

# Giáo trình HTML5

## MỤC LỤC

1. Giới thiệu giáo trình HTML5 .....	3
(1) HTML5 tag ( Thẻ mới trong HTML5) .....	3
(2) HTML5 Attribute ( thuộc tính mới trong HTML5).....	3
(3) HTML5 Graphics (HTML5Giao diện đồ họa ) .....	4
(4)HTML5 Media (Đa phương tiện trên HTML5).....	4
(5) HTML API.....	4
2. HTML5 Tag (thẻ ) .....	4
(1) Các thẻ định nghĩa nhóm nội dung .....	5
(2) Các thẻ định nghĩa nội dung cụ thể, control html.....	7
(1) Thẻ details : .....	7
(2) Thẻ datalist.....	8
(3) Thẻ dialog .....	9
(4) Thẻ embed.....	9
(5) Danh sách các thẻ HTML5 định nghĩa nội dung.....	9
(3) Các thẻ đặc biệt khác .....	10
(4) Những Tag không được HTML5 hỗ trợ .....	11
3. HTML5 Attribute ( thuộc tính HTML5) .....	11
3.1 Attribute hỗ trợ nhập liệu .....	12
3.2 Attribute hỗ trợ kiểm tra nhập liệu .....	12
3.3Các ví dụ.....	13
(1) Form nhập liệu .....	13
(2) Xây dựng biểu thức chính qui (Regular) kiểm tra nhập liệu trong HTML5 .....	16
(3) Form tự động điền với Autocomplete.....	18
(4) Thuộc tính form .....	19
4. HTML 5 đồ họa Canvas - SVG.....	20
4.1 HTML5 Canvas .....	20
(1) Vẽ đường thẳng trong canvas .....	20
(2) Vẽ 1 đường tròn trong canvas : .....	21
(3) Vẽ ảnh trong canvas.....	23
(4) Di chuyển ảnh trên canvas .....	26
(5) Viết chữ lên canvas .....	27

(6) Vẽ đường cong bậc hai Quadratic Curve.....	28
(7) Vẽ đường cong Bizier Curve .....	30
(8) Vẽ đường Canvas Path.....	31
(9) Vẽ đường tùy chỉnh (Sharp) .....	32
(10) Xây dựng thư viện lập trình OOP JavaScript hỗ trợ lập trình HTML5.....	33
(11) Lập trình cờ tướng bằng canvas HTML5 .....	34
4.2 HTML5 SVG.....	47
5. HTML 5 đa phương tiện (Media).....	48
5.1 Phát Video trong HTML5 .....	48
(1) Trình xem video mặc định.....	48
(2) Tùy chỉnh trình mở video .....	49
5.2 Phát nhạc trong HTML5.....	50
6. HTML5 API .....	50
6.1 Xác định vùng địa lý với HTML5 Geolocation .....	50
(1) Giới thiệu về Geolocation.....	50
(2) Hàm xử lý lỗi cho Geolocaltion.....	52
(3) Hiển thị vị trí người dùng lên bản đồ google maps .....	54
6.2. HTML5 Drag và Drop (kéo - thả trong HTML5) .....	55
(1) Kéo thả 1 chiều .....	55
(2) Kéo thả 2 chiều .....	56
6.3 HTML5 Web Storage(lưu trữ dữ liệu tại Client trong HTML5) .....	59
(1) Đối tượng Web Storage .....	59
(2) Đối tượng localStorage .....	60
(3) Đối tượng sessionStorage .....	60
(4) Lưu trữ mảng đối tượng vào web Storage.....	61
6.4 Bộ nhớ Cache cho ứng dụng trong HTML5 .....	62
7. HTML5 Web Worker.....	63
8. Tổng kết HTML5 và sự thay thế Flash, Silverlight của HTML5 .....	65
9. Phụ lục các tag trong HTML5 .....	65

# 1. Giới thiệu giáo trình HTML5

- HTML5 mở ra 1 cuộc cách mạng trong lập trình giao diện web , từ giao diện đồ họa ,API, nhập liệu,tag ...

- Đó là lý do đáng để chúng ta học HTML5 để áp dụng vào website thay cho các công nghệ truyền thống như flash , silverlight

- Tài liệu cung cấp đầy đủ các tính năng của công nghệ HTML5 cho tới thời điểm này (HTML5 vẫn đang được nghiên cứu và hoàn thiện phát triển ) .Chúng tôi cung cấp các kiến thức tổng quan , cốt lõi để các bạn hiểu và có thể áp dụng vào ứng dụng của mình .

- Ngoài ra nền tảng về HTML5 là khá lớn .Tài liệu cung cấp được các khái niệm về tag HTML5 , HTML5 API , HTML Media , HTML5 Attribute để các bạn có thể áp dụng ngay vào thực tiễn .

- Với nền tảng HTML5 đồ họa ( canvas , svg) chúng tôi cũng cung cấp các khái niệm cơ bản và rất chi tiết .Cuối chương chúng tôi còn cung cấp chi tiết thiết kế ứng dụng game cờ tướng bằng HTML5 .Tuy nhiên mã javascript còn khá nặng nhưng khá cần thiết cho các bạn mới tìm hiểu về HTML5 game hơi là dùng thêm các thư viện bên ngoài .Vấn đề phát triển game với HTML5 khá rộng và bạn đọc nào có nhu cầu tìm hiểu có thể đọc tài liệu thiết kế game với HTML5 của chúng tôi.

- HTML 5 cung cấp 1 số tính năng mới như sau :

## (1) HTML5 tag ( Thẻ mới trong HTML5)

- HTML5 cung cấp thêm 1 thẻ mới (tag) để định nghĩa nội dung tài liệu HTML như :

main , nav ,aside , datalist , progress...

- Các thẻ datalist , progress ... thực ra là các Control HTML mới giúp xây dựng giao diện tương tác người dùng tốt hơn .

- Các thẻ như main , nav , aside ... lại là các thẻ trung tính như các thẻ div .Chúng ra đời với mục đích chuẩn hóa tài liệu HTML.Như là trong 1 trang nhìn vào các tag HTML thì có thể biết đâu là nội dung chính (mai) , đâu là thanh điều hướng menu ( nav ) hay nội dung liên quan (aside) .Các thẻ này có ý nghĩa khá lớn trong việc thiết kế website theo chuẩn SEO .Các cỗ máy tìm kiếm google , binding dễ dàng index nội dung website .Do đó các thẻ này là 1 thay thế đáng được áp dụng trong việc thiết kế layout website thay cho thẻ div

## (2) HTML5 Attribute ( thuộc tính mới trong HTML5)

- HTML5 cung cấp 1 số attribute hỗ trợ nhập liệu cho thẻ form như : number , email , datetime...

- Đây cũng là 1 tính năng khá mạnh .Trong lập trình web chúng ta thường phải tạo ra các form nhập liệu để người dùng nhập và gửi dữ liệu lên server xử lý

- Trong nhập liệu về bản chất là chúng ta dùng các ô textbox nhập liệu .Tuy nhiên trong 1 vài yêu cầu chúng ta muốn hỗ trợ người dùng nhập liệu các kiểu dữ liệu như màu sắc , ngày tháng thì trước kia chúng ta thường dùng các thư viện Javascript tạo datetime Control , Color Control .Nhưng hiện nay HTML5 đã hỗ trợ rất nhiều thẻ nhập liệu như vậy và giúp chúng ta không cần phải dùng thêm thư viện bên ngoài

- Ngoài ra trong nhập liệu chúng ta còn có yêu cầu là kiểm tra nhập liệu (client validation) xem người dùng nhập có đúng kiểu dữ liệu không như là : các ô phải nhập dữ liệu , các ô nhập dữ liệu là số , email .Trước kia để làm điều này chúng ta phải viết biểu thức chính qui (Regular Expression) bằng javascript để kiểm tra nhập liệu .Hoặc dùng thêm các thư viện javascript như jquery validate , kedo ui ... Nhưng hiện tại HTML5 đã hỗ trợ biểu thức chính qui khá đầy đủ

### **(3) HTML5 Graphics (HTML5Giao diện đồ họa )**

- HTML5 cung cấp 2 kỹ thuật đồ họa thông qua 2 thẻ canvas và svg

- canvas là hệ thống giao diện đồ họa pixel .Dùng javascript để vẽ

- svg là hệ thống giao diện đồ họa vecto dùng xml để vẽ

- Với canvas và svg chúng ta có thể vẽ bất cứ thứ gì trên web : Điểm , đường thẳng , đường cong , hình ảnh .

- Đồng thời kết hợp với các kỹ thuật lập trình chúng ta hoàn toàn có thể tạo nên các di chuyển đồ họa ( thực ra di chuyển chỉ là việc biến đổi vị trí hình ảnh)

- Do đó html5 canvas , svg hứa hẹn tạo nên nền tảng lập trình game khá tốt thay cho flash , silverlight

### **(4)HTML5 Media (Đa phương tiện trên HTML5)**

- HTML5 cung cấp 2 thẻ video và audio để cho phép mở nhạc và video trên nền tảng web

- Ngoài ra kết hợp với các HTML5 API chúng ta có thể tùy biến trình nghe nhạc , xem mang phong cách riêng .

- Các bạn có thể xem việc trình diễn video dùng HTML5 từ trang <http://www.youtube.com>

thì có thể nhận ra rằng chúng ta có thể hoàn toàn tự xây dựng lên các Control xem nhạc , video từ HTML5

### **(5) HTML API**

- Application Programming Interface :Giao diện ứng dụng lập trình

- API là các hàm , phương thức để cho các ứng dụng bên ngoài có thể gọi , tương tác để trao đổi thông tin , tính toán.

- Việc trao đổi này giúp các nhà lập trình tạo ra các service hỗ trợ những lập trình viên khác có thể tương tác với ứng dụng của chính mình

- Hiện nay trên web các dịch vụ của google , facebook cung cấp rất nhiều api để lập trình viên có thể xây dựng tương tác giữa website của họ với google ,facebook

- Mỗi phần mềm có các cung cấp các API để các ứng dụng khác có thể tương tác với nó

- HTML5 cung cấp rất nhiều API dưới dạng các đối tượng javascript để chúng ta có thể tương tác được với các ứng dụng web .

- Với HTML5 API chúng ta có các tương tác về xử lý lưu dữ liệu tại client ( Web Storage) , cache , kéo thả đối tượng ...

## **2. HTML5 Tag (thẻ )**

- HTML5 cung cấp nhiều thẻ html mới với 2 ý nghĩa :

+) Thẻ dùng để qui định nội dung hiển thị : main , nav , aside

+) Thẻ dùng tạo ra các Control mới :datalist,progress

- Việc dùng các Control mới giúp tạo ra nhiều kiểu trình bày nội dung và làm website thêm sinh động

- Các thẻ qui định nội dung để làm code html trở nên rõ ràng , mạch lạc và đặc biệt giúp ích rất nhiều cho các công cụ tìm kiếm .Đó là 1 lợi thế khi làm SEO website với HTML5

### (1) Các thẻ định nghĩa nhóm nội dung

HTML5 Tag	Mô tả
header	Thẻ hiện nội dung trên cùng .Thường là tiêu đề website , tiêu đề 1 nội dung
footer	Thẻ hiện nội dung cuối cùng .Chân trang website , nội dung cuối của tài liệu
nav	Nơi chứa các link điều hướng website .Menu chính của website thường đặt trong thẻ này
aside	Định nghĩa phần cố định,nội dung chính (menu dọc). Ta hiểu aside như là 1 phần ghi chú , phần nội dung phụ Aside thường chứa nội dung bên ngoài nội dung chính (thường là phần sidebar)
article	Định nghĩa nội dung độc lập riêng biệt ,có thể chứa nhiều section và ngược lại
section	Chứa một nội dung cụ thể của website
hgroup	Định nghĩa một nhóm các tiêu đề.

- Chúng ta sẽ làm 1 ví dụ dùng các thẻ HTML5 để xây dựng Layout ( bộ khung) giao diện website

- Trước kia tất cả các nội dung chúng ta thường dùng thẻ div để nhóm khối .Việc dùng này về ý nghĩa lập trình thì không vấn đề gì cả .Nhưng code trở nên khó đọc và không thân thiện với các công cụ tìm kiếm .

- Dĩ nhiên các thẻ như header , footer ,aside ngoài dùng thiết kế layout cho nội dung website còn có thể dùng cho thiết kế 1 phần nhỏ nội dung cho website .

- Ta tạm hiểu các thẻ này qui định các tiêu đề , chân tiêu đề , nội dung chính , nội dung cố định của 1 nội dung tài liệu html .Trong tài liệu HTML lại chia thành các tài liệu con , và ta lại có thể dùng các thẻ này định nghĩa cho các nội dung con

### Dựng layout HTML5:

Mã HTML :

```
<!DOCTYPEhtml>  
<htmlxmlns="http://www.w3.org/1999/xhtml">  
<head>  
<title></title>  
</head>  
<body>  
<header>  
<h1>Tiêu đề wesite</h1>  
<nav>  
<a href="#">Trang Chủ</a>  
<a href="#">Giới Thiệu</a>  
<a href="#">Liên Hệ</a>  
</nav>
```

```

</header>
<divclass="main">
<main>
<article><p>Phần 1 :Nội dung chính website</p></article>
<article><p>Phần 2 :Nội dung chính website</p></article>
</main>
<aside>
<h2> Nội dung sidebar</h2>
<h3>Link 1</h3>
<h3>Link 2</h3>
<h3>Link 3</h3>
</aside>
</div>
<footer>
<h1>Chân trang website</h1>
</footer>
</body>
</html>

```

Mã CSS :

```

.main{
display:table;
width:95%;
}
main{
width:70%;
display:table-cell;
background:#b6ff00;
}
aside{
width:30%;
display:table-cell;
background:#00ffff;
}

```

Chạy ứng dụng được kết quả như hình sau :



- Ta thấy mã html5 định dạng layout rất đơn giản, ta chỉ thêm duy nhất đó là class của thẻ chứa nội dung chính

- Ở đây ta dùng thuộc tính display:table để định dạng Layout .Cũng như thuộc tính display:inline-block chưa được các trình duyệt cũ hỗ trợ .

- Để nhiều trình duyệt hỗ trợ ta vẫn dùng thuộc tính float:left để định dạng layout
- Với CSS3 ta có 1 kỹ thuật làm layout động , rất mềm dẻo đó là dùng thuộc tính display:box
- Các bạn có thể xem tài liệu CSS3 của chúng tôi để hiểu hết tất cả các kỹ thuật làm layout trong CSS
- Chú ý thẻ main chưa được trình duyệt IE (Internet Explorer) hỗ trợ .Do đó chúng ta nên thay chúng bằng thẻ article và trong thẻ article ta thay bằng các thẻ section như sau :

```

<header>
<h1>Tiêu đề website</h1>
<nav>
<a href="#">Trang Chủ</a>
<a href="#">Giới Thiệu</a>
<a href="#">Liên Hệ</a>
</nav>
</header>
<div class="main">
<article>
<section><p>Phần 1 :Nội dung chính website</p></section>
<section><p>Phần 2 :Nội dung chính website</p></section>
</article>
<aside>
<h2> Nội dung sidebar</h2>
<h3>Link 1</h3>
<h3>Link 2</h3>
<h3>Link 3</h3>
</aside>
</div>
<footer>
<h1>Chân trang website</h1>
</footer>

```

- Thông qua việc định nghĩa layout các bạn thấy tài liệu HTML5 được định nghĩa bằng các tag có ý nghĩa cụ thể , rất dễ đọc code.Không phải các thuộc tính chung chung như trước kia !.

## (2) Các thẻ định nghĩa nội dung cụ thể, control html

- Ngoài các tag gom nhóm có ý nghĩa mô tả nội dung bên trong nó , HTML5 còn hỗ trợ các tag định nghĩa 1 nội dung cụ thể , hay các control mới để tăng tính tương tác trực quan trong html .

- Chúng tôi ví dụ mẫu 1 vài thẻ dưới đây

### (1) Thẻ details :

- Tag details dùng tạo ra 1 Control gồm 2 phần tiêu đề và nội dung.Mặc định phần nội dung bị ẩn đi .Khi click vào tiêu đề thì hiện ra nội dung chi tiết

- Tag summary đi cùng tag details để mô tả tiêu đề cho tag details

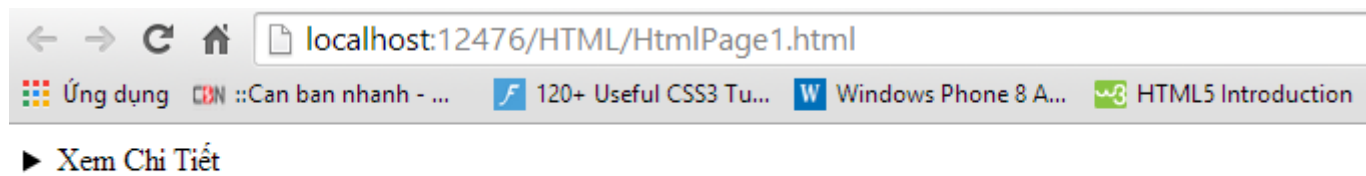
Ví dụ :

```

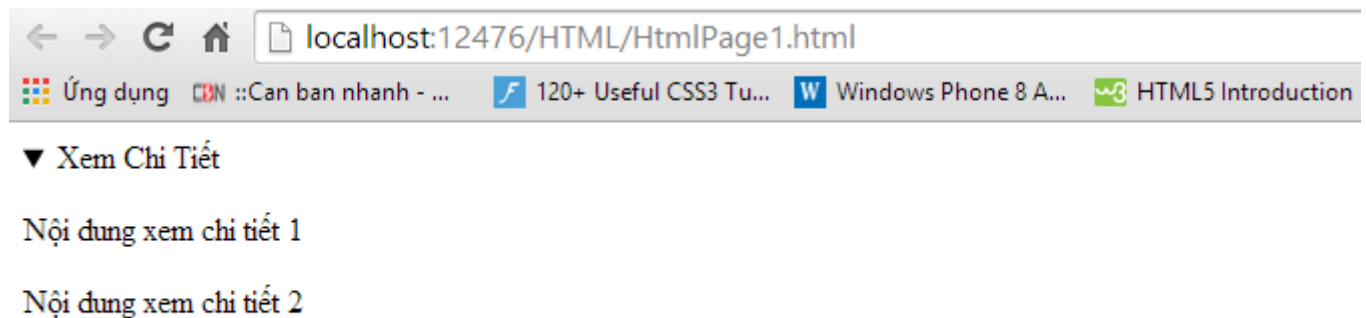
<details>
<summary>Xem Chi Tiết</summary>
<p>
        Nội dung xem chi tiết 1
</p>
<p>
        Thẻ detail chỉ hỗ trợ Chrome, Opera,Safari
</p>
</details>

```

Chạy ví dụ được kết quả như hình sau :



- Khi Click vào biểu tượng tại tiêu đề sẽ hiển thị nội dung chi tiết như sau :



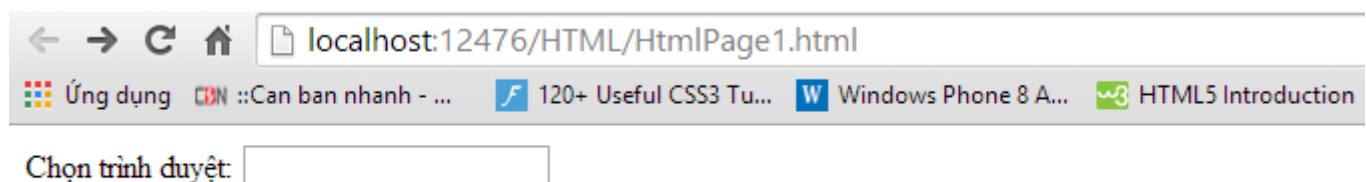
## (2) Thẻ datalist

- Thẻ datalist tương tự như thẻ select , dùng hiển thị ra danh sách nhập liệu (ComboBox)
- Thẻ datalist thường được dùng trong các form nhập liệu (input form)

HTML :

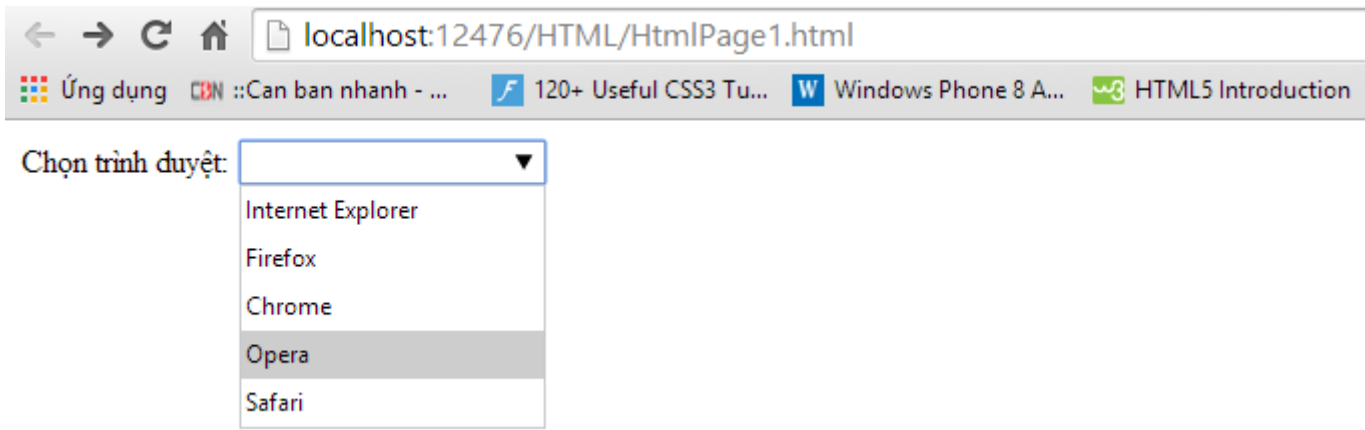
```
<label>Chọn trình duyệt:</label>
<inputlist="browsers" name="browser">
<datalistid="browsers">
<optionvalue="Internet Explorer">
<optionvalue="Firefox">
<optionvalue="Chrome">
<optionvalue="Opera">
<optionvalue="Safari">
</datalist>
```

Chạy ví dụ được kết quả như hình sau :



- Click vào ô nhập liệu sẽ hiển thị danh sách như hình sau :





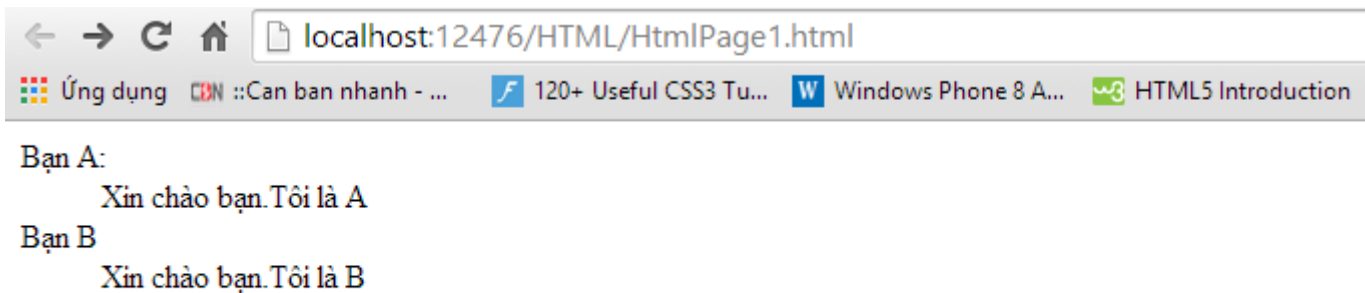
### (3) Thẻ dialog

-Thẻ dialog cho phép đánh dấu một cuộc đàm thoại giữa nhiều người trên mạng. Trong đó <dt> xác định người nói và <dd> chứa nội dung đàm thoại.

HTML :

```
<dialog>
<dt>Bạn A:</dt>
<dd>Xin chào bạn.Tôi là A</dd>
<dt>Bạn B </dt>
<dd>Xin chào bạn.Tôi là B</dd>
</dialog>
```

- Chạy ví dụ được kết quả như hình sau :



### (4) Thẻ embed

- Thẻ embed được dùng để nhúng các plugin cho ứng dụng web .Giải sử với thẻ embed chúng ta có thể chèn flash , silverlight vào trang web như ví dụ sau :

```
<embedsrc="helloworld.swf">
```

scr chỉ định khai báo tới file flash

### (5) Danh sách các thẻ HTML5 định nghĩa nội dung

- Ngoài các thẻ hay dùng chúng tôi ví dụ ở trên , HTML5 cung cấp khá nhiều thẻ mới .Các bạn có thể tham khảo ở bảng dưới đây

Tag	Mô tả
<a href="#">&lt;details&gt;</a>	Xác định thêm chi tiết hoặc điều khiển có thể được ẩn hoặc hiển thị theo yêu cầu.

