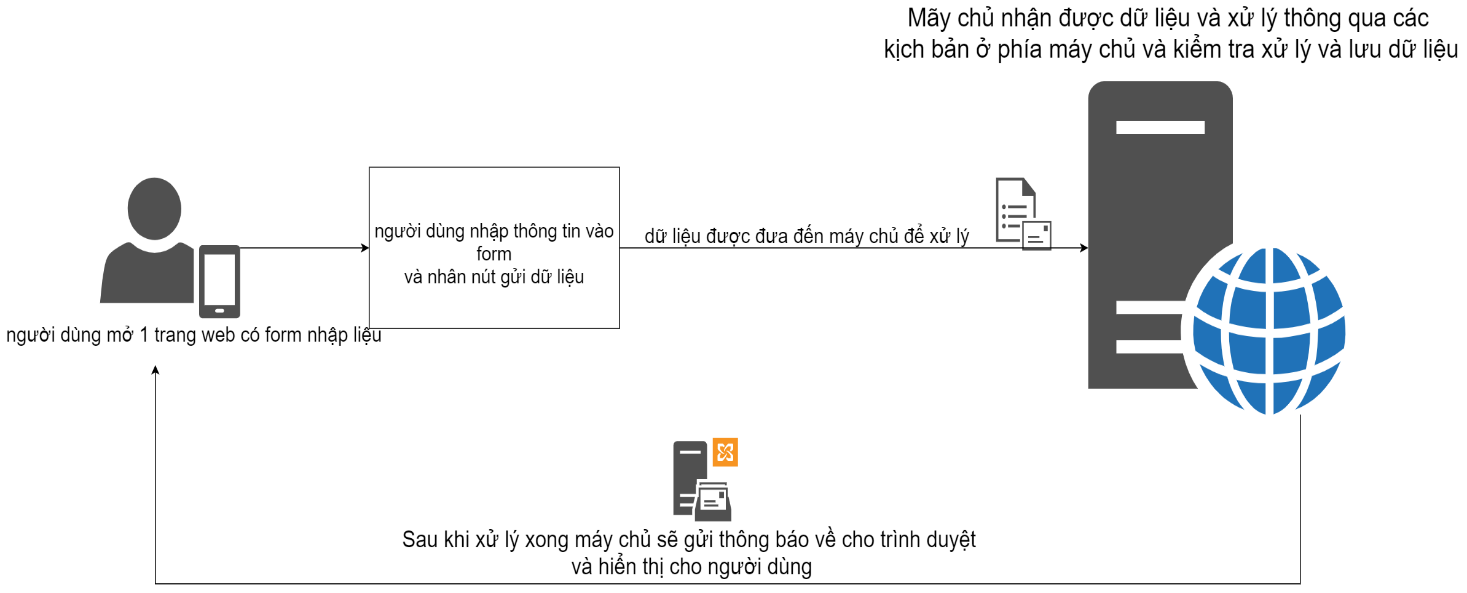
**Tìm Hiểu Về HTML (Bài 4)**

**1. Form là gì? Trong ứng dụng web thì form hoạt động như thế nào? (trình bày hiểu biết, vẽ sơ đồ các hoạt động, …)**

* **Form** là khu vục hình thành nên sự tương tác giữa người dùng và ứng dụng web.
* **Form** là một phần của trang web có chứa các điều khiển như: các trường văn bản, các trường mật khẩu, các trường checkbox, các radio button, submit button, các menu
* **Form** là phương tiện để người dùng nhập dữ liệu và gửi đến máy chủ để sử lý.
* **Form** được sử dụng khi bạn muốn nhập một dữ liệu của người dùng truy cập trang web.

