less');
  } else {
    $firstPara.fadeOut('slow');
    $(this).text('read more');
  }
  return false;
});
});

```

Như chúng ta từng làm ở trên, chúng ta lưu selector vào một biến, trong ví dụ này là `$firstPara = $('p:eq(1)')`, để tránh phải lập lại việc lên xuống cây DOM. Chúng ta cũng không còn ẩn đi đường liên kết mà thay đổi chữ của nó.

Sử dụng mệnh đề `if ... else` là hoàn toàn hợp lý để đảo trạng thái ẩn hiện của một phần tử. Nhưng với hiệu ứng đa hợp của jQuery, chúng ta không cần phải sử dụng dạng điều kiện này nữa. JQuery có một phương thức là `.toggle()`, hoạt động giống như `.show()` và `.hide()`, nó cũng có thể nhận hoặc không nhận tham số tốc độ. Một phương thức đa hợp nữa là `.slideToggle()`, nó được sử dụng để ẩn hoặc hiện một phần tử bằng cách từ từ tăng hoặc giảm độ cao của nó. Đoạn mã dưới đây chúng ta sẽ sử dụng phương thức `.slideToggle()`.

```

$(document).ready(function() {
  var $firstPara = $('p:eq(1)');
  $firstPara.hide();
  $('a.more').click(function() {
    $firstPara.slideToggle('slow');
    var $link = $(this);
    if ( $link.text() == "read more" ) {
      $link.text('read less');
    } else {
      $link.text('read more');
    }
    return false;
  });
});

```

Lần này từ khoá `$(this)` được dùng nhiều lần, cho nên chúng ta lưu nó lại vào một biến `$link` để đoạn mã hoạt động hiệu quả hơn và cũng dễ đọc hơn. Hơn nữa, mệnh đề điều kiện chỉ kiểm tra nội dung chữ của đường liên kết thay vì xem xem đoạn văn thứ 2 có hiện hay không. Bởi vì chúng ta chỉ sử dụng mệnh đề này để thay đổi chữ của nó.

### Tự tạo hiệu ứng động

Ngoài những phương thức có sẵn, jQuery cung cấp thêm một phương thức rất mạnh nữa là `.animate()`. Nó cho phép chúng ta tự tạo ra những hiệu ứng động theo ý thích của mình. Phương thức `.animate()` có hai dạng. Dạng thứ nhất có thể nhận bốn đối số.

- 1. Cặp thuộc tính và giá trị – giống như `.css()` mà chúng ta đã thảo luận ở trên.
- 2. Tốc độ tùy chọn – có thể là một trong những tốc độ có sẵn hoặc một số dưới dạng millisecond.
- 3. Kiểu di chuyển – sẽ được bàn kỹ hơn ở chương sau

- 4. Một hàm gọi ngược sẽ được bàn ở phần dưới.

Kết hợp lại thì bốn đối số trên sẽ có dạng công thức chung như sau

```
.animate({property1: 'value1', property2: 'value2'},
speed, easing, function() {
alert('đã tiến hành xong');
});
```

Dạng thứ hai lấy vào 2 đối số, thuộc tính và tùy chọn.

```
.animate({properties}, {options})
```

Khi chúng ta xuống dòng để dễ đọc hơn thì dạng thứ 2 nhìn như sau:

```
.animate({
    property1: 'value1',
    property2: 'value2'
}, {
duration: 'value',
easing: 'value',
complete: function() {
alert('The animation is finished.');
```

Trước hết chúng ta sẽ sử dụng dạng thứ nhất của .animate(), sau đó chúng ta sẽ sử dụng dạng thứ 2 ở phần sau của chương này khi chúng ta bàn tới xếp hàng hiệu ứng.

### Đảo trạng thái Fade

Khi chúng ta nói về hiệu ứng đa hợp, bạn có thấy rằng không phải phương thức nào cũng có những phương thức đảo trạng thái đi kèm. Ví dụ phương thức .slide() thì có .slideToggle(), nhưng không có .fadeToggle() cho .fadeIn() và .fadeOut(). Nhưng chúng ta có thể dễ dàng sử dụng phương thức .animate() để tạo ra hiệu ứng đảo trạng thái fade. Ở đoạn mã dưới đây, chúng ta sẽ thay thế phương thức .slideToggle() với hiệu ứng động tự tạo.

```
$(document).ready(function() {
$('p:eq(1)').hide();
$('a.more').click(function() {
$('p:eq(1)').animate({opacity: 'toggle'}, 'slow');
var $link = $(this);
if ( $link.text() == "read more" ) {
$link.text('read less');
} else {
$link.text('read more');
}
return false;
});
});
```

Xem ví dụ bạn sẽ thấy, phương thức `.animate()` cũng cho phép sử dụng những từ khoá như 'show', 'hide' và 'toggle'. Khi mà những phương thức rút gọn khác không phù hợp với tác vụ.

### Hiệu ứng động đa thuộc tính

Với phương thức `.animate()`, chúng ta có thể cùng một lúc sửa đổi bất cứ sự kết hợp nào của các thuộc tính. Ví dụ khi bạn muốn cùng một lúc tạo ra hai hiệu ứng trượt và mờ đi khi đảo trạng thái của đoạn văn thứ 2, chúng ta chỉ việc thêm cặp thuộc tính – giá trị chiều cao vào bản đồ thuộc tính `.animate()`.

```
$(document).ready(function() {
  $('p:eq(1)').hide();
  $('a.more').click(function() {
    $('p:eq(1)').animate({
      opacity: 'toggle',
      height: 'toggle'
    },
    'slow');
    var $link = $(this);
    if ( $link.text() == "read more" ) {
      $link.text('read less');
    } else {
      $link.text('read more');
    }
    return false;
  });
});
```

Hơn nữa, chúng ta không chỉ có những thuộc tính định dạng để sử dụng cho những phương thức rút gọn mà chúng ta còn có những thuộc tính khác như: `left`, `top`, `fontSize`, `margin`, `padding` và `borderWidth`. Hãy nhớ lại đoạn mã thay đổi kích thước chữ của đoạn văn ở trên. Chúng ta có thể tăng hoặc giảm kích thước bằng cách dùng `.animate()` thay cho `.css()`.

```
$(document).ready(function() {
  var $speech = $('div.speech');
  var defaultSize = $speech.css('fontSize');
  $('#switcher button').click(function() {
    var num = parseFloat( $speech.css('fontSize'), 10 );
    switch (this.id) {
      case 'switcher-large':
        num *= 1.4;
        break;
      case 'switcher-small':
        num /= 1.4;
        break;
      default:
        num = parseFloat(defaultSize, 10);
    }
    $speech.animate({fontSize: num + 'px'},
    'slow');
  });
});
```

Những thuộc tính vừa nêu ở trên còn cho phép chúng ta tạo ra nhiều hiệu ứng phức tạp khác. Ví dụ chúng ta có thể di chuyển một thành phần từ trái sang phải của trang web và cùng một



lúc tăng chiều cao của nó lên 20px và thay đổi độ dày của border lên thành 5px. Chúng ta sẽ thực nghiệm với thẻ <div id='switcher'>.

Với những giao diện co giãn, chúng ta phải tính toán được khoảng cách mà hộp div sẽ di chuyển trước khi nó chạm vào đường biên bên phải của trang. Giả sử rằng độ rộng của đoạn văn bản là 100%, chúng ta có thể lấy độ rộng của đoạn văn bản trừ đi độ rộng của hộp Cỡ Chữ. JQuery có một phương thức là .width() có thể sử dụng được trong trường hợp này, tuy nhiên nó lại không tính được padding trái phải hoặc độ rộng đường viền. Với jQuery phiên bản 1.2.6, chúng ta có thêm phương thức .outerWidth(). Đây chính là phương thức chúng ta sẽ sử dụng để tránh phải tính thêm vào padding và border. Trong ví dụ này chúng ta sẽ bắt đầu hiệu ứng động khi mà người dùng nhấp chuột vào từ **Cỡ Chữ**, ở ngay phía trên hàng nút. Đoạn mã sẽ tương tự như sau

```
$(document).ready(function() {
    $('div.label').click(function() {
        var paraWidth = $('div.speech p').outerWidth();
        var $switcher = $(this).parent();
        var switcherWidth = $switcher.outerWidth();
        $switcher.animate({left: paraWidth - switcherWidth,
            height: '+=20px', borderWidth: '5px'}, 'slow');
    });
});
```

Bạn nên chú ý đến thuộc tính height có dấu += trước giá trị pixel. Nó có nghĩa là giá trị tương đối. Nên thay vì nó làm cho hộp biến thành 20px, nó sẽ làm cho hộp to ra 20 px lớn hơn so với kích thước hiện tại.

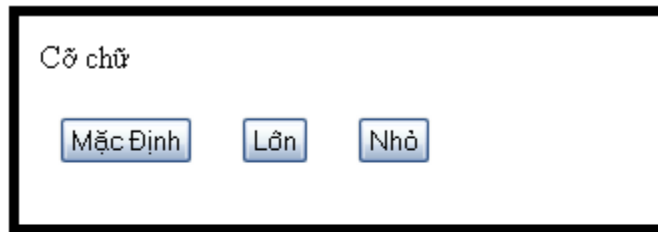
Mặc dù đoạn mã trên làm cho thẻ div cao lên và border dày lên, nhưng nó không di chuyển sang bên tay phải như chúng ta muốn. Chúng ta phải thay đổi thuộc tính position của nó trong CSS.

### Định vị trí với CSS

Khi bạn làm việc với .animate(), bạn nên nhớ đến tầm ảnh hưởng của CSS lên các thành phần chúng ta muốn sửa đổi. Ví dụ khi bạn điều chỉnh thuộc tính left như trên, nó sẽ không tạo ra thay đổi gì với các phần tử đó trừ khi những phần tử bạn muốn thay đổi có thuộc tính position là relative hoặc absolute trong CSS. Vị trí mặc định của CSS cho những thành phần Block-level là static, có nghĩa là tĩnh, điều đó nói lên lý do tại sao nó vẫn giữ nguyên vị trí khi bạn cố gắng di chuyển nó. Do vậy nếu bạn muốn nó được thay đổi, bạn phải sửa lại giá trị position của nó trong CSS.

```
#switcher {
    position: relative;
}
```

Sau khi đã thay đổi thuộc tính position trong CSS, hộp nút của chúng ta đã di chuyển sang bên tay phải như hình



nectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam,  
met, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi

nectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam,  
met, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi

Xem [Demo online](#) – Example 6

Hiệu ứng đồng bộ và theo thứ tự

Phương thức `.animate()` như chúng ta đã biết được sử dụng để tạo ra hiệu ứng đồng bộ cho một tập hợp các phần tử. Nhưng cũng có lúc chúng ta lại muốn các hiệu ứng xảy ra theo thứ tự hết cái này rồi mới đến cái kia.

Làm việc với một tập hợp các phần tử đơn lẻ

Khi bạn muốn áp dụng nhiều hiệu ứng cho cùng một tập hợp các phần tử, queuing dễ dàng được tạo ra bằng cách kết nối các hiệu ứng lại. Để minh họa cho việc này, chúng ta lại sẽ di chuyển hộp **Cỡ Chữ** sang bên tay phải và tăng chiều cao và đường biên của nó. Tuy nhiên lần này chúng ta sẽ cho 3 hiệu ứng đó xảy ra lần lượt theo thứ tự. Chúng ta chỉ cần đặt mỗi hiệu ứng trong một phương thức `.animate()` và sau đó thì kết nối chúng lại với nhau

```
$(document).ready(function() {  
  $('div.label').click(function() {  
    var paraWidth = $('div.speech p').outerWidth();  
    var $switcher = $(this).parent();  
    var switcherWidth = $switcher.outerWidth();  
    $switcher  
      .animate({left: paraWidth - switcherWidth},  
        'slow')  
      .animate({height: '+=20px'}, 'slow')  
      .animate({borderWidth: '5px'}, 'slow');  
  });  
});
```

Tất nhiên khi kết nối các phương thức lại, chúng ta có thể để cả ba phương thức `.animate()` trên cùng một dòng, nhưng vì như thế nó hơi khó đọc do vậy chúng ta để mỗi phương thức trên một dòng cho nó dễ đọc hơn.

Chúng ta có thể xếp hàng bất cứ phương thức jQuery nào bằng cách kết nối chúng với nhau. Ví dụ chúng ta cũng có thể kết nối các hiệu ứng cho thẻ `<div id='switcher'>` với thứ tự sau:

1. Làm giảm độ trong suốt của nó xuống .5 với `.fadeTo()`.
2. Di chuyển nó sang tay phải với `.animate()`.
3. Tăng lại độ trong suốt thành 1 với `.fadeTo()`.

4.Ấn nó đi với `.slideUp()`.

5.Cho hiện lại nó với `.slideDown()`.

Tất cả những gì chúng ta phải làm là kết nối những hiệu ứng trên với thứ tự tương tự trong đoạn mã của chúng ta.

```
$(document).ready(function() {
  $('div.label').click(function() {
    var paraWidth = $('div.speech p').outerWidth();
    var $switcher = $(this).parent();
    var switcherWidth = $switcher.outerWidth();
    $switcher
      .fadeTo('fast',0.5)
      .animate({
        'left': paraWidth - switcherWidth
      }, 'slow')
      .fadeTo('slow',1.0)
      .slideUp('slow')
      .slideDown('slow');
  });
});
```

Nhưng nếu bây giờ chúng ta muốn di chuyển thẻ `<div>` sang bên tay phải và cùng một lúc làm giảm độ trong suốt của nó đi một nửa thì sao? Nếu hai hiệu ứng này cùng xảy ra với cùng một tốc độ, thì chúng ta đơn giản chỉ cần kết hợp nó vào một phương thức `.animate()` là đủ. Nhưng trong ví dụ này, phương thức `fade` sử dụng tốc độ là ‘fast’ trong khi đó di chuyển sang phải lại sử dụng tốc độ là ‘slow’. Đây chính là lúc chúng ta cần sử dụng đến dạng thứ 2 của phương thức `.method()`.

```
$(document).ready(function() {
  $('div.label').click(function() {
    var paraWidth = $('div.speech p').outerWidth();
    var $switcher = $(this).parent();
    var switcherWidth = $switcher.outerWidth();
    $switcher
      .fadeTo('fast',0.5)
      .animate({
        'left': paraWidth - switcherWidth
      }, {duration: 'slow', queue: false})
      .fadeTo('slow',1.0)
      .slideUp('slow')
      .slideDown('slow');
  });
});
```

Đối số thứ 2 ở đây là một hoạ đồ tùy chọn, cho ta tùy biến `queue`. Nếu `queue` có giá trị là `false` sẽ làm cho hiệu ứng động xảy ra cùng một lúc với cái trước nó.

Điều cuối bạn cần biết về dãy hiệu ứng trên một tập hợp các phần tử đơn là `queuing` không tự động gắn lên những phương thức phi hiệu ứng khác như `.css()`. Ví dụ chúng ta muốn thay đổi màu nền của thẻ `<div id='switcher'>` thành đỏ sau khi `.slideUp()` nhưng trước `.slideDown()`. Chúng ta có thể làm như sau:

```
$(document).ready(function() {
  $('div.label').click(function() {
    var paraWidth = $('div.speech p').outerWidth();
```

```

var $switcher = $(this).parent();
var switcherWidth = $switcher.outerWidth();
$switcher
    .fadeTo('fast',0.5)
    .animate({
        'left': paraWidth - switcherWidth
    }, 'slow')
    .fadeTo('slow',1.0)
    .slideUp('slow')
    .css('backgroundColor','f00')
    .slideDown('slow');
});
});

```

Tuy đoạn mã để thay đổi màu nền được đặt đúng thứ tự trong mã, nhưng nó lại thay đổi màu nền ngay khi bạn nhấp chuột.

Xem [Demo Online](#) – Example 7

Có một cách mà bạn có thể thêm phương thức phi hiệu ứng vào đây là bằng cách sử dụng phương thức `.queue()`. Đây sẽ là đoạn mã mà bạn có được.

```

$(document).ready(function() {
    $('div.label').click(function() {
        var paraWidth = $('div.speech p').outerWidth();
        var $switcher = $(this).parent();
        var switcherWidth = $switcher.outerWidth();
        $switcher
            .fadeTo('fast',0.5)
            .animate({
                'left': paraWidth - switcherWidth
            }, 'slow')
            .fadeTo('slow',1.0)
            .slideUp('slow')
            .queue(function() {
                $switcher
                    .css('backgroundColor', 'f00')
                    .dequeue();
            })
            .slideDown('slow');
    });
});

```

Xem [Demo Online](#) – Example 8

Ở đoạn mã trên, khi bạn cho phương thức `.queue()` một hàm hồi truy, nó sẽ thêm hàm đó vào dãy hiệu ứng của phần tử phù hợp. Ở trong hàm này, chúng ta đặt cho màu nền là màu đỏ và sau đó thì thêm vào phương thức hệ quả `.dequeue()`. Khi có sự xuất hiện của `.dequeue()`, nó cho phép dãy hiệu ứng tiếp tục nơi mà nó bị dừng lại và hoàn thành cả chuỗi hiệu ứng với dòng `.slideDown()`. Nếu chúng ta không sử dụng `.dequeue()`, thì hiệu ứng động đã dừng lại rồi.

Chúng ta sẽ tìm hiểu thêm một cách khác để xếp hàng những phương thức phi hiệu ứng. Trong phần tới chúng ta sẽ tìm hiểu hiệu ứng với đa hợp phần tử.

## Làm việc với đa hợp phần tử

Không giống như khi làm việc với nhóm phần tử đơn, khi chúng ta sử dụng hiệu ứng cho các nhóm khác nhau, chúng gần như xảy ra cùng một lúc. Để thấy những hiệu ứng xảy ra cùng một lúc diễn ra như thế nào, chúng ta sẽ di chuyển một đoạn văn xuống dưới đồng thời kéo đoạn văn khác lên. Đầu tiên chúng ta sẽ thêm vào hai thẻ `<p>` nữa.