<a href="#">&lt;summary&gt;</a>	Xác định một tiêu đề cho các thành phần details, được sử dụng để mô tả chi tiết về tài liệu, hoặc các bộ phận của tài liệu. Tag summary đi cùng với tag details
<a href="#">&lt;datalist&gt;</a>	Định nghĩa một danh sách tùy chọn, sử dụng thành phần này cùng với các thành phần input.
<a href="#">&lt;dialog&gt;</a>	Định nghĩa một dialog box hoặc window.
<a href="#">&lt;wbr&gt;</a>	Xác định text quá dài sẽ tự động xuống hàng (không tràn layout)
<a href="#">&lt;embed&gt;</a>	Xác định nội dung nhúng như một plugin. Thường dùng để chèn flash , silverlight ...
<a href="#">&lt;figcaption&gt;</a>	Xác định một chú thích cho tag figure.
<a href="#">&lt;figure&gt;</a>	Xác định các nội dung liên quan mạch lạc với nhau, như hình ảnh, sơ đồ, code,...
<a href="#">&lt;keygen&gt;</a>	Xác định một cặp trường khóa chính sử dụng cho form.
<a href="#">&lt;mark&gt;</a>	Xác định văn bản được đánh dấu, sử dụng khi muốn làm nổi bật văn bản của mình. Như là đánh dấu các nội dung lặp đi lặp lại trong 1 cuốn sách để làm nổi bật chúng : <m>HTML5</m> cung cấp nhiều tính năng mạnh mẽ <m>HTML5</m> mở ra nhiều kỹ thuật lập trình web mới
<a href="#">&lt;meter&gt;</a>	Định nghĩa một phép đo. Sử dụng chỉ cho phép đo với giá trị tối thiểu và tối đa.
<a href="#">&lt;output&gt;</a>	Đại diện cho kết quả của phép tính (giống như được thực hiện bởi script).
<a href="#">&lt;command&gt;</a>	Định nghĩa một nút lệnh, giống như một button, Radiobutton, hộp kiểm
<a href="#">&lt;progress&gt;</a>	Mô tả tiến trình làm việc. <progress value="1534602" max="4603807">33%</progress>
<a href="#">&lt;rp&gt;</a>	Hiển thị những nội dung bên trong khi trình duyệt không hỗ trợ ruby.
<a href="#">&lt;rt&gt;</a>	Định nghĩa một lời giải thích hoặc cách phát âm của các ký tự (đối với kiểu chữ Đông Á).
<a href="#">&lt;ruby&gt;</a>	Định nghĩa một chú thích ruby (đối với kiểu chữ Đông Á). Chú thích Ruby được sử dụng trong khu vực Đông Á, hiển thị cách phát âm của các ký tự Đông Á.
<a href="#">&lt;time&gt;</a>	Xác định thời gian, ngày tháng, hoặc năm sinh. <time>5:35 P.M. on April 23rd</time>

### (3) Các thẻ đặc biệt khác

HTML5 cung cấp 1 số thẻ đặc biệt khác như là :

canvas: Vẽ đối tượng đồ họa pixel lên website (dùng javascript)

svg : Vẽ đối tượng đồ họa vecto lên website (dùng xml)

video : trình xem video

audio : trình xem nhạc

source : Xác định nguồn cho một media ( đi kèm các thẻ video , audio)

Đây là các thẻ rất đặc biệt và có ý nghĩa rất lớn trong nền tảng HTML5 , chúng ta sẽ lần lượt khảo sát chi tiết các thẻ này trong những phần tới .

#### (4) Những Tag không được HTML5 hỗ trợ

- HTML5 bổ xung thêm rất nhiều tag , đồng thời cũng loại bỏ đi 1 số tag gây phiền toái trong kỹ thuật lập trình web .

- Với các thẻ HTML5 mới chúng ta hoàn toàn có thể làm thay thế các thẻ html cũ mà không được HTML5 hỗ trợ

- Dưới đây là các thẻ HTML5 không hỗ trợ

Tag	Mô tả	Tag thay thế
<acronym>	Mô tả từ viết tắt.	<a href="#">&lt;abbr&gt;</a>
<applet>	Xác định applet (nhúng JAVA).	<a href="#">&lt;embed&gt;</a> , <a href="#">&lt;object&gt;</a>
<basefont />	Xác định font, màu sắc, hay kích cỡ mặc định của text trong trang.	<a href="#">font</a>
<big>	Hiển thị text lớn.	<a href="#">font-size</a>
<center>	Canh giữa text.	<a href="#">text-align</a>
<dir>	Xác định danh sách thư mục.	<a href="#">&lt;ul&gt;</a>
<font>	Xác định font, màu sắc, và kích cỡ cho text.	<a href="#">font</a>
<frame>	Xác định một frame trong một khung (frameset).	<a href="#">&lt;iframe&gt;</a>
<frameset>	Xác định một khung (frameset).	<a href="#">&lt;iframe&gt;</a>
<noframes>	Xác định một nội dung thay thế khi trình duyệt không hỗ trợ hoặc người dùng vô hiệu hóa frame.	-
<s>	Hiển thị gạch ngang text.	<a href="#">text-decoration</a>
<strike>	Hiển thị gạch ngang text.	<a href="#">&lt;del&gt;</a>
<tt>	Xác định teletype text (kiểu chữ văn bản máy).	-
<u>	Hiển thị gạch dưới text.	<a href="#">text-decoration</a>

### 3. HTML5 Attribute ( thuộc tính HTML5)

- Nhập liệu là 1 phần không thể thiếu trong website .Trước HTML5 để nhập liệu các dữ liệu kiểu ngày tháng , màu sắc ... chúng ta cần nhờ cậy thêm các thư viện javascript .

- Ngoài ra do nhập liệu phía người dùng nên chúng ta còn thấy phát sinh việc kiểm tra việc nhập liệu đúng không , như những trường yêu cầu nhập dữ liệu ( không để trống ), trường chỉ cho nhập dữ liệu là số , email hay nhập số trong 1 khoảng nào đó .Trước kia để làm điều này chúng ta cần viết regular Javascript hoặc dùng các thư viện tương tự như : jQuery validate,kendo UI .

- HTML5 cung cấp khá đầy đủ các thẻ nhập liệu cũng như regular hỗ trợ việc nhập liệu

- Với HTML 5 chúng ta có 1 cách code hoàn toàn gọn gàng và thống nhất hơn là dùng thêm các thư viện

- Với các trình duyệt nếu chưa hỗ trợ các thẻ qui ước nhập liệu thì các thẻ trở thành các ô nhập liệu thông thường và không hề báo lỗi nếu người dùng nhập liệu sai .

- Do đó cần kiểm tra trình duyệt người dùng trước khi áp dụng các thẻ này

### 3.1 Attribute hỗ trợ nhập liệu

number	Thẻ nhập số
email	Thẻ nhập liệu Email
url	Nhập 1 link web
tel	Định nghĩa số điện thoại
date	Chọn ngày, tháng, và năm
datetime	Chọn thời gian, ngày, tháng, và năm (thời gian UTC)
datetime-local	Chọn thời gian, ngày, tháng, và năm (thời gian địa phương)
time	Chọn thời gian (giờ và phút)
month	Chọn tháng và năm
week	Chọn tuần và năm
color	Thẻ cho phép chọn màu sắc
range	thẻ cho phép kéo chọn giá trị như Slider Control với thuộc tính range ta còn có khai báo thêm các thuộc tính sau : +) max : giá trị lớn nhất cho phép +) min : giá trị nhỏ nhất cho phép +) step : giá trị mỗi lần di chuyển +) value : giá trị mặc định
search	Thẻ định nghĩa trường tìm kiếm

### 3.2 Attribute hỗ trợ kiểm tra nhập liệu

- Mỗi thuộc tính của trường nhập liệu (email , number ...) lại sinh ra các thuộc tính để kiểm tra nhập liệu xem người dùng có nhập đúng giá trị không

required	Trường không được để trống. Bắt buộc nhập liệu
placeholder	tiêu đề thông báo nội dung nhập liệu
min	Qui định giá trị thấp nhất ( dùng với thẻ input là number)
max	Qui định giá trị cao nhất ( dùng với thẻ input là number)
autofocus	Chỉ định trường hiện con trỏ chuột
autocomplete	Tự động điền, áp dụng cho thẻ input và thẻ form
pattern	Định nghĩa biểu thức chính qui
height	
form	Thuộc tính form qui định input thuộc về form nào Áp dụng cho các thẻ input
formmethod	
formaction	Thuộc tính formaction khai báo URL xử lý form - Thuộc tính formaction chỉ được sử dụng với 2 loại thẻ input là : submit và image - Với thẻ submit thì thuộc tính formaction sẽ ghi đè thuộc tính Action của thẻ form
formenctype	
formnovalidate	- thuộc tính novalidate để khai báo không kiểm tra tính đúng sai của các trường nhập liệu trong form .

	<ul style="list-style-type: none"> <li>- Giả sử ta có 1 trường cần nhập vào là số trong form , nếu khai báo thuộc tính novalidate cho form , khi này nếu người dùng có nhập chuỗi vào thẻ input rồi nhấn submit gửi về server thì form hoàn toàn không báo lỗi</li> <li>- Mặc định khi không khai báo thuộc tính novalidate thì sẽ có kiểm tra lỗi nhập liệu</li> </ul>
formtarget	
multiple	
step	
data-XXXX	Thuộc tính tùy chỉnh. Người lập trình có thể tự định nghĩa các thuộc tính của chính họ, song phải bắt đầu bằng "data-". Ví dụ: data-id="1"

### 3.3 Các ví dụ

- Chúng tôi đưa ra 1 vài ví dụ để các bạn có thể hiểu ý nghĩa sử dụng các thuộc tính cũng như kỹ thuật dùng cho các trường hợp khác nhau

#### (1) Form nhập liệu

- Ví dụ sau sử dụng các thuộc tính thường dùng trong form nhập liệu như là : nhập ngày tháng , email , tuổi .

- Có kiểm tra nhập liệu , các thuộc tính yêu cầu không được để trống hay nhập đúng kiểu dữ liệu yêu cầu

```

<form action="/" method="post">
<div>
<input type="text" name="Name" placeholder="Họ Và Tên" required>
</div>
<div>
<input type="number" name="Age" placeholder="Tuổi" required/>
</div>
<div>
    Năm sinh: <input type="date" name="BirthDay" required/>
</div>
<div>
<input type="email" name="Email" placeholder="Email" required/>
</div>
<div>
<input type="url" name="WebSite" placeholder="Website" required/>
</div>
<div>
    Màu yêu thích : <input type="color" name="Color" value=" " />
</div>
<p>
<input type="submit" name="name" value="Gửi Yêu Cầu"/>
</p>
</form>

```

- Chạy ví dụ ta được kết quả như hình sau :

localhost:12476/HTML/HtmlPage1.html

Ứng dụng Can ban nhanh - ... 120+ Useful CSS3 Tu... Windows Phone 8 A... HTML5 Introduction

Họ Và Tên

Tuổi

Năm sinh: dd/mm/yyyy

Email

Website

Màu yêu thích :

OK

- Không nhập dữ liệu gì nhấn OK thì sẽ báo yêu cầu nhập Họ tên như hình sau ( bởi mặc định HTML5 sẽ kiểm tra lỗi từ trên xuống )

localhost:12476/HTML/HtmlPage1.html

Ứng dụng Can ban nhanh - ... 120+ Useful CSS3 Tu... Windows Phone 8 A... HTML5 Introduction

Họ Và Tên

Tuổi

Năm sinh: dd/mm/yyyy

Email

Website

Màu yêu thích :

OK

Vui lòng điền vào trường này.

- Khi nhập Họ tên , phần tuổi yêu cầu không để trống , và dữ liệu nhập là số .Nếu nhập chữ , nhấn OK sẽ báo lỗi sau :

localhost:12476/HTML/HtmlPage1.html

Ứng dụng Can ban nhanh - ... 120+ Useful CSS3 Tu... Windows Phone 8 A... HTML5 Introduction

Nguyễn Văn A

mười

Năm sinh: dd/mm/yyyy

Email

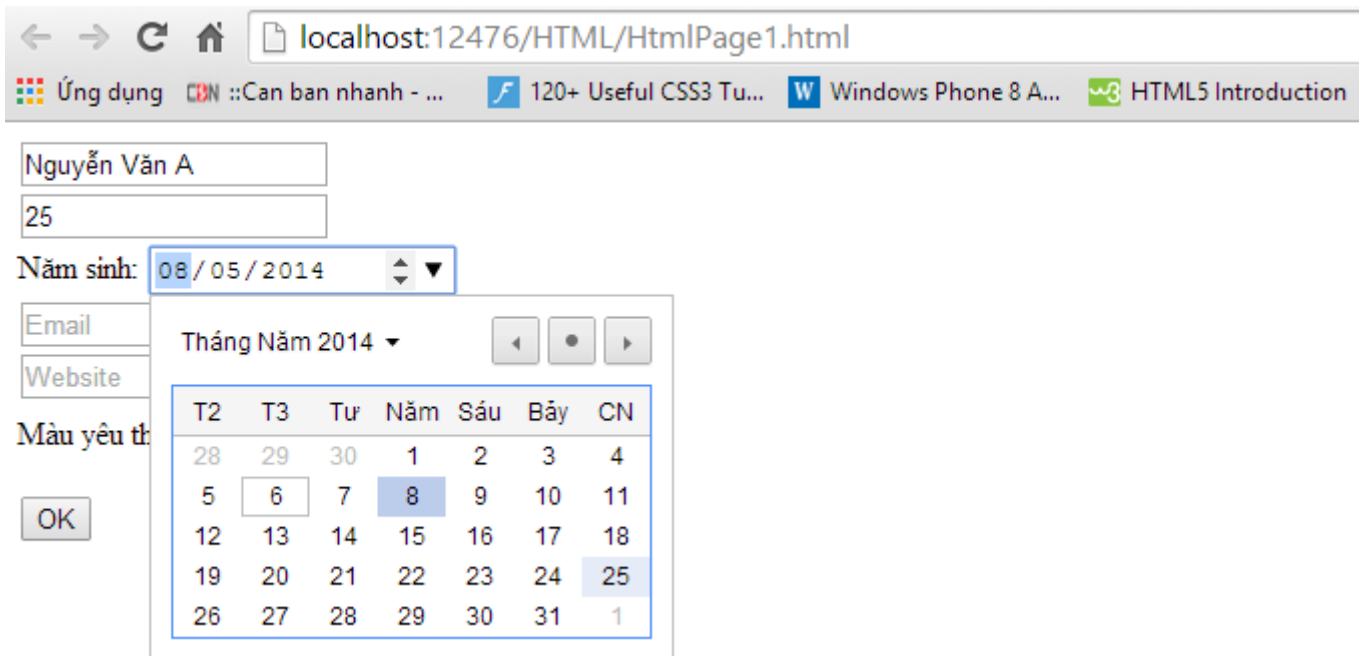
Website

Màu yêu thích :

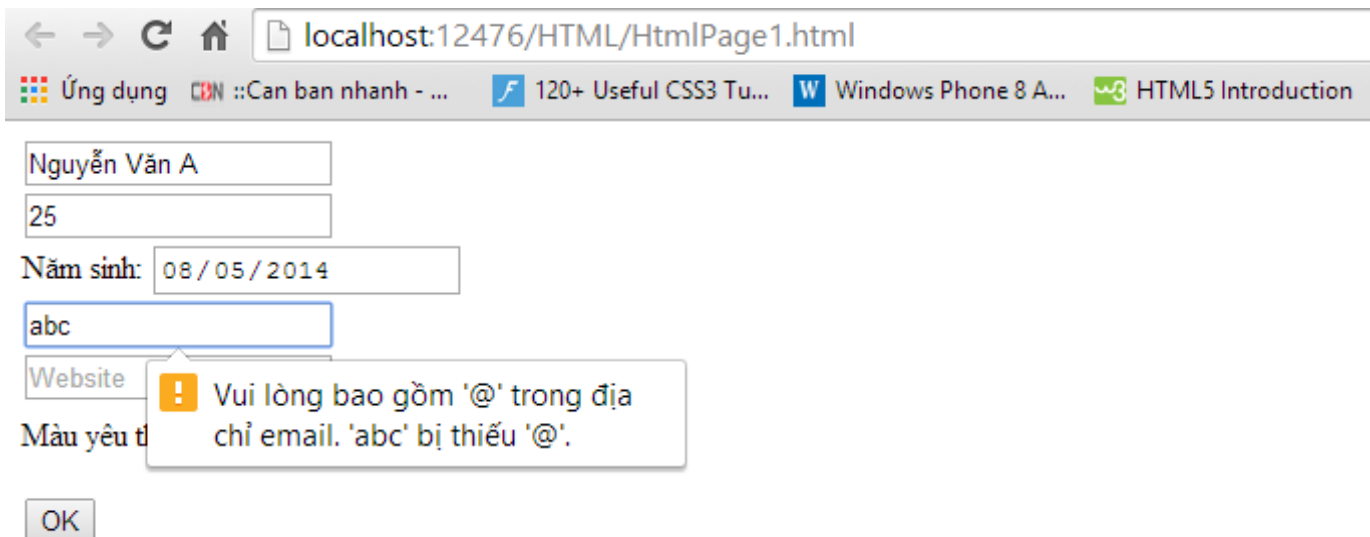
OK

Vui lòng nhập một số.

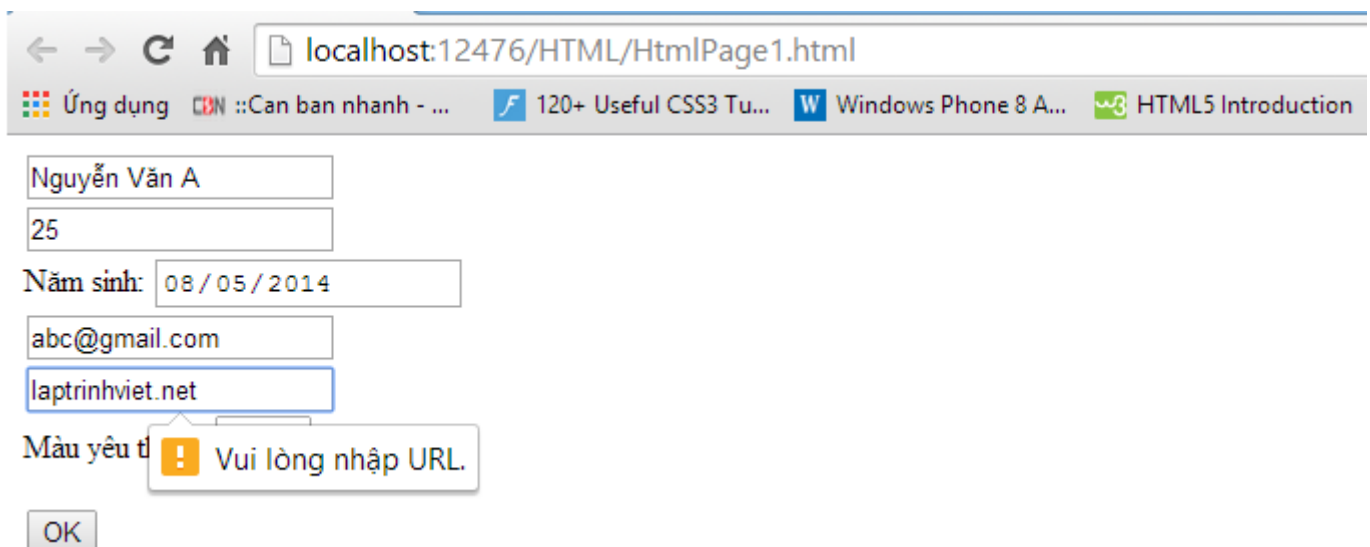
- Năm sinh là Control cho phép chọn ngày tháng năm như sau :



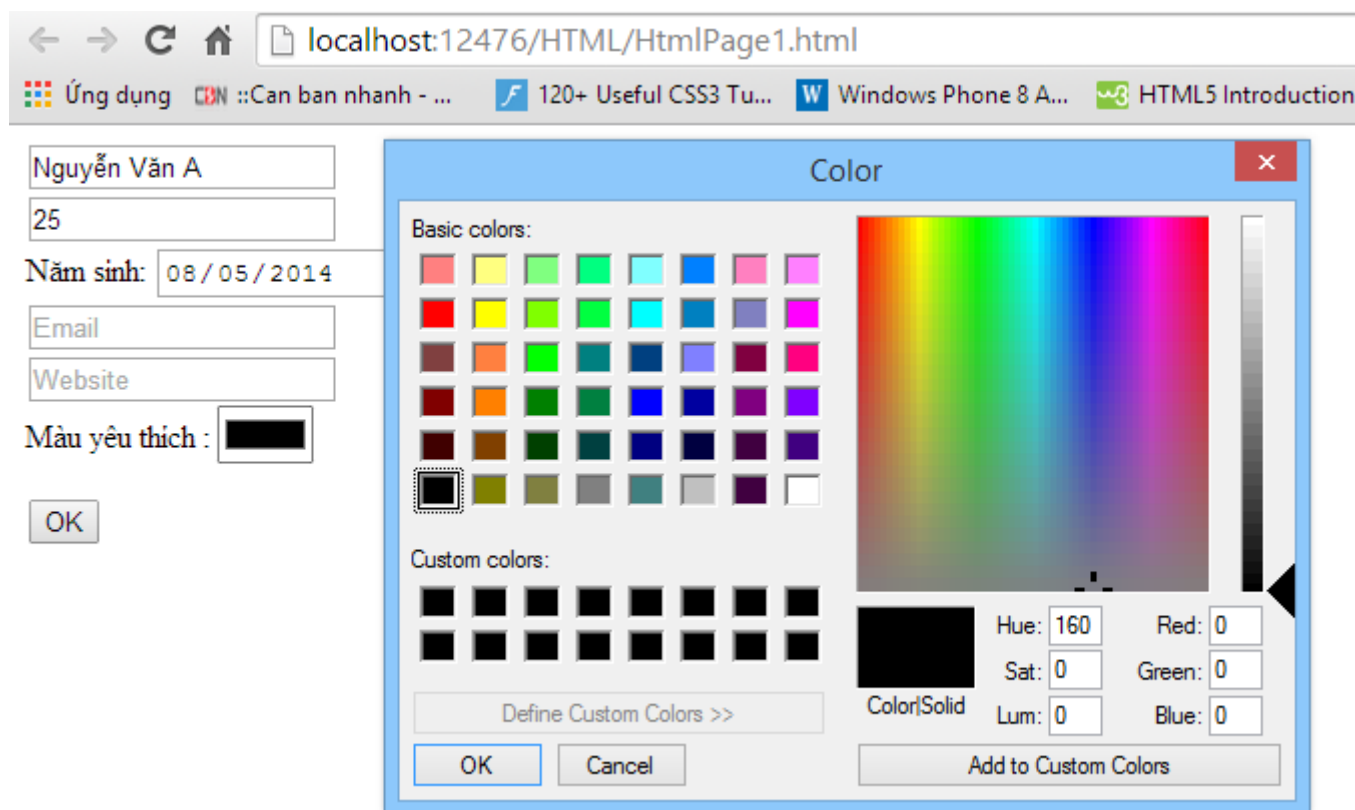
- Nhập Email không đúng định dạng (không có @) thì sẽ báo lỗi như sau :



- Nhập website là URL cần có http , nếu không có sẽ báo lỗi sau :



- Hộp chọn màu sắc như sau :



- Ta thấy HTML5 cung cấp cho chúng ta khá nhiều ô nhập liệu quan trọng, giúp nhập liệu đơn giản hơn. Điều mà trước kia chúng ta chỉ có thể nhờ cậy tới javascript hoặc thư viện javascript.
- Tính năng kiểm tra nhập liệu cũng khá mạnh. Chúng ta đủ công cụ để kiểm tra các ô trống, các ô nhập dữ liệu là số, email, url.