**Thẻ <form> cơ bản có hai thuộc tính cần lưu tâm là action và method:**

* action thuộc tính để thiết lập URL, là địa chỉ mà dữ liệu của form gửi đến (submit đến, post đến). Thiếu tham số này thì action bằng URL đang truy cập.
* Form này có 2 method chính: **get**,**post**
* Sử dụng **method="get"** thì khi submit dữ liệu được biểu diễn thông qua URL.
* Sử dụng **method="post"** thì khi submit thì dữ liệu được biểu diễn trong nội dung của Request gửi đến Server và là ẩn với người dùng, sử dụng post an toàn hơn.
* **Form hoạt động như thế nào:**
  + Người dùng mở trang web có form nhập liệu trong trình duyệt
  + Người dùng điền thông tin vào các phần form nhập liệu và nhấn nút (submit) để gửi dữ liệu lên máy chủ. Trong giai đoạn này dữ liệu được nhập vào đã có thể được kiểm tra tại máy tính người dùng thông qua các đoạn mã Javascript.
  + Tại máy chủ, thông tin của người dùng gửi lên được sử lý thông qua các ngôn ngữ kịch bản máy chủ như: PHP, ASP.NET, JSP, .... Các đoạn mã kịch bản máy chủ sẽ thực hiện kiểm tra dữ liệu trong form, thực hiện các logic, ghi nhận dữ liệu vào database...
  + Sau đó máy chủ web sau khi sử lý xong thông tin dữ liệu sẽ gửi lại một thông báo thành công hoặc thông báo lỗi mời nhập lại cho người dùng và kết thúc chu trình sử lý form nhập liệu
* **Sơ đồ hoạt động người dùng sơ tác với form gửi dũ liêu lên máy chủ.**



**2.Tìm hiểu về các form control input?**

* 1. **Input kiểu văn bản**

<input type=”text”> xác định trường nhập văn bản một dòng:

Ví dụ:

<form>

  First name:<br>

  <input type="text" name="firstname"><br>

  Last name:<br>

  <input type="text" name="lastname">

</form>

* 1. **Input kiểu password**

<input type="password"> xác định một trường nhập mật khẩu:

Ví dụ:

<form>

  User name:<br>

  <input type="text" name="username"><br>

  User password:<br>

  <input type="password" name="psw">

</form>

* 1. **Input kiểu submit**

<input type="submit"> xác định một nút để gửi đi dữ liệu form tới bộ xử lý form.

* + Bộ xử lý này thường là một trang server với những đoạn script chuyên dụng để xử lý dữ liệu đầu vào.
  + Bộ xử lý form được chỉ ra trong thuộc tính action của form.
  + Nếu bỏ qua thuộc tính value của nút submit thì nó sẽ nhận một giá trị mặc định, tùy trình duyệt.

Ví dụ:

<form action="action.php">

    First name:<br>

    <input type="text" name="firstname" value="Mickey"><br>

    Last name:<br>

    <input type="text" name="lastname" value="Mouse"><br><br>

    <input type="submit" value="Submit">

</form>

* 1. **Input kiểu reset**

<input type="reset"> xác định một nút để reset (đặt lại) mọi giá trị của form về giá trị mặc định.

Ví dụ:

<form action="action.php">

    First name:<br>

    <input type="text" name="firstname" value="Mickey"><br>

    Last name:<br>

    <input type="text" name="lastname" value="Mouse"><br><br>

    <input type="submit" value="Submit">

    <input type="reset">

</form>

* 1. **Input kiểu radio**

<input type="radio"> xác định một nút radio (nút tròn).

* Nút radio cho phép người dùng lựa chọn một giá trị duy nhất trong danh sách cách lựa chọn giới hạn.

Ví dụ:

<form>

    <input type="radio" name="gender" value="male" checked> Male<br>

    <input type="radio" name="gender" value="female"> Female<br>

    <input type="radio" name="gender" value="other"> Other

</form>

* 1. **Input kiểu checkbox**

<input type="checkbox"> xác định một checkbox (hộp kiểm).

* Checkbox cho phép người dùng lựa chọn tùy ý trong danh sách lựa chọn (không chọn, hoặc chọn nhiều đều được)

Ví dụ:

<form>

    <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>

    <input type="checkbox" name="vehicle2" value="Car"> I have a car

</form>

* 1. **Input kiểu button**

<input type="button"> xác định một button (nút có thể bấm)

Ví dụ:

<input type="button" onclick="alert('Hello World!')" value="Click Me!">

* 1. **Input kiểu email**

<input type="email"> được sử dụng cho trường nhập email.

* Tùy trình duyệt hỗ trợ, địa chỉ email có thể được tự động xác thực khi gửi đi.
* Một số smartphone nhận dạng kiểu email, và thêm ".com" vào gợi ý để khớp với input email.

Ví dụ:

<form>

    E-mail:

    <input type="email" name="email">

</form>

* 1. **Input kiểu file**

<input type="file"> xác định một trường để chọn tệp từ máy tính và một nút "Browse" để duyệt và tải tệp lên.