```
<div id="wrapper">
  <div id="switcher">
    <div class="label">Cỡ chữ</div>
    <button id="switcher-default">Mặc Định</button>
    <button id="switcher-large">Lớn</button>
    <button id="switcher-small">Nhỏ</button>
  </div><!--End #switcher-->

  <div class="speech">
    <p>
      Pellentesque habitant morbi tristique senectus et netus et
      malesuada
      fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae,
      ultricies eget,
      tempor sit amet, ante. Donec eu libero sit amet quam egestas
      semper.
      Aenean ultricies mi vitae est.
    </p>
    <p>
      Pellentesque habitant morbi tristique senectus et netus et
      malesuada
      fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae,
      ultricies eget,
      tempor sit amet, ante. Donec eu libero sit amet quam egestas
      semper.
      Aenean ultricies mi vitae est.
    </p>

    <a href="#" class="more">Read More</a>
    <p>
      Pellentesque habitant morbi tristique senectus et netus et
      malesuada
      fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae,
      ultricies eget,
      tempor sit amet, ante. Donec eu libero sit amet quam egestas
      semper.
      Aenean ultricies mi vitae est.
    </p>
    <p>
      Pellentesque habitant morbi tristique senectus et netus et
      malesuada
      fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae,
      ultricies eget,
      tempor sit amet, ante. Donec eu libero sit amet quam egestas
      semper.
      Aenean ultricies mi vitae est.
    </p>
  </div>
</div>
```

Tiếp theo để cho dễ quan sát, chúng ta sẽ cho đoạn văn thứ 3 một đường viền 1 px và đoạn văn thứ 4 có màu nền xám. Chúng ta cũng sẽ ẩn đoạn văn thứ 4 khi DOM sẵn sàng.

```
$(document).ready(function() {
  $('p:eq(2)').css('border', '1px solid #333');
  $('p:eq(3)').css('backgroundColor', '#ccc').hide();
});
```

Cuối cùng chúng ta thêm phương thức `.click()` vào đoạn văn thứ 3 để khi nhấp chuột vào, thì đoạn văn thứ 3 sẽ chạy lên và ra khỏi tầm nhìn, trong khi đó đoạn văn thứ 4 sẽ chạy xuống và vào tầm nhìn.

```
$(document).ready(function() {
  $('p:eq(2)')
    .css('border', '1px solid #333')
    .click(function() {
      $(this).slideUp('slow')
      .next().slideDown('slow');
    });
  $('p:eq(3)').css('backgroundColor', '#ccc').hide();
});
```

Pellentesque habitant morbi tristique senectus et netus  
feugiat vitae, ultricies eget, tempor sit amet, ante. Donec  
vitae est.

Pellentesque habitant morbi tristique senectus et netus  
feugiat vitae, ultricies eget, tempor sit amet, ante. Donec  
vitae est.

[Read More](#)

Pellentesque habitant morbi tristique senectus et netus

Pellentesque habitant morbi tristique senectus et netus

Kết quả bạn đã thấy hai hiệu ứng xảy ra gần như cùng một lúc. Đoạn văn thứ 3 chạy lên trên đồng thời đoạn văn thứ 4 đang bị ẩn cũng lò rò chạy lên trên.

### Hàm hồi truy

Để có thể xếp hàng các hiệu ứng trên những phần tử khác nhau, jQuery cung cấp một hàm hồi truy cho mỗi phương thức hiệu ứng. Như chúng ta đã thấy với các bộ xử lý sự kiện và phương thức `.queue()`, hàm hồi truy chỉ đơn giản là đối số của một phương thức. Trong trường hợp hiệu ứng, chúng xuất hiện là đối số cuối cùng của phương thức.

Nếu chúng ta sử dụng hàm hồi truy để xếp hàng hai hiệu ứng trượt, chúng ta có thể cho đoạn văn thứ 4 trượt xuống trước khi đoạn văn thứ 3 trượt lên. Ở ví dụ dưới đây chúng ta sẽ xem xét cách thiết lập phương thức `.slideDown()` với hàm hồi truy.

```
$(document).ready(function() {
  $('p:eq(2)')
    .css('border', '1px solid #333')
    .click(function() {
      $(this).next().slideDown('slow', function() {
        // đoạn mã này sẽ chạy sau khi đoạn văn thứ 3 dừng lại
      });
    });
});
```

```
});
$('p:eq(3)').css('backgroundColor', '#ccc').hide();
});
```

Tuy nhiên chúng ta cũng cần phải thận trọng ở đây và phải biết chắc cái gì sẽ thực sự trượt lên. Ngữ cảnh cũng đã thay đổi với từ khoá \$(this) bởi vì hàm hồi truy đang nằm trong phương thức .slideDown(). Khi \$(this) được đặt trong hàm hồi truy thì nó không còn chỉ đoạn văn thứ 3 nữa như khi nó còn ở ngoài với phương thức .click(). Bởi vì phương thức .slideDown() được gán cho

\$(this).next(), tất cả những gì nằm trong phương thức này sẽ nhìn nhận \$(this) là anh em họ của nó hoặc nói khác đi chính là đoạn văn thứ 4. Cho nên nếu chúng ta thêm \$(this).slideUp('slow') vào trong hàm hồi truy, chúng ta sẽ kết thúc với việc cho ẩn đi đoạn văn mà chúng ta vừa cho hiện lên.

Một cách đơn giản để giữ \$(this) không bị thay đổi là lưu nó ngay vào một biến trong phương thức .click() như sau: var \$thirdPara = \$(this). Bây giờ biến \$thirdPara chính là đoạn văn thứ 3, ở cả trong và ngoài hàm hồi truy. Đây là đoạn mã sử dụng biến vừa tạo của chúng ta.

```
$(document).ready(function() {
var $thirdPara = $('p:eq(2)');
$thirdPara
.css('border', '1px solid #333')
.click(function() {
$(this).next().slideDown('slow', function() {
$thirdPara.slideUp('slow');
});
});
$('p:eq(3)').css('backgroundColor', '#ccc').hide();
});
```

Sử dụng biến \$thirdPara ở trong hàm hồi truy .slideDown() phụ thuộc vào thuộc tính của **Closures**. Chúng ta sẽ bàn thêm về thuộc tính rất quan trọng nhưng hơi khó hiểu này trong phần sau.

Khi bạn cho chạy đoạn mã trên bạn sẽ thấy cả hai đoạn văn đều xuất hiện, khi đoạn văn thứ 4 đã trượt xuống hết thì đoạn văn thứ 3 mới bắt đầu trượt lên.

Pellentesque habitant morbi tristique senectus et netus et n  
feugiat vitae, ultricies eget, tempor sit amet, ante. Donec e  
vitae est.

Pellentesque habitant morbi tristique senectus et netus et n  
feugiat vitae, ultricies eget, tempor sit amet, ante. Donec e  
vitae est.

[Read More](#)

Pellentesque habitant morbi tristique senectus et netus et n  
quam, feugiat vitae, ultricies eget, tempor sit amet, ante. D  
ultricies mi vitae est.

Pellentesque habitant morbi tristique senectus et netus et n  
feugiat vitae, ultricies eget, tempor sit amet, ante. Donec e  
vitae est.

Bây giờ bạn đã hiểu được hàm hồi truy, chúng ta có thể trở lại đoạn mã ở trên trong chương này. Đoạn mã chúng ta xếp hàng cho hiệu ứng đổi màu nền ở gần cuối của dãy hiệu ứng. Thay vì chúng ta sử dụng phương thức `.queue()` như trước đây, chúng ta có thể sử dụng hàm hồi truy để thay thế.

```
$(document).ready(function() {
  $('div.label').click(function() {
    var paraWidth = $('div.speech p').outerWidth();
    var $switcher = $(this).parent();
    var switcherWidth = $switcher.outerWidth();
    $switcher
      .fadeTo('slow', 0.5)
      .animate({
        'left': paraWidth - switcherWidth
      }, 'slow')
      .fadeTo('slow', 1.0)
      .slideUp('slow', function() {
        $switcher
          .css('backgroundColor', '#f00');
      })
      .slideDown('slow');
  });
});
```

Khi cho chạy đoạn mã bạn sẽ thấy màu nền của thẻ `<div id='switcher'>` đổi thành màu đỏ sau khi nó đã trượt lên và trước khi nó trượt xuống.

#### Tóm lược

Bằng cách sử dụng những phương thức hiệu ứng chúng ta vừa học ở chương này, chúng ta có thể tăng hoặc giảm dần cỡ chữ bằng cách sử dụng `.css()` hoặc `.animate()`. Chúng ta cũng có thể áp dụng nhiều hiệu ứng để ẩn hoặc hiện một phần tử bất kỳ bằng nhiều cách và tạo hiệu ứng động cho nó xảy ra cùng một lúc hoặc theo thứ tự.

Ở 4 chương đầu này, tất cả những ví dụ của chúng ta đều là dạng sửa đổi những phần tử đã được viết mã HTML trước. Trong chương 5, chúng ta sẽ học cách sử dụng jQuery để tạo ra các phần tử mới và thêm chúng vào cây DOM bất cứ khi nào ta muốn.

## Chương 6 – Sửa đổi DOM

Bạn đã từng xem ảo thuật và thấy những ảo thuật gia có thể với tay lên không trung và cho xuất hiện một bó hoa, jQuery cũng có thể tạo ra các thành phần, thuộc tính, và cả chữ trên một trang web giống với cách mà ảo thuật gia trình diễn vậy. Hơn nữa, jQuery cũng có thể làm biến mất tất cả những thứ nó tạo ra. Và chúng ta cũng có thể lấy bó hoa kia và biến nó thành `<div class='magic' id='flowerToDove'>Dove</div>`

#### Sửa đổi thuộc tính

Qua 4 chương đầu của giáo trình này, chúng ta đã biết cách sử dụng phương thức `.addClass()` và `.removeClass()` để làm thay đổi giao diện của các thành phần trên trang web. Thực chất thì những phương thức này sửa đổi thuộc tính của class. Phương thức `.addClass()` thì tạo ra hoặc thêm vào cho thuộc tính, trong khi phương thức `.removeClass()` thì lại xóa hoặc giảm thuộc



tính. Còn có một phương thức nữa là `.toggleClass()`, nó có thể vừa loại bỏ và vừa thêm vào một class. Như thế với 3 phương thức trên chúng ta đã có những công cụ đủ mạnh để làm việc với class.

Tuy nhiên, thuộc tính class chỉ là một trong số những thuộc tính mà chúng ta cần dùng tới hoặc thay đổi. Ví dụ, id, rel và href. Để sửa đổi những thuộc tính này, jQuery cung cấp phương thức `.attr()` và `.removeAttr()`. Chúng ta thậm chí có thể sử dụng `.attr()` và `.removeAttr()` để sửa đổi thuộc tính class. Nhưng phương thức chuyên dụng là `.addClass()` và `.removeClass()` thì phù hợp hơn trong trường hợp này bởi vì nó có thể xử lý chính xác những trường hợp một phần tử có nhiều class như: `<div class='first second'>`.

### Thuộc tính phi class

Có những thuộc tính không đơn giản để sửa đổi nếu không có sự trợ giúp của jQuery. Hơn nữa, jQuery cho phép chúng ta sửa đổi nhiều thuộc tính cùng một lúc, tương tự như cách mà chúng ta làm việc với nhiều thuộc tính CSS khi sử dụng phương thức `.css()` ở chương 4.

Ở ví dụ này, chúng ta có thể dễ dàng thiết lập id, rel và thuộc tính title cho đường liên kết cùng một lúc. Dưới đây là mã HTML

```
<h1>jQuery DOM Manipulation</h1>
  <h3>An example at izwebz.com</h3>
  <div class="chapter">
    <p class="first">Qua 4 chương đầu của giáo trình này, chúng ta đã
biết cách sử
    dụng phương thức .addClass() và .removeClass() để làm thay đổi giao
diện của
    các thành phần trên trang web. <a href="http://www.izwebz.com">Thực
chất</a> thì những phương thức này sửa đổi thuộc tính của class. Phương
thức .addClass() thì tạo ra hoặc thêm vào cho
    thuộc tính, trong khi phương thức .removeClass() thì lại xoá hoặc
giảm thuộc
    tính.
  </p>

  <p class="second method">Tuy nhiên, thuộc tính class chỉ là một
trong số
    những thuộc tính mà chúng ta cần dùng tới hoặc thay đổi. Ví dụ, id,
rel và href.
    Để sửa đổi những thuộc tính này, <a
href="http://www.izwebz.com">jQuery</a> cung cấp phương thức .attr()
    và .removeAttr(). Chúng ta thậm chí có thể sử dụng .attr() và
.removeAttr()
    để sửa đổi <a href="http://www.izwebz.com">thuộc tính</a> class.
Nhưng phương thức chuyên dụng là .addClass()
    và .removeClass() thì phù hợp hơn trong trường hợp này bởi vì nó
<a href="http://www.izwebz.com">có thể</a> xử lý
    chính xác những trường hợp một phần tử có nhiều class như:
  </p>

  <p><span class="pull-quote">
    Có những thuộc tính <span class="drop">không đơn giản</span> để sửa
đổi nếu không có sự trợ giúp của
    jQuery. Hơn nữa, <strong>jQuery</strong> cho phép chúng ta sửa đổi
nhiều thuộc tính cùng một lúc,
    tương tự như cách mà chúng ta làm việc với nhiều thuộc tính
CSS</span> khi sử dụng
```

phương thức `.css()` ở chương 4. Ở ví dụ này, chúng ta có thể dễ dàng thiết lập `id`, `rel` và thuộc tính `title` cho đường liên kết cùng một lúc. Dưới đây là mã HTML

```
</p>
</div><!--End .chapter-->
```

Bây giờ chúng ta có thể đi qua từng đường liên kết trong thẻ `<div class='chapter'>` và áp dụng thuộc tính cho chúng từng thẻ một. Nếu bạn chỉ muốn tạo ra một giá trị thuộc tính giống nhau cho tất cả các đường liên kết, thì bạn chỉ cần một dòng mã đơn giản sau:

```
$(document).ready(function() {
$('div.chapter a').attr({'rel': 'external'});
});
```

Cách này có thể dùng được bởi vì chúng ta muốn giá trị của thuộc tính `rel` vừa tạo là như nhau ở tất cả các đường liên kết. Tuy nhiên, thường thì những thuộc tính ta thêm vào hoặc thay đổi phải có giá trị khác nhau cho mỗi một thành phần. Ví dụ với bất cứ tài liệu nào, mỗi một `id` đều phải là duy nhất nếu ta muốn mã JavaScript của mình làm việc theo ý muốn. Để tạo được một `id` duy nhất cho mỗi đường liên kết, chúng ta không sử dụng phương pháp ở trên nữa mà thay vào đó sử dụng phương thức `.each()`.

```
$(document).ready(function() {
$('div.chapter a').each(function(index) {
$(this).attr({
'rel': 'external',
'id': 'izwebz-' + index
});
});
});
```

Phương thức `.each()` hoạt động như vòng lặp hiện, nó có nguyên lý hoạt động như vòng lặp `for` nhưng thuận tiện hơn. Người ta thường sử dụng phương thức này khi mà đoạn mã chúng ta sử dụng trên mỗi phần tử của bộ chọn quá phức tạp cho vòng lặp `for`. Trong trường hợp này, hàm ẩn của phương thức `.each()` được gán một số `index` để chúng ta có thể gán nó cho mỗi `id`. Đối số `index` này hoạt động như một bộ đếm, bắt đầu từ số 0 cho đường liên kết đầu tiên và tăng dần 1 đơn vị cho mỗi đường liên kết kế tiếp. Cho nên khi ta thiết lập `id` thành `'izwebz-' + index`, thì đường liên kết đầu tiên sẽ có `id` là `izwebz-0`, đường liên kết thứ 2 sẽ là `izwebz-1`, v.v..

Xem [Demo Online](#) – Example 1 (dùng firebug để inspect link)

Chúng ta sẽ sử dụng thuộc tính `title` để cho người đọc biết thêm thông tin về đường liên kết ở Izwebz. Ở ví dụ dưới đây, tất cả các đường liên kết đều hướng tới `izwebz.com`. Tuy nhiên, chúng ta nên để cho biểu thức bộ chọn được cụ thể hơn, chúng ta chỉ nên chọn những đường liên kết có chứa `izwebz` trong phần `href`. Để phòng sau này chúng ta lại thêm những đường liên kết khác không phải là `izwebz`.

```
$(document).ready(function() {
$('div.chapter a[href*=izwebz]').each(function(index) {
var $thisLink = $(this);
$thisLink.attr({
```

```

        'rel': 'external',
        'id': 'izwebzLink-' + index,
        'title': 'know more about ' + $thisLink.text() + ' at
izwebz'
    });
    });
});

```

Ở đây có điểm bạn cần chú ý là chúng ta đã lưu lại từ khoá \$(this) vào một biến gọi là \$thisLink, bởi vì chúng ta sử dụng nó nhiều hơn một lần.

Với cả 3 giá trị thuộc tính được thiết lập như trên, bây giờ đường liên kết của chúng ta sẽ có dạng như sau:

```

<a href="http://www.izwebz.com" rel="external" id="izwebzLink-0"
title="know more about Thực chất at izwebz">Thực chất</a>

```

Xem [Demo Online](#) – Example 2 (dùng firebug để inspect link)

Ôn lại hàm \$()

Từ khi bắt đầu làm quen với jQuery, chúng ta đã biết cách sử dụng hàm \$() để tiếp cận các thành phần trên trang. Thực tế thì hàm này là trọng tâm của thư viện jQuery, bởi vì nó được gọi mỗi khi chúng ta cần gán một hiệu ứng, sự kiện hoặc thuộc tính cho một phần tử.

Nhưng hàm \$() còn một chức năng khác nữa nằm trong hai dấu ngoặc – tính năng này rất đối mạnh mẽ đến nỗi nó không những có thể thay đổi giao diện của một thành phần mà nó còn có thể thay đổi nội dung của một trang web. Chỉ đơn giản bằng cách chèn một đoạn mã HTML nằm giữa hai dấu ngoặc, chúng ta có thể tạo ra một cấu trúc DOM mới từ hư vô.

**Chú ý:** *Bạn cũng nên chú ý khi tạo ra những hiệu ứng để cải thiện giao diện hoặc nội dung phụ thuộc vào JavaScript. Bởi vì không phải ai cũng bật JavaScript, nên những thông tin quan trọng phải được nhìn thấy bởi tất cả mọi người, chứ không phải chỉ nhóm người có trình duyệt hiện đại hoặc bật JavaScript.*

Một chức năng thường thấy trong những trang FAQs là đường liên kết **Back to top** ở dưới mỗi câu hỏi và trả lời. Bởi vì cái này nếu có bỏ đi hoặc không được hiển thị ở một số trình duyệt thì cũng không ảnh hưởng đến nội dung chính của trang. Do vậy chúng ta có thể dùng JavaScript để thêm vào. Chúng ta sẽ thêm vào đường liên kết **Back to top** ở cuối mỗi đoạn văn, và điểm dừng mà đường liên kết **Back to top** sẽ dẫn tới. Chúng ta tạo ra các thành phần mới như sau:

```

$(document).ready(function() {
$('

```

Khi cho chạy thử đoạn mã trên, bạn vẫn không thấy những đường liên kết back to top và các điểm dừng xuất hiện, cho dù ta đã tạo nó ở đoạn code trên. Vấn đề là dòng mã ở trên đã tạo ra các thành phần ta muốn, nhưng nó chưa được thêm vào trang. Để làm được điều này, chúng ta có thể sử dụng một trong rất nhiều phương thức chèn của jQuery.

## Chèn các thành phần mới

jQuery có hai phương thức dùng để chèn phần tử này vào trước phần tử kia là: `.insertBefore()` và `.before()`. Hai phương thức này có cùng chức năng, nhưng khác nhau ở điểm là nó sẽ được kết hợp với các phương thức khác như thế nào. Còn hai phương pháp nữa là, `.insertAfter()` và `.after()`, cũng có nguyên lý hoạt động như nhau nhưng nó được sử dụng để chèn phần tử này vào sau phần tử kia. Với ví dụ về back to top của ta, chúng ta sẽ sử dụng phương pháp `.insertAfter()`.