## (2) Xây dựng biểu thức chính quy (Regular) kiểm tra nhập liệu trong HTML5

- HTML5 cung cấp khá nhiều tính năng kiểm tra nhập liệu cho kiểu số, email, url ...
- Tuy nhiên HTML5 cũng cung cấp thêm 1 thuộc tính khá mạnh đó là pattern dùng để cho bạn viết các biểu thức chính quy kiểm tra nhập liệu.
- Trường pattern mô phỏng các chức năng của một biểu thức JavaScript chính quy (regex) truyền thống. Đầu vào phải khớp với cấu trúc mẫu của biểu thức chính quy được định nghĩa trước khi nó được xác nhận hợp lệ. Trường này làm việc với các kiểu text, search, url, telephone, email, và password.
- Nếu bạn là người lập trình thì đã khá quen thuộc với biểu thức chính quy Regular. Được dùng để qui định những ký tự nào được phép nhập là hợp lệ theo 1 luật nhất định.
- Ở đây chúng tôi không đi sâu vào chúng, mà đưa ra 1 ví dụ dùng biểu thức chính quy trong HTML5

HTML :

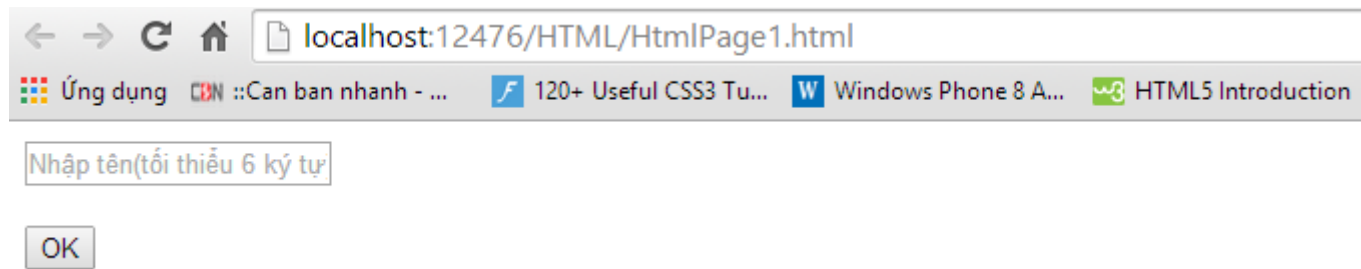
```
<form action="/" method="post">
<input name="Name" pattern="[a-zA-Z0-9]{6,}" title="Bạn phải nhập tối thiểu 6 ký tự(A-Z;0-9)" placeholder="nhập tên(tối thiểu 6 ký tự)">
<p>
<input type="submit" name="name" value="OK"/>
</p>
</form>
```



- Thẻ input trên qui định trường nhập liệu tối thiểu 6 ký tự .Các ký tự chỉ bao gồm từ a - z ( cả chữ hoa) và các số từ 0 - 9

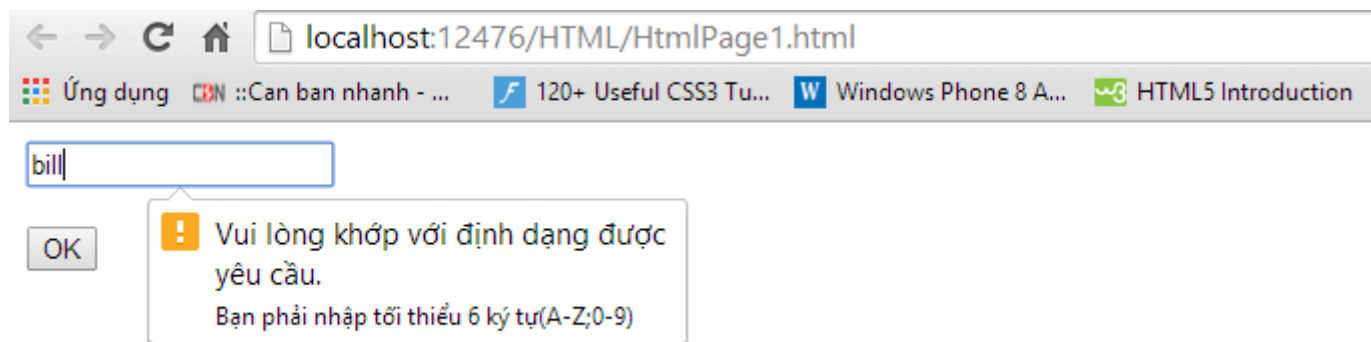
- Thuộc tính title là nội dung thông báo khi người dùng nhập sai

Chạy ví dụ được kết quả như sau

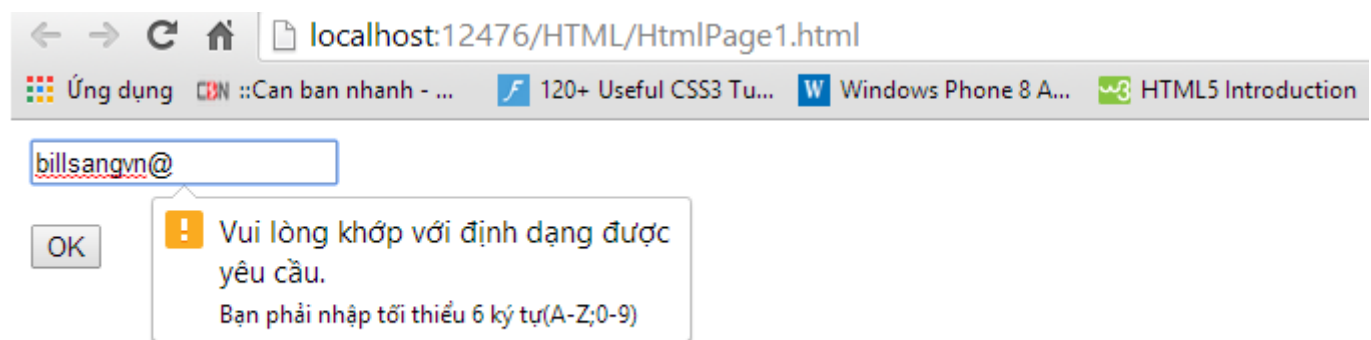


- Nếu ta nhập không đủ 6 ký tự hoặc nhập thêm ký tự lạ như ( ? ; / ) sau đó nhấn OK thì sẽ báo lỗi nhập không đúng định dạng như sau :

+) Nhập không đủ ký tự



+) Nhập ký tự không có trong mẫu :



- Các bạn có thể tìm hiểu và tự viết những biểu thức chính qui cho riêng mình.Biểu thức chính qui tồn tại hầu như trong tất cả các ngôn ngữ lập trình nên các bạn có thể dễ dàng tìm cho mình những biểu thức theo yêu cầu thông thường ( email , số điện thoại)

- Trong 1 vài trường hợp riêng bạn có thể tự tìm hiểu cấu trúc biểu thức chính qui để tự viết cho riêng mình

- Ví dụ 1 vài biểu thức chính qui hay dùng :

[a-z0-9\_]{4,20} : Nhập từ 4 - 20 ký tự a - z ; 0 - 9

("^[a-zA-Z0-9\_\\.-]+@[([a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}\$") :Biểu thức qui định nhập vào 1 email

Số	Chỉ chứa các chữ số thập phân; ví dụ 5, hoặc 5683874674.	^\d+\$
PIN	Chứa 4 chữ số thập phân, ví dụ 1234.	^\d{4}\$
Mật khẩu đơn giản	Chứa từ 6 đến 8 ký tự; ví dụ ghtd6f hoặc b8c7hohg.	^\w{6,8}\$
Số thẻ tín dụng	Chứa dữ liệu phù hợp với cấu trúc của hầu hết các loại số thẻ tín dụng, ví dụ 4921835221552042 hoặc 4921-8352-2155-2042.	^\d{4}-?\d{4}-?\d{4}-?\d{4}\$
Địa chỉ e-mail	[\\w-]+ nghĩa là chứa một hoặc nhiều ký tự word hoặc dấu gạch ngang, ví dụ somebody@adatum.com	^[\\w-]+@[\\w-]+\\.+[\\w-]+\$
HTTP hoặc HTTPS URL	Dữ liệu là một URL dựa trên HTTP hay dựa trên HTTPS, ví dụ http://www.microsoft.com	^https?:\\/([\\w-]+\\.)+[\\w-]+(\\/([\\w- . / ? % = ]*))?\$

### (3) Form tự động điền với Autocomplete

- Thuộc tính autocomplete cung cấp tính năng cho phép trình duyệt tự động điền ( hiển thị ) các thông tin đã được nhập trước đó sau mỗi Request .

- Ta lấy 1 ví dụ đơn giản :

Có 1 form đăng nhập gồm 2 giá trị input là : UserName và Passwords . Lần đầu người dùng đăng nhập ( gọi đến trang đăng nhập ) . Điền các thông tin UserName và Passwords .

Khi đăng xuất khỏi ứng dụng , sau đó đăng nhập lại thì form đăng nhập tự động điền các thông số đăng nhập ( UserName và Passwords ) trước đó . Nếu là tên đăng nhập cũ thì người dùng chỉ cần nhấn vào nút submit để đăng nhập .

- thuộc tính autocomplete nhận 2 giá trị là on hoặc off tương ứng với chức năng bật hoặc tắt tính năng tự động điền

- Thuộc tính autocomplete hỗ trợ thẻ form và các thẻ khác trong form sau : text, search, url, tel, email, password, datepickers, range, and color

Ví dụ :

```
<form action="#" autocomplete="on">
    First name: <input type="text" name="fname"><br>
    Last name: <input type="text" name="lname"><br>
    E-mail: <input type="email" name="email" autocomplete="on"><br>
<input type="submit">
</form>
```

- Với khai báo trên thì thuộc tính autocomplete của form được bật và thẻ email được bật autocomplete , các thẻ khác không được bật tính năng này khi người dùng nhập dữ liệu

#### (4) Thuộc tính form

Thuộc tính form qui định input thuộc về form nào

Áp dụng cho các thẻ input

- Thông thường khi khai báo 1 form thì ta khai báo các thẻ input trong form đó .Tuy nhiên html 5 có cung cấp 1 thuộc tính form cho thẻ input để chỉ định thẻ input ở ngoài form thuộc về form nào .

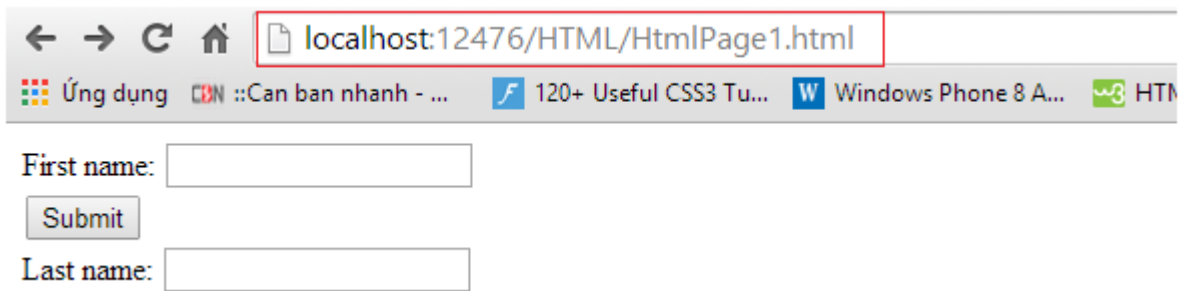
- Khi nhấn submit gửi về server thì form sẽ gửi tất cả các input nằm trong form và các input nằm ngoài form có thuộc tính form là form gửi submit

Ví dụ :

```
<formaction="#"id="form1">  
    First name: <inputtype="text"name="fname"><br>  
<inputtype="submit"value="Submit">  
</form>  
    Last name: <inputtype="text"name="lname"form="form1">
```

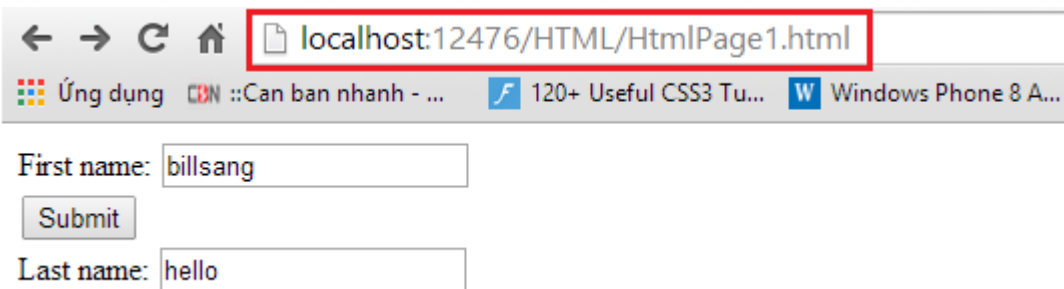
Ví dụ trên thẻ input Last Name cũng thuộc form1 .Khi nhấn submit gửi về server thì form1 gửi cả 2 thuộc tính là FirstName và LastName

- Chạy ví dụ được kết quả như hình sau :

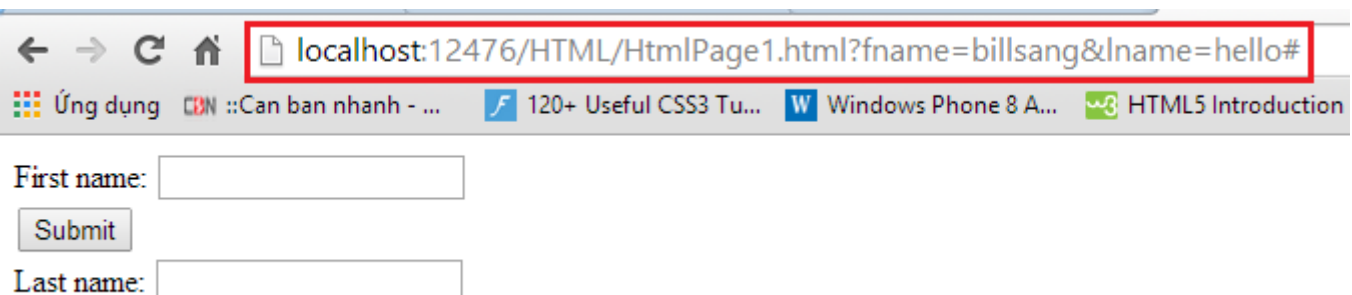


The screenshot shows a web browser window with the address bar containing `localhost:12476/HTML/HtmlPage1.html`. The browser tabs include "Ứng dụng", "Can ban nhanh - ...", "120+ Useful CSS3 Tu...", "Windows Phone 8 A...", and "HTML5 Introduction". The form contains two text input fields: "First name:" and "Last name:". Below the "First name:" field is a "Submit" button.

- Các bạn quan sát kỹ URL , khi nhập First Name và Last Name rồi nhấn Submit URL đã thay đổi , và First Name và Last Name được post lên server ( Do server ở đây là luôn file hiện tại nên chưa có xử lý gì cả )



The screenshot shows the same web browser window after the form has been submitted. The address bar now contains `localhost:12476/HTML/HtmlPage1.html?fname=billsang&lname=hello#`. The "First name:" field contains the text "billsang" and the "Last name:" field contains the text "hello". The "Submit" button is still present.



The screenshot shows the same web browser window after the form has been submitted. The address bar now contains `localhost:12476/HTML/HtmlPage1.html?fname=billsang&lname=hello#`. The "First name:" field is empty and the "Last name:" field is empty. The "Submit" button is still present.

## 4. HTML 5 đồ họa Canvas - SVG

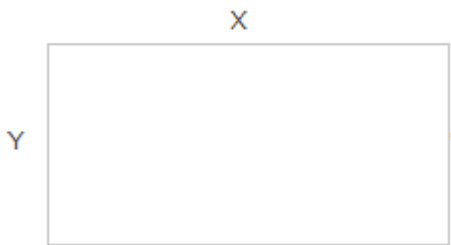
### 4.1 HTML5 Canvas

- Canvas là hệ thống đồ họa pixel dùng để vẽ các đối tượng bằng javascript
- Canvas xây dựng sẵn các phương thức cơ bản để vẽ các đường thẳng, hộp, văn bản, đường tròn, đường cong và hình ảnh
- Với canvas chúng ta có thể vẽ bất cứ thứ gì trên HTML bằng javascript

Dưới đây ta khai báo 1 thẻ canvas như sau :

```
<canvasid="myCanvas"width="200"height="100" style="border:1pxsolid#000000;"></canvas>
```

- Khung canvas được chia theo tọa độ hai chiều như hình sau :



- Trong html5 có 1 phương thức đối trục tọa độ như sau :

```
ctx.translate(50, 50);
```

// Đổi gốc tọa độ sang vị trí (50,50)

#### (1) Vẽ đường thẳng trong canvas

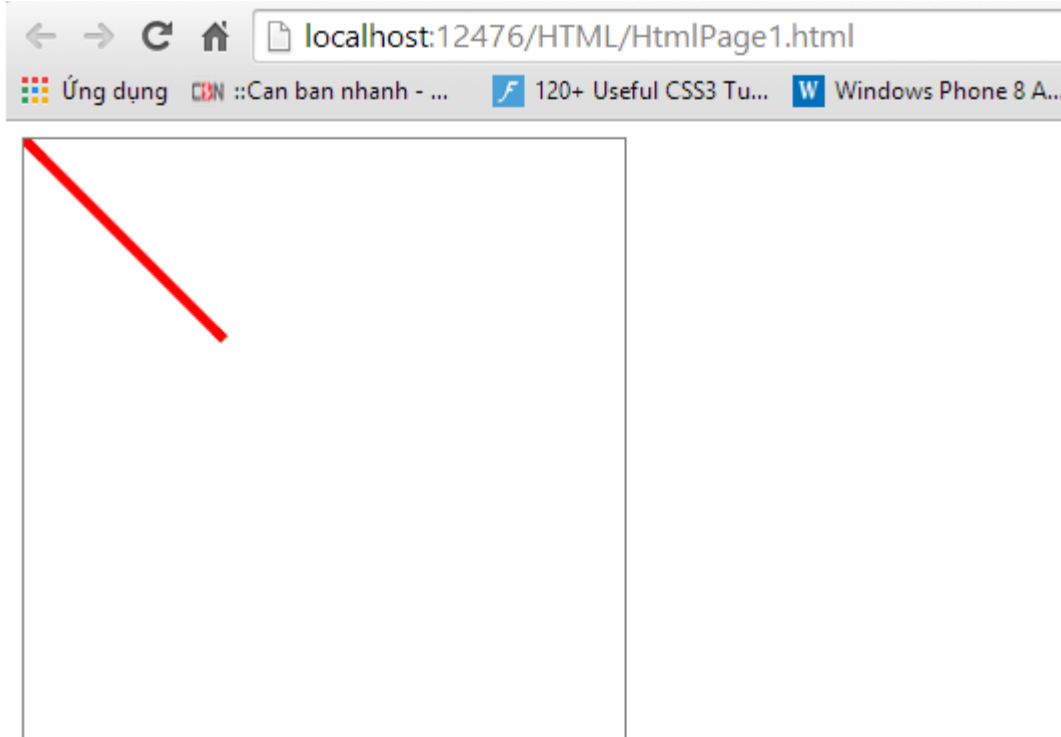
- Vẽ đường thẳng trong Canvas dùng 2 phương thức moveTo (điểm đầu) và.lineTo (Điểm cuối)
- Phương thức stroke() để thực hiện lệnh vẽ
- Để vẽ 1 đường thẳng chúng ta cần khai báo điểm đầu và điểm cuối bằng cách sử dụng 2 phương thức là moveTo và.lineTo .
- Các phương thức này nhận vào 2 tham số là hoành độ và tung độ điểm
- Lệnh dưới đây vẽ 1 đường thẳng nối 2 điểm (0,0) tới (100,100)

```
<canvasid="canvas"width="300"height="300"style="border:1pxsolid#808080;"></canvas>
<script>
var c = document.getElementById("canvas")
var ctx = c.getContext("2d");
// tọa độ điểm đầu
  ctx.moveTo(0, 0);
  // tọa độ điểm cuối
  ctx.lineTo(100, 100);
// thực hiện lệnh vẽ
  ctx.stroke();
</script>
```

Chúng ta có thể thêm các yếu tố định dạng đường vẽ như sau :

```
ctx.lineWidth = 5; // định dạng độ lớn đường vẽ
ctx.strokeStyle = "#f00"; // định dạng màu đường vẽ
```

- Chạy ví dụ được kết quả như hình sau :



## (2) Vẽ 1 đường tròn trong canvas :

- Ta có thể vẽ 1 đường tròn bằng phương thức arc như sau :

```
context.arc(x,y,r,sAngle,eAngle,counterclockwise);
```

Trong đó :

x , y là hoành độ và tung độ của tâm đường tròn

r là bán kính đường tròn

sAngle:góc bắt đầu

eAngle :góc kết thúc

counterclockwise :tùy chọn vẽ cung tròn .Có 2 giá trị là true và false .Mặc định là false

- Lệnh dưới đây vẽ 1 đường tròn có tâm (50,50) bán kính 40 .Ở đây ta vẽ đường tròn nên góc bắt đầu sẽ là 0 và góc kết thúc là  $2\pi$

```
<canvasid="myCanvas"width="300"height="300"style="border:1pxsolid#808080;"></canvas>
```

```
<script>
```

```
var c = document.getElementById("myCanvas");
```

```
var ctx = c.getContext("2d");
```

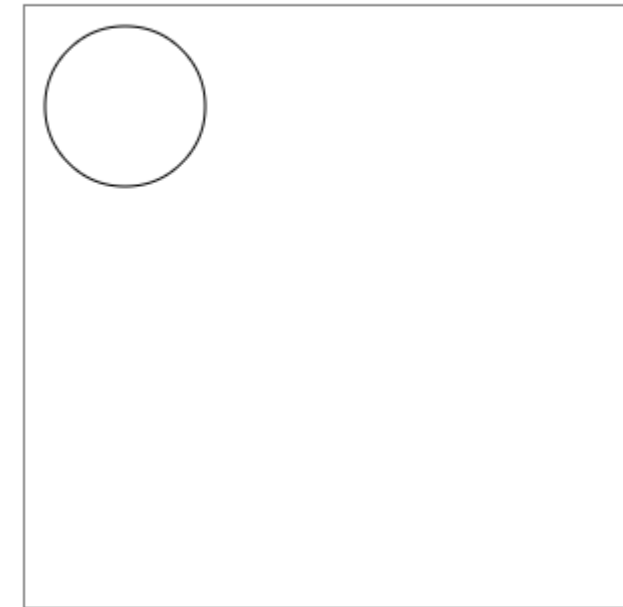
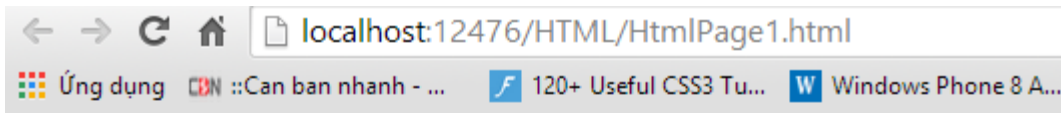
```
ctx.beginPath();
```

```
ctx.arc(50, 50, 40, 0, 2 * Math.PI);
```

```
ctx.stroke();
```

```
</script>
```

- Kết quả :

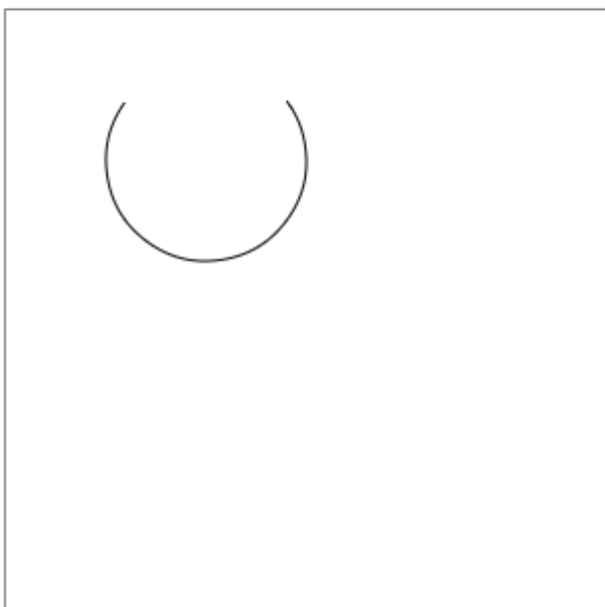
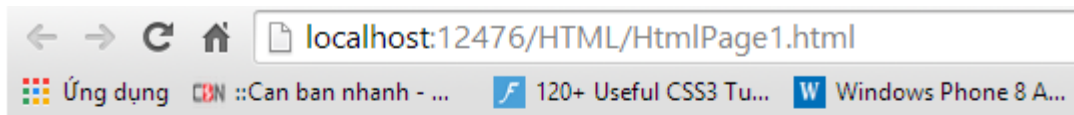


- Đọc tham số của phương thức arc thì ta có thể thấy phương thức này ta có thể vẽ 1 cung tròn bất kỳ

- Lệnh dưới đây vẽ 1 cung tròn lớn :

```
<canvasid="myCanvas"width="300"height="300"style="border:1pxsolid#808080;"></canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
    ctx.beginPath();
    ctx.arc(100, 75, 50, 1.2 * Math.PI, 1.8 * Math.PI, true);
    ctx.stroke();
</script>
```

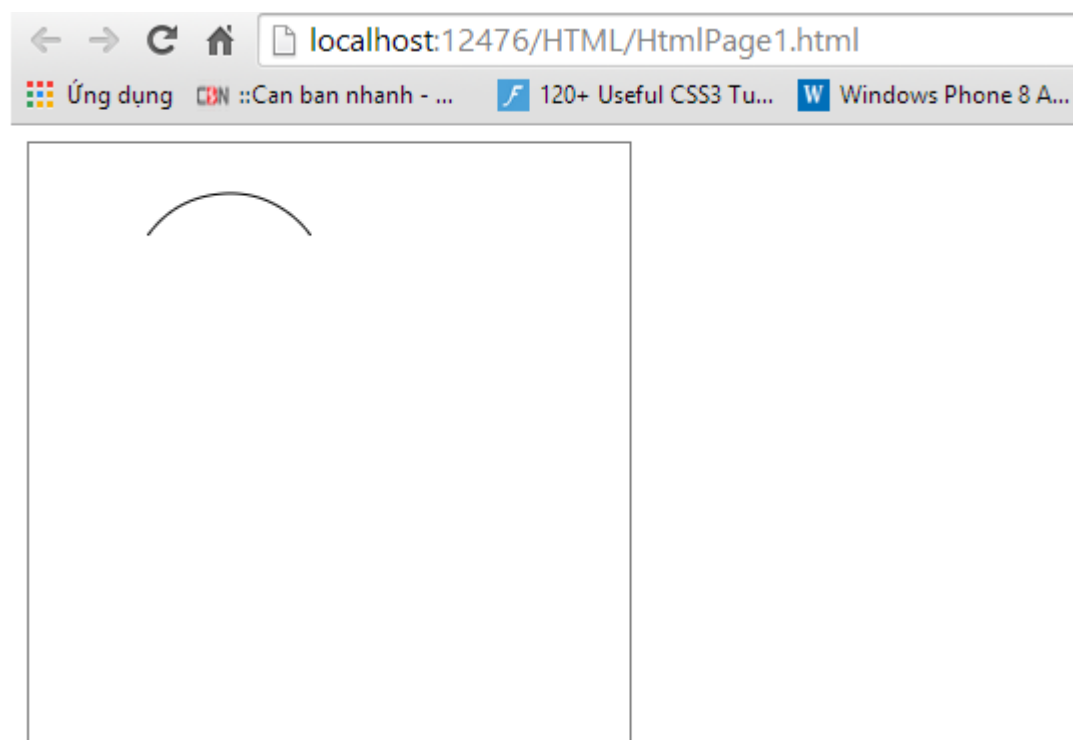
- Kết quả :