Ví dụ:

<form>

    Select a file: <input type="file" name="myFile">

</form>

* 1. **Input kiểu month**

<input type="month"> cho phép người dùng chọn tháng và năm.

Ví dụ:

<form>

    Birthday (month and year):

    <input type="month" name="bdaymonth">

</form>

* 1. **Input kiểu number**

<input type="number"> xác định một trường nhập số.

* Bạn cũng có thể hạn chế khoảng số được cho phép bằng thuộc tính min và max.

Ví dụ:

<form>

    Số lượng (Từ 1 tới 10):

    <input type="number" name="quantity" min="1" max="10">

</form>

* 1. **Input kiểu time**

<input type="time"> cho phép người dùng chọn một thời gian (không có múi giờ).

Ví dụ:

<form>

    Select a time:

    <input type="time" name="usr\_time">

</form>

* 1. **Input kiểu url**

<input type="url"> được dùng cho trường nhập chứa địa chỉ URL.

Ví dụ:

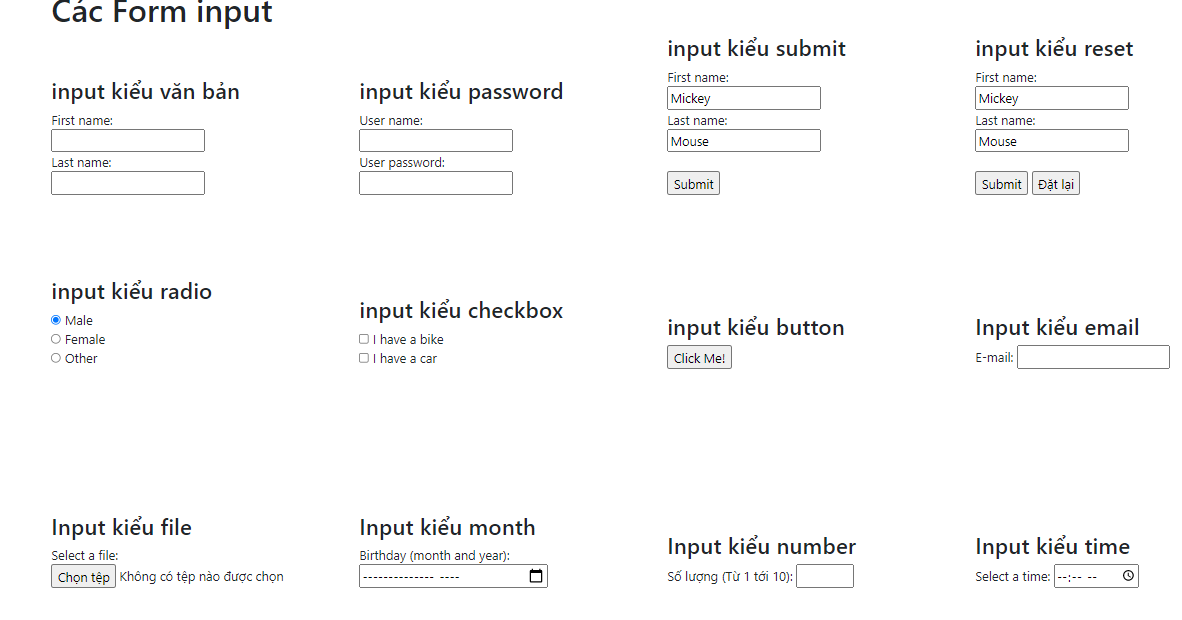
<form>

    Add your homepage:

    <input type="url" name="homepage">

</form>

**2.1 Tạo 1 trang giao diện HTML có chứa tất cả các form input**

****

**2.2. Đăng ký tài khoản tài codepen.io**

[**https://codepen.io/HungTK/project/editor/ABYjzV#**](https://codepen.io/HungTK/project/editor/ABYjzV)

**3. HTTP là gì? Có bao nhiêu phương thức giao tiếp trong HTTP (Request Method)? Trình bày chi tiết sự hiểu biết về từng phương thức**