```
$(document).ready(function() {  
  $('<a href="#top">back to top</a>')  
    .insertAfter('div.chapter p');  
  $('<a id="top"></a>');  
});
```

Phương thức `.after()` cũng có thể cho kết quả tương tự với `.insertAfter()`, nhưng với biểu thức bộ chọn nằm trước phương thức thay vì theo sau nó. Nếu sử dụng `.after()`, thì dòng mã đầu tiên trong `$(document).ready()` sẽ là như sau:

```
$('div.chapter').after('<a href="#top">back to top</a>');
```

Với `.insertAfter()` thì bạn vẫn có thể thêm vào đằng sau nó những phương thức khác để tiếp tục làm việc với thẻ `<a>`. Nhưng với `.after()`, những phương thức bạn thêm vào sau này sẽ chỉ có tác dụng với những phần tử phù hợp với bộ chọn – trong trường hợp này là – `$(div.chapter p)`. Nói cách khác, thẻ `<a>` của bạn sẽ không chịu ảnh hưởng bởi những phương thức thêm vào sau nó.

Bây giờ chúng ta đã chính thức chèn đường liên kết vào trang web (và vào trong DOM) sau mỗi một đoạn văn nằm trong thẻ `<div class='chapter'>`, đường liên kết back to top sẽ xuất hiện như hình.

Qua 4 chương đầu của giáo trình này, chúng ta đã biết cách sử dụng phương tiện web. Thực chất thì những phương thức này sửa đổi thuộc tính của class. Phương thức `.removeClass()` thì lại xóa hoặc giảm thuộc tính.

back to top

Tuy nhiên, thuộc tính class chỉ là một trong số những thuộc tính mà chúng ta cần cập nhật. Phương thức `.attr()` và `.removeAttr()`. Chúng ta thậm chí có thể sử dụng `.addClass()` và `.removeClass()` thì phù hợp hơn trong trường hợp này bởi vì

back to top

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac t  
Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. N  
erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.

back to top

Nhưng hiện tại những đường liên kết vẫn chưa hoạt động được. Chúng ta vẫn còn phải chèn điểm dừng với id='top'. Chúng ta có thể sử dụng một trong những phương thức dùng để chèn một phần tử vào một phần tử khác.

```
$(document).ready(function() {  
    $('<a href="#top">back to top</a>')  
    .insertAfter('div.chapter p');  
    $('<a id="top" name="top"></a>')  
    .prependTo('body');  
});
```

Đoạn mã trên chèn điểm dừng ngay trên phần bắt đầu của thẻ <body>, hay nói cách khác là trên cùng của trang. Với phương thức .insertAfter() cho đường liên kết và .prependTo() được sử dụng cho điểm dừng, những đường liên kết back to top của chúng ta đã hoạt động được.

Một điểm thường thấy nữa của những đường liên kết back to top là nó không có tác dụng gì khi nằm trên cùng của trang vì phần đầu người đọc vẫn nhìn thấy được. Chúng ta cần chỉnh sửa lại mã một chút sao cho những đường liên kết chỉ bắt đầu sau đoạn văn thứ 4. Để đạt được điều này, chúng ta chỉ cần thay đổi biểu thức bộ chọn một chút:

.insertAfter('div.chapter:gt(2)'). Tại sao lại có giá trị là 2 ở đây? Bởi vì JavaScript đánh số bắt đầu từ 0, cho nên đoạn văn đầu tiên sẽ là số 0, đoạn văn thứ 2 là số 1, thứ 3 là số 2 và thứ tư là số 3. Biểu thức bộ chọn của chúng ta sẽ chèn đường liên kết vào sau mỗi đoạn văn khi mà giá trị chỉ mục là 3, bởi vì nó là số đầu tiên lớn hơn 2.

Hình dưới đây cho bạn thấy kết quả của biểu thức bộ chọn ở trên.

Qua 4 chương đầu của giáo trình này, chúng ta đã biết cách sử dụng jQuery để thao tác với DOM. Thực chất thì những phương thức này sửa đổi thuộc tính của các phần tử. Ví dụ, phương thức .removeClass() thì lại xóa hoặc giảm thuộc tính.

Tuy nhiên, thuộc tính class chỉ là một trong số những thuộc tính mà chúng ta có thể thao tác. jQuery cung cấp thêm các phương thức .attr() và .removeAttr(). Chúng ta thậm chí có thể sử dụng .addClass() và .removeClass() thì phù hợp hơn trong trường hợp này.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac luctus et viverra. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae erat. wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.

Có những thuộc tính không đơn giản để sửa đổi nếu không có sự trợ giúp của jQuery. Ví dụ, chúng ta làm việc với nhiều thuộc tính CSS khi sử dụng jQuery để thao tác với DOM cùng một lúc. Dưới đây là mã HTML:

[back to top](#)

Xem [Demo Online](#) – Example 3

Di chuyển các phần tử

Với ví dụ về đường liên kết back to top ở trên, chúng ta đã tạo ra những phần tử mới và chèn chúng vào trang. Nhưng chúng ta cũng có thể di chuyển một phần tử từ nơi này qua nơi khác.

Một ứng dụng thực tế của cách chèn này là dạng tự động hoá cách chèn phần ghi chú ở cuối trang. Một phần chú thích đã có trong đoạn văn mẫu và chúng ta sẽ sử dụng nó trong ví dụ này. Chúng ta cũng đã tạo ra một số phần ghi chú khác cho ví dụ này.

```
<p><span class="pull-quote">
    Có những thuộc tính <span class="drop">không đơn giản</span> để sửa
đổi nếu không có sự trợ giúp của
    jQuery. Hơn nữa, <strong>jQuery</strong> cho phép chúng ta</span>
sửa đổi nhiều thuộc tính cùng một lúc,
    tương tự như cách mà chúng ta làm việc với nhiều thuộc tính CSS
khi sử dụng
    phương thức .css() ở chương 4. Ở ví dụ này, chúng ta có thể dễ
dàng thiết lập
    id, rel và thuộc tính title cho đường liên kết cùng một lúc. Dưới
đây là mã
    HTML
</p>

<p>
Pellentesque habitant morbi tristique senectus et netus et
malesuada fames
    ac turpis egestas. <span class="footnote">Vestibulum tortor quam,
feugiat vitae, ultricies eget,
    tempor sit amet, ante. Donec eu libero sit amet quam egestas
semper.
    Aenean ultricies mi vitae est</span>. Mauris placerat eleifend
leo. Quisque
    sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi,
condimentum sed, commodo vitae, ornare sit amet, wisi.
</p>

<p>
Pellentesque habitant morbi tristique senectus et netus et
malesuada fames
    ac turpis egestas. Vestibulum tortor quam, feugiat vitae,
ultricies eget,
    tempor sit amet, ante. <span class="footnote">Donec eu libero sit
amet quam egestas semper.
    Aenean ultricies mi vitae est. Mauris placerat eleifend leo.
Quisque
    sit amet est et sapien ullamcorper pharetra.</span> Vestibulum
erat wisi,
    condimentum sed, commodo vitae, ornare sit amet, wisi.
</p>

<p>
Pellentesque habitant morbi tristique senectus et netus et
malesuada fames
    ac turpis egestas. Vestibulum tortor quam, feugiat vitae,
ultricies eget,
    tempor sit amet, ante. Donec eu libero sit amet quam egestas
semper.
    Aenean ultricies mi vitae est. <span class="footnote">Mauris
placerat eleifend leo. Quisque
    sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi,
condimentum sed, commodo vitae, ornare sit amet, wisi.</span>
</p>
```

Mỗi một đoạn văn này có một đoạn ghi chú được gói trong thẻ `<span class='footnote'></span>`. Khi viết mã HTML như thế này, chúng ta đã bảo tồn được ngữ

cảnh của ghi chú. Sau đó trong CSS stylesheet, chúng ta cho nó in nghiêng và được hình dưới đây.

Bây giờ chúng ta có thể lấy phần ghi chú và chèn chúng vào giữa hai thẻ `<div class='chapter'>` và `<div id='footer'>`. Bạn cũng nên biết rằng kể cả trong trường hợp của vòng lặp ẩn, thứ tự chèn đã được định trước, bắt đầu từ trên cây DOM đi xuống dưới. Bởi vì chúng ta cũng cần bảo tồn thứ tự của phần ghi chú ở vị trí mới của nó trên trang, cho nên chúng ta sẽ dùng `.insertBefore('#footer')`.

Làm như vậy sẽ chèn từng phần ghi chú ngay trên thẻ `<div id='footer'>`. Do đó ghi chú thứ nhất sẽ được đặt nằm giữa `<div class='chapter'>` và `<div id='footer'>`, ghi chú thứ 2 sẽ được nằm giữa ghi chú thứ nhất và thẻ `<div id='footer'>`, v.v.. Mặt khác, nếu chúng ta sử dụng `.insertAfter('div.chapter')`, thì thứ tự sẽ bị đảo lộn. Cho nên mã của chúng ta sẽ như sau

```
$(document).ready(function() {  
    $('span.footnote').insertBefore('#footer');  
});
```

Ở đây chúng ta lại có thêm một vấn đề nữa. Đó là các dòng ghi chú được nằm trong thẻ `<span>`, mà bản thân thẻ `<span>` có `display: inline` theo mặc định, do vậy các dòng ghi chú nối liền nhau mà không xuống dòng.

Chúng ta sẽ chỉnh sửa CSS để giải quyết vấn đề này, chúng ta sẽ làm cho những phần tử `<span>` trở thành block-level, nhưng chỉ áp dụng với những thẻ nằm trong `<div class='chapter'>`

```
span.footnote {  
    font-style: italic;  
    font-family: "Times New Roman", Times, serif;  
    display: block;  
    margin: 1em 0;  
}  
.chapter span.footnote {  
    display: inline;  
}
```

Bây giờ thì phần ghi chú của chúng ta đã xuống dòng.

*Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis eges  
Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.*

*Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis eges  
Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est.*

*Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec*

*Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris*

*Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra.*

Xem [Demo Online](#) – Example 4



Ít ra thì phần ghi chú của chúng ta nhìn cũng đã tạm được rồi, nhưng còn nhiều việc chúng ta có thể làm để cải thiện nó. Một số điểm cần làm như sau:

1. Đánh dấu vị trí trong tài liệu nơi mà ghi chú được sử dụng
2. Đánh số cho từng vị trí và cung cấp một số phù hợp với bản thân từng dòng ghi chú.
3. Tạo đường liên kết giữa vị trí văn bản đến điểm ghi chú, và từ điểm ghi chú ngược lại đoạn văn bản.

Những bước này có thể làm được nhờ phương thức `.each()`, nhưng trước hết chúng ta phải tạo ra một nơi chứa những dòng ghi chú ở dưới cùng của trang.

```
$(document).ready(function() {  
  $('<ol id="notes"></ol>').insertAfter('div.chapter');  
});
```

Chúng ta sử dụng một danh sách đánh số `<ol id='notes'> </ol>` cho phần ghi chú, bởi vì mình muốn chúng tự động được đánh số thứ tự. Chúng ta đã cho danh sách này một `id='notes'` và chèn nó vào sau thẻ `<div class='chapter'>`.

### Đánh dấu, đánh số và liên kết văn bản

Bây giờ chúng ta đã có thể đánh dấu và đánh số vị trí mà phần ghi chú được trích dẫn.

```
$(document).ready(function() {  
  $('<ol id="notes"></ol>').insertAfter('div.chapter');  
  $('span.footnote').each(function(index) {  
    $(this)  
      .before(  
        ['<a href="#foot-note-',  
          index+1,  
          ' " id="context-',  
          index+1,  
          ' " class="context">',  
          '<sup>' + (index+1) + '</sup>',  
          '</a>'  
        ].join('')  
      )  
    });  
  });
```

Ở đây chúng ta sử dụng bộ chọn giống với bộ chọn ở ví dụ trước, nhưng khác cái là chúng ta gán phương thức `.each()` cho nó. Ở trong `.each()` chúng ta bắt đầu bằng `$(this)`, nó là đại diện của từng dòng ghi chú liên tiếp nhau, và chúng ta gán phương thức `.method()` cho nó.

Kết quả của cả đoạn mã “loằng ngoằng” nằm trong dấu ngoặc đơn của phương thức `.before()` sẽ là một đường siêu liên kết. Nó sẽ được chèn vào trước từng thẻ `<span>` của phần ghi chú. Dưới đây là mã HTML của một trong những dòng ghi chú khi nó được chèn vào DOM.

```
<a href="#foot-note-1" id="context-1"  
class="context"><sup>1</sup></a>
```

Cú pháp nhìn có vẻ khá phức tạp, nên chúng ta dành vài phút để tìm hiểu xem nó như thế nào. Bên trong dấu ngoặc đơn của phương thức `.before()`, chúng ta bắt đầu với một cặp ngoặc vuông `[ ]` – nó chính là đại diện của mảng trực kiện (array literal). Mỗi phần tử nằm trong array sẽ có một dấu phẩy theo sau (trừ phần tử cuối cùng, nếu không mã sẽ không chạy). Để cho dễ đọc hơn, chúng ta đã viết mỗi lệnh trên một dòng. Sau khi mảng đã lập xong, chúng ta



chuyển nó lại thành dạng chuỗi bằng cách sử dụng phương thức của JavaScript là `.join()`. Phương thức này lấy vào một chuỗi rỗng làm đối số, chuỗi rỗng được biểu thị bởi một cặp dấu nháy (‘ ‘), bởi vì chúng ta không muốn bất cứ thứ gì xuất hiện giữa các phần tử mảng khi nó được xuất ra dạng HTML.

Lưu ý bạn đến phần `index+1` trên đoạn mã trên. Bởi vì JavaScript đánh số bắt đầu từ 0, do vậy để thuộc tính `href` có giá trị là `#footnote-1`, thì chúng ta phải cộng 1 đơn vị cho nó. Thuộc tính `id` là `#context-1` và tên của đường link sẽ là 1. Thuộc tính `href` đặc biệt quan trọng, bởi vì nó phải tuyệt đối phù hợp với thuộc tính `id` của phần ghi chú (tất nhiên là không gồm dấu #).

Cách thứ 2 chúng ta cũng có thể sử dụng chuỗi nối thay vì dùng mảng ghép:

```
.before('<a href="#foot-note-' + (index+1) +  
'" id="context-' + (index+1) +  
'" class="context"><sup>' +  
(index+1) + '</sup></a>');
```

Nhưng trong trường hợp này, cách sử dụng mảng có vẻ dễ quản lý hơn.

**Lưu ý:** Đã có nhiều tài liệu nói về sự khác nhau về mặt hiệu năng làm việc giữa mảng ghép và chuỗi nối. Nếu bạn còn phân vân, bạn có thể đọc tài liệu sau <http://www.sitepen.com/blog/2008/05/09/string-performance-an-analysis/>

Tuy nhiên, trong hầu hết các trường hợp, những khác biệt này là không đáng kể. Nếu hiệu năng làm việc của đoạn mã là điều cần quan tâm thì còn nhiều yếu tố khác có tầm ảnh hưởng còn lớn hơn như là cách lưu bộ chọn mà chúng ta đã bàn.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. <sup>1</sup> eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condim vitae, ornare sit amet, wisi.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. V feugiat vitae, ultricies eget, tempor sit amet, ante. <sup>2</sup> Vestibulum erat wisi, condimentum sed, coi sit amet, wisi.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. V feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper vitae est. <sup>3</sup>

Xem [Demo Online](#) – example 5

Gắn phần ghi chú

Bước kế tiếp là di chuyển phần tử `<span class='footnote'>` như chúng ta đã làm. Tuy nhiên, lần này chúng ta sẽ đặt nó vào trong thẻ `<ol id='note'>` vừa mới được tạo. Chúng ta sẽ sử dụng `.appendTo()`, nhưng cũng để giữ đúng thứ tự, những dòng ghi chú kế tiếp sẽ được chèn vào cuối của hàng.

```
$(document).ready(function() {
$(<ol id="notes"></ol>').insertAfter('div.chapter');
$('span.footnote').each(function(index) {
$(this)
.before(
['<a href="#foot-note-',
index+1,
'" id="context-',
index+1,
'" class="context">',
'<sup>' + (index+1) + '</sup>',
'</a>']
).join('')
)
.appendTo('#notes')
});
});
```

Bạn cũng nên nhớ rằng `.appendTo()` đang được gắn với `$(this)`, cho nên nói theo ngôn ngữ của jQuery thì là “gắn phần ghi chú span vào phần tử với id là ‘notes’”.

Đối với mỗi dòng ghi chú mà chúng ta vừa di chuyển, chúng ta sẽ gắn cho nó một đường liên kết khác. Đường liên kết này sẽ quay lại số thứ tự nằm trong văn bản.