- Nếu không khai báo hoặc khai báo tham số là false thì ta được cung tròn nhỏ như sau :

```
<canvasid="myCanvas"width="300"height="300"style="border:1pxsolid#808080;"></canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
    ctx.beginPath();
    ctx.arc(100, 75, 50, 1.2 * Math.PI, 1.8 * Math.PI, false);
// hoặc không khai báo
// ctx.arc(100, 75, 50, 1.2 * Math.PI, 1.8 * Math.PI);
    ctx.stroke();
</script>
```

- Kết quả :



### (3) Vẽ ảnh trong canvas

- Ta có thể lấy 1 hình ảnh rồi vẽ lại chúng lên thẻ canvas bằng phương thức drawImage

- Phương thức drawImage nhận 3 tham số là : 1 bức ảnh theo đường dẫn .Hoàn độ , tung độ điểm bắt đầu vẽ ảnh

- Ví dụ dưới đây vẽ 1 bức ảnh lên canvas bắt đầu từ tọa độ góc (0,0)

```
<canvasid="myCanvas"width="300"height="300"style="border:1pxsolid#808080;"></canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = new Image();
    img.src = 'Billgates(1).jpg';
    img.onload = function () {
        ctx.drawImage(img, 0, 0);
    };
</script>
```

Hoặc có thể gọi phương thức vẽ ảnh trong sự kiện onload của thẻ body như sau:

```
<bodyonload="GetCanvas()">
<canvasid="myCanvas"width="300"height="300"style="border:1pxsolid#808080;"></canvas>
```

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = new Image();
    img.src = 'Billgates(1).jpg';
function GetCanvas() {
    ctx.drawImage(img, 0, 0);
}
</script>
</body>
```

- Chạy ví dụ được kết quả như hình sau :

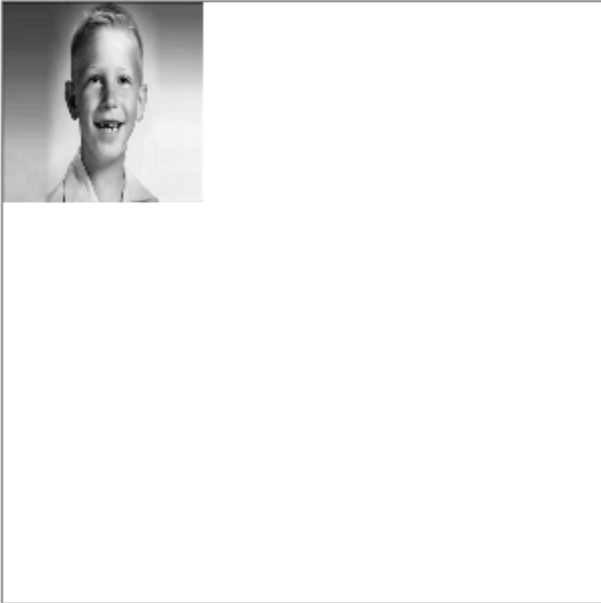
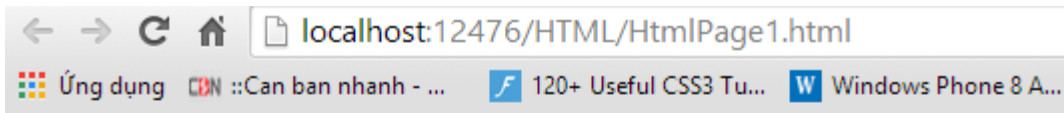


- Ta có thể định kích thước bức ảnh bằng cách truyền tham số chiều rộng , chiều cao bức ảnh như sau :

```
ctx.drawImage(img, 0, 0,100,100);
```

- Ta được kết quả hiển thị ảnh trên như sau :





- Việc vẽ 1 bức ảnh lên canvas được dùng rất nhiều trong lập trình giao diện .Do đó ta thường viết sẵn 1 hàm load ảnh như sau :

- Thường ta viết sẵn 1 hàm load ảnh như sau :

```
// đối tượng điểm
function Point(a, b) {
  this.x = a;
  this.y = b;
}

// hàm vẽ ảnh
function DrawImg(ctx, _src, Point1) {
  var w = 50;
  var h = 50;
  var img = new Image();
  img.src = _src;
  img.onload = function () {
    ctx.drawImage(img, Point1.x, Point1.y, w, h);
  }
}
```

Đây là cách viết hướng đối tượng trong JavaScript.Các kỹ thuật này rất hay được dùng trong lập trình canvas HTML5 .Các bạn có thể tìm hiểu kỹ hơn về kỹ thuật lập trình hướng đối tượng trong javascript .Và hiện tại các bạn có thể dùng ngôn ngữ TypeScript để lập trình hướng đối tượng mạnh hơn javascript để thay thế .Các kiến thức này được chúng tôi đề cập trong giao trình về javascript

## Vẽ ảnh vào tâm

- thường phần đầu tiên của ảnh sẽ hiển thị vào tọa độ bắt đầu .Để tâm của ảnh hiển thị vào giữa tọa độ cần vẽ ta cần tính toán lại tọa độ hiển thị

```
function DrawImgCenter(ctx,_src,Point1) {
  var w = 50;
  var h = 50;
  var img = new Image();
  img.src = _src;
  img.onload = function () {
    ctx.drawImage(img, Point1.x-w/2, Point1.y-h/2, w, h);
  }
}
```

```
}  
}
```

#### (4) Di chuyển ảnh trên canvas

- Trong lập trình giao diện html hoặc thiết kế game ta thường có yêu cầu di chuyển đối tượng nhận vật hay hình ảnh .

- Trong lập trình game HTML5 chúng ta cũng dùng thêm các thư viện Framework khác để hỗ trợ lập trình giao diện .Tuy nhiên ở đây ta có thể xây dựng ví dụ đơn giản để di chuyển ảnh trong canvas mà không dùng thêm thư viện khác với mục đích hiểu bản chất vấn đề

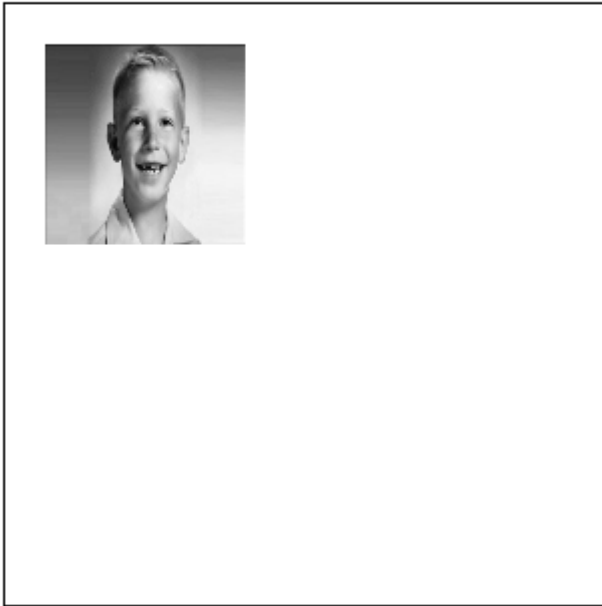
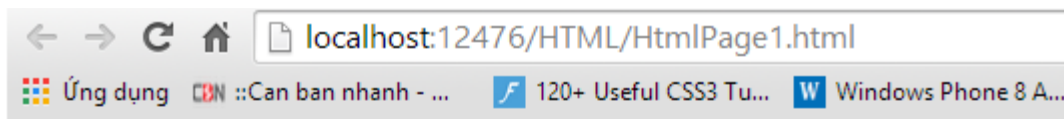
- Để tạo ra di chuyển chúng ta làm 2 nhiệm vụ chính sau : Xóa bức ảnh cũ và vẽ lại chúng trên tọa độ mới

- Với các Framework hỗ trợ lập trình canvas thì thường dùng kỹ thuật tạo layer cho các đối tượng trên canvas để dễ dàng quản lý việc xóa đối tượng nào ( tương tự như trong photoshop)

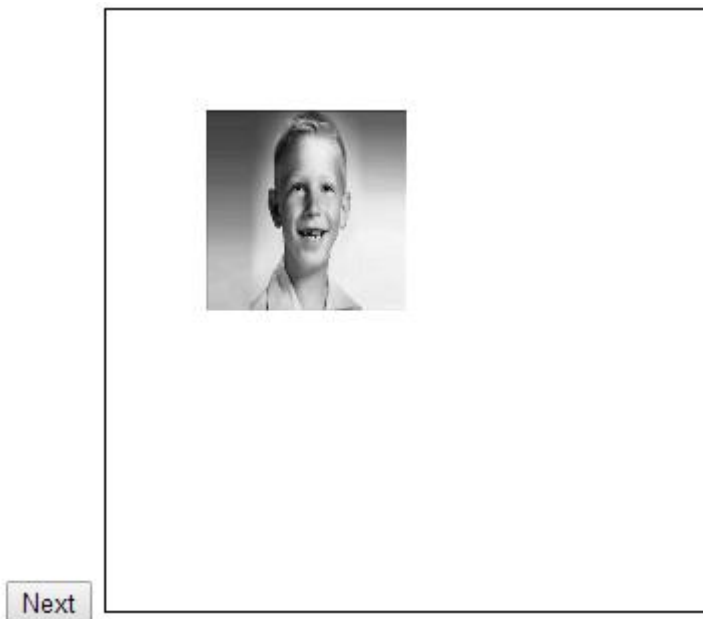
HTML :

```
<bodyonload="GetCanvas()">  
<buttononclick="NextCommand()">Next</button>  
<canvasid="Mycanvas"width="300"height="300"style="border: 1pxsolid#000"></canvas>  
<script>  
    img = new Image();  
    img.src = "Billgates(1).jpg"  
var ctx = document.getElementById("Mycanvas").getContext('2d');  
function GetCanvas() {  
    // hiện ảnh cũ (100,100) tại tọa độ (20,20)  
    ctx.drawImage(img, 20, 20, 100, 100);  
}  
function NextCommand() {  
    // xóa đối tượng img  
    ctx.clearRect(20, 20, 100, 100);  
    ctx.drawImage(img, 50, 50, 100, 100);  
}  
</script>  
</body>
```

Chạy ví dụ được kết quả như hình sau :



- Nhấn vào button Next ảnh sẽ di chuyển sang vị trí mới được kết quả như hình sau :



### (5) Viết chữ lên canvas

- Ngoài vẽ các đường ( đường thẳng , đường cong ) , ảnh lên canvas .Chúng ta có thể vẽ chữ (text) lên canvas

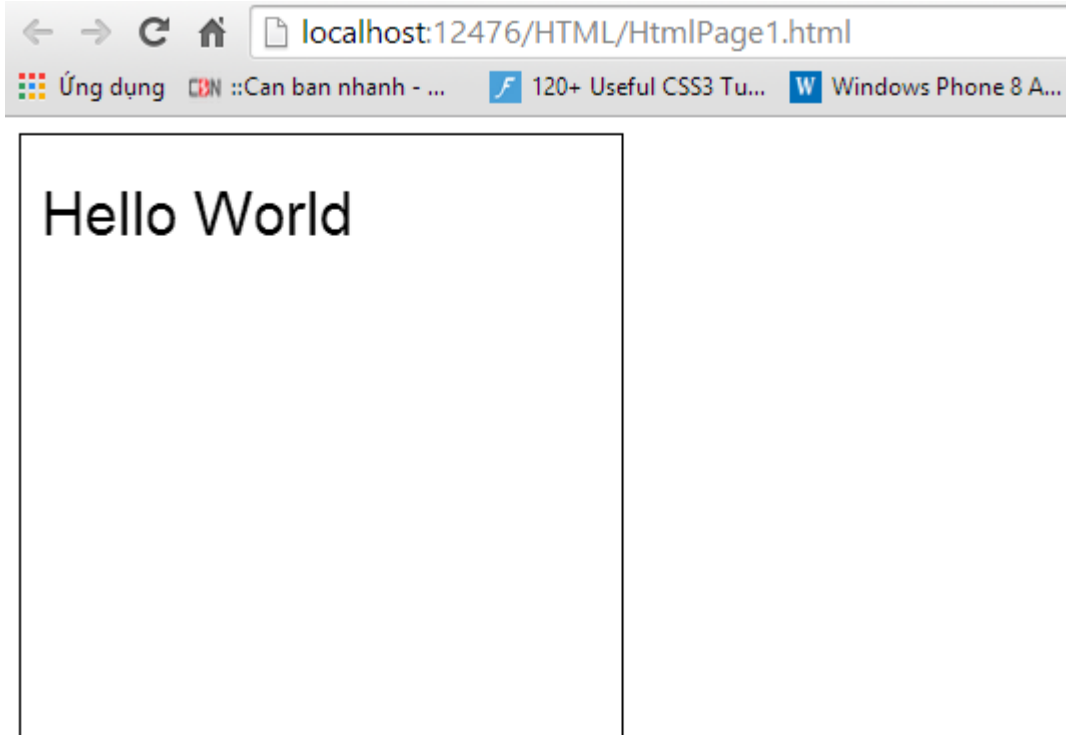
- Để vẽ chữ chúng ta dùng phương thức fillText và truyền vào tham số là đoạn text muốn vẽ và tọa độ bắt đầu vẽ

- Ví dụ sau viết chữ hello world lên thẻ canvas

```
<canvasid="Mycanvas"width="300"height="300"style="border: 1pxsolid#000"></canvas>
<script>
var ctx = document.getElementById("Mycanvas").getContext('2d');
ctx.font = "30px Arial";
```

```
ctx.fillText("Hello World", 10, 50);  
</script>
```

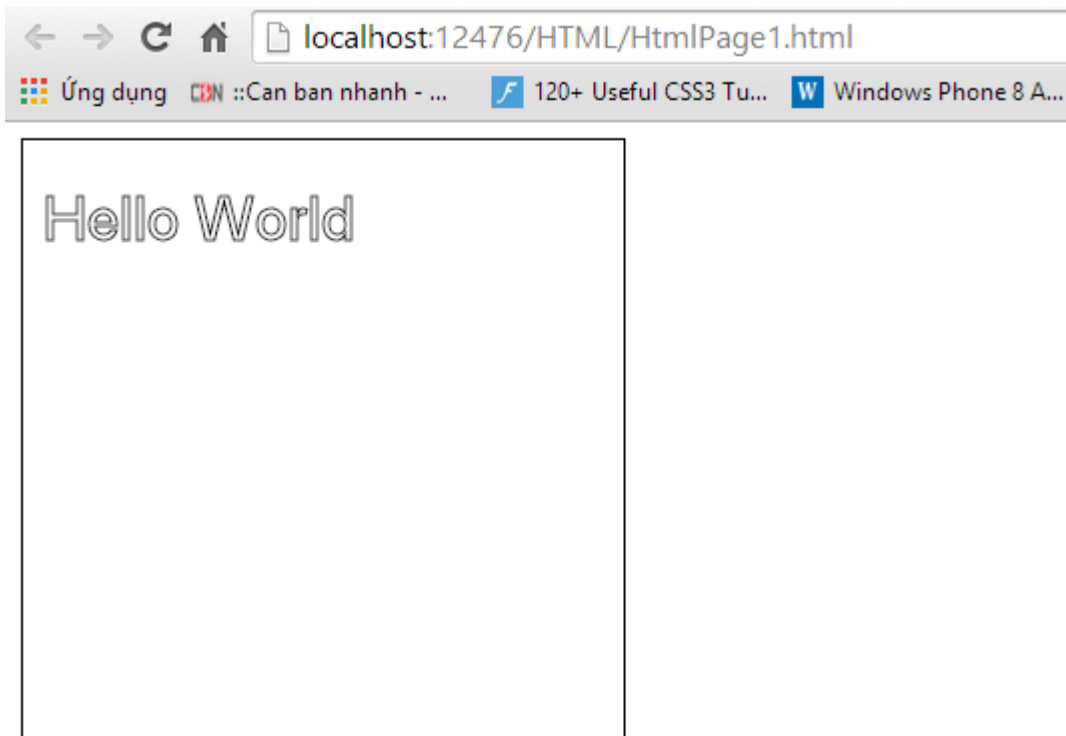
- Chạy ví dụ được kết quả như hình sau :



- Chúng ta có thể vẽ chữ nghệ thuật bằng cách thay phương thức `fillText` thành `strokeText`

```
ctx.strokeText("Hello World", 10, 50);
```

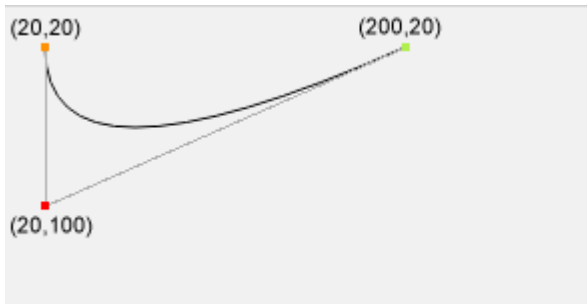
Được kết quả như hình sau :



## (6) Vẽ đường cong bậc hai Quadratic Curve

- Để tạo ra một đường cong bậc hai với HTML5 Canvas, chúng ta có thể sử dụng phương thức `quadraticCurveTo()`.

- Đường cong bậc hai được xác định bởi ba điểm : Điểm bắt đầu , điểm điều khiển và điểm kết thúc như mô tả sau trên hình sau :



- Hình trên mô tả điểm bắt đầu có tọa độ là : (20,20)

Điểm điều khiển có tọa độ là : (20,100)

Điểm kết thúc có tọa độ là : (200,20)

- Để vẽ đường cong thực hiện hai lệnh sau :

// xác định điểm bắt đầu

```
context.moveTo(startX, startY)
```

// bắt đầu vẽ đường cong với điểm điều khiển và điểm kết thúc

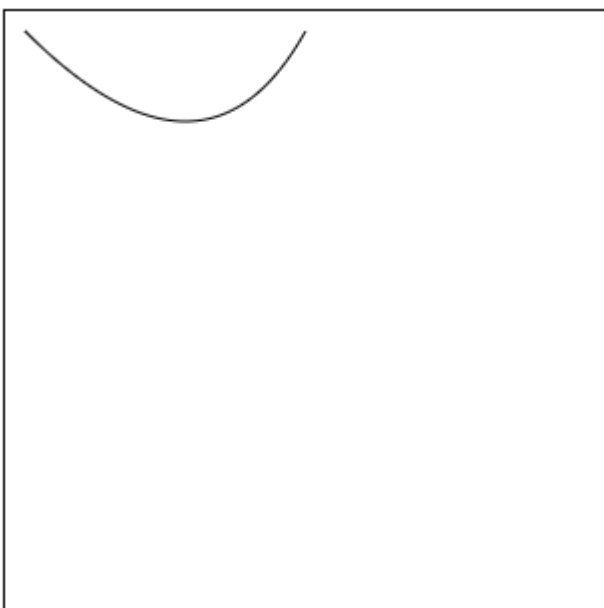
```
context.quadraticCurveTo(controlX, controlY, endX, endY);
```

- Để vẽ đường cong như hình trên ta có mã như sau :

```
<canvasid="Mycanvas"width="300"height="300"style="border: 1pxsolid#000"></canvas>
<script>
var ctx = document.getElementById("Mycanvas").getContext('2d');
    ctx.moveTo(10, 10);

    ctx.quadraticCurveTo(100, 100, 150, 10);
    ctx.stroke();
</script>
```

Chạy ví dụ được như hình sau :

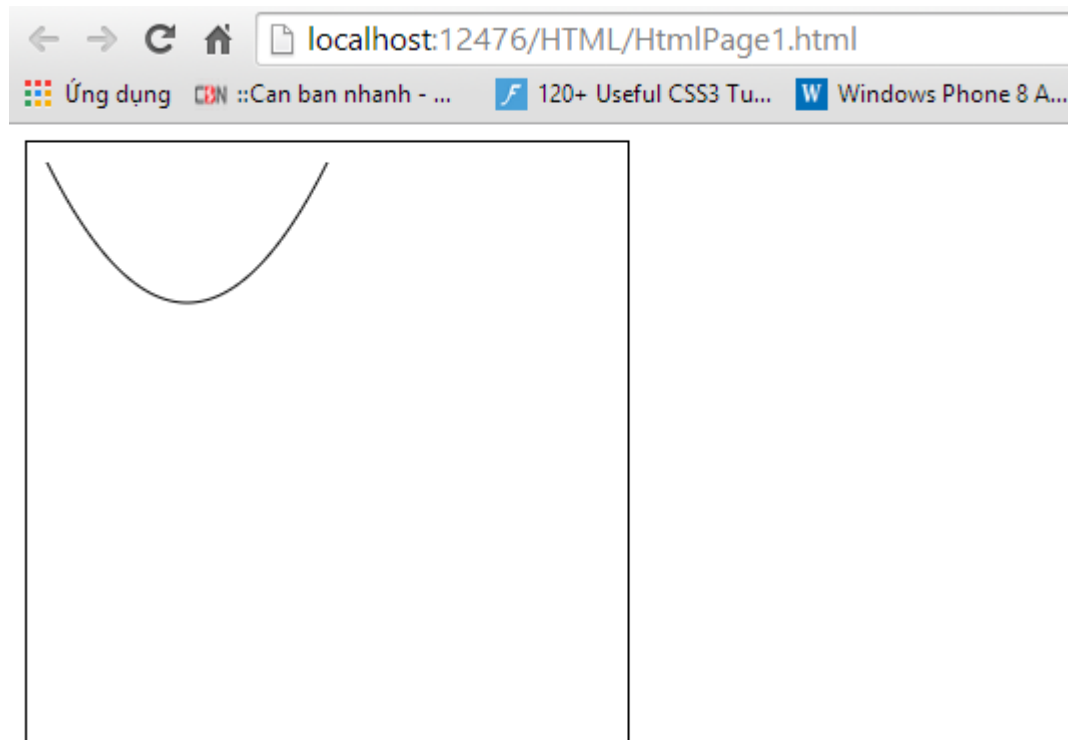


- muốn vẽ hàm số bậc hai thì hoành độ điểm trung tâm là trung bình cộng 2 hoành độ biên . Tung độ là đỉnh Parabol

- Ta có mã vẽ đường cong Parabol như sau :

```
<canvasid="Mycanvas"width="300"height="300"style="border: 1pxsolid#000"></canvas>
<script>
var ctx = document.getElementById("Mycanvas").getContext('2d');
    ctx.moveTo(10, 10);
    ctx.quadraticCurveTo(80, 150, 150, 10);
    ctx.stroke();
</script>
```

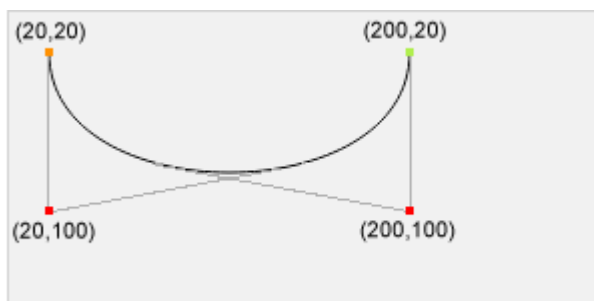
Chạy ví dụ được kết quả như hình sau :



### (7) Vẽ đường cong Bizier Curve

- Phần trước chúng ta đã học cách vẽ đường cong bậc 2 (Quadrati) , với tham số là 3 điểm .

- Phần này chúng ta sẽ vẽ đường cong Bizier với tham số là 4 điểm được mô tả như hình sau:



- Để vẽ đường cong Bizier ta thực hiện các bước sau :

// xác định điểm bắt đầu