* Http (HyperText Transfer Protocol) là giao thức truyền tải siêu văn bản được sử dụng trong www dùng để truyền tải dữ liệu giữa Web server đến các trình duyệt Web và ngược lại. Giao thức này sử dụng cổng 80 (port 80) là chủ yếu.
* Khi dùng trình duyệt truy cập Web chúng ta thường gặp nhưng thông báo lỗi khác nhau như:
  1. Lỗi 404 hay Http 404 là lỗi không tồn tại địa chỉ đang truy cập
  2. Lỗi 401 lỗi này ta truy cập vào nơi yêu cầu xác thực, nhưng không vượt qua được sẽ có lỗi này.
  3. Lỗi 500 là lỗi thường do Web sever mà ta truy cập bị lỗi nên không thể truy cập vào được.
  4. Ngoài ra Http 200 tức là truy cập thành công.
* **Các phương thức giao tiếp trong Http? Trình bày chi tiết các giao thức.**

**Có 2 phương thức chính: GET và POST:**

* + **Phương thức GET:**
  + Phương thức GET gửi dữ liệu từ client lên server thông qua các paramter (tham số) trên URL, từ đó server sẽ phân tích dữ liệu để thực thi một hành động nào đó. Các tham số mà phương thức GET gửi lên sẽ bắt đầu sau dấu (?). Và sẽ nối nhau giữa các khóa thông qua dấu &
  + Băng thông của nó chỉ khoảng 1024 kí tự vì vây GET hạn chế về số kí tự được gửi đi. GET không thể gửi dữ liệu nhị phân, hình ảnh ... Có thể cached và được bookmark (đánh dấu trên trình duyệt). Lưu trong browser history.
  + Truy vấn GET có thể được lưu lại (cached)
  + Truy vấn GET vẫn được lưu lại trong lịch sử trình duyệt
  + Truy vấn GET có thể được bookmark (đánh dấu rồi xem lại sau)
  + Truy vấn GET không bao giờ được sử dụng để gửi đi các dữ liệu nhạy cảm
  + Truy vấn GET có những hạn chế về chiều dài dữ liệu
  + Truy vấn GET chỉ nên sử dụng cho việc lấy dữ liệu GET
  + phương thức này có độ bảo mật kém
    - **SERVER nhận dữ liệu**
      * Để thông báo cho server nhận dữ liệu GET thì khi đó chúng ta chỉ cần sử dụng **$\_GET** đây là một biến toàn cục lưu trữ dưới dạng mảng bất tuần tự.
  + **Phương thức POST:**
* Khác với phương thức GET phương thức POST không gửi dữ liệu thồng qua paramter trên thanh URL nên chúng ta không thấy được dữ liệu và phương thức này có độ bảo mật cao.
* Truy vấn POST không bao giờ được lưu trữ (cached)
* Truy vấn POST không được lưu lại trong lịch sử tình duyệt
* Truy vấn POST không thể bookmark
* Truy vấn POST không hạn chế chiều dài dữ liệu
* **SERVER nhận dữ liệu**
  + - * + Để thông báo cho sever nhận được dữ liệu POST thì khi đó chúng ta chỉ cần sử dụng **$\_POST** đây là một biến toàn cục lưu trữ dưới dạng mảng bất tuần tự.

## **Các phương thức giao tiếp khác trong HTTP**

* **HEAD**: Tương tự như GET nhưng chỉ trả về HTTP header và không có thân tài liệu
* **PUT**: Tải lên một đại diện xác định của URI
* **DELETE**: Xoá tài nguyên xác định
* **OPTIONS**: Quay trở lại phương thức HTTP mà server hỗ trợ
* **CONNECT**: Chuyển đổi các yêu cầu kết nối đến một tunnel TCP/IP
* **PATCH**: PATCH khác PUT nhưng lại giống POST, nó không phải là idempotent
* **COPY**: Phương thức này không còn được phát triển. Vì vậy nó không đảm bảo yêu cầu về an toàn trong kết nối
* **LINK**: Nó dùng để kết nối các tài nguyên lại với nhau bởi các URI
* **UNLINK**: Đơn giản, nó chỉ là phương thức ngược lại với phương thức LINK
* **PURGE**: Nó không phải là phương thức chính thức, có một số máy chủ có phương **PURGE** như: Squid, Varnish
* **LOCK**: Nó cũng giống như COPY đã bị khai tử
* **UNLOCK**: Nó cũng đã bị khai tử
* **PROPFIND**: Nó cũng đã không còn được hỗ trợ như các phương thức LOCK, UNLOCK, COPY
* **TRACE**: Nó là một vòng lặp dữ liệu đến máy chủ

**4. Trình bày sự khác nhau giữa phương thức GET và POST? Khi nào thì nên sử dụng GET? Khi nào thì nên sử dụng POST?**

- Điểm chung: là các HTTP method dùng để trao đổi dữ liệu giữa client và server.