```
$(document).ready(function() {
$(<ol id="notes"></ol>').insertAfter('div.chapter');
$('span.footnote').each(function(index) {
$(this)
.before(
['<a href="#foot-note-',
index+1,
'" id="context-',
index+1,
'" class="context">',
'<sup>' + (index+1) + '</sup>',
'</a>']
).join('')
)
.appendTo('#notes')
.append( ' &nbsp; (<a href="#context-' + (index+1) +
'">context</a>)' );
});
});
```

Chú ý đến thuộc tính href quay ngược lại id phù hợp với chỗ đánh dấu trước. Dưới đây bạn sẽ thấy phần ghi chú đã được gắn đường liên kết.

*Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, c  
semper. Aenean ultricies mi vitae est (context)* ←

*Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vit  
amet est et sapien ullamcorper pharetra. (context)* ←

*Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorpe  
sed, commodo vitae, ornare sit amet, wisi. (context)* ←

Tuy nhiên phần ghi chú vẫn chưa có số thứ tự, mặc dù chúng đã được đặt trong thẻ <ol>, là bởi vì mỗi một dòng sẽ phải được đặt nằm giữa thẻ <li>.

### Gói phần tử

Phương thức dùng để gói một phần tử này bên trong phần tử khác có tên là .wrap(). Bởi vì chúng ta muốn mỗi một \$(this) sẽ được gói trong cặp thẻ <li></li>, chúng ta có thể hoàn thiện mã cho phần ghi chú như sau:

```
$(document).ready(function() {
$(<ol id="notes"></ol>`).insertAfter('div.chapter');
$('span.footnote').each(function(index) {
$(this)
.before(
['<a href="#foot-note-',
index+1,
' " id="context-',
index+1,
' " class="context">',
'<sup>' + (index+1) + '</sup>',
'</a>']
).join('')
)
.appendTo('#notes')
.append( ' &nbsp; (<a href="#context-' + (index+1) +
'">context</a>)' )
.wrap('<li id="foot-note-' + (index+1) +
'"></li>');
});
});
```

Bây giờ mỗi phần tử <li> đều có một id bằng với href của nó. Cuối cùng chúng ta đã có một danh sách các đoạn ghi chú được đánh dấu và liên kết với nhau.

1. *Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet semper. Aenean ultricies mi vitae est (context)*
2. *Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae amet est et sapien ullamcorper pharetra. (context)*
3. *Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper, commodo vitae, ornare sit amet, wisi. (context)*

© 2010 - izwebz.com

Xem [Demo Online](#) – Example 6

Tất nhiên, số thứ tự có thể được chèn vào trước từng dòng ghi chú như cách mà chúng được chèn trong đoạn văn. Tuy nhiên bạn có thể thấy rất “khoái chí” khi chúng ta có những đoạn mã hợp chuẩn được tự động tạo ra bởi JavaScript.

## Sao chép phần tử

Từ đầu chương cho tới giờ chúng ta đã chèn phần tử được tạo ra, di chuyển nó từ vị trí này sang vị trí khác của tài liệu, gói phần tử đã có bằng một phần tử mới. Nhưng cũng có khi chúng ta muốn sao chép một phần tử nào đó. Ví dụ khi bạn có một thanh di chuyển ở trên cùng của trang, bạn muốn copy nó và chèn nó xuống phần footer. Thực ra nếu bạn muốn copy lại phần nào của trang để tăng tính thẩm mỹ của nó thì bạn nên sử dụng mã để cho nó làm tự động cho mình. Mà suy cho cùng thì tại sao chúng ta phải mất thời gian và công sức để viết lại một đoạn mã y chang làm gì? Tại sao không để jQuery làm giúp vừa lẹ mà lại vừa tránh sai sót.

Để sao chép phần tử, phương thức `.clone()` là tất cả những gì chúng ta cần. Phương thức này sẽ tạo ra một bản sao của phần tử mà chúng ta chọn để dùng sau này. Cũng như những phương thức tạo ra các phần tử mới mà chúng ta đã học ở trên, phương thức này cũng không tạo ra thay đổi gì cho đến khi chúng ta sử dụng một trong những phương thức chèn. Cũng giống như khi bạn copy một đoạn văn trong Microsoft Word, khi bạn vừa bôi đen và chọn copy. Thì MS Word sẽ lưu đoạn văn đó trong bộ nhớ của máy, nhưng trên tài liệu chưa có gì xuất hiện. Cho đến khi bạn chọn Paste thì đoạn văn copy mới được hiển thị. Phương thức `.clone()` cũng hoạt động theo nguyên lý đó.

Đoạn mã dưới đây sẽ tạo ra một bản sao của đoạn văn nằm trong thẻ `<div class='chapter'>`.

```
$('div.chapter p:eq(0)').clone();
```

Như đã nói ở trên, khi bạn cho chạy đoạn code trên sẽ không có gì xảy ra hết. Bởi vì jQuery mới chỉ copy đoạn văn đó thôi chứ nó chưa làm gì với nó cả.

Để cho đoạn văn chúng ta vừa copy xuất hiện, chúng ta sẽ cho nó xuất hiện ở trên thẻ `<div class='chapter'>`

```
$('div.chapter p:eq(0)').clone().insertBefore('div.chapter');
```

Bây giờ nếu cho chạy đoạn mã trên, bạn sẽ thấy đoạn văn đầu được xuất hiện 2 lần và bởi vì nó không còn nằm trong thẻ `<div class='chapter'>` nữa cho nên những style nào bạn áp dụng trong CSS sẽ không có tác dụng với nó. Ở đây bạn thấy rõ nhất là nó rộng hơn những đoạn văn còn lại.

## Sao chép kèm sự kiện

Mặc định của phương thức `.clone()` là không sao chép bất cứ sự kiện nào hoặc “họ hàng nội ngoại” của đối tượng được chọn. Nhưng nó cũng có thể lấy vào một tham số Boolean, mà khi giá trị này là true, nó sẽ sao chép cả sự kiện đi kèm: `.clone(true)`. Như vậy chúng ta đỡ phải mất công gán lại sự kiện cho nó trong trường hợp chúng ta muốn nó đi kèm.

## Sao chép cho phân trích dẫn

Khi bạn đọc báo giấy, đôi khi bạn thấy có phần khung in nhỏ, trong đó có những mẫu trích dẫn những đoạn quan trọng để gây sự chú ý. Chúng ta có thể làm được việc này bằng cách sử dụng phương thức `.clone()`. Hãy xem lại đoạn mã HTML của đoạn văn thứ 3 của chúng ta.

```
<p>
<span class="pull-quote">It is a Law of Nature
<span class="drop">with us</span> that a male child shall
have <strong>one more side</strong> than his father</span>
so that each generation shall rise (as a rule) one step in
the scale of development and nobility. Thus the son of a
Square is a Pentagon; the son of a Pentagon, a Hexagon; and
so on.
</p>
```

Đoạn văn bắt đầu bằng một thẻ `<span class='pull-quote'>`. Đây sẽ là class chúng ta sẽ dùng để sao chép. Một khi đoạn văn nằm trong thẻ `<span>` đó được copy, chúng ta sẽ dán nó vào một nơi khác và chúng ta cũng cần chỉnh sửa lại style cho nó bắt mắt hơn.

Chúng ta sẽ gán một class là `'pulled'` cho thẻ `<span>` vừa được copy và khai báo những thuộc tính sau trong CSS

```
.pulled {
background: #e5e5e5;
position: absolute;
width: 145px;
top: -20px;
right: -180px;
padding: 12px 5px 12px 10px;
font: italic 1.4em "Times New Roman", Times, serif;
}
```

Đoạn trích dẫn có màu nền xanh nhạt, một ít padding và font khác. Chúng ta cũng sử dụng absolute position để định vị cho thành phần mới này.

Bây giờ chúng ta sẽ quay lại phần jQuery. Chúng ta sẽ bắt đầu với biểu thức bộ trọng cho tất cả các thẻ `<span class='pull-quote'>`, sau đó chúng ta sẽ gán vào phương thức `.each()` để chạy qua từng phần tử một.

```
$(document).ready(function() {
$('span.pull-quote').each(function(index) {
//...
});
});
Tiếp theo chúng ta tìm đoạn văn "cha mẹ" của từng đoạn trích dẫn và áp dụng
thuộc tính CSS
$(document).ready(function() {
$('span.pull-quote').each(function(index) {
var $parentParagraph = $(this).parent('p');
$parentParagraph.css('position', 'relative');
});
});
```

Một lần nữa, chúng ta lưu lại bộ chọn mà chúng ta sẽ sử dụng hơn một lần vào một biến để giúp cho mã của chúng ta hoạt động hiệu quả hơn và cũng dễ đọc hơn.

Bây giờ sau khi đã thiết lập xong hai giá trị định vị quan trọng của CSS là relative cho đoạn văn chứa phần trích dẫn và absolute cho đoạn trích dẫn. Tiếp theo chúng ta có thể copy từng thẻ `<span>` và thêm class là pulled cho từng bản sao, và cuối cùng là chèn nó vào phần bắt đầu của đoạn văn bản.

Lưu ý: nếu bạn chưa hiểu kỹ khái niệm này, bạn có thể xem video về [Absolute Position](#) trong CSS

```
$(document).ready(function() {
  $('span.pull-quote').each(function(index) {
    var $parentParagraph = $(this).parent('p');
    $parentParagraph.css('position', 'relative');
    $(this).clone()
      .addClass('pulled')
      .prependTo($parentParagraph);
  });
});
```

Bởi vì chúng ta sử dụng absolute position cho đoạn trích dẫn, cho nên nó sẽ định vị theo đoạn văn bản chứa nó. Dưới đây là hình mà chúng ta có được đến bước này.

Về cơ bản thì đoạn trích dẫn nhìn cũng tạm ổn rồi, nhưng thường thì phần trích dẫn không có cùng định dạng văn bản như là đoạn văn được copy. Ở đây đoạn trích dẫn của chúng ta có một vài chữ được tô đậm. Điều chúng ta muốn là đoạn văn sau khi được copy sẽ loại bỏ hết những thẻ <strong>, <em> và <a href> hoặc bất cứ thẻ inline nào đi. Hơn nữa chúng ta cũng muốn sau khi copy thì cũng có thể sửa đổi nó chút chút như là xóa đi vài từ và thay thế nó bằng dấu ba chấm .... Quay lại đoạn mã HTML, bạn sẽ thấy có một vài từ được đặt nằm trong cặp thẻ <span class='drop'> không đơn giản </span>.

Chúng ta sẽ làm dấu 3 chấm trước sau đó sẽ loại bỏ hết các thuộc tính HTML đi và chỉ giữ lại phiên bản chữ không.

```
$(document).ready(function() {
  $('span.pull-quote').each(function(index) {
    var $parentParagraph = $(this).parent('p');
    $parentParagraph.css('position', 'relative');
    var $clonedCopy = $(this).clone();
    $clonedCopy
      .addClass('pulled')
      .find('span.drop')
      .html('&hellip;');
    $clonedCopy
      .prependTo($parentParagraph);
    var clonedText = $clonedCopy.text();
    $clonedCopy.html(clonedText);
  });
});
```

Chúng ta bắt đầu quá trình sao chép bằng cách lưu bản sao vào một biến. Lần này biến được tạo ra là cần thiết bởi vì chúng ta không thể làm việc hoàn toàn với nó trong cùng một dòng lệnh. Bạn cũng đã thấy là sau khi chúng ta tìm hết những thẻ <span class='drop'> và thay thế nó với dấu ba chấm, chúng ta sử dụng một phương thức .end(). Phương thức này nói cho jQuery biết, chúng ta muốn quay lại một bước trước. Trong trường hợp này chúng ta muốn quay lại đến bước .find('span.drop'). Như thế chúng ta sẽ chèn cả đoạn copy chứ không phải chỉ là dấu ba chấm vào phần đầu của đoạn văn.

Cuối cùng chúng ta tạo thêm một biến nữa là clonedText, biến này sẽ chứa phiên bản chữ không của đoạn văn chúng ta cần copy. Sau đó chúng ta sử dụng phiên bản chữ không có định dạng này để thay thế cho phiên bản HTML. Bây giờ đoạn trích dẫn sẽ như hình

trợ giúp của jQuery. Hơn nữa, jQuery cho phép  
mà chúng ta làm việc với nhiều thuộc tính CSS khi  
thể dễ dàng thiết lập id, rel và thuộc tính title cho

la fames ac turpis egestas.<sup>1</sup> Mauris placerat  
Vestibulum erat wisi, condimentum sed, commodo

la fames ac turpis egestas. Vestibulum tortor quam,  
t wisi, condimentum sed, commodo vitae, ornare

la fames ac turpis egestas. Vestibulum tortor quam,  
sit amet quam egestas semper. Aenean ultricies mi

por sit amet, ante. Donec eu libero sit amet quam e

ltricies mi vitae est. Mauris placerat eleifend leo. Qu

*Có những thuộc  
tính ... để sửa  
đổi nếu không  
có sự trợ giúp  
của jQuery.  
Hơn nữa,  
jQuery cho  
phép chúng*

*Pellentesque  
habitant ... et  
netus et  
malesuada  
fames ac turpis  
egestas.*

Ở hình trên tôi đã thêm một đoạn `<span class='pull-quote'>` vào để xem mã của chúng ta có  
thể tự động tạo ra một đoạn trích dẫn nữa không.

Bo tròn góc cho đoạn trích

Bây giờ đoạn trích dẫn đã hoạt động như mình muốn là những phần tử con bị loại bỏ, dấu 3  
chấm được thêm vào để thay cho chữ. Nếu bây giờ chúng ta muốn cho đoạn trích dẫn được  
bo tròn góc thì sao. Chúng ta có thể tạo ra một thẻ `<div>` bao quanh lấy đoạn trích.

```
$(document).ready(function() {  
  $('span.pull-quote').each(function(index) {  
    var $parentParagraph = $(this).parent('p');  
    $parentParagraph.css('position', 'relative');  
    var $clonedCopy = $(this).clone();  
    $clonedCopy  
      .addClass('pulled')  
      .find('span.drop')  
      .html('&hellip;')  
      .end()  
      .prependTo($parentParagraph)  
      .wrap('<div class="pulled-wrapper"></div>');  
    var clonedText = $clonedCopy.text();  
    $clonedCopy.html(clonedText);  
  });  
});
```



Chúng ta cũng cần phải chỉnh sửa mã CSS để có thể tạo ra hiệu ứng bo tròn góc.

```
.pulled-wrapper {  
background: url(top-bg.png) no-repeat left top;  
position: absolute;  
width: 160px;  
right: -180px;  
padding-top: 18px;  
}  
.pulled {  
background: url(bottom-bg.png) no-repeat left bottom;  
position: relative;  
display: block;  
width: 140px;  
padding: 0 10px 24px 10px;  
font: italic 1.4em "Times New Roman", Times, serif;  
}
```

jQuery. Hơn nữa, **jQuery** cho phép  
a làm việc với nhiều thuộc tính CSS khi  
; thiết lập id, rel và thuộc tính title cho

turpis egestas.<sup>1</sup> Mauris placerat  
erat wisi, condimentum sed, commodo

turpis egestas. Vestibulum tortor quam,  
mentum sed, commodo vitae, ornare

turpis egestas. Vestibulum tortor quam,  
m egestas semper. Aenean ultricies mi

t, ante. Donec eu libero sit amet quam

vitae est. Mauris placerat eleifend leo.

*Có những thuộc  
tính ... để sửa  
đổi nếu không  
có sự trợ giúp  
của jQuery. Hơn  
nữa, jQuery cho  
phép chúng*

*Pellentesque  
habitant ... et  
netus et  
malesuada  
fames ac turpis  
egestas.*

Xem [Demo Online](#) – Example 7

Tóm lược các phương thức sửa đổi DOM

Sự khác biệt giữa những phương thức sửa đổi DOM mà jQuery cung cấp cho chúng ta phụ thuộc vào nhiệm vụ và vị trí của nó. Phần này sẽ tóm lược cho bạn để nhớ phương thức nào nên được sử dụng để làm gì và khi nào thì nên sử dụng chúng



1. Để tạo một phần tử mới từ HTML, sử dụng hàm `$()`.
2. Để chèn một hoặc nhiều phần tử vào trong một phần tử khác sử dụng
  - `.append()`
  - `.appendTo()`
  - `.prepend()`
  - `.prependTo()`
3. Để chèn một phần tử mới vào bên cạnh một phần tử khác sử dụng
  - `.after()`
  - `.insertAfter()`
  - `.before()`
  - `.insertBefore()`
4. Để chèn một phần tử mới xung quanh một phần tử khác, sử dụng
  - `.wrap()`
  - `.wrapAll()`
  - `.wrapInner()`
5. Để thay thế một phần tử này với phần tử khác hoặc chữ sử dụng
  - `.html()`
  - `.text()`
  - `.replaceAll()`
  - `.replaceWith()`
6. Để loại bỏ một phần tử nằm trong một phần tử khác dùng
  - `.empty()`
7. Để loại bỏ một phần tử và con cái của nó trong một tài liệu mà không thực sự xoá nó dùng
  - `.remove()`

#### Tóm tắt

Trong chương này chúng ta đã tạo, sao chép, tái kết hợp và làm đẹp cho phần nội dung sử dụng phương pháp sửa đổi DOM của jQuery. Chúng ta đã áp dụng những phương thức này cho một trang web để biến những đoạn văn bình thường thành phân chú thích tự động, phân trích dẫn, đường liên kết v.v..

Phần tutorial của cuốn sách cũng sắp hết, nhưng trước khi chúng ta học thêm những ví dụ phức tạp hơn và mở rộng hơn. Chúng ta hãy dành chút thời gian để nghiên cứu phương thức AJAX của jQuery. Trong chương kế tiếp, chúng ta hãy dạo một vòng lên server nhờ “phi thuyền” AJAX.

## Chương 7: AJAX – Phần 1

Trong những năm gần đây, người ta hay đánh giá một trang web dựa vào công nghệ mà trang đó đang ứng dụng. Một trong những công nghệ trở nên rất đình đám trong thời gian gần đây là ứng dụng web được gọi là AJAX. Nó là tổng hợp của nhiều công nghệ khác nhau.

AJAX là chữ viết tắt của Asynchronous JavaScript and XML. Những công nghệ có trong một giải pháp AJAX bao gồm

- JavaScript dùng để tương tác với người dùng hoặc các sự kiện liên quan đến trình duyệt.

- Đối tượng XMLHttpRequest, cho phép những câu lệnh truy vấn được gửi đến server mà không làm gián đoạn những tác vụ khác của trình duyệt
- XML ở trên server, hoặc những định dạng dữ liệu tương tự như HTML và JSON
- Thêm JavaScript, dùng để chuyển đổi dữ liệu từ server và hiển thị nó lên trang web.

Công nghệ AJAX được ca tụng như là vị cứu tinh của thế giới web, nó biến những trang web tĩnh thành những ứng dụng có tính tương tác cao. Rất nhiều frameworks được tạo ra để giúp các lập trình viên học cách sử dụng nó, chính bởi sự không nhất quán của trình duyệt trong việc ứng dụng đối tượng XMLHttpRequest, jQuery cũng không phải là ngoại lệ. Chúng ta sẽ xem AJAX có thực sự kỳ diệu như người ta hay nói không.

### Tải dữ liệu khi được yêu cầu

Đằng sau ánh hoàng quang, thì AJAX thực sự chỉ là một công cụ dùng để tải dữ liệu từ server về trình duyệt mà không cần phải refresh lại trang web. Những dữ liệu này có nhiều định dạng và chúng ta cũng có nhiều lựa chọn để làm việc với nó khi nó được tải xong.

Chúng ta sẽ xây dựng một trang web hiển thị những từ mới trong cuốn từ điển, các nhóm từ được gom lại dưới một chữ cái như trong từ điển. Mã HTML để định dạng vùng nội dung của trang sẽ như sau:

```
<div id="dictionary">
</div>
```

Yep! Chỉ có vậy thôi. Trang web của chúng ta sẽ không có nội dung nào hết. Chúng ta sẽ sử dụng những phương thức AJAX của jQuery để hiển thị thẻ <div> với cuốn từ điển.

Chúng ta sẽ cần một nơi để kích hoạt quá trình tải dữ liệu, do vậy chúng ta sẽ thêm vào vài đường liên kết để sau này mình có nơi để gán bộ xử lý sự kiện.