```
context.moveTo(startX, startY)
```

// bắt đầu vẽ đường cong với 2 điểm điều khiển và điểm kết thúc

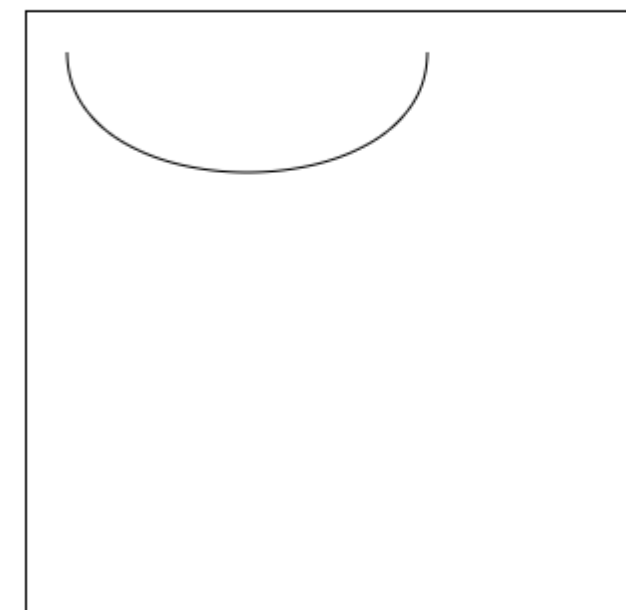
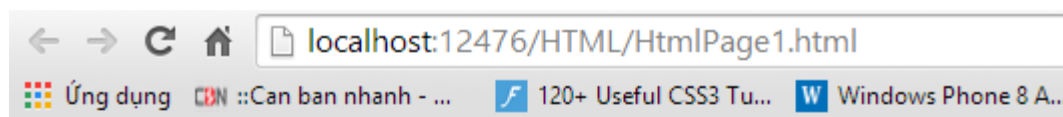
```
context.quadraticCurveTo(Point1X, Point1Y, Point2X, Point2Y, endX, endY);
```

- Ví dụ để vẽ đường cong như hình minh họa trên ta có mã như sau :

HTML :

```
<canvasid="Mycanvas"width="300"height="300"style="border: 1pxsolid#000"></canvas>
<script>
var ctx = document.getElementById("Mycanvas").getContext('2d');
    ctx.beginPath();
    ctx.moveTo(20, 20);
    ctx.bezierCurveTo(20, 100, 200, 100, 200, 20);
    ctx.stroke();
</script>
```

- Chạy ví dụ được kết quả như hình sau :



## (8) Vẽ đường Canvas Path

- Đường canvas path thực ra là nối các đường thẳng , đường cong với nhau .

- Chúng ta có thể sử dụng các phương thức `lineTo()`, `arcTo()`, `quadraticCurveTo ()` và `bezierCurveTo()` để xây dựng mỗi đường line để vẽ nối tiếp và tạo thành 1 đường cong tùy ý .

- Điểm kết thúc của đường này trở thành bối cảnh điểm bắt đầu đường kế tiếp

- Mỗi lần vẽ 1 đường path mới chúng ta gọi phương thức `beginPath()`

- Khi kết thúc đường path chúng ta gọi phương thức `closePath()` để đóng đường Path trở về ngữ cảnh mới. Tức là khi đóng đường path mà muốn vẽ thêm 1 đường bất kỳ chúng ta cần khai báo điểm bắt đầu của đường đó

Ví dụ :

HTML :

```
<canvasid="Mycanvas"width="350"height="300"style="border: 1pxsolid#000"></canvas>
<script>
var ctx = document.getElementById("Mycanvas").getContext('2d');
    ctx.beginPath();
    ctx.moveTo(10, 10);
    ctx.lineTo(100, 100); // đường thẳng
```

```

ctx.quadraticCurveTo(150, 50, 200, 100); // đường quadratic
ctx.bezierCurveTo(250, 50, 270, 40, 300, 150); // đường bezier
ctx.stroke();

```

```
</script>
```

## (9) Vẽ đường tùy chỉnh (Sharp)

- Tương tự như vẽ đường path , với đường sharp thì chúng ta gọi thêm phương thức `closePath()` để tạo thành 1 đường đóng kín.

- Chúng ta có thể tạo các đường thẳng , tròn , cong sau đó nối lại và gọi phương thức `closePath` để tạo thành đường đóng kín .

- Có thể tô màu cho đường cong khép kín bằng phương thức `fillStyle`

```
<canvasid="Mycanvas"width="500"height="300"style="border: 1pxsolid#000"></canvas>
<script>
```

```

var ctx = document.getElementById("Mycanvas").getContext('2d');
ctx.beginPath(); // bắt đầu đường vẽ
ctx.moveTo(170, 80);
ctx.bezierCurveTo(130, 100, 130, 150, 230, 150);
ctx.bezierCurveTo(250, 180, 320, 180, 340, 150);
ctx.bezierCurveTo(420, 150, 420, 120, 390, 100);
ctx.bezierCurveTo(430, 40, 370, 30, 340, 50);
ctx.bezierCurveTo(320, 5, 250, 20, 250, 50);
ctx.bezierCurveTo(200, 5, 150, 20, 170, 80);
ctx.closePath(); // kết thúc đường vẽ

```

```
// trang trí đường viền
```

```
// độ lớn viền
```

```
ctx.lineWidth = 3;
```

```
// màu viền
```

```
ctx.strokeStyle = "blue";
```

```
// màu nền đường cong kín
```

```
ctx.fillStyle = "red";
```

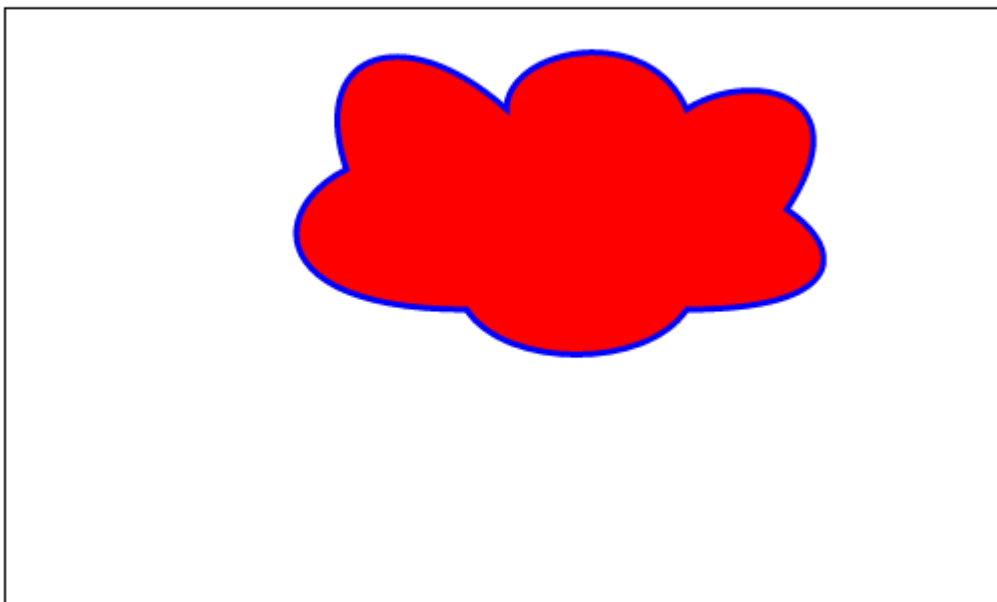
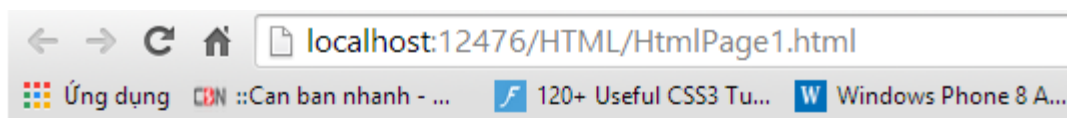
```
ctx.fill();
```

```
ctx.stroke();
```

- javascript hỗ trợ đổ màu bằng gọi tên màu hoặc gọi theo mã màu :

- Ở trên ta có thể gọi màu nền đỏ là : red hoặc thay bằng mã màu sau #ff000

- Chạy ví dụ được kết quả như hình sau :





## (10) Xây dựng thư viện lập trình OOP JavaScript hỗ trợ lập trình HTML5

- Qua một loạt các ví dụ chúng ta thấy có thể vẽ , di chuyển bất kỳ 1 đối tượng nào trên nền canvas trong HTML5 .

- Tuy nhiên với việc xử dụng thuần các phương thức mà HTML5 API cung cấp thì có thể việc lập trình giao diện đồ họa sẽ vô cùng phức tạp và khó khăn .

- Có rất nhiều thư viện javascript hỗ trợ lập trình giao diện đồ họa canvas 2D cũng như 3D rất mạnh như :

WebGL (Web Graphics Library)

Enchant

- Các thư viện này cung cấp khá nhiều lớp đối tượng , hay kỹ thuật lập trình xây dựng giao diện cũng như tương tác trên giao diện cực kỳ hữu ích .

- Mà nếu bạn có ý định tự viết 1 thư viện xử lý đồ họa canvas thì có lẽ cũng phải viết lại rất nhiều lớp giống các thư viện này .

- Ví dụ như với mỗi 1 đối tượng ( điểm , đường , hình ảnh ) các thư viện trên đều sinh ra 1 layer ( như trong photoshop) để chúng ta dễ dàng thao tác đối tượng trên layer đó ( sửa , xóa) mà không ảnh hưởng tới toàn bộ nền canvas

- Tuy nhiên đó là việc làm khi đi sâu phát triển 1 ứng dụng đồ họa (game chẳng hạn) trên nền html5 .

- Hiện tại chúng ta nên đi tìm hiểu bản chất các đối tượng canvas HTML5 sau đó đi sâu vào viết nên các đối tượng

- Chúng tôi sẽ trở lại vấn đề này trong tài liệu hướng dẫn phát triển game HTML5

- Còn ở đây chúng tôi giới thiệu qua các kỹ thuật cơ bản .

- Game là sự kết hợp ngoạn mục giữa toán học ( hình học ) và va chạm vật lý (chúng tôi không nói đến vấn đề đồ họa) .Do đó chúng ta đi xây dựng các đối tượng toán học là điều tất yếu

- Giả sử chúng ta có yêu cầu vẽ 1 đường thẳng thì cần xác định hai điểm , và ta đi định nghĩa đối tượng điểm trong javascript như sau :

```
// định nghĩa 1 điểm , vecto
function Point(X, Y) {
  this.x = X;
  this.y = Y;
}
```

- Và đây là phương thức vẽ 1 đường thẳng từ hai điểm :

```
// vẽ 1 đường thẳng từ 1 điểm
function GetLine(ctx, Point1, Point2) {
  ctx.moveTo(Point1.x, Point1.y);
  ctx.lineTo(Point2.x, Point2.y);
  ctx.stroke();
}
```

- Hay phương thức vẽ ảnh lên canvas như sau :

```
// hàm vẽ ảnh
```

```
function DrawImg(ctx, _src, Point1) {
var w = 50;
var h = 50;
var img = new Image();
    img.src = _src;
    img.onload = function () {
        ctx.drawImage(img, Point1.x, Point1.y, w, h);
    }
}
```

- Các bạn thấy phương pháp xây dựng đối tượng JavaScript có giống đối tượng toán học không ?

- Từ gợi ý đó các bạn có thể tạo ra các phương thức vẽ đường cong bậc hai , đường tròn, cung tròn , các đối tượng mặt cầu , mặt phẳng trong không gian ( để hỗ trợ lập trình 3D)

- Dĩ nhiên với việc xử dụng Javascript thuần thì mã nguồn se rất rối và khó bảo trì .Bản thân javascript là ngôn ngữ không hướng đối tượng .Do đó các bạn có thể dùng các ngôn ngữ như TypeScript để thay thế viết mã Javascript ( TypeScript sẽ biên dịch ra mã JavaScript ) .Và TypeScript là ngôn ngữ hoàn toàn hướng đối tượng như C# ( TypeScript do Microsoft phát triển ) .Các bạn có thể tìm đọc tài liệu về TypeScript của chúng tôi để hiểu rõ hơn về kỹ thuật lập trình này !

## (11) Lập trình cờ tướng bằng canvas HTML5

- Phần cuối về canvas chúng tôi đưa ra một ví dụ rất gần gũi để giúp các bạn có thể lập trình nền game cờ tướng bằng canvas HTML5

- Tất nhiên là chúng tôi đưa ra các phương thức thuần javascript để các bạn hiểu bản chất để có thể từ ý tưởng trên phát triển lên các thư viện của mình .Còn trên thực tế các bạn dùng thêm các thư viện như WebGL , Merchant cũng là như giải pháp khá hữu ích

- Dĩ nhiên là code thuần javascript nên mã nguồn khá là phức tạp , các bạn có thể xem mã nguồn cũng như ảnh các quan cờ trong file đính kèm

- Để dễ hiểu chúng tôi hướng dẫn các bạn 3 bước cơ bản là : vẽ bàn cờ , đặt quân cờ lên bàn cờ và di chuyển quân cờ .

### Bước 1 : Vẽ bàn cờ bằng canvas

- Như đã thảo luận phần trước , chúng ta cần xây dựng 1 số lớp đối tượng hình học để có thể vẽ các đường thẳng cho bàn cờ

- Trong thực tế các bạn có thể chèn 1 bức ảnh làm bàn cờ .Nhưng ở đây chúng tôi vẽ bàn cờ bằng canvas

- Tạo file javascript có tên là : xiangqui.js và có đoạn mã như sau :

(Mã khá dài các bạn có thể tham khảo file đính kèm theo tài liệu)

```
// lớp định nghĩa điểm
function Point(X, Y) {
this.x = X;
this.y = Y;
}
// hàm tịnh tiến
function MoveVecto(Point1, Point2) {
returnnew Point(Point2.x + Point1.x, Point2.y + Point1.y);
}
// hàm vẽ đường thẳng
function DrawLine(ctx, Point1, Point2) {
```

```

    ctx.moveTo(Point1.x, Point1.y);
    ctx.lineTo(Point2.x, Point2.y);
    ctx.stroke();
}

```

- 3 hàm trên 1 hàm định nghĩa 1 đối tượng đường thẳng .Một hàm dùng để vẽ đường thẳng , và một hàm dùng để thực hiện phép tịnh tiến ( phép tính tiến trong hình học)

- Phần tiếp chúng ta sẽ xây dựng các phương thức vẽ quân cờ , định nghĩa và di chuyển quân cờ .Để mã đơn giản , chúng tôi thực hiện từng bước 1

- Việc vẽ bàn cờ thực ra chỉ dùng vòng lặp để vẽ các đường ngang , dọc .Nếu không thích các bạn có thể vẽ trực tiếp 10 đường kẻ ngang và 9 đường kẻ dọc để hình thành bàn cờ

- Đánh dấu các ô tướng , ô tốt , pháo ta cần đúng tọa độ và vẽ các đường đánh dấu

- Hàm vẽ bàn cờ và đánh dấu vị trí các quân ( tướng , pháo , tốt)

```

function Chess(ctx, h) {
var k = h / 10;
// lề padding
var p = h / 15;
// chiều ngang (width)
var w = 8 * k + 2 * p;
function GetLine(Point1, Point2) {
    DrawLine(ctx, Point1, Point2);
}

//#region Vẽ đánh dấu vị trí 1 ô pháo , tốt
function GetFlagI() {
var t1 = 5; var t2 = 10;
function GetFlagI(Point1) {
var O = new Point(Point1.x, Point1.y);
// O ->A : x tăng , y giảm
var A = MoveVecto(O, new Point(t1, -t1));
// A ->A1 :x nguyên , y giảm
var A1 = MoveVecto(A, new Point(0, -t2))
// A->A2 : x tăng , y nguyên
var A2 = MoveVecto(A, new Point(t2, 0));
    GetLine(A, A1);
    GetLine(A, A2);
}
function GetFlagII(Point1) {
var O = new Point(Point1.x, Point1.y);
// O ->A : x giảm , y giảm
var A = MoveVecto(O, new Point(-t1, -t1));
// A ->A1 :x nguyên , y giảm
var A1 = MoveVecto(A, new Point(0, -t2))
// A->A2 : x giảm , y nguyên
var A2 = MoveVecto(A, new Point(-t2, 0));
    GetLine(A, A1);
    GetLine(A, A2);
}
function GetFlagIII(Point1) {
var O = new Point(Point1.x, Point1.y);
// O ->A : x giảm , y tăng
var A = MoveVecto(O, new Point(-t1, t1));
// A ->A1 :x nguyên , y tăng
var A1 = MoveVecto(A, new Point(0, t2))
// A->A2 : x giảm , y nguyên
var A2 = MoveVecto(A, new Point(-t2, 0));
    GetLine(A, A1);
    GetLine(A, A2);
}
function GetFlagIV(Point1) {
var O = new Point(Point1.x, Point1.y);

```

```

// 0 ->A : x tăng , y tăng
var A = MoveVecto(0, new Point(t1, t1));
// A ->A1 :x nguyên , y tăng
var A1 = MoveVecto(A, new Point(0, t2))
// A->A2 : x tăng , y nguyên
var A2 = MoveVecto(A, new Point(t2, 0));
    GetLine(A, A1);
    GetLine(A, A2);
}
this.GetFlagFull = function (Point1) {
    GetFlagI(new Point(Point1.x, Point1.y));
    GetFlagII(new Point(Point1.x, Point1.y));
    GetFlagIII(new Point(Point1.x, Point1.y));
    GetFlagIV(new Point(Point1.x, Point1.y));
}
this.GetFlagLeft = function (Point1) {
    GetFlagII(new Point(Point1.x, Point1.y));
    GetFlagIII(new Point(Point1.x, Point1.y));
}
this.GetFlagRight = function (Point1) {
    GetFlagI(new Point(Point1.x, Point1.y));
    GetFlagIV(new Point(Point1.x, Point1.y));
}
}
}
//#endregion
this.Getboard = function () {
// Tịnh tiến hệ tọa độ về (10,10)
    ctx.translate(p, p / 1.5);
//#region Vẽ dòng , cột
// Vẽ dòng
for (var i = 0; i < 10; i++) {
    GetLine(new Point(0, k * i), new Point(8 * k, k * i));
}
// Vẽ cột
for (var i = 0; i < 9; i++) {
if ((i == 0) | (i == 8)) {
    GetLine(new Point(i * k, 0), new Point(i * k, 9 * k));
}
else {
    GetLine(new Point(i * k, 0), new Point(i * k, 4 * k));
    GetLine(new Point(i * k, 5 * k), new Point(i * k, 9 * k));
}
}
}
//#endregion
//#region Vẽ chỉ số dòng , cột
    ctx.font = 'italic 14px Calibri';
// Chỉ số cột
for (var i = 0; i < 9; i++) {
    ctx.fillText(10 - (i + 1), k * i, h - 25);
}
// Chỉ số dòng
var Alphabet = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J"]
for (var i = 0; i <= Alphabet.length; i++) {
// có thể đưa về hệ (0,0) ban đầu để có chỉ số dương
    ctx.fillText(Alphabet[Alphabet.length - (i + 1)], -20, k * i);
}
}
//#endregion
//#region Danh dau o tuong
// Vẽ ô tướng 1
    GetLine(new Point(3 * k, 0), new Point(5 * k, 2 * k));
    GetLine(new Point(3 * k, 2 * k), new Point(5 * k, 0));
// Vẽ ô tướng 2
    GetLine(new Point(3 * k, 7 * k), new Point(5 * k, 9 * k));
    GetLine(new Point(3 * k, 9 * k), new Point(5 * k, 7 * k));
}
//#endregion

```

```

//#region Đánh dấu ô tốt , pháo
// Mảng lưu vị trí cần đánh dấu
var _FlagFull = [
new Point(k, 2 * k),
new Point(7 * k, 2 * k),
new Point(k, 7 * k),
new Point(7 * k, 7 * k),

new Point(0, 3 * k),
new Point(2 * k, 3 * k),
new Point(4 * k, 3 * k),
new Point(6 * k, 3 * k),
new Point(8 * k, 3 * k),

new Point(0, 6 * k),
new Point(2 * k, 6 * k),
new Point(4 * k, 6 * k),
new Point(6 * k, 6 * k),
new Point(8 * k, 6 * k)
];
var _GetFlag = new GetFlag();
for (var i = 0; i <= _FlagFull.length; i++) {
switch (_FlagFull[i].x) {
case (0): _GetFlag.GetFlagRight(_FlagFull[i]); break;
case (8 * k): _GetFlag.GetFlagLeft(_FlagFull[i]); break;
default: _GetFlag.GetFlagFull(_FlagFull[i]);
}
}
}
}
//#endregion
}
//#region GetPosition
function GetPosition() {

for (var i = 0; i < k; i++) {

}
}
}
}
//#endregion
}

```

- Đối tượng Chess nhận vào 2 tham số là tag canvas và chiều cao bàn cờ

- Phương thức Getboard() để vẽ bàn cờ

- Sử dụng các đối tượng đã tạo ra như sau :

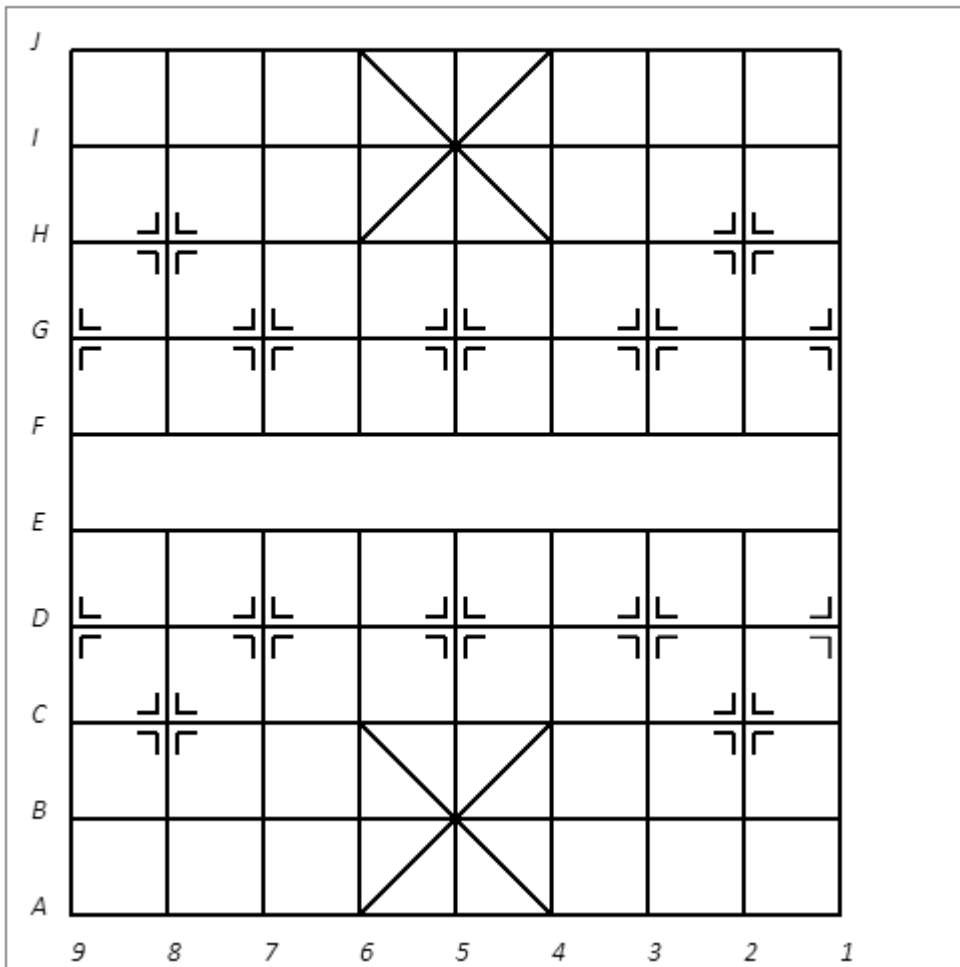
Mã HTML :

```

<scriptsrc="xiangqui.js"></script>
<canvasid="canvas"width="480"height="480"style="border: 1pxsolidgray;"></canvas>
<script>
var ctx = document.getElementById("canvas").getContext("2d");
var Chess1 = new Chess(ctx,480);
    Chess1.Getboard();
</script>

```

- Chạy ví dụ được kết quả như hình sau :



- Nói chung mã javascript là khá nhiều , nhưng khi bạn đã hiểu về các phép biến đổi hình học giải tích thì nó rất đơn giản .Sau này các bạn có thể xây dựng thành các thư viện dùng cho nhiều dự án khác nhau .

- Và các bạn thấy bàn cờ đã được hiển thị lên thẻ canvas rất chính xác .Tất các các đường thẳng , trang trí đều được vẽ bằng canvas .

- Nếu muốn bàn cờ đẹp hơn , bạn có thể vẽ các đối tượng hình ảnh lên thẻ canvas để bàn cờ thêm sinh động

### Bước 2 : Đặt quân cờ lên bàn cờ

- Chúng ta đã vẽ được bàn cờ lên thẻ canvas , phần này chúng ta sẽ đặt các quân cờ lên bàn cờ

- Kỹ thuật thì đơn giản thôi , ta lấy các bức ảnh quân cờ , rồi vẽ chính xác lên các tọa độ

- Ta lưu trữ vị trí các quân cờ trên bàn cờ vào 1 mảng, sau đó duyệt các phần tử trong mảng để hiển thị lên bàn cờ

- Ta có thư mục chứa ảnh các quân cờ trong thư mục ChessManImage

### Bước 3 : Di chuyển quân cờ

- Việc di chuyển quân cờ ta áp dụng kỹ thuật tương tự như việc di chuyển 1 bức ảnh .

- Ở đây ta thực hiện giải pháp sau :

+ ) Mỗi nước di chuyển , ta sẽ lưu tọa độ các quân sau khi di chuyển vào 1 mảng

+) Xóa bàn cờ

+) Đặt lại các quân cờ ở vị trí mới lên bàn cờ

- Nếu dùng các thư viện javascript như WebGL , Mechant thì việc di chuyển 1 đối tượng bằng layer thì sẽ đơn giản hơn

Mã toàn bộ file xiangqui.js như sau :

( Có thể xem mã trong file đính kèm tài liệu )

```
//#region Lớp cơ bản
// lớp định nghĩa điểm
function Point(X, Y) {
  this.x = X;
  this.y = Y;
}
// hàm tịnh tiến
function MoveVecto(Point1, Point2) {
  return new Point(Point2.x + Point1.x, Point2.y + Point1.y);
}
// hàm vẽ đường thẳng
function DrawLine(ctx, Point1, Point2) {
  ctx.moveTo(Point1.x, Point1.y);
  ctx.lineTo(Point2.x, Point2.y);
  ctx.stroke();
}
function RemoveObj(ctx, Point1, w, h) {
  ctx.clearRect(Point1.x - w / 2, Point1.y - h / 2, w, h);
}
//#endregion

//#region Vẽ ảnh Canvas
// hàm vẽ ảnh tại tọa độ bất kỳ
function DrawImg(ctx, _src, Point1) {
  var img = new Image();
  img.src = _src;
  img.onload = function () {
    ctx.drawImage(img, Point1.x, Point1.y, 50, 50);
  }
}
// vẽ ảnh vào tâm
function DrawImgCenter(ctx, _src, Point1) {
  var w = 50;
  var h = 50;
  var img = new Image();
  img.src = _src;
  img.onload = function () {
    ctx.drawImage(img, Point1.x - w / 2, Point1.y - h / 2, w, h);
  }
}
//#endregion
//#region Lớp định nghĩa quân cờ , nước đi
// Định nghĩa quân cờ
function Piece(_name, _IsRead, _img) {
  this.Name = _name;
  this.Img = _img;
  this.IsRed = _IsRead;
}
// Định nghĩa quân cờ và tọa độ
function PieceXy(_piece, Point1) {
  this.Piece = _piece;
  this.Position = Point1;
}
function ConvertChessMove() {
}
}
```

```

//#endregion

//#region khai báo các quân cờ
// quân đỏ

var King1 = new Piece("Tướng", true, "ChessManImage/Tg1.png");
var Bishop1 = new Piece("Sỹ", true, "ChessManImage/Sy1.png");
var Elephant1 = new Piece("Tượng", true, "ChessManImage/Tuong1.png");
var Rook1 = new Piece("Xe", true, "ChessManImage/Xe1.png");
var Cannon1 = new Piece("Pháo", true, "ChessManImage/Phao1.png");
var Knight1 = new Piece("Mã", true, "ChessManImage/Ma1.png");
var Pawn1 = new Piece("Tốt", true, "ChessManImage/Tot1.png");

// quân đen
var King2 = new Piece("Tướng", false, "ChessManImage/Tg2.png");
var Bishop2 = new Piece("Sỹ", false, "ChessManImage/Sy2.png");
var Elephant2 = new Piece("Tượng", false, "ChessManImage/Tuong2.png");
var Rook2 = new Piece("Xe", false, "ChessManImage/Xe2.png");
var Cannon2 = new Piece("Pháo", false, "ChessManImage/Phao2.png");
var Knight2 = new Piece("Mã", false, "ChessManImage/Ma2.png");
var Pawn2 = new Piece("Tốt", false, "ChessManImage/Tot2.png");
//#endregion

//#region Hàm định tọa độ quân cờ
// định tọa độ kiểu Ai (A,4)
function GetXY(row, column, k) {
var x = (9 - column) * k;
var y;
switch (row) {
case ("A"): y = 9 * k; break;
case ("B"): y = 8 * k; break;
case ("C"): y = 7 * k; break;
case ("D"): y = 6 * k; break;
case ("E"): y = 5 * k; break;
case ("F"): y = 4 * k; break;
case ("G"): y = 3 * k; break;
case ("H"): y = 2 * k; break;
case ("I"): y = k; break;
case ("J"): y = 0; break;
}
return new Point(x, y);
}
//#endregion

//#region Đối tượng nước đi
// nước đi
function MovePiece(Point1, Point2) {
this.OldPosition = Point1;
this.NewPosition = Point2;
}
// Nước cờ : Gồm 2 nước đi cho bên đen và trắng
function MoveChess(_light, _dark) {
// nước đi quân đỏ
this.Light = _light;
// nước đi quân đen
this.Dark = _dark;
}
//#endregion

//#region Vẽ bàn cờ
function Chess(ctx, h) {
var k = h / 10;
// lề padding
var p = h / 15;
// chiều ngang (width)
var w = 8 * k + 2 * p;
function GetLine(Point1, Point2) {
DrawLine(ctx, Point1, Point2);
}
}

```



```

//#region Vẽ đánh dấu vị trí 1 ô pháo , tốt
function GetFlag() {
var t1 = 5; var t2 = 10;
function GetFlagI(Point1) {
var O = new Point(Point1.x, Point1.y);
// O ->A : x tăng , y giảm
var A = MoveVecto(O, new Point(t1, -t1));
// A ->A1 :x nguyên , y giảm
var A1 = MoveVecto(A, new Point(0, -t2))
// A->A2 : x tăng , y nguyên
var A2 = MoveVecto(A, new Point(t2, 0));
        GetLine(A, A1);
        GetLine(A, A2);
    }
function GetFlagII(Point1) {
var O = new Point(Point1.x, Point1.y);
// O ->A : x giảm , y giảm
var A = MoveVecto(O, new Point(-t1, -t1));
// A ->A1 :x nguyên , y giảm
var A1 = MoveVecto(A, new Point(0, -t2))
// A->A2 : x giảm , y nguyên
var A2 = MoveVecto(A, new Point(-t2, 0));
        GetLine(A, A1);
        GetLine(A, A2);
    }
function GetFlagIII(Point1) {
var O = new Point(Point1.x, Point1.y);
// O ->A : x giảm , y tăng
var A = MoveVecto(O, new Point(-t1, t1));
// A ->A1 :x nguyên , y tăng
var A1 = MoveVecto(A, new Point(0, t2))
// A->A2 : x giảm , y nguyên
var A2 = MoveVecto(A, new Point(-t2, 0));
        GetLine(A, A1);
        GetLine(A, A2);
    }
function GetFlagIV(Point1) {
var O = new Point(Point1.x, Point1.y);
// O ->A : x tăng , y tăng
var A = MoveVecto(O, new Point(t1, t1));
// A ->A1 :x nguyên , y tăng
var A1 = MoveVecto(A, new Point(0, t2))
// A->A2 : x tăng , y nguyên
var A2 = MoveVecto(A, new Point(t2, 0));
        GetLine(A, A1);
        GetLine(A, A2);
    }
this.GetFlagFull = function (Point1) {
        GetFlagI(new Point(Point1.x, Point1.y));
        GetFlagII(new Point(Point1.x, Point1.y));
        GetFlagIII(new Point(Point1.x, Point1.y));
        GetFlagIV(new Point(Point1.x, Point1.y));
    }
this.GetFlagLeft = function (Point1) {
        GetFlagII(new Point(Point1.x, Point1.y));
        GetFlagIII(new Point(Point1.x, Point1.y));
    }
this.GetFlagRight = function (Point1) {
        GetFlagI(new Point(Point1.x, Point1.y));
        GetFlagIV(new Point(Point1.x, Point1.y));
    }
}
//#endregion

function AddPiece(_Piece, Point1) {
    DrawImgCenter(ctx, _Piece.Img, Point1);
}

```

```

function GetPosition(row, column) {
return GetXY(row, column, k);
}
this.Getboard = function () {
// Tịnh tiến hệ tọa độ về (10,10)
ctx.translate(p, p / 1.5);

//#region Vẽ dòng , cột
// Vẽ dòng
for (var i = 0; i < 10; i++) {
GetLine(new Point(0, k * i), new Point(8 * k, k * i));
}

// Vẽ cột
for (var i = 0; i < 9; i++) {
if ((i == 0) | (i == 8)) {
GetLine(new Point(i * k, 0), new Point(i * k, 9 * k));
}
else {
GetLine(new Point(i * k, 0), new Point(i * k, 4 * k));

GetLine(new Point(i * k, 5 * k), new Point(i * k, 9 * k));
}
}
}
//#endregion

//#region Vẽ chỉ số dòng , cột
ctx.font = 'italic 14px Calibri';
// Chỉ số cột
for (var i = 0; i < 9; i++) {
ctx.fillText(10 - (i + 1), k * i, h - 25);
}

// Chỉ số dòng
var Alphabet = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J"]
for (var i = 0; i <= Alphabet.length; i++) {
// có thể đưa về hệ (0,0) ban đầu để có chỉ số dương
ctx.fillText(Alphabet[Alphabet.length - (i + 1)], -20, k * i);
}
}
//#endregion

//#region Danh dau o tuong
// Vẽ ô tướng 1
GetLine(new Point(3 * k, 0), new Point(5 * k, 2 * k));
GetLine(new Point(3 * k, 2 * k), new Point(5 * k, 0));

// Vẽ ô tướng 2
GetLine(new Point(3 * k, 7 * k), new Point(5 * k, 9 * k));
GetLine(new Point(3 * k, 9 * k), new Point(5 * k, 7 * k));
}
//#endregion

//#region Đánh dấu ô tốt , pháo
// Mảng lưu vị trí cần đánh dấu
var _FlagFull = [
new Point(k, 2 * k),
new Point(7 * k, 2 * k),
new Point(k, 7 * k),
new Point(7 * k, 7 * k),

new Point(0, 3 * k),
new Point(2 * k, 3 * k),
new Point(4 * k, 3 * k),
new Point(6 * k, 3 * k),
new Point(8 * k, 3 * k),

new Point(0, 6 * k),
new Point(2 * k, 6 * k),
new Point(4 * k, 6 * k),
new Point(6 * k, 6 * k),

```

```

new Point(8 * k, 6 * k)
    ];
var _GetFlag = new GetFlag();
for (var i = 0; i <= _FlagFull.length; i++) {
switch (_FlagFull[i].x) {
case (0): _GetFlag.GetFlagRight(_FlagFull[i]); break;
case (8 * k): _GetFlag.GetFlagLeft(_FlagFull[i]); break;
default: _GetFlag.GetFlagFull(_FlagFull[i]);
    }
}
}
}

//#region phương thức Thêm quân cờ vào bàn cờ
// thêm 1 quân
this.AddPiece = function (_Piece, Point1) {
    AddPiece(_Piece, Point1);
}
// thêm nhiều quân
this.AddPieces = function (_Pieces) {
for (var i in _Pieces) {
    AddPiece(_Pieces[i].Piece, _Pieces[i].Position);
}
}
}
}

//#region Di chuyển quân cờ

/*
    Tham số :
    - Danh sách các quân cờ trên bàn cờ
    - Tọa độ cũ
    - Tọa độ mới
*/

this.MoveChess = function (_Pieces, OldPoint, NewPoint) {
// Lỗi , khi này giá trị biến tham số _Piece cũng bị thay đổi

var _NewPiece = [];
for (var i in _Pieces) {
    _NewPiece.push(new PieceXy(_Pieces[i].Piece, _Pieces[i].Position));
}
var CurrentPiece = _Pieces[i].Position;
if ((CurrentPiece.x == OldPoint.x) & (CurrentPiece.y == OldPoint.y)) {
    _NewPiece[i].Position = NewPoint;
}
}
return _NewPiece;
}

}

//#endregion
// k = 60
this.GetPosition = function (row, column) {
return GetPosition(row, column);
}

}

//#region Phương thức khai báo tọa độ các quân cờ xuất phát
this.GetStartChess = function () {
// khai báo vị trí mặc định
var StartChessMan = [
new PieceXy(Rook1, GetPosition("A", 1)),
new PieceXy(Knight1, GetPosition("A", 2)),
new PieceXy(Elephant1, GetPosition("A", 3)),
new PieceXy(Bishop1, GetPosition("A", 4)),
new PieceXy(King1, GetPosition("A", 5)),
new PieceXy(Bishop1, GetPosition("A", 6)),
new PieceXy(Elephant1, GetPosition("A", 7)),
new PieceXy(Knight1, GetPosition("A", 8)),

```

```

new PieceXy(Rook1, GetPosition("A", 9)),

new PieceXy(Cannon1, GetPosition("C", 2)),
new PieceXy(Cannon1, GetPosition("C", 8)),

new PieceXy(Pawn1, GetPosition("D", 1)),
new PieceXy(Pawn1, GetPosition("D", 3)),
new PieceXy(Pawn1, GetPosition("D", 5)),
new PieceXy(Pawn1, GetPosition("D", 7)),
new PieceXy(Pawn1, GetPosition("D", 9)),

new PieceXy(Rook2, GetPosition("J", 1)),
new PieceXy(Knight2, GetPosition("J", 2)),
new PieceXy(Elephant2, GetPosition("J", 3)),
new PieceXy(Bishop2, GetPosition("J", 4)),
new PieceXy(King2, GetPosition("J", 5)),
new PieceXy(Bishop2, GetPosition("J", 6)),
new PieceXy(Elephant2, GetPosition("J", 7)),
new PieceXy(Knight2, GetPosition("J", 8)),
new PieceXy(Rook2, GetPosition("J", 9)),

new PieceXy(Cannon2, GetPosition("H", 2)),
new PieceXy(Cannon2, GetPosition("H", 8)),

new PieceXy(Pawn2, GetPosition("G", 1)),
new PieceXy(Pawn2, GetPosition("G", 3)),
new PieceXy(Pawn2, GetPosition("G", 5)),
new PieceXy(Pawn2, GetPosition("G", 7)),
new PieceXy(Pawn2, GetPosition("G", 9))
    ];
return StartChessMan;
}
//#endregion
}
//#endregion

//#region ký hiệu quân cờ
/*
Quân cờ việt :
Tướng : T
Tượng :V (Voi)
Tốt : C(Chốt )
*/
*/
//#endregion

```

## Sử dụng Mã

```

<scriptsrc="xiangqui.js"></script>
<canvasid="canvas"width="480"height="480"style="border: 1pxsolidgray;"></canvas>
<div>
<buttononclick="Begin()"><</button>
<buttononclick="PreView()"><</button>
<buttononclick="Next()">></button>
<buttononclick="End()">></button>
</div>
<scripttype="text/javascript">
var ctx = document.getElementById("canvas").getContext("2d");
var Chess1 = new Chess(ctx, 480);
//#region dữ liệu ván cờ
// mảng lưu trữ nước đi
var MoveChess = [
new MoveChess(new MovePiece(Chess1.GetPosition("C", 2), Chess1.GetPosition("C", 5)), new
MovePiece(Chess1.GetPosition("J", 2), Chess1.GetPosition("H", 3))),
new MoveChess(new MovePiece(Chess1.GetPosition("A", 2), Chess1.GetPosition("C", 3)), new
MovePiece(Chess1.GetPosition("J", 8), Chess1.GetPosition("H", 7))),

```

```

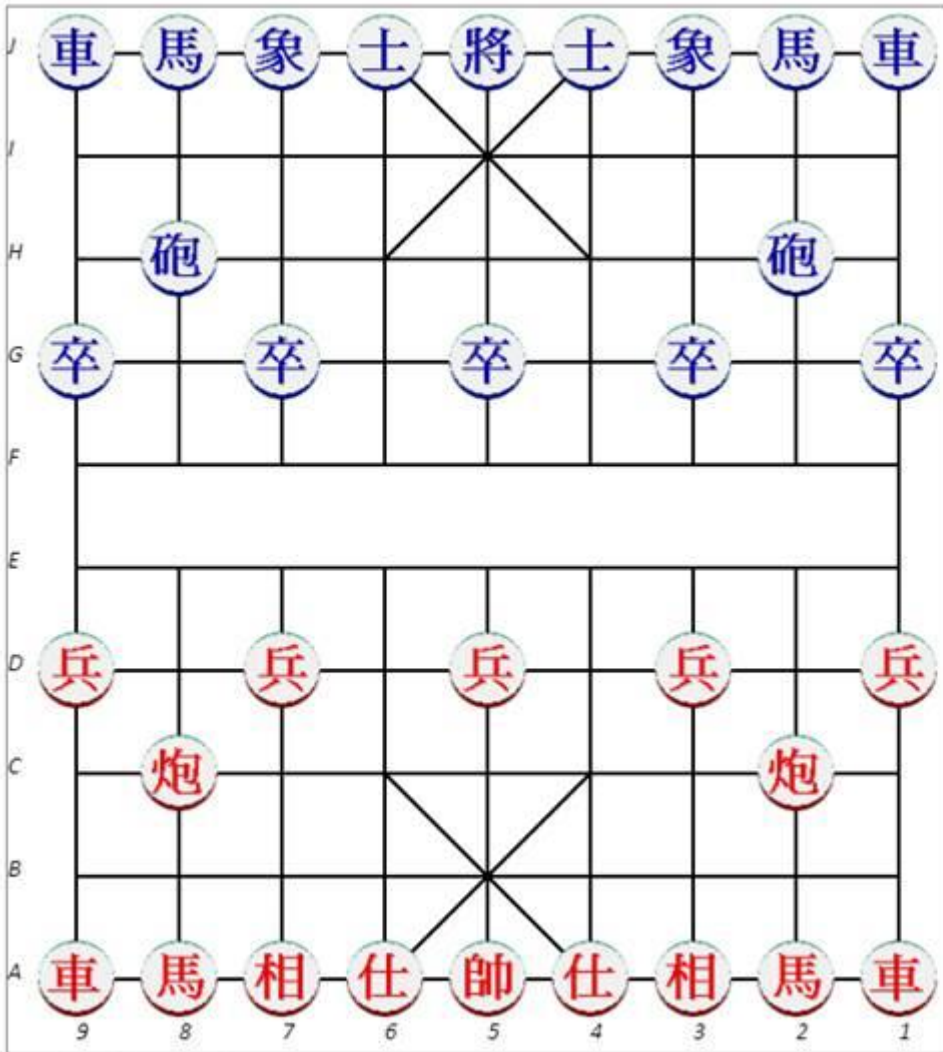
new MoveChess(new MovePiece(Chess1.GetPosition("A", 1), Chess1.GetPosition("A", 2)), new
MovePiece(Chess1.GetPosition("J", 1), Chess1.GetPosition("J", 2)))
    ];
// mảng lưu trữ vị trí bàn cờ sau mỗi nước đi, mỗi nước đi sinh ra 1 vị trí
var _StartChess = Chess1.GetStartChess();
var BoardChess = [Chess1.GetStartChess()];
for (var i in MoveChess) {
var _Light = MoveChess[i].Light;
var _Dark = MoveChess[i].Dark;
var _StartChess1 = Chess1.MoveChess(_StartChess, _Light.OldPosition, _Light.NewPosition);
    BoardChess.push(_StartChess1);
var _StartChess2 = Chess1.MoveChess(_StartChess1, _Dark.OldPosition, _Dark.NewPosition);
    BoardChess.push(_StartChess2);
    _StartChess = _StartChess2;
}
//#endregion
var BoardIndex = 0;
    Chess1.AddPieces(BoardChess[BoardIndex]);
    Chess1.Getboard();

//#region Điều hướng nước đi
function ShowCurrentBoard(k) {
    ctx.translate(-600 / 15, -600 / 15 / 1.5);
    ctx.clearRect(0, 0, c.width, c.height);
    ctx.beginPath();
    Chess1.AddPieces(BoardChess[k]);
    Chess1.Getboard();
}
function DiChuyen(k) {
    BoardIndex = k;
    ShowCurrentBoard(BoardIndex);
}
function Next() {
if (BoardIndex < BoardChess.length - 1) {
    BoardIndex += 1;
    ShowCurrentBoard(BoardIndex);
}
}
function PreView() {
if (BoardIndex >= 1) {
    BoardIndex -= 1;
    ShowCurrentBoard(BoardIndex);
}
}
function End() {
    BoardIndex = BoardChess.length;
    ShowCurrentBoard(BoardIndex - 1);
}
function Begin() {
    BoardIndex = 0;
    ShowCurrentBoard(BoardIndex);
}
//#endregion
</script>

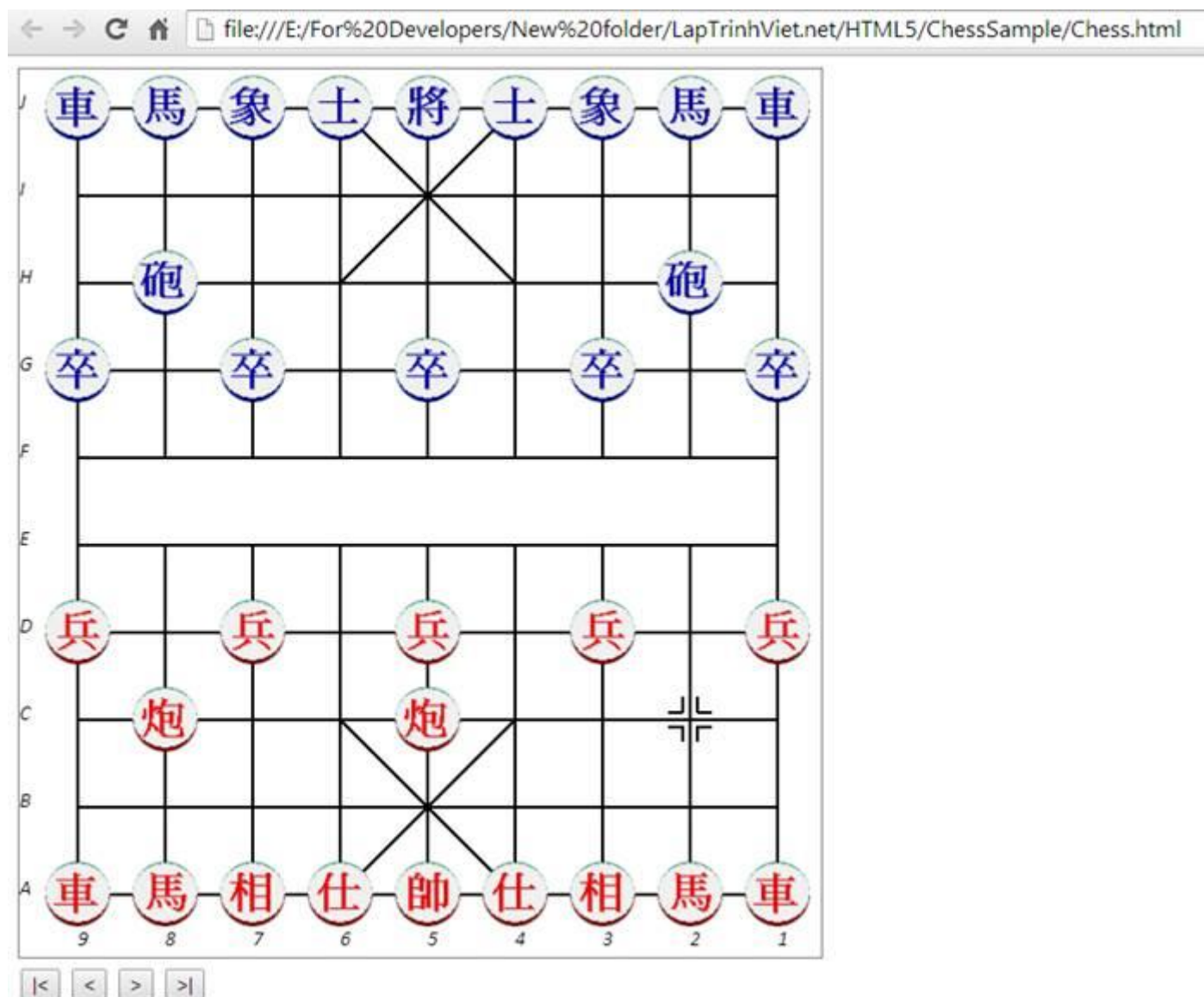
```

- Chạy ví dụ được kết quả như hình sau

( Các bạn có thể tham khảo mã đi cùng tài liệu để hiểu rõ hơn )



- Click vào các button Next , Preview ... để xem sự di chuyển của các quân cờ  
Các quân cờ di chuyển như hình sau :



## 4.2 HTML5 SVG

- SVG là viết tắt của từ : Scalable Vector Graphics

- SVG dùng để vẽ các đối tượng đồ họa vecto trên web dựa trên các định dạng xml

- Khác với canvas vẽ các đối tượng đồ họa dùng javascript , còn svg vẽ các đối tượng dựa vào các khai báo của chúng trong các thuộc tính kiểu xml

- Vẽ các đối tượng đồ họa trong svg tương tự vẽ các đối tượng đồ họa trong xaml của wpf

- Do SVG vẽ đối tượng dựa trên thuộc tính xml , lên xây các đối tượng trở nên tĩnh , rất khó áp dụng cho việc xây dựng các đối tượng động như trong các game , các giao diện động .

- Tuy nhiên SVG cũng rất hữu ích khi định nghĩa các đối tượng đồ họa vecto

Ví dụ sau vẽ một ngôi sao năm cánh :