```
<div class="letters">
<div class="letter" id="letter-a">
<h3><a href="#">A</a></h3>
</div>
<div class="letter" id="letter-b">
<h3><a href="#">B</a></h3>
</div>
<div class="letter" id="letter-c">
<h3><a href="#">C</a></h3>
</div>
<div class="letter" id="letter-d">
<h3><a href="#">D</a></h3>
</div>
</div>
```

Thêm một chút CSS, chúng ta có một trang như sau

## Izwebz - jQuery

by: Demon Warlock

---

[A](#)

[B](#)

[C](#)

[D](#)

[E](#)

Bây giờ chúng ta tập trung vào phần lấy nội dung cho trang.

Gán HTML vào

Ứng dụng AJAX thường chỉ là những truy vấn để có được những đoạn mã HTML. Kỹ thuật này đôi khi còn được gọi là AHAH (Asynchronous HTTP and HTML), lại quá đơn giản với jQuery. Trước hết chúng ta cần một đoạn mã HTML để chèn, chúng ta sẽ tạo một file mới đặt tên là a.html. File HTML này sẽ có mã như sau:

```
<div class="entry">
<h3 class="term">ABDICATION</h3>
<div class="part">n.</div>
<div class="definition">
An act whereby a sovereign attests his sense of the high
temperature of the throne.
<div class="quote">
<div class="quote-line">Poor Isabella's Dead, whose
abdication</div>
<div class="quote-line">Set all tongues wagging in the
Spanish nation.</div>
<div class="quote-line">For that performance 'twere
unfair to scold her:</div>
<div class="quote-line">She wisely left a throne too
hot to hold her.</div>
<div class="quote-line">To History she'll be no royal
riddle &mdash;</div>
<div class="quote-line">Merely a plain parched pea that
jumped the griddle.</div>
<div class="quote-author">G.J.</div>
</div>
</div>
</div>
<div class="entry">
<h3 class="term">ABSOLUTE</h3>
<div class="part">adj.</div>
<div class="definition">
Independent, irresponsible. An absolute monarchy is one
in which the sovereign does as he pleases so long as he
pleases the assassins. Not many absolute monarchies are
left, most of them having been replaced by limited
monarchies, where the sovereign's power for evil (and for
good) is greatly curtailed, and by republics, which are
```

governed by chance.  
</div>  
</div>

Đây là hình mà chúng ta sẽ có được, tất nhiên nó nhìn hơi “cùi” vì chưa có định dạng gì hết.

## ABDICATION

n.

An act whereby a sovereign attests his sense of the high temperature of the throne.

Poor Isabella's Dead, whose abdication

Set all tongues wagging in the Spanish nation.

For that performance 'twere unfair to scold her:

She wisely left a throne too hot to hold her.

To History she'll be no royal riddle —

Merely a plain parched pea that jumped the griddle.

G.J.

## ABSOLUTE

adj.

Independent, irresponsible. An absolute monarchy is one in which the sovereign does as he pleases so long as limited monarchies, where the sovereign's power for evil (and for good) is greatly curtailed, and by republics, v

## ACKNOWLEDGE

v.t.

To confess. Acknowledgement of one another's faults is the highest duty imposed by our love of truth.

Bạn cũng nên chú ý là file a.html không phải là một tài liệu HTML thực sự, bởi vì nó không có thể <html>, <head> và <body>. Đây là những thẻ bắt buộc phải có cho một tài liệu HTML. Những file như thế này được gọi là mảnh hoặc đoạn mã, mục đích tồn tại của nó chỉ dùng để chèn vào những tài liệu HTML khác, đây chính là việc chúng ta sẽ làm.

```
$(document).ready(function() {  
  $('#letter-a a').click(function() {  
    $('#dictionary').load('a.html');  
    return false;  
  });  
});
```

Phương thức .load() sẽ làm tất cả những việc còn lại cho chúng ta. Chúng ta chỉ cho nó đường dẫn đến đoạn mã cần chèn bằng cách sử dụng những bộ chọn jQuery thông thường, và sau đó đưa URL của tên file mà chúng ta cần tải dưới dạng tham số của phương thức. Bây giờ nếu bạn nhấp chuột vào đường liên kết đầu tiên, tệp tin đó sẽ được tải và đặt vào trong <div id='dictionary'>. Trình duyệt sẽ xử lý đoạn mã HTML mới ngay khi nó được chèn vào.

[A](#)

[B](#)

[C](#)

[D](#)

## ABDICATION *n.*

An act whereby a sovereign attests his sense of the high temperature of the throne.

Poor Isabella's Dead, whose abdication  
Set all tongues wagging in the Spanish nation.  
For that performance 'twere unfair to scold her:  
She wisely left a throne too hot to hold her.  
To History she'll be no royal riddle —  
Merely a plain parched pea that jumped the griddle.  
G.J.

## ABSOLUTE *adj.*

Independent, irresponsible. An absolute monarchy is one in which the sovereign does as he pleases so long as he pleases the assassins. Not many absolute monarchies are left, most them having been replaced by limited monarchies, where the sovereign's power for evil (and good) is greatly curtailed, and by republics, which are governed by chance.

## ACKNOWLEDGE *v.t.*

To confess. Acknowledgement of one another's faults is the highest duty imposed by our love truth.

Bạn nhận thấy rằng mã HTML của chúng ta đã tự động có định dạng CSS còn trước đây thì nó không có định dạng gì. Bởi vì ngay sau khi đoạn mã HTML này được chèn vào trang thì nó sẽ chịu ảnh hưởng bởi các luật CSS của trang nó được chèn vào.

Khi bạn thử nhấn một chữ thì định nghĩa của từ đó sẽ xuất hiện gần như ngay lập tức. Đây chính là điểm nhằm lẫn khi bạn làm việc local. Bạn sẽ không thấy được thời gian phải đợi để truyền tải tài liệu trên mạng. Giả sử chúng ta thêm một thông báo khi định nghĩa của từ đã tải xong

```
$(document).ready(function() {  
    $('#letter-a a').click(function() {  
        $('#dictionary').load('a.html');  
        alert('Loaded!');  
        return false;  
    });  
});
```

Khi bạn nhìn vào đoạn mã jQuery ở trên bạn có thể nghĩ rằng hộp thông báo chỉ xuất hiện sau khi tài liệu đã được tải xong. Những lệnh của JavaScript là đồng bộ, làm xong với tác vụ này mới đến tác vụ khác theo trật tự nghiêm ngặt.

Nhưng nếu đoạn mã này được chạy thử trên host thật thì bảng thông báo xuất hiện và biến mất trước khi quá trình tải hoàn thành, đó chính là do sự chậm trễ của mạng. Điều này xảy ra là vì những cuộc gọi của AJAX là không đồng bộ. Nếu không thì ta phải gọi nó là SJAX, nghe đã không thấy phê rồi.

Tải dữ liệu không đồng bộ có nghĩa là một khi truy vấn HTTP gửi đi để lấy đoạn mã HTML về được sử dụng, đoạn mã vừa gửi truy vấn đó lập tức quay lại hoạt động mà không chờ thêm gì nữa. Khoảng một lúc sau, trình duyệt nhận được phản hồi từ server và xử lý nó. Thường thì đây là điều mình muốn bởi vì bạn không muốn khóa cửa sổ duyệt web của người dùng trong khi chờ tải dữ liệu.

Nhưng nếu bạn muốn đoạn mã phải chờ cho đến khi quá trình tải hoàn thành, jQuery cung cấp một hàm truy hồi cho vấn đề này. Chúng ta hãy xem ví dụ dưới đây

### Làm việc với đối tượng JavaScript

Để tải được một trang HTML được định dạng đầy đủ rất đơn giản, nhưng cũng có lúc chúng ta muốn đoạn mã của mình có thể xử lý dữ liệu trước khi nó được hiển thị. Trong trường hợp này, chúng ta cần lấy dữ liệu ra với cấu trúc mà chúng ta có thể dùng JavaScript để thao tác.

Với bộ chọn jQuery, chúng ta có thể di chuyển qua lại trong HTML và thao tác với nó, nhưng trước hết nó phải được chèn vào tài liệu đã. Định dạng dữ liệu thuần JavaScript hơn có nghĩa là bạn ít phải viết ít mã hơn.

Lấy ra một đối tượng JavaScript

Như chúng ta thường thấy, đối tượng JavaScript chỉ là tập hợp của những cặp key-value, và có thể được định nghĩa ngắn gọn với cặp ngoặc cong {}. Trái lại, mảng JavaScript lại được định nghĩa bằng cặp ngoặc vuông []. Kết hợp hai khái niệm này, chúng ta có thể biểu đạt được những cấu trúc phức tạp và giàu dữ liệu.

Khái niệm JavaScript Object Notation (JSON) được giới thiệu bởi Douglas Crockford để tận dụng thế mạnh về cú pháp đơn giản này. Bản ký pháp này cho chúng ta một sự thay thế hoàn hảo cho định XML, mà có lúc rất cồng kềnh.

```
{
  "key": "value",
  "key 2": [
    "array",
    "of",
    "items"
  ]
}
```

**Lưu ý:** Nếu bạn muốn biết thêm những thông tin về thế mạnh của JSON và những ứng dụng của nó cho những ngôn ngữ lập trình khác, bạn có thể vào trang web [www.json.org](http://www.json.org)

Chúng ta có thể mã hóa dữ liệu của chúng ta bằng cách sử dụng định dạng này bằng nhiều cách. Chúng ta sẽ để vài mục từ trong từ điển ở một file JSON và đặt tên là b.json. Đoạn mã sẽ như sau

```
[
{
  "term": "BACCHUS",
  "part": "n.",
  "definition": "A convenient deity invented by the...",
  "quote": [
    "Is public worship, then, a sin,",
    "That for devotions paid to Bacchus",
  ]
}
```

```

"The lictors dare to run us in,",
"And resolutely thump and whack us?"
],
"author": "Jorace"
},
{
"term": "BACKBITE",
"part": "v.t.",
"definition": "To speak of a man as you find him when..."
},
{
"term": "BEARD",
"part": "n.",
"definition": "The hair that is commonly cut off by..."
},

```

Để lấy dữ liệu này ra, chúng ta sẽ sử dụng phương thức `$.getJSON()`, phương thức này sẽ tìm nạp tệp tin và xử lý nó, kết quả của đoạn mã được gọi sẽ là đối tượng JavaScript.

### Hàm jQuery toàn cục

Cho đến thời điểm này, những phương thức mà chúng ta sử dụng được gán vào một đối tượng jQuery mà chúng ta tạo ra bằng cách sử dụng hàm `$()`. Bộ chọn cho phép chúng ta chọn ra một điểm trong DOM để các phương thức của chúng ta làm việc trên chúng. Nhưng hàm `$.getJSON` thì lại khác. Nó sẽ không được áp dụng lên bất cứ phần tử DOM nào, đối tượng trả về phải được sử dụng cho đoạn mã chứ không phải là chèn vào trang. Chính vì lý do này mà hàm `getJSON()` được định nghĩa là phương thức đối tượng jQuery toàn cục (một đối tượng được gọi bởi jQuery hoặc được `$` xác định một lần bởi jQuery) chứ không phải một phiên bản đối tượng jQuery (đối tượng được chúng ta tạo với hàm `$()`).

Nếu JavaScript có class như những ngôn ngữ lập trình hướng đối tượng khác, thì chúng ta sẽ gọi `$.getJSON()` là một phương thức class. Do vậy chúng ta gọi phương pháp dạng này là hàm toàn cục, trong thực tế, nó là những hàm sử dụng dấu cách jQuery để tránh bị xung đột với tên của các hàm khác.

Để sử dụng hàm này, chúng ta truyền qua tên file như trước:

```

$(document).ready(function() {
$('#letter-b a').click(function() {
$.getJSON('b.json');
return false;
});
});

```

Đoạn mã trên không tạo ra thay đổi gì rõ ràng khi bạn nhấp vào đường liên kết. Hàm được gọi sẽ tải tệp tin, nhưng chúng ta chưa bảo JavaScript phải làm gì với dữ liệu có được. Do vậy chúng ta phải sử dụng hàm truy hồi.

Hàm `$.getJSON()` lấy vào một tham số thứ 2, tham số này cũng chính là một hàm được gọi khi quá trình tải hoàn thành. Như đã nói trước đây, những cuộc gọi của AJAX là dạng không đồng bộ, cho nên hàm truy hồi sẽ đợi cho dữ liệu được tải hết thay vì chạy đoạn mã ngay lập tức. Hàm truy hồi này cũng lấy vào một tham số nữa dùng để chứa dữ liệu thu về. Nên chúng ta có thể viết:

```
$(document).ready(function() {
$('#letter-b a').click(function() {
$.getJSON('b.json', function(data) {
});
return false;
});
});
```

Ở đây chúng ta sử dụng một hàm ẩn như là hàm truy hồi, như một cách viết tắt phổ biến trong jQuery. Một hàm có thể được sử dụng làm hàm truy hồi.

Bên trong hàm này, chúng ta có thể sử dụng biến số data để di chuyển trong cấu trúc dữ liệu nếu cần. Chúng ta cần phải chạy lên mảng trên cùng, xây dựng HTML cho từng phần tử. Chúng ta cũng có thể làm việc này với một vòng for, nhưng thay vào đó, chúng ta sẽ làm quen với một hàm toàn cục nữa của jQuery là \$.each(). Chúng ta đã biết một hàm gần giống nó là phương thức .each() trong chương 5. Thay vì chỉ làm việc với một đối tượng jQuery, hàm này lấy vào một mảng hoặc một biểu đồ làm tham số thứ nhất và một hàm truy hồi làm tham số thứ 2. Mỗi lần vòng lặp chạy thì chỉ số lặp hiện tại và phần tử hiện tại trong mảng hoặc biểu đồ được chuyển vào như hai tham số cho hàm truy hồi.

```
$(document).ready(function() {
$('#letter-b a').click(function() {
$.getJSON('b.json', function(data) {
$('#dictionary').empty();
$.each(data, function(entryIndex, entry) {
var html = '<div class="entry">';
html += '<h3 class="term">' + entry['term'] + '</h3>';
html += '<div class="part">' + entry['part'] + '</div>';
html += '<div class="definition">';
html += entry['definition'];
html += '</div>';
html += '</div>';
$('#dictionary').append(html);
});
});
return false;
});
});
```

Trước khi vòng lặp bắt đầu, chúng ta đã làm rỗng thẻ <div id='dictionary'>, do vậy chúng ta có thể chèn vào mã HTML vừa tạo được. Sau đó chúng ta sử dụng hàm \$.each() để kiểm tra từng phần tử một, xây dựng cấu trúc HTML dựa vào nội dung của biểu đồ. Cuối cùng chúng ta biến đoạn mã HTML thành cây DOM bằng cách gán nó vào thẻ <div>

**Lưu ý:** cách này giả sử rằng dữ liệu tải về là an toàn để sử dụng với HTML, nó không được có những ký hiệu như kiểu <.

Bây giờ chỉ còn phần trích dẫn của mục từ trong từ điển, bằng cách sử dụng một vòng lặp \$.each() nữa.

```
$(document).ready(function() {
$('#letter-b a').click(function() {
$.getJSON('b.json', function(data) {
$('#dictionary').empty();
$.each(data, function(entryIndex, entry) {
var html = '<div class="entry">';
```



```

html += '<h3 class="term">' + entry['term'] + '</h3>';
html += '<div class="part">' + entry['part'] + '</div>';
html += '<div class="definition">';
html += entry['definition'];
if (entry['quote']) {
html += '<div class="quote">';
$.each(entry['quote'], function(lineIndex, line) {
html += '<div class="quote-line">' + line + '</div>';
});
if (entry['author']) {
html += '<div class="quote-author">' + entry['author'] + '</div>';
}
html += '</div>';
}
html += '</div>';
html += '</div>';
$('#dictionary').append(html);
});
});
return false;
});
});

```

Bây giờ bạn có thể thử nhấp chuột vào chữ B để xem thử kết quả

## Izwebz - jQuery

by: Demon Warlock

[A](#)

[B](#)

[C](#)

[D](#)

### BACCHUS *n.*

A convenient deity invented by the ancients as an excuse for getting drunk.

Is public worship, then, a sin,  
That for devotions paid to Bacchus  
The lictors dare to run us in,  
And resolutely thump and whack us?

Jorace

### BACKBITE *v.t.*

To speak of a man as you find him when he can't find you.

### BEARD *n.*

The hair that is commonly cut off by those who justly execrate the absurd Chinese custom of shaving the head.

### BEGGAR *n.*

One who has relied on the assistance of his friends.

**Lưu ý:** định dạng JSON rất ngắn gọn nhưng nghiêm ngặt. Mỗi dấu ngoặc, dấu nhảy hay dấu phải đều phải đầy đủ và chính xác, nếu không tập tin sẽ không được tải. Trong phần lớn các

*trình duyệt, thậm chí nó còn không báo lỗi mà cả đoạn mã hoàn toàn không chạy một cách âm thầm.*

### Chạy một đoạn mã

Đôi khi chúng ta không muốn lấy về tất cả mã JavaScript khi trang web được tải lần đầu tiên. Chúng ta không biết đoạn mã nào là cần thiết cho đến khi có những tương tác của người dùng. Chúng ta cũng có thể sử dụng thẻ `<script>` nếu cần nhưng có một cách khác hay hơn để chèn thêm mã vào là dùng jQuery để tải trực tiếp tệp tin .js

Để chèn vào một đoạn mã cũng đơn giản như khi chèn một đoạn HTML. Trong trường hợp này chúng ta sử dụng hàm toàn cục `$.getScript()`, hàm này cũng như những hàm cùng chức năng của nó, chấp nhận một địa chỉ URL trở đến vị trí của tệp tin.

```
$(document).ready(function() {  
  $('#letter-c a').click(function() {  
    $.getScript('c.js');  
    return false;  
  });  
});
```

Ở ví dụ cuối cùng của chúng ta, chúng ta cần phải xử lý dữ liệu trả về để mình có thể làm một cái gì đó với tệp tin được tải về. Nhưng với những tệp tin chứa mã, quá trình xử lý là hoàn toàn tự động, một khi được tải đoạn mã sẽ tự chạy.

Mã được tải bằng cách này sẽ chạy trong ngữ cảnh toàn cục của trang hiện tại. Điều đó có nghĩa là chúng ta có thể đến được tất cả những hàm và các biến số được khai báo toàn cục, kể cả bản thân jQuery. Cho nên chúng ta có thể bắt chước ví dụ về JSON để chuẩn bị và chèn HTML vào trang khi đoạn mã được thực thi, và đặt đoạn mã này vào tệp `c.js`:

```
var entries = [  
  {  
    "term": "CALAMITY",  
    "part": "n.",  
    "definition": "A more than commonly plain and..."  
  },  
  {  
    "term": "CANNIBAL",  
    "part": "n.",  
    "definition": "A gastronome of the old school who..."  
  },  
  {  
    "term": "CHILDHOOD",  
    "part": "n.",  
    "definition": "The period of human life intermediate..."  
  },  
  {  
    "term": "CLARINET",  
    "part": "n.",  
    "definition": "An instrument of torture operated by..." },  
  {  
    "term": "COMFORT",  
    "part": "n.",  
    "definition": "A state of mind produced by..."  
  },  
  {  
    "term": "CORSAIR",
```

```

"part": "n.",
"definition": "A politician of the seas."
}
];
var html = '';
$.each(entries, function() {
html += '<div class="entry">';
html += '<h3 class="term">' + this['term'] + '</h3>';
html += '<div class="part">' + this['part'] + '</div>';
html += '<div class="definition">' + this['definition'] + '</div>';
html += '</div>';
});
$('#dictionary').html(html);
},
{
"term": "COMFORT",
"part": "n.",
"definition": "A state of mind produced by..."
},
{
"term": "CORSAIR",
"part": "n.",
"definition": "A politician of the seas."
}
];
var html = '';
$.each(entries, function() {
html += '<div class="entry">';
html += '<h3 class="term">' + this['term'] + '</h3>';
html += '<div class="part">' + this['part'] + '</div>';
html += '<div class="definition">' + this['definition'] + '</div>';
html += '</div>';
});
$('#dictionary').html(html);

```

Bạn thử nhấn vào chữ cái C để xem kết quả.

# Izwebz - jQuery

by: Demon Warlock

[A](#)

[B](#)

[C](#)

[D](#)

## CALAMITY *n.*

A more than commonly plain and unmistakable reminder that the affairs of this life are not of our own ordering. Calamities are of two kinds: misfortune to ourselves, and good fortune to others.

## CANNIBAL *n.*

A gastronome of the old school who preserves the simple tastes and adheres to the natural diet of the pre-pork period.

## CHILDHOOD *n.*

The period of human life intermediate between the idiocy of infancy and the folly of youth — two removes from the sin of manhood and three from the remorse of age.

## CLARIONET *n.*

An instrument of torture operated by a person with cotton in his ears. There are two instruments that are worse than a clarionet — two clarionets.

## COMFORT

### Tải tài liệu XML

XML là một phần trong những chữ cái viết tắt của AJAX, nhưng chúng ta vẫn chưa tải XML lần nào. Cách tải tệp XML cũng khá đơn giản và rất giống với cách mà chúng ta làm với JSON. Trước hết chúng ta cần một tệp XML là đặt tên là d.xml và chứa những dữ liệu chúng ta cần hiển thị.

```
<?xml version="1.0" encoding="UTF-8"?>
<entries>
<entry term="DEFAME" part="v.t.">
<definition>
To lie about another. To tell the truth about another.
</definition>
</entry>
<entry term="DEFENCELESS" part="adj.">
<definition>
Unable to attack.
</definition>
</entry>
<entry term="DELUSION" part="n.">
<definition>
The father of a most respectable family, comprising
Enthusiasm, Affection, Self-denial, Faith, Hope,
Charity and many other goodly sons and daughters.
</definition>
<quote author="Mumfrey Mappel">
<line>All hail, Delusion! Were it not for thee</line>
<line>The world turned topsy-turvy we should see;
</line>
<line>For Vice, respectable with cleanly fancies,
```

```

</line>
<line>Would fly abandoned Virtue's gross advances.
</line>
</quote>
</entry>
<entry term="DIE" part="n.">
<definition>
The singular of "dice." We seldom hear the word,
because there is a prohibitory proverb, "Never say
die." At long intervals, however, some one says: "The
die is cast," which is not true, for it is cut. The
word is found in an immortal couplet by that eminent
poet and domestic economist, Senator Depew:
</definition>
<quote>
<line>A cube of cheese no larger than a die</line>
<line>May bait the trap to catch a nibbling mie.</line>
</quote>
</entry>
</entries>

```

Tất nhiên dữ liệu này có thể được biểu thị bằng nhiều cách, và một số phần rất giống với cấu trúc mà chúng ta đã làm với HTML và JSON trước đây. Nhưng trong ví dụ này bạn sẽ được làm quen với một vài chức năng của XML được thiết kế để con người còn có thể hiểu được, như là cách sử dụng thuộc tính cho term và part thay vì dùng thẻ. Chúng ta cũng bắt đầu hàm với cách quen thuộc

```

$(document).ready(function() {
$('#letter-d a').click(function() {
$.get('d.xml', function(data) {
});
return false;
});
});

```

Lần này chúng ta sử dụng hàm \$.get(). Nói chung, hàm này chỉ đơn thuần là truy xuất tệp tin ở địa chỉ URL cho trước và cung cấp một đoạn chữ trắng không định dạng cho hàm truy hồi. Nhưng nếu phản hồi lại là định dạng XML dựa vào MINE type của server cung cấp, hàm truy hồi sẽ nhận được cây XML DOM.

Cũng may cho chúng ta là jQuery có khả năng di chuyển rất tốt trong DOM. Chúng ta có thể sử dụng phương thức .find(), .filter() và những phương thức di chuyển khác trong tài liệu XML y như cách mà chúng ta làm việc với HTML.

```

$(document).ready(function() {
$('#letter-d a').click(function() {
$.get('d.xml', function(data) {
$('#dictionary').empty();
$(data).find('entry').each(function() {
var $entry = $(this);
var html = '<div class="entry">';
html += '<h3 class="term">' + $entry.attr('term')
+ '</h3>';
html += '<div class="part">' + $entry.attr('part')
+ '</div>';
html += '<div class="definition">';
html += $entry.find('definition').text();

```

```

var $quote = $entry.find('quote');
if ($quote.length) {
html += '<div class="quote">';
$quote.find('line').each(function() {
html += '<div class="quote-line">'
+ $(this).text() + '</div>';
});
if ($quote.attr('author'))
html += '<div class="quote-author">'
+ $quote.attr('author') + '</div>';
}
html += '</div>';
}
html += '</div>';
html += '</div>';
$('#dictionary').append($ (html));
});
});
return false;
});
});

```

Bạn nhận thử chữ D để xem kết quả

**A**

**B**

**C**

**D**

### **DANCE** *v.i.*

To leap about to the sound of tittering music, preferably with arms about your neighbor's daughter. There are many kinds of dances, but all those requiring the participation of the dancer have two characteristics in common: they are conspicuously innocent, and warmly love vicious.

### **DAY** *n.*

A period of twenty-four hours, mostly misspent. This period is divided into two parts, the day and the night, or day improper the former devoted to sins of business, the latter consecrated to other sort. These two kinds of social activity overlap.

### **DEBT** *n.*

An ingenious substitute for the chain and whip of the slave-driver.

As, pent in an aquarium, the troutlet  
Swims round and round his tank to find an outlet,  
Pressing his nose against the glass that holds him,  
Nor ever sees the prison that enfolds him;  
So the poor debtor, seeing naught around him,  
Yet feels the narrow limits that impound him,  
Grieves at his debt and studies to evade it,  
And finds at last he might as well have paid it.

**Barlow S. Vode**

Đây là một cách mới trong những phương thức di chuyển trong DOM mà chúng ta đã biết, cho ta thấy tính linh động của bộ chọn CSS trong jQuery. Cú pháp của CSS thường được sử dụng để làm đẹp cho trang HTML, cho nên bộ chọn tiêu chuẩn trong file .css sử dụng tên thẻ HTML như div và body để tìm đến nội dung. Tuy nhiên, jQuery cũng có thể sử dụng những thẻ XML thông thường như là entry và definition, như cách mà chúng ta sử dụng HTML.

Những bộ chọn nâng cao của jQuery còn cho phép tìm đến những phần ở tài liệu XML trong những trường hợp phức tạp hơn nhiều. Ví dụ chúng ta muốn giới hạn hiển thị những mục từ có chứa câu trích dẫn và thuộc tính author. Để làm được điều này, chúng ta có thể giới hạn những mục từ có chứa các phần tử <quotes> bằng cách thay đổi entry thành entry:has(quote). Sau đó chúng ta cũng có thể giới hạn thêm những mục từ có chứa thuộc tính author trong phần bằng cách viết entry:has(quote[author]). Bây giờ bộ chọn của chúng ta sẽ như sau:

```
$(data).find('entry:has(quote[author])').each(function() {
```

Biểu thức bộ chọn bây giờ giới hạn những mục từ như hình

<b>A</b>	<b>DEBT</b> <i>n.</i> An ingenious substitute for the chain and whip of the slave-driver.  As, pent in an aquarium, the troutlet Swims round and round his tank to find an outlet, Pressing his nose against the glass that holds him, Nor ever sees the prison that enfolds him; So the poor debtor, seeing naught around him, Yet feels the narrow limits that impound him, Grieves at his debt and studies to evade it, And finds at last he might as well have paid it.  Barlow S. Vode
<b>B</b>	
<b>C</b>	
<b>D</b>	<b>DELUSION</b> <i>n.</i> The father of a most respectable family, comprising Enthusiasm, Affection, Self-Charity and many other goodly sons and daughters.  All hail, Delusion! Were it not for thee The world turned topsy-turvy we should see; For Vice, respectable with cleanly fancies, Would fly abandoned Virtue's gross advances.  Mumfrey Mappel

## Chương 7 – AJAX – Phần 2

Đây là phần thứ 2 của chương 6 – AJAX

Lựa chọn định dạng dữ liệu

Chúng ta đã xem qua 4 định dạng cho dữ liệu bên ngoài, mỗi một dạng đều được xử lý bởi những hàm thuần AJAX của jQuery. Chúng ta cũng đã xác minh cả 4 định dạng đều có thể xử lý được tình huống là tải thông tin cho trang mỗi khi người dùng yêu cầu chứ không phải trước đó. Như vậy thì định dạng nào phù hợp với ứng dụng nào?

HTML không mất nhiều công để tải. Dữ liệu bên ngoài ngoài có thể được tải và chèn vào trang với một phương thức mà thậm chí không cần có hàm truy hồi. Chúng ta cũng không cần sử dụng những phương thức di chuyển trong dữ liệu để thêm một đoạn HTML vào trang. Trái lại, dữ liệu này không có cấu trúc phù hợp để có thể tái sử dụng cho những ứng dụng khác. Mà nó được liên kết chặt chẽ với thành phần mà nó sẽ được chèn vào.

JSON thì được cấu trúc cho việc tái sử dụng đơn giản. Định dạng này cô đọng và dễ đọc. Nhưng chúng ta phải di chuyển trong cấu trúc dữ liệu để lấy thông tin hiển thị ra trang web, nhưng điều này cũng dễ dàng được thực hiện bởi những kỹ thuật JavaScript tiêu chuẩn. Bởi vì tệp tin có thể được tải chỉ bằng một cuộc gọi đến phương thức JavaScript eval(), phương thức này đọc tệp JSON hết sức mau lẹ. Tuy nhiên cách sử dụng eval() cũng chứa đựng một chút rủi ro. Những lỗi lập trình trong tệp JSON có gây ra lỗi ẩn và tạo ra hiệu ứng phụ không mong muốn trên trang, cho nên dữ liệu phải được viết hết sức cẩn thận.

Tệp JavaScript có tính linh động nhất nhưng nó lại không thực sự là một cơ chế lưu trữ dữ liệu. Bởi vì nó mang hơi hướng của ngôn ngữ lập trình, nó không thể cung cấp cùng một loại thông tin cho những hệ thống khác nhau. Thực tế việc tải một tệp JavaScript có nghĩa là những bộ xử lý mà ít khi được dùng tới có thể để tách rời ở một tệp bên ngoài, như thế chúng ta có thể giảm được dung lượng của mã và chỉ tải nó khi cần thiết.

Tài liệu XML cực kỳ cơ động. Bởi vì XML đã trở thành ngôn ngữ chung cho thế giới mạng, cung cấp dữ liệu dưới dạng này thì nó rất có thể được tái sử dụng ở đâu đó. Ví dụ Flickr, del.icio.us và Upcoming đều xuất dữ liệu của họ dưới dạng XML, và được rất nhiều các trang khác tái sử dụng rất sáng tạo. Tuy nhiên định dạng XML hơi cồng kềnh và mất nhiều thời gian để tải và thao tác hơn những định dạng khác.

Với những tính năng như ở trên, thì bạn thấy cách dễ nhất để cung cấp dữ liệu từ bên ngoài là dưới dạng HTML miễn là dữ liệu đó không cần phải được sử dụng cho những ứng dụng khác. Trong trường hợp dữ liệu sẽ được tái sử dụng và những ứng dụng khác có thể bị ảnh hưởng, thì JSON thường là lựa chọn tốt bởi vì nó có hiệu suất làm việc cao và dung lượng nhỏ. Nhưng khi ứng dụng là điều mà bạn không chắc chắn, thì XML là lựa chọn an toàn nhất để có tính tương kết cao nhất.

Trên tất cả những điều trên, chúng ta phải xác định xem dữ liệu đã có sẵn chưa. Nếu nó đã có sẵn rồi thì rất có thể nó rơi vào một trong những định dạng trên và như thế thì bạn khỏi cần phải mất công tự quyết định.

### Truyền dữ liệu đến server

Những ví dụ của chúng ta cho đến giờ chỉ tập trung vào việc lấy dữ liệu tĩnh từ web server. Tuy nhiên, AJAX chỉ thực sự mạnh mẽ khi mà server có thể tự động truyền dữ liệu dựa vào thông tin được nhập từ trình duyệt web. JQuery có thể giúp chúng ta rất nhiều trong quá trình này, những phương thức chúng ta đã học đến nay có thể được cải tiến một chút để cho quá trình truyền tải dữ liệu trở thành đường 2 chiều.

**Lưu ý:** những ví dụ sắp tới đòi hỏi phải tương tác với web server, cho nên đây là lần đầu tiên trong cuốn sách chúng ta sẽ sử dụng mã server-side. Trong những phần tới chúng ta sẽ sử dụng ngôn ngữ lập trình PHP, đây là ngôn ngữ được sử dụng rộng rãi và hoàn toàn miễn phí. Chúng ta sẽ không đề cập đến cách tạo web server trong khuôn khổ của cuốn sách này. Bạn có thể tìm các nguồn hướng dẫn như ở trang [Apache.org](http://Apache.org) hoặc [php.net](http://php.net). Trên [izwebz](http://izwebz.com) cũng có một vài bài hướng dẫn cách tạo localhost để làm việc với PHP.



## Thực hiện lệnh truy vấn GET

Để minh họa cho quá trình giao tiếp giữa người dùng và server, chúng ta sẽ viết một đoạn mã mà nó có thể chỉ gửi một mục từ trong từ điển đến trình duyệt cho mỗi một lệnh truy vấn. Mục từ được chọn sẽ dựa vào tham số được gửi qua trình duyệt. Mã của chúng ta sẽ lấy dữ liệu từ cấu trúc dữ liệu trong như sau:

[?](#)

```
1 <?php
2 $entries = array(
3   'EAVESDROP' => array(
4     'part' => 'v.i.',
5     'definition' => 'Secretly to overhear a catalogue of the
6 crimes and vices of another or yourself.',
7     'quote' => array(
8       'A lady with one of her ears applied',
9       'To an open keyhole heard, inside,',
10      'Two female gossips in converse free &mdash;',
11      'The subject engaging them was she.',
12      '"I think," said one, "and my husband thinks',
13      'That she\'s a prying, inquisitive minx! "',
14      'As soon as no more of it she could hear',
15      'The lady, indignant, removed her ear.',
16      '"I will not stay," she said, with a pout,',
17      '"To hear my character lied about! "',
18    ),
19    'author' => 'Gopete Sherany',
20  ),
21  'EDIBLE' => array(
22    'part' => 'adj.',
23    'definition' => 'Good to eat, and wholesome to digest, as
a worm to a toad, a toad to a snake, a snake to a pig,
```

```

24a pig to a man, anda man to a worm.',
25),
26'EDUCATION' => array(
27'part' => 'n.',
28'definition' => 'That which discloses to the wise and
29disguises from the foolish their lack of
30understanding.',
31),
32);
33?>
34

```

Ở trong những ứng dụng thật sự thì dữ liệu phải được lưu trữ trong cơ sở dữ liệu và chỉ được tải khi hỏi. Bởi vì dữ liệu là một phần của đoạn mã trên cho nên việc viết mã để lấy dữ liệu ra khá đơn giản. Chúng ta sẽ xem xét dữ liệu đã được tạo ra và viết mã HTML để hiển thị nó:

[?](#)

```

1 <?php
2 $term = strtoupper($_REQUEST['term']);
3 if (isset($entries[$term])) {
4     $entry = $entries[$term];
5     $html = '<div class="entry">';
6     $html .= '<h3 class="term">';
7     $html .= $term;
8     $html .= '</h3>';
9     $html .= '<div class="part">';
10    $html .= $entry['part'];
11    $html .= '</div>';
12    $html .= '<div class="definition">';
13    $html .= $entry['definition'];
14    if (isset($entry['quote'])) {

```

```

14$html .= '<div class="quote">';
15foreach ($entry['quote'] as $line) {
16$html .= '<div class="quote-line">'. $line .'
```

Bây giờ khi đoạn mã được truy vấn thì tệp e.php được gọi, và nó sẽ trả về một đoạn HTML phù hợp với điều kiện được gửi qua tham số của GET. Ví dụ khi bạn truy cập đoạn mã với e.php?term=eavesdrop, chúng ta sẽ có được.

## EAVESDROP

v.i.  
 Secretly to overhear a catalogue of the crimes and vices of another or yourself.  
 A lady with one of her ears applied  
 To an open keyhole heard, inside,  
 Two female gossips in converse free —  
 The subject engaging them was she.  
 "I think," said one, "and my husband thinks  
 That she's a prying, inquisitive minx!"  
 As soon as no more of it she could hear  
 The lady, indignant, removed her ear.  
 "I will not stay," she said, with a pout,  
 "To hear my character lied about!"  
 Gopete Sherany

Một lần nữa chúng ta thấy được trang kết quả không có chút định dạng nào bởi vì CSS chưa được áp dụng vào trang này.

Bởi vì chúng ta sẽ nghiên cứu dữ liệu được truyền tải đến server như thế nào, chúng ta sẽ sử dụng một phương thức khác để lấy mục từ thay vì chỉ sử dụng một dạng nút đơn từ trước tới giờ. Dưới đây là đoạn mã HTML

[?](#)