```
<!DOCTYPEhtml>
<htmlxmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
</head>
<body>
<svgwidth="300"height="200">
```

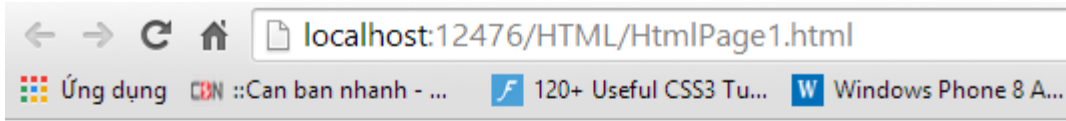
```

<polygonpoints="100,10 40,180 190,60 10,60 160,180"
style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;"/>
    Sorry, your browser does not support inline SVG.
</svg>
</body>

</html>

```

- Chạy ví dụ được kết quả như hình sau :



- Các bạn thử vẽ giao diện trên bằng canvas và cảm nhận độ khó của nó so với dùng svg, qua đó so sánh và nhận xét được khi nào nên dùng canvas và khi nào nên dùng svg .

## 5. HTML 5 đa phương tiện (Media)

- HTML5 cung cấp các tính năng phát nhạc , video thông qua các thẻ video , audio

- Kết hợp với javascript chúng ta có thể xây dựng được các trình phát nhạc rất đẹp và chuyên nghiệp.

- Bạn có thể xem các trình phát nhạc trên youtube được code bằng HTML5 ( chỉ chạy youtube trên các trình duyệt mới )

- Mặc định trình phát nhạc , video của HTML5 rất đơn giản tuy nhiên chúng ta có thể code thêm bằng javascript để cho trình media trở lên đẹp và chuyên nghiệp hơn. Hoặc bạn có thể download rất nhiều mã nguồn mở html5 cho trình Media

- Đây là tính năng khá mạnh của HTML5 , điều mà trước đây để xây dựng các trình phát nhạc , video chúng ta cần dùng tới flash , nhưng với HTML 5 thì chúng ta không cần

### 5.1 Phát Video trong HTML5

#### (1) Trình xem video mặc định

- HTML5 phát video thông qua thẻ video , thẻ source chỉ định đường dẫn tới file video

```

<videowidth="320"height="240"controls>
<sourcesrc="movie.mp4"type="video/mp4">
</video>

```

- Chạy ví dụ được kết quả như hình sau :





- HTML5 video hỗ trợ mở các định dạng file video là : .MP4 ,.Ogg , WebM

- Ta có thể mở nhiều video trong thẻ video như ví dụ sau :

```
<videowidth="320"height="240"controls>  
<sourcesrc="movie.mp4"type="video/mp4">  
<sourcesrc="movie.ogg"type="video/ogg">  
</video>
```

## (2) Tùy chỉnh trình mở video

- HTML5 cung cấp các API để chúng ta có thể tùy chỉnh trình xem video theo cách riêng của bạn

- HTML5 Video cung cấp các API hỗ trợ các điều khiển cơ bản như : play , pause và volume

Ví dụ :

HTML :

```
<divstyle="text-align:center">  
<buttononclick="playPause()">Play/Pause</button>  
<buttononclick="makeBig()">To</button>  
<buttononclick="makeSmall()">Nhỏ</button>  
<buttononclick="makeNormal()">Bình Thường</button>  
<br>  
<videoid="video1"width="420">  
<sourcesrc="mov_bbb.mp4"type="video/mp4">  
<sourcesrc="mov_bbb.ogg"type="video/ogg">  
    Trình duyệt không hỗ trợ HTML5 Video  
</video>  
</div>  
<script>  
var myVideo=document.getElementById("video1");  
  
function playPause()  
{  
if (myVideo.paused)  
    myVideo.play();  
else  
    myVideo.pause();  
}  
  
function makeBig()  
{  
    myVideo.width=560;  
}  
  
function makeSmall()  
{
```

```

myVideo.width=320;
}

function makeNormal()
{
myVideo.width=420;
}
</script>

```

- Chạy ví dụ được kết quả như hình sau :



## 5.2 Phát nhạc trong HTML5

- HTML5 hỗ trợ phát nhạc thông qua tag audio .Các sử dụng và tùy chỉnh trình nghe nhạc HTML5 tương tự thẻ video

Ví dụ :

HTML :

```

<audiocontrols>
<source src="horse.ogg" type="audio/ogg">
<source src="horse.mp3" type="audio/mpeg">
Trình duyệt không hỗ trợ thẻ audio
</audio>

```

Chạy ví dụ được kết quả như hình sau :



- Tương tự trình phát video , trình phát nhạc cũng hỗ trợ các điều khiển cơ bản như :play,pause và volume

- Trình phát nhạc HTML5 hỗ trợ các định dạng file sau : MP3 , WAV và Ogg

## 6. HTML5 API

### 6.1 Xác định vùng địa lý với HTML5 Geolocation

#### (1) Giới thiệu về Geolocation

- Geolocation là API HTML5 dùng để xác định vị trí địa lý của người dùng

- Với Geolocation chúng ta dễ dàng lấy được vị trí ( kinh độ , vĩ độ ) của người dùng .

- Kết hợp với các API của google maps chúng ta dễ dàng xác định vị trí trên bản đồ của người dùng
- Vấn đề cung cấp thông tin vị trí của người dùng là vấn đề cá nhân .Nên để hiển thị các thông tin thì cần được người dùng cho phép
- Ví dụ dưới đây sử dụng phương thức `getCurrentPosition()` để lấy về kinh độ và vĩ độ của người dùng ( người truy cập website)

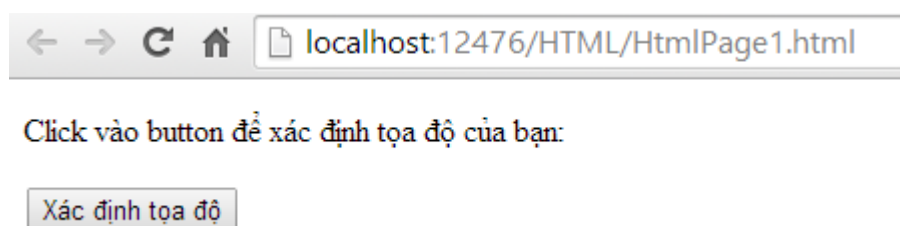
HTML :

```
<pid="demo">Click vào button để xác định tọa độ của bạn:</p>
<buttononclick="getLocation()">Xác định tọa độ</button>
<script>
var x = document.getElementById("demo");
function getLocation()
{
if (navigator.geolocation)
{
navigator.geolocation.getCurrentPosition(showPosition);
}
else{x.innerHTML = "Trình duyệt không hỗ trợ Geolocation API";}
}
function showPosition(position)
{
x.innerHTML = "Vĩ độ: " + position.coords.latitude +
"<br>Kinh độ: " + position.coords.longitude;
}
</script>
```

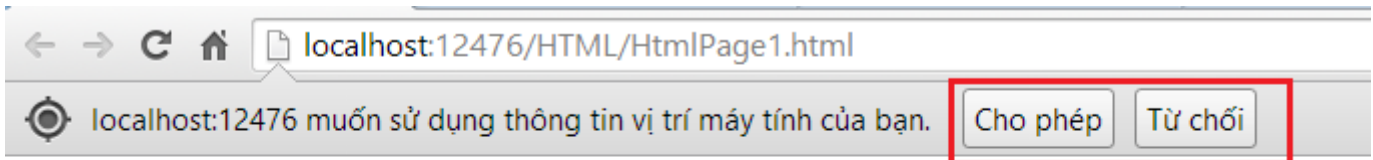
### Giải thích code :

Đoạn code trên thực hiện các công việc sau :

- Kiểm tra Geolocation có được hỗ trợ không ?Nếu có thì chạy phương thức `getCurrentPosition()` , nếu không thì đưa ra thông báo cho người sử dụng là trình duyệt không hỗ trợ .
- Phương thức `getCurrentPosition()` trả về kinh độ và vĩ độ của người dùng .
- Và hiển thị thông tin kinh độ , vĩ độ lên tài liệu html
- Chạy ví dụ được kết quả như hình sau :



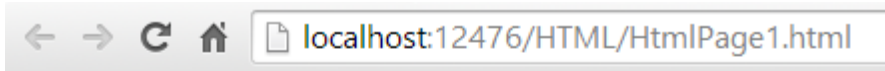
- Click vào button sẽ hiện ra 1 thông báo có cho phép hoặc không cho phép lấy thông tin Geolocation như hình sau :



Click vào button để xác định tọa độ của bạn:

Xác định tọa độ

- Người dùng có thể cho phép hoặc không cho phép lấy thông tin vị trí .Nếu chọn cho phép được kết quả như hình sau :



Vĩ độ: 20.9904703

Kinh độ: 105.8498151

Xác định tọa độ

- Kết quả hiển thị vị trí máy của chúng tôi .Với máy của các bạn sẽ hiển thị vị trí vĩ độ và kinh độ khác .

## (2) Hàm xử lý lỗi cho Geolocation

- Chúng ta có thể đưa ra các thông báo lỗi không thể truy cập geolocation API bằng hàm dưới đây

```
function showError(error) {
  switch (error.code) {
    case error.PERMISSION_DENIED:
      x.innerHTML = "Người dùng không cho phép theo dõi vị trí"
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML = "Không thể tìm vị trí người dùng"
      break;
    case error.TIMEOUT:
      x.innerHTML = "Quá thời gian cho phép"
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML = "Lỗi không xác định"
      break;
  }
}
```

- Hàm trên xử lý 3 lỗi chính đó là :

Permission denied : Người dùng không cho phép tính năng theo dõi vị trí

Position unavailable : Không thể tìm được vị trí hiện tại

Timeout : Quá thời gian cho phép

Sử dụng hàm trên như sau :

HTML :

```
<pid="demo">Click button để xác định tọa độ</p>
<buttononclick="getLocation()">Xác định tọa độ</button>
<script>
```

```

var x = document.getElementById("demo");
function getLocation() {
if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(showPosition, showError);
}
else { x.innerHTML = "Trình duyệt không hỗ trợ Geolocation API";}
}
function showPosition(position) {
x.innerHTML = "Kinh Độ: " + position.coords.latitude +
"<br>Vĩ Độ: " + position.coords.longitude;
}
function showError(error) {
switch (error.code) {
case error.PERMISSION_DENIED:
x.innerHTML = "Người dùng không cho phép theo dõi vị trí"
break;
case error.POSITION_UNAVAILABLE:
x.innerHTML = "Không thể tìm vị trí người dùng"
break;
case error.TIMEOUT:
x.innerHTML = "Quá thời gian cho phép"
break;
case error.UNKNOWN_ERROR:
x.innerHTML = "Lỗi không xác định"
break;
}
}
}
</script>

```

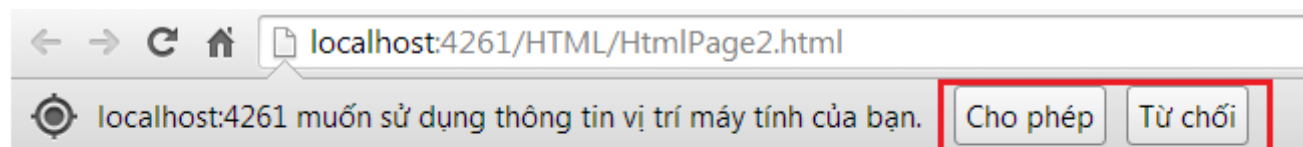
Chạy ví dụ được kết quả sau :



Click button để xác định tọa độ

Xác định tọa độ

- Click vào button hiện ra bảng thông báo :

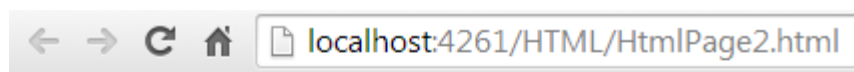


Click button để xác định tọa độ

Xác định tọa độ

- Bạn click vào cho phép hay từ chối sẽ hiện kết quả tương ứng

- Click từ chối



Người dùng không cho phép theo dõi vị trí

Xác định tọa độ

- Click cho phép :

Kinh Độ: 20.9905375

Vĩ Độ: 105.8498595

Xác định tọa độ

### (3) Hiển thị vị trí người dùng lên bản đồ google maps

- Google maps cung cấp api hiển thị 1 vị trí dựa vào kinh độ , vĩ độ

- Chúng ta có thể dùng Geolocation lấy về kinh độ , vĩ độ rồi hiển thị lên google maps như ví dụ sau

HTML :

```
<pid="demo">Click button để hiển thị vị trí trên bản đồ</p>
<buttononclick="getLocation()">Hiển thị vị trí trên bản đồ</button>
<divid="mapholder"></div>
<script>
var x = document.getElementById("demo");
function getLocation() {
if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(showPosition, showError);
}
else { x.innerHTML = "Geolocation is not supported by this browser."; }
}

function showPosition(position) {
var latlon = position.coords.latitude + "," + position.coords.longitude;

var img_url = "http://maps.googleapis.com/maps/api/staticmap?center="
+ latlon + "&zoom=14&size=400x300&sensor=false";
document.getElementById("mapholder").innerHTML = "<img src='" + img_url + "'>";
}

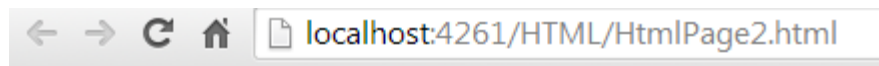
function showError(error) {
switch (error.code) {
case error.PERMISSION_DENIED:
x.innerHTML = "Người dùng không cho phép theo dõi vị trí"
break;
case error.POSITION_UNAVAILABLE:
x.innerHTML = "Không thể tìm vị trí người dùng"
break;
case error.TIMEOUT:
x.innerHTML = "Quá thời gian cho phép"
break;
case error.UNKNOWN_ERROR:
x.innerHTML = "Lỗi không xác định"
break;
}
}
</script>
```

Chạy ví dụ được kết quả như hình sau :

Click button để hiển thị vị trí trên bản đồ

Hiển thị vị trí trên bản đồ

- Click vào button được kết quả như hình sau :



Click button để hiển thị vị trí trên bản đồ



## 6.2. HTML5 Drag và Drop (kéo - thả trong HTML5)

- Kéo thả là 1 tính năng rất thú vị trong HTML5 .

- Chúng ta có thể kéo rồi thả bất kể 1 đối tượng nào : hình ảnh , thẻ div vào 1 khung chứa nó

- Để làm điều này thì đối tượng kéo thả cần khai báo thuộc tính draggable là true

### (1) Kéo thả 1 chiều

- Ví dụ sau khai báo 1 kéo 1 thẻ div vào 1 thẻ div

HTML :

```
<div id="id1" ondrop="drop(event)" ondragover="allowDrop(event)">Thả vào đây</div>
<div id="id2" draggable="true" ondragstart="drag(event)">
<p>Nhấn kéo và thả</p>
</div>
```

- Ta định dạng một chút CSS để dễ hình dung như sau :

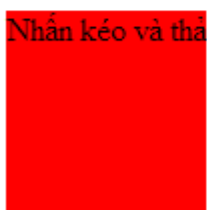
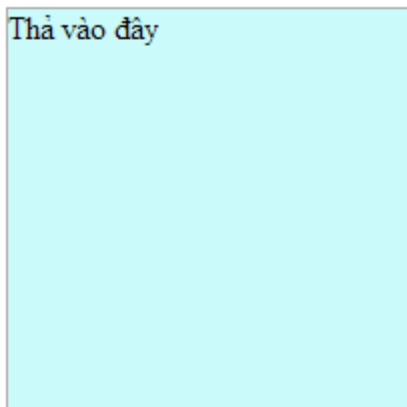
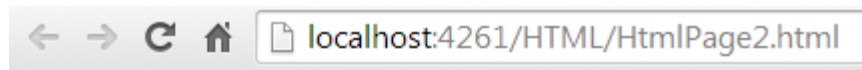
```
#id1 {
width:200px;
height:200px;
background:#cbfafa;
border:1px solid #aaaaaa;}
#id2{
width:100px;
height:100px;
background:#ff0000;
}
```

- Javascript :

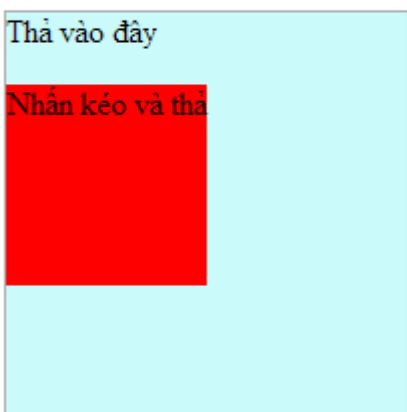
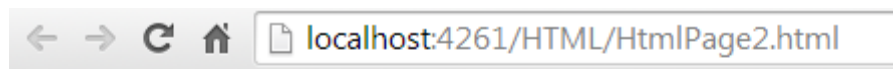
```
<script>
function allowDrop(ev) {
ev.preventDefault();
}
```

```
function drag(ev) {
    ev.dataTransfer.setData("Text", ev.target.id);
}
function drop(ev) {
    ev.preventDefault();
var data = ev.dataTransfer.getData("Text");
    ev.target.appendChild(document.getElementById(data));
}
```

- Chạy ví dụ được kết quả như hình sau :



- Ta chọn và kéo div (màu đỏ) vào vùng màu xanh được kết quả như hình sau:



- Ta có kéo bất kỳ đối tượng nào ( kể cả ảnh )

## (2) Kéo thả 2 chiều

- Phần trước ta kéo và thả đối tượng vào vị trí mới , và không kéo trả lại được vị trí cũ

- Ví dụ này chúng ta sẽ tạo đối tượng có thể kéo thả đối tượng trở về vị trí cũ :

HTML :

```
<div id="id1" ondrop="drop(event)" ondragover="allowDrop(event)">Thả vào đây</div>
<div id="id3" ondrop="drop(event)" ondragover="allowDrop(event)">
```



```
<div id="id2" draggable="true" ondragstart="drag(event)">
<p>Nhấn kéo và thả</p>
</div>
</div>
```

## CSS :

```
#id1 {
width: 200px;
height: 200px;
background: #cbfafa;
border: 1px solid #aaaaaa;
}
```

```
#id2 {
width: 100px;
height: 100px;
background: #ff0000;
}
```

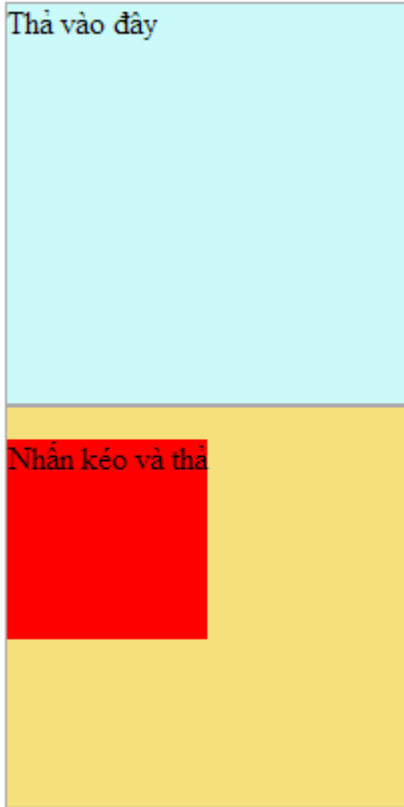
```
#id3 {
width: 200px;
height: 200px;
background: #f5e07c;
border: 1px solid #aaaaaa;
}
```

## Javascript

```
<script>
function allowDrop(ev) {
    ev.preventDefault();
}
function drag(ev) {
    ev.dataTransfer.setData("Text", ev.target.id);
}
function drop(ev) {
    ev.preventDefault();
var data = ev.dataTransfer.getData("Text");
    ev.target.appendChild(document.getElementById(data));
}
</script>
```

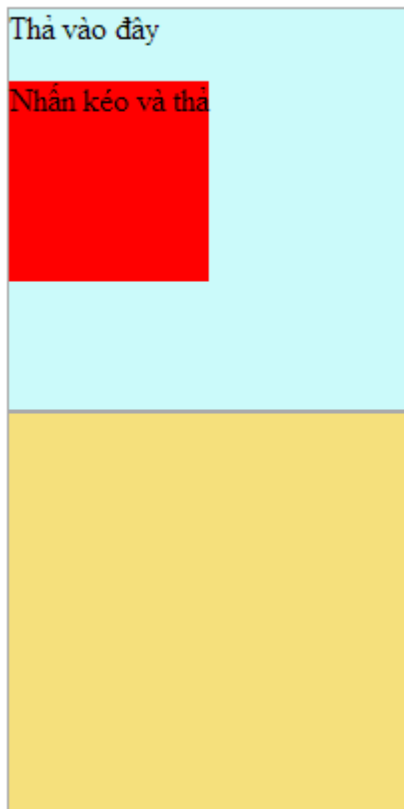
- Chạy ví dụ được kết quả như hình sau :

← → ↻ 🏠 localhost:4261/HTML/HtmlPage2.html

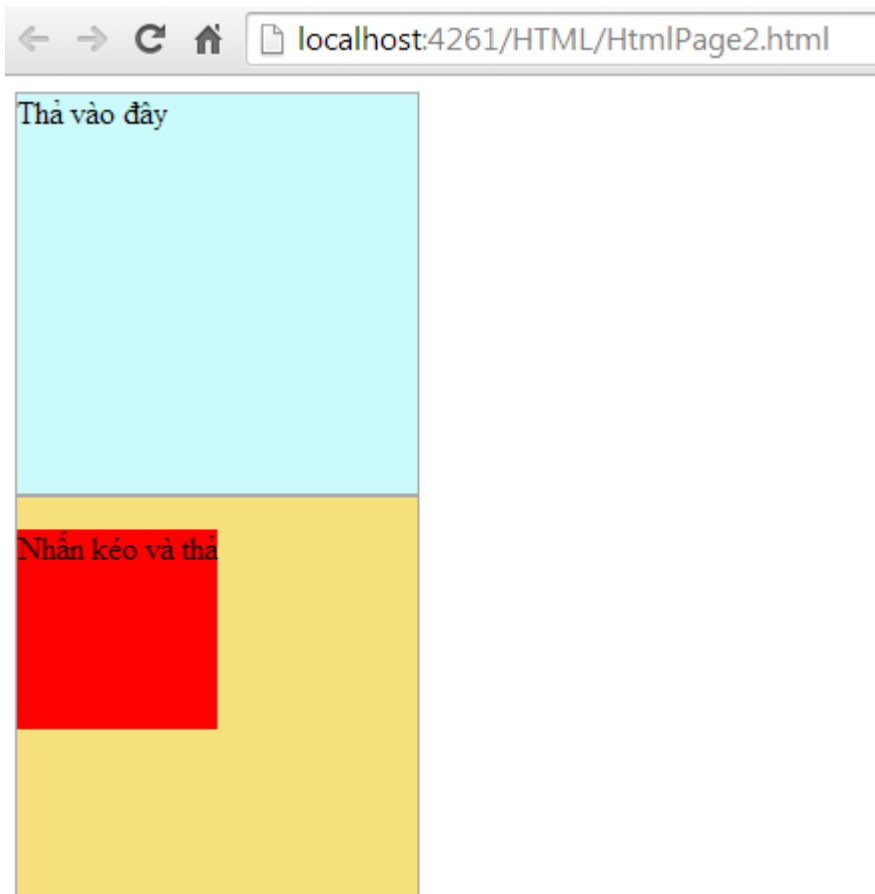


- Ta có thể kéo thả chuyển đổi vị trí ( đối tượng có nền đỏ)
- Kéo sang vị trí mới

← → ↻ 🏠 localhost:4261/HTML/HtmlPage2.html



Kéo trở về vị trí cũ



## 6.3 HTML5 Web Storage(lưu trữ dữ liệu tại Client trong HTML5)

### (1) Đối tượng Web Storage

HTML5 cung cấp một tính năng lưu trữ dữ liệu tại client với dung lượng giới hạn lớn hơn nhiều so với cookie. Tính năng này được gọi là Web Storage và được chia thành hai đối tượng là localStorage và sessionStorage.

- Trước khi sử dụng tính năng này ta có câu lệnh JavaScript để kiểm tra trình duyệt có hỗ trợ Web Storage hay không như sau :

```
if (typeof (Storage) !== "undefined") {  
  // Code for localStorage/sessionStorage.  
}  
else {  
  // Sorry! No Web Storage support..  
}
```

- Đối tượng localStorage và sessionStorage có các thành viên được mô tả trong 1 Interface sau:

```
interface Storage {  
  // số lượng cặp key/value có trong đối tượng  
  readonly attribute unsigned long length;  
  // trả về tên của key thứ n trong danh sách  
  DOMString? key(unsigned long index);  
  // trả về value được gán với key  
  getter DOMString getItem(DOMString key);  
  // thêm hoặc gán một cặp key/value vào danh sách  
  setter creator void setItem(DOMString key, DOMString value);  
  // xóa cặp key/value khỏi danh sách  
  deleter void removeItem(DOMString key);  
  // xóa toàn bộ dữ liệu trong danh sách  
  void clear();  
};
```

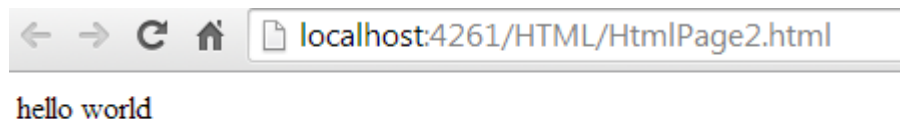
- Từ các thành viên của Interface trên chúng ta dễ dàng hình dung ra các tạo và quản lý các localStorage hay sessionStorage .
- Việc sử dụng localStorage và sessionStorage tương tự nhau , chỉ khác ở thời gian lưu trữ dữ liệu .
- Dữ liệu lưu trên localStorage và sessionStorage khá lớn ( từ 5 - 10 MB) tùy mỗi trình duyệt
- Do đó dùng localStorage hay sessionStorage là 1 giải pháp mới thay cho cookies

## (2) Đối tượng localStorage

- window.localStorage : Lưu trữ dữ liệu không giới hạn thời gian ( không có ngày hết hạn ).localStorage có thể truy xuất lẫn nhau giữa các cửa sổ trình duyệt
- Ví dụ sau sẽ tạo mới 1 localStorage ,sau đó đọc giá trị localStorage và ghi lên màn hình

```
<script>
    localStorage.Hello = "hello world";
    document.write(localStorage.Hello);
</script>
```

Chạy ví dụ được kết quả như hình sau :



- Ngoài ra chúng ta có thể tạo (set) và đọc (get) giá trị localStorage theo khóa Key như sau:
- Ví dụ sau dùng 2 phương thức setItem("key","value") và getItem("key") để đặt và lấy giá trị cho localStorage, kết quả đạt được như ví dụ trước :

```
<script>
    localStorage.setItem("Hello", "Hello world")
    document.write(localStorage.getItem("Hello"));
</script>
```

- Chú ý là cặp name/value luôn có kiểu là string , và bạn cần chuyển đổi sang dạng khác nếu bạn cần !

- Chúng ta có thể xóa 1 localStorage như sau :

```
localStorage.removeItem("Hello");
```

## (3) Đối tượng sessionStorage

- Đối tượng sessionStorage tương tự đối tượng localStorage , chỉ khác biệt một chút đó là dữ liệu lưu trên sessionStorage sẽ bị mất nếu trình duyệt bị đóng ( kết thúc phiên làm việc )
- Ta có 1 ví dụ tương tự phần localStorage là lưu 1 câu chào vào sessionStorage

```
<script>
    sessionStorage.setItem("Hello", "Hello world")
    document.write(sessionStorage.getItem("Hello"));
</script>
```

- Chạy ví dụ ta được kết quả tương tự như phần localStorage .

Một điểm khác biệt đó là nếu trong cùng 1 trang web , ta truy vấn biến localStorage ở nhiều nơi khác nhau và không bị mất đi khi đóng trình duyệt. Tức là người dùng tắt trình duyệt , rồi mở trình duyệt nên thì giá trị localStorage vẫn tồn tại mà không cần khởi tạo .

- Ngược lại với sessionStorage thì khi đóng trình duyệt thì sẽ mất luôn dữ liệu lưu trên nó

#### (4) Lưu trữ mảng đối tượng vào web Storage

- Chúng ta không thể gán 1 giá trị mảng cho đối tượng localStorage hay sessionStorage

( do các đối tượng này có các cặp key , value đều là string )

- Một giải pháp để lưu 1 mảng vào localStorage hay sessionStorage là chúng ta dùng đối tượng JSON convert 1 mảng sang chuỗi rồi lưu vào localStorage hay sessionStorage khi nào sử dụng chỉ việc convert đối tượng string về đối tượng mảng

- Ví dụ dưới đây lưu 1 mảng cho đối tượng localStorage ( với đối tượng sessionStorage làm tương tự )

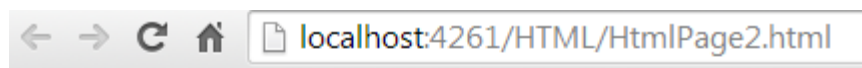
```
<script>
function Product(_id,_name) {
this.ID = _id;
this.Name = _name;
}
var Products = [new Product("1", "Product 1"),
new Product("2", "Product 2"),
new Product("3", "Product 3"),
new Product("4", "Product 4")
];
localStorage["Products"] = JSON.stringify(Products);
var Productslocal = JSON.parse(localStorage["Products"]);
Productslocal.forEach(function (item) {
document.write("<p>"+item.ID + "-" +item.Name+"</p>");
});
</script>
```

- Ví dụ trên ta tạo 1 lớp Product , và mảng danh sách Product

- Lưu mảng danh sách Product vào localStorage

- Đọc giá trị localStorage rồi hiển thị lên màn hình

- Chạy ví dụ được kết quả như hình sau :



1-Product 1

2-Product 2

3-Product 3

4-Product 4

## 6.4 Bộ nhớ Cache cho ứng dụng trong HTML5

- Bộ nhớ cache trong HTML5 dùng để lưu trữ ứng dụng web và có thể truy cập chúng mà không cần kết nối Internet

- Bộ nhớ cache của HTML5 có 3 lợi ích sau :

+) Người dùng có thể sử dụng ứng dụng khi offline ( không cần kết nối Internet)

+) Tốc độ tải tài nguyên sẽ nhanh hơn

+) Giảm tải cho máy chủ , trình duyệt chỉ tải những tài nguyên thay đổi và do đó chỉ yêu cầu máy chủ trả về những thông tin thay đổi nên sẽ giảm rất nhiều công việc cho máy chủ

- HTML5 cung cấp 1 tính năng để tạo ra phiên bản Offline của ứng dụng web bằng cách tạo ra 1 file manifest cho bộ nhớ cache

Ví dụ dưới đây trình bày một trang HTML dùng cache manifest (không có kết nối internet):

```
<!DOCTYPE html>
<html manifest="demo.appcache">
<body>
  The content of the document.....
</body>
</html>
```

Để kích hoạt application cache, chèn thuộc tính manifest vào thẻ <html>:

```
<!DOCTYPE html>
<html manifest="demo.appcache">

</html>
```

Một trang có thuộc tính manifest sẽ được lưu lại trong bộ nhớ cache khi người dùng truy cập đến nó. Nếu như trang web không thiết lập thuộc tính manifest thì nó sẽ không được lưu cache trừ khi trong file manifest có dòng lệnh thiết lập thuộc tính cho nó.

Phần đuôi mở rộng của files manifest là: ".appcache"

Lưu ý: Bạn phải cấu hình trên máy chủ "text/cache-manifest" để chạy được file manifest.

- Chúng ta cần bật tính năng cache trong trang web bằng thuộc tính manifest .Tất cả các trang được khai báo thuộc tính manifest sẽ được lưu trữ lại khi người dùng truy cập nó

### Tệp manifest

- Tệp Manifest là 1 tệp tin văn bản thông báo với trình duyệt những gì được lưu và những gì không được lưu trong bộ nhớ cache của 1 trang web

- Tệp này gồm 3 phần :

+) CACHE MANIFEST : gồm những file nguồn như javascript (.js) , CSS ( .css) hay các file ảnh ( .jpg , .gif , .png ...)

Khi file Manifest được nạp trình duyệt sẽ tải về lưu các file trên tại thư mục gốc của website.Và khi các không có kết nối Internet các tệp tin này sẽ được sử dụng

+) NETWORK : Các tệp tin kết nối tới máy chủ và không bao giờ được lưu trữ như những file:

NETWORK:

login.asp

Một khai báo dấu hoa thị sau để biểu diễn tất cả các tệp tin cần 1 kết nối Internet

NETWORK:

\*

FALLBACK : Xác định các trang dự phòng trong 1 thư mục nào đó .Trong trường hợp không có kết nối mạng các trang dự phòng sẽ được dùng để hiển thị .

### **Cập nhật bộ nhớ Cache**

- Trình duyệt vẫn đọc thông tin từ các tệp tin của bộ nhớ cache cho tới khi 1 trong các vấn đề sau xảy ra :

+ ) Người sử dụng xóa bộ nhớ cache của trình duyệt

+ ) File manifest bị sửa đổi

+ ) Bộ nhớ Cache được cập nhật theo ứng dụng .Tức là ứng dụng sẽ chỉ định cập nhật bộ nhớ cache theo 1 thời gian nhất định

### **Chú ý khi sử dụng bộ nhớ Cache cho ứng dụng của bạn**

- Các thông tin , tệp tin được lưu trữ dưới bộ nhớ cache nên trình duyệt luôn hiển thị phiên bản lưu trữ ngay cả khi trên Server có thay đổi thông tin .Do đó người lập trình cần thêm những đoạn code thông báo mỗi khi thay đổi nội dung , tệp tin trên server để người dùng cập nhật hay tự động cập nhật .Để đảm bảo trình duyệt cập nhật bộ nhớ cache bạn cần thay đổi file manifest

- Các trình duyệt có những giới hạn khác nhau cho bộ nhớ Cache .Một số trình duyệt giới hạn 5 MB bộ nhớ Cache cho 1 website

## **7. HTML5 Web Worker**

- Với mã kịch bản JavaScript khi đang chạy thì trang web sẽ bị khóa lại cho đến khi kịch bản chạy xong thì web được mở và chúng ta có thể thao tác tiếp .

- HTML5 web worker là đoạn mã javascript chạy ở chế độ nền , độc lập với kịch bản khác mà không ảnh hưởng tới tương tác cũng như hiệu suất của trang .Chúng ta có thể làm các việc khác (tương tác ) với website như : click , select ,trong khi mã web worker vẫn chạy ngầm .

- Đối tượng Web Worker được tạo ra sẽ thực thi trong một thread độc lập và chạy ở chế độ nền nên không ảnh hưởng đến giao diện tương tác của trang web với người dùng. Với đặc điểm này, bạn có thể sử dụng Web Workert các công việc đòi hỏi thời gian xử lý lâu nạp dữ liệu, tạo cache,...

- Điểm hạn chế của Web Worker là không thể truy xuất được thành phần trên DOM, và cả các đối tượng window, document hay parent. Mã lệnh các công việc cần thực thi cũng phải được cách ly trong một tệp tin script.

- Chúng ta thường áp dụng tính năng web worker trong các trường hợp cần chạy các đoạn Script làm những công việc nặng nhọc như post dữ liệu lên server và xử lý dữ liệu trên server về , cache

nội dung .Khi này các đoạn ma javascript vẫn âm thầm chạy , và người dùng vẫn có thể tương tác với website một cách thông thường .

- Hay một trường hợp các bạn có thể xử dụng ngay đó là việc kiểm tra đăng nhập hệ thống .Trong trường hợp server quá tải chúng ta thường dùng kỹ thuật TimeOut , nhưng với web worker bạn có thể âm thầm kiểm tra thông tin đăng nhập trong khi đó người dùng vẫn đang lướt website của các bạn , và chúng ta cũng âm thầm xác nhận việc đăng nhập, nếu không thành công thì nên có thông báo cho người dùng

## Ví dụ

- Ví dụ sau tạo 1 bộ đếm số vô hạn .Trong khi đếm số , người dùng vẫn có thể tương tác với website.Ta hiểu vẫn có thể tương tác tức là vẫn có thể click , select ... các nội dung trên website

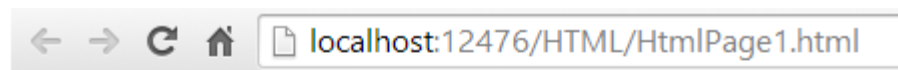
Tạo mới file demo\_workers.js có code như sau :

```
var i = 0;
function timedCount() {
    i = i + 1;
    postMessage(i);
    setTimeout("timedCount()", 500);
}
timedCount();
```

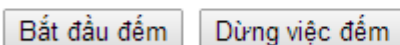
- Mã HTML :

```
<p>Đếm số: <output id="result"></output></p>
<button onclick="startWorker()">Bắt đầu đếm</button>
<button onclick="stopWorker()">Dừng việc đếm</button>
<br><br>
<script>
    var w;
    function startWorker() {
        if (typeof (Worker) !== "undefined") {
            if (typeof (w) == "undefined") {
                // khai báo trở tới file javascript tạo ở bước trước
                w = new Worker("demo_workers.js");
            }
            w.onmessage = function (event) {
                document.getElementById("result").innerHTML = event.data;
            };
        }
        else {
            document.getElementById("result").innerHTML = "Trình duyệt không hỗ trợ web worker";
        }
    }
    function stopWorker() {
        w.terminate();
    }
</script>
```

Chạy ví dụ được kết quả như hình sau :

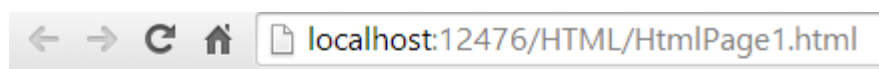


Đếm số:





- Nhấn vào bắt đầu đếm , ma javascript bắt đầu chạy ngầm .Khi này ta vẫn có thể tương tác và làm các việc khác với website .Khi muốn dừng thì nhấn vào button dừng việc đếm



Đếm số: 8

Bắt đầu đếm

Dừng việc đếm

## 8. Tổng kết HTML5 và sự thay thế Flash, Silverlight của HTML5

- Chúng ta đã đi qua tất cả các thành phần của HTML5 .Và các bạn hiểu được rằng những thứ mà HTML5 có thể làm và không thể làm.Và tất nhiên trả lời được câu hỏi là HTML5 có đáng là công nghệ lập trình web có thể thay thế được các công nghệ flash , Silverlight hay không ?

- Và trong một chừng mực nào đó các bạn có thể hiểu được bản chất của công nghệ HTML5 và sẽ định hình được nên áp dụng vào project của mình phần nào , và không áp dụng phần nào

- Các thẻ mới , các tính năng đa phương tiện (video , audio), kéo thả là những tính năng mà có lẽ chúng ta nên dung ngay vào website của mình

- Phần HTML5 canvas , svg là 1 phần khá mạnh của HTML5, điều mà được kỳ vọng xây dựng được các giao diện tương tác trên nền web thay cho công nghệ Flash , Silverlight

- HTML5 vẫn là công nghệ web Client , để áp dụng vào thực tế bạn cần kết hợp với các công nghệ web server như asp.net , php .

- Nhưng dù sao hiểu bản chất công nghệ Client là khá quan trọng .Bạn cần hiểu Client hiểu những thông tin gì , hiển thị , xử lý được những kiểu dữ liệu gì,để từ đó có thể lập trình nên hệ thống tương tác giữa client - server một cách nhịp nhàng , mềm dẻo .

- Các kiến thức về HTML5 tag , HTML5 API , HTML5 Media , HTML5 Web Storage , HTML5 web worker các bạn có thể vận dụng và từ từ áp dụng lượng thích hợp ngay vào website .

- Riêng về HTML5 canvas (svg) là 1 mảng khá rộng lớn trong nền tảng HTML5 .Trong tài liệu chúng tôi trình bày cũng khá dài nhưng thế nói hết được các khía cạnh cũng như kỹ thuật.Chúng tôi đã giới thiệu kỹ thuật xây dựng và xử lý giao diện game cờ tướng trên nền HTML5 với thẻ canvas.Tuy nhiên để xây dựng được game hoàn chỉnh thì đòi hỏi chúng ta phải gia công thêm đoạn code đó .Còn phần đó chính là những kỹ thuật cũng như thuật toán cơ bản nhất để game cờ tướng có thể chạy được .Mời các bạn quan tâm tới HTML5 game trở lại đọc tài liệu về phát triển game trên HTML5 của chúng tôi

- Trong 1 phần xây dựng website thực sự bằng HTML5 các bạn cũng có thể đón đọc các tài liệu về xây dựng ứng dụng web MVC trên nền HTML5 của chúng tôi để các bạn thực sự thấy sự tương tác trong lập trình giữa Client và server có sử dụng các kỹ thuật giao diện của HTML5

## 9. Phụ lục các tag trong HTML5

- Dưới đây là bảng tất cả các thẻ HTML5 , để các bạn tìm hiểu về HTML5 có thể dễ dàng tra cứu .

Tag	Description
-----	-------------

<u>&lt;!--...--&gt;</u>	Defines a comment
<u>&lt;!DOCTYPE&gt;</u>	Defines the document type
<u>&lt;a&gt;</u>	Defines a hyperlink
<u>&lt;abbr&gt;</u>	Defines an abbreviation
<u>&lt;acronym&gt;</u>	<b>Not supported in HTML5. Use &lt;abbr&gt; instead.</b> Defines an acronym
<u>&lt;address&gt;</u>	Defines contact information for the author/owner of a document
<u>&lt;applet&gt;</u>	<b>Not supported in HTML5. Use &lt;object&gt; instead.</b> Defines an embedded applet
<u>&lt;area&gt;</u>	Defines an area inside an image-map
<u>&lt;article&gt;</u>	Defines an article
<u>&lt;aside&gt;</u>	Defines content aside from the page content
<u>&lt;audio&gt;</u>	Defines sound content
<u>&lt;b&gt;</u>	Defines bold text
<u>&lt;base&gt;</u>	Specifies the base URL/target for all relative URLs in a document
<u>&lt;basefont&gt;</u>	<b>Not supported in HTML5. Use CSS instead.</b> Specifies a default color, size, and font for all text in a document
<u>&lt;bdi&gt;</u>	Isolates a part of text that might be formatted in a different direction from other text outside it
<u>&lt;bdo&gt;</u>	Overrides the current text direction
<u>&lt;big&gt;</u>	<b>Not supported in HTML5. Use CSS instead.</b> Defines big text
<u>&lt;blockquote&gt;</u>	Defines a section that is quoted from another source
<u>&lt;body&gt;</u>	Defines the document's body
<u>&lt;br&gt;</u>	Defines a single line break
<u>&lt;button&gt;</u>	Defines a clickable button
<u>&lt;canvas&gt;</u>	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<u>&lt;caption&gt;</u>	Defines a table caption
<u>&lt;center&gt;</u>	<b>Not supported in HTML5. Use CSS instead.</b> Defines centered text
<u>&lt;cite&gt;</u>	Defines the title of a work
<u>&lt;code&gt;</u>	Defines a piece of computer code
<u>&lt;col&gt;</u>	Specifies column properties for each column within a <colgroup> element
<u>&lt;colgroup&gt;</u>	Specifies a group of one or more columns in a table for formatting
<u>&lt;datalist&gt;</u>	Specifies a list of pre-defined options for input controls
<u>&lt;dd&gt;</u>	Defines a description/value of a term in a description list
<u>&lt;del&gt;</u>	Defines text that has been deleted from a document

<u>&lt;details&gt;</u>	Defines additional details that the user can view or hide
<u>&lt;dfn&gt;</u>	Defines a definition term
<u>&lt;dialog&gt;</u>	Defines a dialog box or window
<u>&lt;dir&gt;</u>	<b>Not supported in HTML5. Use &lt;ul&gt; instead.</b> Defines a directory list
<u>&lt;div&gt;</u>	Defines a section in a document
<u>&lt;dl&gt;</u>	Defines a description list
<u>&lt;dt&gt;</u>	Defines a term/name in a description list
<u>&lt;em&gt;</u>	Defines emphasized text
<u>&lt;embed&gt;</u>	Defines a container for an external (non-HTML) application
<u>&lt;fieldset&gt;</u>	Groups related elements in a form
<u>&lt;figcaption&gt;</u>	Defines a caption for a <figure> element
<u>&lt;figure&gt;</u>	Specifies self-contained content
<u>&lt;font&gt;</u>	<b>Not supported in HTML5. Use CSS instead.</b> Defines font, color, and size for text
<u>&lt;footer&gt;</u>	Defines a footer for a document or section
<u>&lt;form&gt;</u>	Defines an HTML form for user input
<u>&lt;frame&gt;</u>	<b>Not supported in HTML5.</b> Defines a window (a frame) in a frameset
<u>&lt;frameset&gt;</u>	<b>Not supported in HTML5.</b> Defines a set of frames
<u>&lt;h1&gt; to &lt;h6&gt;</u>	Defines HTML headings
<u>&lt;head&gt;</u>	Defines information about the document
<u>&lt;header&gt;</u>	Defines a header for a document or section
<u>&lt;hr&gt;</u>	Defines a thematic change in the content
<u>&lt;html&gt;</u>	Defines the root of an HTML document
<u>&lt;i&gt;</u>	Defines a part of text in an alternate voice or mood
<u>&lt;iframe&gt;</u>	Defines an inline frame
<u>&lt;img&gt;</u>	Defines an image
<u>&lt;input&gt;</u>	Defines an input control
<u>&lt;ins&gt;</u>	Defines a text that has been inserted into a document
<u>&lt;kbd&gt;</u>	Defines keyboard input
<u>&lt;keygen&gt;</u>	Defines a key-pair generator field (for forms)
<u>&lt;label&gt;</u>	Defines a label for an <input> element
<u>&lt;legend&gt;</u>	Defines a caption for a <fieldset> element
<u>&lt;li&gt;</u>	Defines a list item
<u>&lt;link&gt;</u>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<u>&lt;main&gt;</u>	Specifies the main content of a document
<u>&lt;map&gt;</u>	Defines a client-side image-map
<u>&lt;mark&gt;</u>	Defines marked/highlighted text
<u>&lt;menu&gt;</u>	Defines a list/menu of commands

<u>&lt;menuitem&gt;</u>	Defines a command/menu item that the user can invoke from a popup menu
<u>&lt;meta&gt;</u>	Defines metadata about an HTML document
<u>&lt;meter&gt;</u>	Defines a scalar measurement within a known range (a gauge)
<u>&lt;nav&gt;</u>	Defines navigation links
<u>&lt;noframes&gt;</u>	<b>Not supported in HTML5.</b> Defines an alternate content for users that do not support frames
<u>&lt;noscript&gt;</u>	Defines an alternate content for users that do not support client-side scripts
<u>&lt;object&gt;</u>	Defines an embedded object
<u>&lt;ol&gt;</u>	Defines an ordered list
<u>&lt;optgroup&gt;</u>	Defines a group of related options in a drop-down list
<u>&lt;option&gt;</u>	Defines an option in a drop-down list
<u>&lt;output&gt;</u>	Defines the result of a calculation
<u>&lt;p&gt;</u>	Defines a paragraph
<u>&lt;param&gt;</u>	Defines a parameter for an object
<u>&lt;pre&gt;</u>	Defines preformatted text
<u>&lt;progress&gt;</u>	Represents the progress of a task
<u>&lt;q&gt;</u>	Defines a short quotation
<u>&lt;rp&gt;</u>	Defines what to show in browsers that do not support ruby annotations
<u>&lt;rt&gt;</u>	Defines an explanation/pronunciation of characters (for East Asian typography)
<u>&lt;ruby&gt;</u>	Defines a ruby annotation (for East Asian typography)
<u>&lt;s&gt;</u>	Defines text that is no longer correct
<u>&lt;samp&gt;</u>	Defines sample output from a computer program
<u>&lt;script&gt;</u>	Defines a client-side script
<u>&lt;section&gt;</u>	Defines a section in a document
<u>&lt;select&gt;</u>	Defines a drop-down list
<u>&lt;small&gt;</u>	Defines smaller text
<u>&lt;source&gt;</u>	Defines multiple media resources for media elements (<video> and <audio>)
<u>&lt;span&gt;</u>	Defines a section in a document
<u>&lt;strike&gt;</u>	<b>Not supported in HTML5. Use &lt;del&gt; instead.</b> Defines strikethrough text
<u>&lt;strong&gt;</u>	Defines important text
<u>&lt;style&gt;</u>	Defines style information for a document
<u>&lt;sub&gt;</u>	Defines subscripted text
<u>&lt;summary&gt;</u>	Defines a visible heading for a <details> element
<u>&lt;sup&gt;</u>	Defines superscripted text
<u>&lt;table&gt;</u>	Defines a table

<u>&lt;tbody&gt;</u>	Groups the body content in a table
<u>&lt;td&gt;</u>	Defines a cell in a table
<u>&lt;textarea&gt;</u>	Defines a multiline input control (text area)
<u>&lt;tfoot&gt;</u>	Groups the footer content in a table
<u>&lt;th&gt;</u>	Defines a header cell in a table
<u>&lt;thead&gt;</u>	Groups the header content in a table
<u>&lt;time&gt;</u>	Defines a date/time
<u>&lt;title&gt;</u>	Defines a title for the document
<u>&lt;tr&gt;</u>	Defines a row in a table
<u>&lt;track&gt;</u>	Defines text tracks for media elements (<video> and <audio>)
<u>&lt;tt&gt;</u>	<b>Not supported in HTML5. Use CSS instead.</b> Defines teletype text
<u>&lt;u&gt;</u>	Defines text that should be stylistically different from normal text
<u>&lt;ul&gt;</u>	Defines an unordered list
<u>&lt;var&gt;</u>	Defines a variable
<u>&lt;video&gt;</u>	Defines a video or movie
<u>&lt;wbr&gt;</u>	Defines a possible line-break