```
1 <div class="letter" id="letter-e">
2 <h3>E</h3>
3 <ul>
4 <li><a href="e.php?term=Eavesdrop">Eavesdrop</a></li>
5 <li><a href="e.php?term=Edible">Edible</a></li>
6 <li><a href="e.php?term=Education">Education</a></li>
7 <li><a href="e.php?term=Eloquence">Eloquence</a></li>
8 <li><a href="e.php?term=Elysium">Elysium</a></li>
9 <li><a href="e.php?term=Emancipation">Emancipation</a>
10</li>
11<li><a href="e.php?term=Emotion">Emotion</a></li>
12<li><a href="e.php?term=Envelope">Envelope</a></li>
13<li><a href="e.php?term=Envy">Envy</a></li>
14<li><a href="e.php?term=Epitaph">Epitaph</a></li>
15<li><a href="e.php?term=Evangelist">Evangelist</a></li>
16</ul>
17</div>
```

Bây giờ chúng ta cần mã JavaScript gọi đến PHP với tham số phù hợp. Chúng ta có thể làm được việc này với cơ chế `.load()`, gán chuỗi truy vấn vào URL và sau đó thì truy xuất dữ liệu trực tiếp với địa chỉ như kiểu `e.php?term=eavesdrop`. Tuy nhiên, thay vì làm như vậy chúng ta sẽ sử dụng jQuery để xây dựng chuỗi truy vấn dựa vào biểu đồ mà ta cung cấp cho hàm `$.get()`:

[?](#)

```
1$(document).ready(function() {
2$('#letter-e a').click(function() {
3$.get('e.php', {'term': $(this).text()}, function(data) {
```

```

4$('#dictionary').html(data);
5});
6return false;
7});
8});

```

Tới giờ chúng ta đã thấy những giao tác AJAX mà jQuery cung cấp, cách làm việc của hàm này rất quen thuộc. Điều khác biệt duy nhất là ở tham số thứ 2, nó cho phép chúng ta cung cấp một biểu đồ key-value và nó là một phần của chuỗi truy vấn. Trong trường hợp trên, giá trị key luôn là term nhưng value sẽ được lấy từ chữ của mỗi đường liên kết. Nên nếu bây giờ bạn nhấp chuột vào đường liên kết đầu tiên trong danh sách thì định nghĩa của từ đó sẽ xuất hiện.

[A](#)

[B](#)

[C](#)

[D](#)

[E](#)

[Eavesdrop](#)

[Edible](#)

[Education](#)

[Eloquence](#)

[Elysium](#)

[Emancipation](#)

[Emotion](#)

[Envelope](#)

[Envy](#)

[Epitaph](#)

[Evangelist](#)

## EAVESDROP *v.i.*

Secretly to overhear a catalogue of the crimes and vices of another

A lady with one of her ears applied  
To an open keyhole heard, inside,  
Two female gossips in converse free —  
The subject engaging them was she.  
"I think," said one, "and my husband thinks  
That she's a prying, inquisitive minx!"  
As soon as no more of it she could hear  
The lady, indignant, removed her ear.  
"I will not stay," she said, with a pout,  
"To hear my character lied about!"

Gopete She

Tất cả những đường liên kết đều có địa chỉ dù cho chúng ta không sử dụng nó trong mã. Điều này cho phép những người dùng không có hoặc không bật JavaScript vẫn có thể xem được thông tin trên trang. Để tránh đường liên kết di chuyển theo mặc định, bộ xử lý sự kiện phải là return false.

## Thực hiện lệnh truy vấn POST

Truy vấn HTTP sử dụng phương thức POST gần như tương đồng với phương thức GET. Một trong những khác biệt dễ thấy nhất đó là phương thức GET đặt tham số của nó vào chuỗi truy vấn của URL. Còn POST thì không. Tuy nhiên trong các cuộc gọi của AJAX, điểm khác biệt này cũng bị ẩn đi với người dùng. Nói chung, lý do chính để chọn phương thức này thay vì phương thức khác là để phù hợp với chuẩn của server, hoặc để truyền tải một lượng dữ liệu lớn. Phương thức GET có giới hạn nghiêm khắc hơn. Chúng ta đã viết mã PHP trong ví dụ này sao cho nó có thể làm việc được với cả 2 phương thức, để chúng ta có thể chuyển từ GET sang POST chỉ bằng cách thay đổi phương thức jQuery mà chúng ta gọi:

[?](#)

```
1$(document).ready(function() {  
2$('#letter-e a').click(function() {  
3$.post('e.php', {'term': $(this).text()}, function(data) {  
4$('#dictionary').html(data);  
5});  
6return false;  
7});  
8});
```

Các tham số thì vẫn vậy nhưng lệnh truy vấn sẽ được gửi qua POST. Chúng ta cũng có thể đơn giản hóa đoạn mã hơn nữa bằng cách sử dụng phương thức `.load()`. Phương thức này theo mặc định là sử dụng POST khi nó được cung cấp một biểu đồ tham số.

[?](#)

```
1$(document).ready(function() {  
2$('#letter-e a').click(function() {  
3$('#dictionary').load('e.php', {'term': $(this).text()});  
4return false;  
5});  
6});
```

Phiên bản mã ngắn hơn này vẫn có tác dụng tương tự khi chữ a được nhấp chuột.

**A**

**B**

**C**

**D**

**E**

[Eavesdrop](#)

[Edible](#)

[Education](#)

[Eloquence](#)

[Elysium](#)

[Emancipation](#)

[Emotion](#)

[Envelope](#)

[Envy](#)

[Epitaph](#)

[Evangelist](#)

## EMANCIPATION *n.*

A bondman's change from the tyranny of another to the despotism of himself.

He was a slave: at word he went and came;  
His iron collar cut him to the bone.  
Then Liberty erased his owner's name,  
Tightened the rivets and inscribed his own.

G.J.

Sắp xếp thứ tự form

Thường khi bạn muốn gửi dữ liệu đến server bạn được yêu cầu phải điền vào form. Thay vì phải phụ thuộc vào những cơ chế gửi form bình thường như kiểu tải toàn bộ câu trả lời vào một cửa sổ trình duyệt, chúng ta có thể sử dụng AJAX của jQuery để gửi một form theo thứ tự và đặt câu trả lời vào trang hiện tại. Dưới đây chúng ta sẽ tạo một form đơn giản:

[?](#)

```
1<div class="letter" id="letter-f">
2<h3>F</h3>
3<form>
4<input type="text" name="term" value="" id="term" />
5<input type="submit" name="search" value="search"
6id="search" />
7</form>
8</div>
```

Lần này chúng ta sẽ trả về một tập hợp các mục từ từ mã PHP bằng cách tìm kiếm từ khóa được cung cấp dưới dạng chuỗi phụ của từ trong từ điển. Cấu trúc dữ liệu sẽ có định dạng giống như trước đây, nhưng logic thì hơi khác một chút.

[?](#)

```
1
2 foreach ($entries as $term=> $entry) {
3   if (strpos($term, strtoupper($_REQUEST['term']))
4     !== FALSE) {
5     $html = '<div class="entry">';
6     $html .= '<h3 class="term">';
7     $html .= $term;
8     $html .= '</h3>';
9     $html .= '<div class="part">';
10    $html .= $entry['part'];
11    $html .= '</div>';
12    $html .= '<div class="definition">';
13    $html .= $entry['definition'];
14    if (isset($entry['quote'])) {
15      foreach ($entry['quote'] as $line) {
16        $html .= '<div class="quote-line">'. $line .'</div>';
17      }
18      if (isset($entry['author'])) {
19        $html .= '<div class="quote-author">'.
20          $entry['author'] .'</div>';
21      }
22    }
23    $html .= '</div>';
24    print($html);
25  }
26 }
```



Hàm `strops()` tìm từ khóa phù hợp với chuỗi tìm kiếm của người dùng. Bây giờ chúng ta có thể phản ứng lại với form gửi về và tạo tham số truy vấn phù hợp bằng cách di chuyển trong cây DOM:

[?](#)

```
1$(document).ready(function() {  
2$('#letter-f form').submit(function() {  
3$('#dictionary').load('f.php',  
4{'term': $('input[name="term"]').val()});  
5return false;  
6});  
7});
```

Tuy đoạn mã của chúng ta đã làm việc như mong muốn, nhưng để tìm từng trường nhập liệu bằng tên và sau đó gán từng cái một cho biểu đồ thì phiền phức quá. Hơn nữa cách này khó xử lý được khi mà form của chúng ta trở nên phức tạp hơn. Cũng may là jQuery có cách giúp chúng ta trong trường hợp này. Phương thức `.serialize()` hoạt động trên một đối tượng jQuery và chuyển những phần tử DOM phù hợp thành chuỗi truy vấn và chuyển nó cùng với AJAX truy vấn. Chúng ta có thể viết mã cho bộ xử lý form như sau:

[?](#)

```
1$(document).ready(function() {  
2$('#letter-f form').submit(function() {  
3$.get('f.php', $(this).serialize(), function(data) {  
4$('#dictionary').html(data);  
5});  
6return false;  
7});  
8});
```

Bây giờ đoạn mã đã có thể gửi form, cho dù số lượng các trường nhập liệu có tăng. Khi chúng ta muốn tìm kiếm, những mục từ phù hợp sẽ được hiển thị.

[A](#)

[B](#)

[C](#)

[D](#)

[E](#)

[Eavesdrop](#)

[Edible](#)

[Education](#)

[Eloquence](#)

[Elysium](#)

[Emancipation](#)

[Emotion](#)

[Envelope](#)

[Envy](#)

[Epitaph](#)

[Evangelist](#)

[F](#)

## The Devil's Dictionary: H

by Ambrose Bierce

### HABEAS CORPUS *n.*

A writ by which a man may be taken out of jail when

### HABIT *n.*

A shackle for the free.

### HALF *n.*

One of two equal parts into which a thing may be d  
In the fourteenth century a heated discussion arose  
philosophers as to whether Omniscience could par  
the pious Father Aldrovinus publicly prayed in the c  
would demonstrate the affirmative of the proposition

Nên chú ý đến lệnh truy vấn

Cho đến giờ chúng ta có thể tạo cuộc gọi đến phương thức AJAX và kiên nhẫn chờ đợi được trả lời. Nhưng cũng có lúc chúng ta muốn biết thêm một chút nữa về lệnh truy vấn HTTP khi nó được thực hiện. jQuery cho bạn một số hàm có thể được sử dụng để đăng ký hàm truy hồi khi nhiều sự kiện liên quan đến AJAX xảy ra.

Hai phương thức `.ajaxStart()` và `.ajaxStop()` là những ví dụ điển hình về chức năng quan sát, và có thể được gán với bất kỳ đối tượng jQuery nào. Khi lệnh gọi AJAX bắt đầu mà không có gì đang được tải, hàm truy hồi `.ajaxStart()` sẽ khởi động. Ngược lại, khi lệnh truy vấn cuối cùng kết thúc, hàm truy hồi được gán với `.ajaxStop()` sẽ bắt đầu. Tất cả những hàm quan sát là dạng hàm toàn cục, chúng được gọi mỗi khi sự giao tiếp AJAX xảy ra, mà không quan tâm đến mã nào gọi nó.

Chúng ta có thể sử dụng những phương thức này để thông báo cho người dùng biết trong trường hợp mạng của họ không được nhanh. Đoạn mã HTML sẽ có một đoạn thông báo “đang tải dữ liệu”:

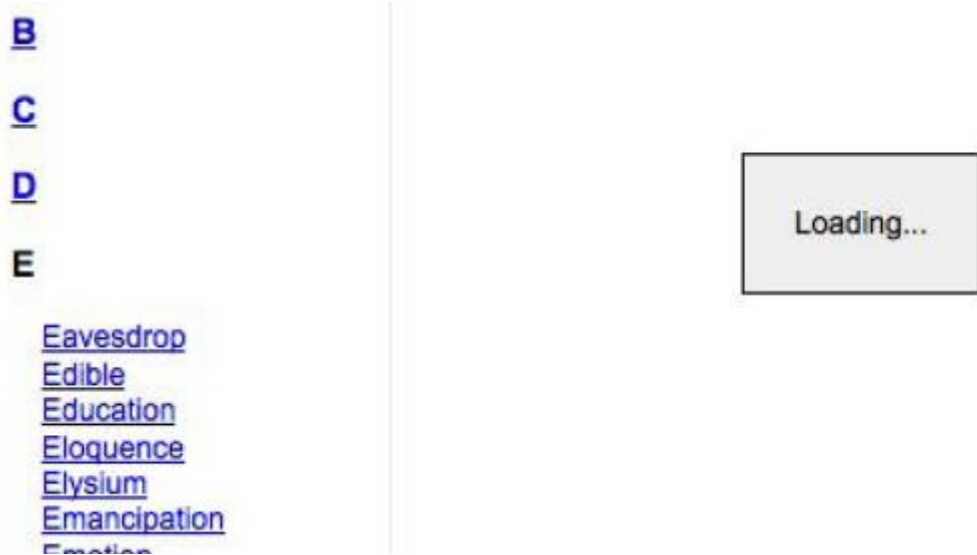
[?](#)

```
1<div id="loading">
```

2Loading...

3

Thông báo này chỉ là một đoạn mã HTML bình thường, nhưng bạn cũng có thể thêm vào một hình GIF động kiểu xoay xoay cho nó chuẩn. Chúng ta sẽ chỉnh sửa CSS một chút để khi thông báo được đưa ra nó được hiển thị như hình dưới.



Bởi vì đây chỉ là tính năng làm cho trang thêm đẹp cho những người dùng có trình duyệt hiện đại, do vậy chúng ta sẽ không chèn đoạn mã HTML này trực tiếp vào trang. Bởi vì chúng ta chỉ muốn nó hiển thị với những ai có bật JavaScript, cho nên chúng ta sẽ chèn nó bằng jQuery.

?

```
1$(document).ready(function() {
2$('<div id="loading">Loading...</div>')
3.insertBefore('#dictionary')
4});
```

Chúng ta sẽ khai báo trong CSS cho thẻ div này có display: none; để cho khi trang được tải, thì thông báo sẽ bị ẩn đi. Để nó hiển thị đúng lúc mình cần, chúng ta chỉ cần đăng ký nó với chức năng quan sát với .ajaxStart():

?

```
$(document).ready(function() {  
1  $('<div id="loading">Loading...</div>')  
2  .insertBefore('#dictionary')
```

```

3.ajaxStart(function() {
4$(this).show();
5});
6});
7

```

Chúng ta kết hợp phương thức `.hide()` luôn vào đây

[?](#)

```

1$(document).ready(function() {
2$('<div id="loading">Loading...</div>')
3.insertBefore('#dictionary')
4.ajaxStart(function() {
5$(this).show();
6}).ajaxStop(function() {
7$(this).hide();
8});
9});

```

Như vậy chúng ta đã có bảng thông báo.

Một lần nữa bạn cũng nên lưu ý rằng những phương thức này không liên quan gì đến cách mà giao tiếp AJAX bắt đầu. Chính phương thức `.load()` được gán cho chữ A và `.getJSON()` được gán cho chữ B đã làm cho giao tiếp AJAX xảy ra.

Trong trường hợp này, tập tính toàn cục là điều chúng ta muốn. Tuy nhiên nếu chúng ta muốn cụ thể hơn nữa, chúng ta có vài lựa chọn để sử dụng. Một vài chức năng quan sát như, `.ajaxError()`, nó sẽ gửi cho hàm truy hồi một tham chiếu đến đối tượng XMLHttpRequest. Cái này có thể dùng để phân biệt giữa các lệnh truy vấn với nhau, và cung cấp những tập tính khác nhau. Để có những cách xử lý cụ thể hơn bạn có thể sử dụng hàm `$.ajax()` cấp thấp, mà chúng ta sẽ bàn tới ở phần dưới.

Tuy nhiên cách phổ biến nhất để giao tiếp với lệnh truy vấn là hàm truy hồi thành công, mà chúng ta đã nói đến ở trên. Chúng ta đã sử dụng nó trong một vài những ví dụ trên để xử lý dữ liệu quay lại từ server và cho hiển thị kết quả lên trang web. Tất nhiên nó cũng có thể được sử dụng cho những thông tin phản hồi khác. Hãy xem lại ví dụ về `.load()`:

[?](#)

```

1$(document).ready(function() {
2$('#letter-a a').click(function() {
3$('#dictionary').load('a.html');
4return false;
5});
6});

```

Chúng ta có thể cải tiến một chút ở đây bằng cách làm cho nội dung từ từ hiện ra thay vì âm một phát. Phương thức `.load()` có thể lấy vào một hàm truy hồi và kích hoạt nó khi đã hoàn thành.

[?](#)

```

1$(document).ready(function() {
2$('#letter-a a').click(function() {
3$('#dictionary').hide().load('a.html', function() {
4$(this).fadeIn();
5});
6return false;
7});
8});

```

Trước tiên ta ẩn đi phần tử đích, và sau đó thì khởi động quá trình tải. Khi quá trình tải hoàn thành, chúng ta sử dụng hàm truy hồi để cho phần tử vừa tạo hiện ra từ từ.

## AJAX và sự kiên

Giả sử chúng ta muốn dùng các mục từ trong từ điển để quyết định ẩn hoặc hiện định nghĩa của từ đó, khi người dùng nhấp chuột vào từ thì nó sẽ ẩn hoặc hiện định nghĩa đi kèm với nó. Với những kỹ thuật ta đã học, thì để làm được việc này rất đơn giản

[?](#)

```

$(document).ready(function() {
1
2  $('#.term').click(function() {
3
4    $(this).siblings('.definition').slideToggle();
5  });
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

4}};

5

Khi mục từ bị nhấp chuột, jQuery sẽ tìm phần tử là “anh em họ” của nó mà có `class='definition'`, và trượt nó lên trên hoặc xuống dưới.

Mọi việc nghe có vẻ hợp lý, nhưng khi ta nhấp chuột vào thì sẽ không xảy ra việc gì. Vấn đề là mục từ chưa được thêm vào tài liệu khi ta gán bộ xử lý sự kiện. Cho dù ta có thể gán được bộ xử lý click vào các phần tử này, một khi mình nhấp chuột vào một chữ cái khác thì bộ xử lý sẽ không còn được gán cho nó nữa.

Đây là vấn đề thường thấy trong phạm vi của trang được đưa vào bởi AJAX. Một giải pháp thông thường là chúng ta sẽ gán lại bộ xử lý mỗi khi vùng của trang được refresh. Nhưng cách này cũng hơi mất thời gian bởi vì đoạn mã gán sự kiện phải được gọi mỗi khi có một thành phần nào làm thay đổi cấu trúc DOM của trang.

Một cách thay thế hay nhất cho vấn đề này được giới thiệu ở chương 3: Chúng ta có thể áp dụng ủy thác sự kiện ở đây, bằng cách gán sự kiện cho thành phần bố mẹ và đây là những thành phần sẽ không bao giờ thay đổi. Trong trường hợp này, chúng ta sẽ gán bộ xử lý sự kiện nhấp chuột vào tài liệu sử dụng phương thức `.live()`

?

```
1$(document).ready(function() {  
2$('.term').live('click', function() {  
3$(this).siblings('.definition').slideToggle();  
4});  
5});
```

Phương thức `.live()` sẽ hướng dẫn cho trình duyệt quan sát tất cả những cú nhấp chuột trên toàn bộ trang web, và chỉ khi một phần tử phù hợp với bộ chọn `.term`, thì bộ xử lý sự kiện mới thực hiện. Bây giờ phương thức `.slideToggle()` sẽ hoạt động dưới bất kỳ term nào, cho dù nó được thêm vào sau này bởi một giao tác AJAX

Hạn chế về bảo mật

Đối với tất cả các tiện ích của nó trong việc tạo các ứng dụng web động, XMLHttpRequest (công nghệ trình duyệt cơ bản đằng sau những ứng dụng jQuery AJAX) được quản lý rất nghiêm ngặt. Thường thì bạn không thể truy vấn được một tài liệu đang nằm ở một server khác với server đang chứa trang gốc của bạn, điều này để tránh những vụ tấn công cross-site.

Đây thực tế lại là một việc tốt. Ví dụ có người cho rằng cách thực thi phân tích JSON bằng cách sử dụng `.eval()` là không an toàn. Nếu có những đoạn mã độc nằm trong tệp dữ liệu, nó sẽ chạy nếu hàm `.eval()` gọi nó. Vậy nên nếu tệp dữ liệu và bản thân trang web cùng nằm trên một server thì khả năng chèn mã độc vào tệp dữ liệu gần như tương đương với việc tự chèn

mã vào trang của mình. Điều đó có nghĩa là, trong trường hợp bạn tải những tệp JSON không mang mã độc, thì hàm `.eval()` không còn là một mối lo cho bảo mật.

Nhưng cũng trong nhiều trường hợp lại có lợi hơn nếu bạn có thể tải dữ liệu từ một nguồn thứ 3. Có vài cách để bạn có thể làm để tránh được khâu giới hạn bảo mật và cho phép việc tải dữ liệu này có thể thực hiện được.

Cách thứ nhất là dựa vào server để tải dữ liệu từ xa và sau đó thì cung cấp nó khi được yêu cầu bởi người dùng. Đây là cách rất mạnh bởi vì server có thể xử lý trước dữ liệu nếu cần. Ví dụ chúng ta có thể tải file XML chứa RSS Feed từ nhiều nguồn, tập hợp chúng lại thành một feed trên server và phát hành tệp mới này đến người dùng khi được yêu cầu.

Để tải dữ liệu từ một vị trí từ xa mà không cần sự can thiệp của server, chúng ta phải ‘gian manh’ một tí. Một cách thường dùng trong trường hợp bạn muốn tải một tệp JavaScript bên ngoài là chèn cặp thẻ `<script>` khi cần. Bởi vì jQuery có thể giúp chúng ta chèn những phần tử DOM mới, nên chúng ta dễ dàng có thể viết:

[?](#)

```
1$(document.createElement('script'))  
2.attr('src', 'http://example.com/example.js')  
3.appendTo('head');
```

Thực tế, phương thức `$.getScript()` cũng tự động thích nghi với kỹ thuật này nếu nó phát hiện một host khác trong tham số URL, vậy nên kể cả việc này cũng đã được giải quyết cho chúng ta.

Trình duyệt sẽ chạy đoạn mã được tải, nhưng không có cơ chế nào có thể lấy về kết quả từ đoạn mã. Chính vì thế kỹ thuật này đòi hỏi sự cộng tác với host ở xa. Đoạn mã được tải về cũng phải làm một cái gì đó như là tạo ra một biến toàn cục và có hiệu lực trên môi trường cục bộ. Những ai tạo ra mã có thể chạy được bằng cách này cũng sẽ cung cấp một API để tương tác với mã từ xa này.

Một cách nữa là sử dụng thẻ HTML `<iframe>` để tải dữ liệu từ xa. Phần tử này cho phép bất cứ URL nào cũng được sử dụng làm nguồn để truy xuất dữ liệu của nó, cho dù nó không cùng một server. Dữ liệu dễ dàng được tải và hiển thị lên trang. Nhưng để thao tác với dữ liệu thì cũng đòi hỏi sự cộng tác như là cách sử dụng thẻ `<script>`. Mã nằm trong `<iframe>` cần phải cung cấp dữ liệu cho đối tượng trong tài liệu gốc một cách rõ ràng.

Sử dụng JSONP cho dữ liệu từ xa

Ý tưởng sử dụng thẻ `<script>` để truy xuất tệp JavaScript từ một nguồn ở xa cũng có thể sử dụng để kéo một file JSON từ một server khác. Nhưng để thực hiện được, chúng ta cần phải chỉnh sửa tệp JSON một chút. Cũng có vài cách để làm việc này, một trong số đó được hỗ trợ trực tiếp bởi jQuery: JSON với Padding hoặc viết tắt là JSONP.

Định dạng của tệp JSONP bao gồm một tệp JSON tiêu chuẩn đã được đặt trong dấu ngoặc đơn và gán vào đằng sau một chuỗi ký tự bình thường. Chuỗi này, hay còn là ‘padding’, được

xác định bởi người dùng đang truy vấn dữ liệu. Bởi vì hai dấu ngoặc này, người dùng có thể hoặc là làm cho hàm được gọi hoặc một biến được thiết lập phụ thuộc vào cái gì được gửi dưới dạng chuỗi padding.

Một ứng dụng PHP của kỹ thuật JSONP khá đơn giản:

?

```
1<?php
2print($_GET['callback'] . '(' . $data . ')');
3?>
```

Ở đây biến \$data chứa một chuỗi làm đại diện cho một tệp JSON. Khi đoạn mã này được gọi, một tham số chuỗi truy vấn callback được gắn vào trước tệp kết quả và sẽ được trả về cho người dùng.

Để minh họa cho kỹ thuật này, chúng ta chỉ cần sửa đổi một chút ví dụ về JSON ở trên để gọi nguồn dữ liệu từ xa. Hàm \$.getJSON() tận dụng ký tự giữ chỗ đặc biệt, ?, để làm được việc này.

?

```
1 $(document).ready(function() {
2   var url = 'http://examples.learningjquery.com/jsonp/g.php';
3   $('#letter-g a').click(function() { $.getJSON(url + '?callback=?',
4     function(data) {
5       $.each(data, function(entryIndex, entry) {
6         var html = '<div class="entry">';
7         html += '<h3 class="term">' + entry['term']
8         + '</h3>';
9         html += '<div class="part">' + entry['part']
10        + '</div>';
11        html += '<div class="definition">';
12        html += entry['definition'];
13        if (entry['quote']) {
14          html += '<div class="quote">';
15          $.each(entry['quote'], function(lineIndex, line) {
```



```

15html += '<div class="quote-line">' + line
16+ '</div>';
17});
18if (entry['author']) {
19    html += '<div class="quote-author">'
20    + entry['author'] + '</div>';
21    }
22    html += '</div>';
23    }
24    html += '</div>';
25    $('#dictionary').append(html);
26    });
27    });
28return false;
29    });
30    });
31
32

```

Thường thì chúng ta không được phép truy xuất JSON từ một server ở xa. Nhưng bởi vì file này được tạo nên để cung cấp dữ liệu của nó dưới dạng JSONP, chúng ta có thể lấy dữ liệu này bằng cách gán một chuỗi truy vấn vào URL, sử dụng dấu ? làm nơi lưu giữ cho giá trị của tham số hàm truy hồi. Khi truy vấn đã được tạo, jQuery sẽ thay thế dấu ? cho chúng ta, phân tích kết quả, và chuyển nó đến hàm dưới dạng dữ liệu như thể đó là một truy vấn JSON nội bộ.

Bạn cũng nên lưu ý về vấn đề bảo mật ở đây như trước, bất cứ cái gì server trả về đến trình duyệt đều sẽ được thực hiện trong máy tính của người dùng. Kỹ thuật JSONP chỉ nên được sử dụng với dữ liệu đến từ các nguồn có đáng tin cậy.

#### Lựa chọn thêm

Bộ công cụ AJAX được cung cấp bởi jQuery rất đầy đủ. Chúng ta đã xem qua một số lựa chọn, nhưng đó mới chỉ là bề nổi của tảng băng. Có quá nhiều thứ để có thể nói đến trong phần này, do vậy chúng ta chỉ khái quát qua một số những cách phổ biến để tùy biến giao tiếp AJAX.

## Phương pháp AJAX cấp thấp

Chúng ta đã thấy một số phương pháp khởi động giao tác AJAX. Nhưng đằng sau hậu trường, jQuery gom mỗi một phương thức này vào những hàm \$.ajax() toàn cục khác nhau. Thay vì phỏng đoán một dạng sự kiện AJAX, hàm này lấy vào một biểu đồ các sự lựa chọn mà có thể được sử dụng để tùy biến chế độ của nó.

Ví dụ đầu tiên mà chúng ta sử dụng \$('#dictionary').load('a.html') để tải một đoạn mã HTML. Cách này có thể được thay thế bằng phương thức \$.ajax() như sau:

[?](#)

```
1$.ajax({
2url: 'a.html',
3type: 'GET',
4dataType: 'html',
5success: function(data) {
6$('#dictionary').html(data);
7}
8});
```

Chúng ta cần phải hết sức cụ thể với phương thức truy vấn, loại dữ liệu trả về, và sẽ làm gì với kết quả dữ liệu đó. Cái này có vẻ hơi tốn công sức nhưng bù lại với công sức bạn bỏ ra là thành quả mỹ mãn. Một vài trong số những khả năng đặc biệt khi sử dụng với những phương thức \$.ajax() cấp thấp bao gồm:

- Ngăn không cho trình duyệt lưu lại sự phản hồi từ server. Điều này có ích khi mà server tự động tạo ra dữ liệu của nó.
- Đăng ký riêng biệt hàm truy hồi cho dù khi lệnh truy vấn thực hiện thành công, bị lỗi hoặc trong tất cả các trường hợp.  
Chặn bộ xử lý toàn cục (như là bộ xử lý đăng ký với \$.ajaxStart()) mà thường được khởi động bởi tất cả các tương tác AJAX.
- Cung cấp Username và mật khẩu để xác nhận với host từ xa.

**Chú ý:** để biết thêm chi tiết về cách sử dụng những lựa chọn khác, xem thêm phần *jQuery Reference Guide* hoặc xem phần *API Reference* tại (<http://docs.jquery.com/Ajax/jQuery.ajax>).

## Chỉnh sửa tùy chọn mặc định

Hàm \$.ajaxSetup() cho phép chúng ta định rõ giá trị mặc định của mỗi tùy chọn được sử dụng khi phương thức AJAX được gọi. Nó cũng lấy vào một biểu đồ tùy chọn giống y như biểu đồ có trong bản thân \$.ajax(), và làm cho những giá trị này được sử dụng cho những truy vấn AJAX về sau trừ khi có cái khác mạnh hơn.

[?](#)

```
1 $.ajaxSetup({
2   url: 'a.html',
3   type: 'POST',
4   dataType: 'html'
5 });
6 $.ajax({
7   type: 'GET',
8   success: function(data) {
9     $('#dictionary').html(data);
10  }
11 });
```

Dãy thao tác này hoạt động giống như ví dụ trước là \$.ajax(). Bạn nên chú ý rằng URL của lệnh truy vấn được xác định làm giá trị mặc định bởi cuộc gọi \$.ajaxSetup(), cho nên giá trị này có thể được bỏ trống khi \$.ajax() được gọi. Ngược lại, tham số type có giá trị mặc định là POST, nhưng nó vẫn bị đề lên bởi cuộc gọi \$.ajax() đến GET.

#### Tải các phần của một trang HTML

Kỹ thuật đầu tiên và cũng là đơn giản nhất mà chúng ta đã thảo luận ở trên là truy xuất một đoạn code HTML và chèn nó vào một trang. Nhưng cũng có khi server đã cung cấp cho ta đoạn HTML mình cần nhưng nó lại bị bao quanh bởi một trang HTML khác mà ta không muốn. Khi mà việc yêu cầu server cung cấp định dạng chúng ta muốn không thuận tiện, jQuery có thể giúp chúng ta ở phía người dùng.

Hãy tưởng tượng trường hợp như ở ví dụ đầu tiên, nhưng trang chứa các từ mục lại là một trang HTML hoàn chỉnh như sau:

[?](#)

```
1 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
2   lang="en">
3   <head>
4     <meta http-equiv="Content-Type"
5       content="text/html; charset=utf-8"/>
6     <title>The Devil's Dictionary: H</title>
```

```
6 <link rel="stylesheet" href="dictionary.css"
7 type="text/css" media="screen" />
8 </head>
9 <body>
10 <div id="container">
11   <div id="header">
12     <h2>The Devil's Dictionary: H</h2>
13     <div class="author">by Ambrose Bierce</div>
14     </div>
15     <div id="dictionary">
16       <div class="entry">
17         <h3 class="term">HABEAS CORPUS</h3>
18         <div class="part">n.</div>
19         <div class="definition">
20           A writ by which a man may be taken out of jail
21           when confined for the wrong crime.
22         </div>
23       </div>
24       <div class="entry">
25         <h3 class="term">HABIT</h3>
26         <div class="part">n.</div>
27         <div class="definition">
28           A shackle for the free.
29         </div>
30       </div>
31     </div>
32   </body>
33 </html>
```

34

35

Chúng ta có thể tải toàn bộ tài liệu vào trong trang bằng cách sử dụng đoạn mã ta viết trước đây:

[?](#)

```
1$(document).ready(function() {  
2$('#letter-h a').click(function() {  
3$('#dictionary').load('h.html');  
4return false;  
5});  
6});
```

Bạn thấy trang web không được bình thường bởi vì nó chứa những đoạn HTML chúng ta không muốn thêm vào.

**A**

**B**

**C**

**D**

**E**

[Eavesdrop](#)

[Edible](#)

[Education](#)

[Eloquence](#)

[Elysium](#)

[Emancipation](#)

[Emotion](#)

[Envelope](#)

[Envy](#)

[Epitaph](#)

[Evangelist](#)

**F**

**FIDDLE** *n.*  
An instrument to tickle human ears by friction of a horse's tail on the entrails of a cat.  
To Rome said Nero: "If to smoke you turn  
I shall not cease to fiddle while you burn."  
To Nero Rome replied: "Pray do your worst,  
'Tis my excuse that you were fiddling first."  
Orm Pludge

**FIDELITY** *n.*  
A virtue peculiar to those who are about to be betrayed.

Để loại bỏ những đoạn dư này, chúng ta có thể sử dụng một tính năng mới của phương thức `.load()`. Khi bạn khai báo URL của tài liệu cần tải, chúng ta cũng có thể cung cấp một biểu thức bộ chọn jQuery. Nếu đã khai báo, biểu thức này sẽ được sử dụng để xác định một phần mã của tài liệu. Chỉ những phần nào phù hợp với bộ chọn mới được chèn vào trang. Trong trường hợp này, chúng ta có thể sử dụng kỹ thuật này để kéo chỉ những mục từ nằm trong tài liệu và chèn nó:

?

```
1$(document).ready(function() {  
2$('#letter-h a').click(function() {  
3$('#dictionary').load('h.html .entry');  
4return false;  
5});  
6});
```

Bây giờ những phần không liên quan của tài liệu đã được loại bỏ khỏi trang

**A**

**B**

**C**

**D**

**E**

[Eavesdrop](#)

[Edible](#)

[Education](#)

[Eloquence](#)

[Elysium](#)

[Emancipation](#)

[Emotion](#)

[Envelope](#)

[Envy](#)

[Epitaph](#)

[Evangelist](#)

**F**

## **HABEAS CORPUS** *n.*

A writ by which a man may be taken out of jail when confined for the

## **HABIT** *n.*

A shackle for the free.

## **HALF** *n.*

One of two equal parts into which a thing may be divided, or considered as such. In the sixteenth century a heated discussion arose among theologians and philosophers as to whether it was possible to part an object into three halves; and the pious Father Aldrovandus, in his cathedral at Rouen that God would demonstrate the affirmative of the matter in an unmistakable way, and particularly (if it should please Him) upon the person of the Manutius Procinus, who maintained the negative. Procinus, however, was a viper.

## **HAND** *n.*

A singular instrument worn at the end of the human arm and commonly used for holding a racket.

## Tóm tắt

Chúng ta đã học được rằng những phương thức AJAX cung cấp bởi jQuery có thể giúp chúng ta tải dữ liệu dưới một số định dạng khác nhau từ server mà không cần phải refresh lại trang. Chúng ta có thể thực hiện mã từ server khi cần và gửi dữ liệu quay lại server.

Chúng ta cũng học được cách để xử lý với những khó khăn thường gặp của kỹ thuật tải không đồng bộ như là giữ bộ xử lý ở nguyên vị trí khi quá trình tải bắt đầu và tải dữ liệu từ một server thứ 3.

Chương này đã khép lại phần tutorial của cuốn sách. Chúng ta đã có đủ những công cụ chủ yếu của jQuery: bộ chọn, sự kiện, hiệu ứng, thao tác DOM và truy vấn server không đồng bộ. Đây không phải là tất cả jQuery có thể hỗ trợ ta, chúng ta sẽ tìm hiểu thêm về một vài tính năng mà các jQuery Plugin đem lại trong chương tới. Nhưng trước hết, chúng ta hãy xem xét những kết hợp của các kỹ thuật đã học để làm cho trang web của chúng ta hấp dẫn hơn.

## Tài liệu tổng hợp từ:

- <http://www.izwebz.com/>
- Microsoft Vietnam – DPE Team