* POST: Bảo mật hơn GET vì dữ liệu được gửi ngầm, không xuất hiện trên URL
* GET: Dữ liệu được gửi tường minh, chúng ta có thể nhìn thấy trên URL, đây là lý do khiến nó không bảo mật so với POST.
* GET thực thi nhanh hơn POST vì những dữ liệu gửi đi luôn được trình duyệt web được lưu vào bộ nhớ đệm lại.
* Khi dùng phương thức POST thì server luôn thực thi và trả về kết quả cho client, còn phương thức GET ứng với cùng một yêu cầu đó trình duyệt web sẽ xem trong cached có kết quả tương ứng với yêu cầu đó không và trả về ngay không cần phải thực thi các yêu cầu đó ở phía server.
* Đối với những dữ liệu luôn được thay đổi thì chúng ta nên sử dụng phương thức POST, còn dữ liệu ít thay đổi chúng ta dùng phương thức GET để truy xuất và xử lý nhanh hơn

**5. Validation là gì? Tại sao chúng ta cần phải sử dụng đến validation trong ứng dụng web?**

* Validation là kiểm tra ràng buộc dữ liệu đó có hợp lệ với format yêu cầu của người lập trình hay không, nếu đúng thì dữ liệu submit thành công, nếu sai thì sẽ hiển thị lỗi cho người dùng.
* Validation thực sự rất cần thiết nó giúp hạn chế những request sai lên server nhưng không thể chống lại những người cố tình phá hoại.

**6. Tìm hiểu tất cả các rule validaion có trong HTML5**

* **type:**

Ngoài giá trị text, chúng ta sẽ có thêm

* **email**: chỉ cho phép nhập địa chỉ email
* **number**: chỉ cho phép nhập số
* **url**: chỉ cho phép nhập dạng đường dẫn url
* **tel**: không nên xài, vì mỗi nước của một kiểu format số điện thoại riêng
* **required:** Một attribute đơn giản nhất, truyền vào một giá trị **boolean**, bắt buộc user phải nhập giá trị nếu đang set **true**

**Ví dụ:**  <input id="choose" name="i\_like" required>

* **minlength, maxlength, min, max:**
  + Với <input type="number"/> chúng ta dùng min và max để đặt ràng buộc khoảng giá trị, các <input/>, <textarea/> còn lại dùng minlength và maxlength
* **pattern:** Truyền vào một regular expression, chỉ có trên <input/>

Ví dụ:

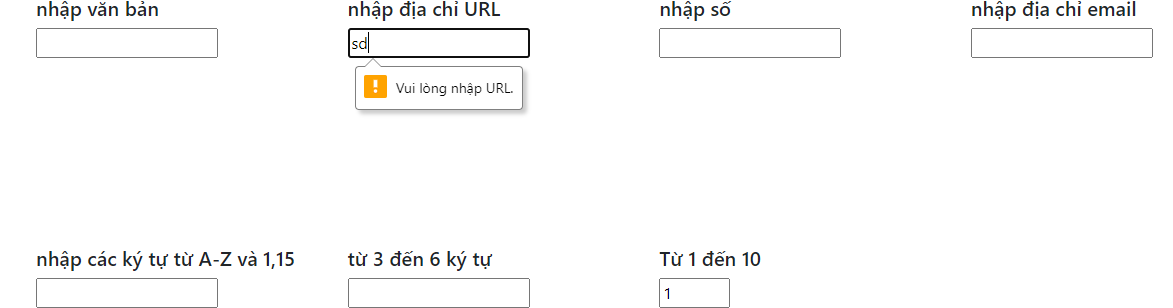
<input

type="text"

pattern="(\+84|0)\d{9,10}"

/>

**6.2 Tạo 1 trang giao diện HTML có sử dụng tất cả các rule trong HTML5**

****

**7. Tìm hiểu sự khác nhau giữa [Block Elements] và [Inline Elements]?**

|  |  |
| --- | --- |
| - Các phần tử khối (**block element**) là các phần tử sẽ kéo dài toàn bộ chiều rộng của phần tử mẹ và sẽ không cho phép bất kỳ phần tử nào khác chiếm cùng một không gian ngang khi nó được đặt trên đó.  - Các phần tử khối bắt đầu bằng một dòng mới.  - Các phần tử khối có tất cả chiều rộng.  - Không cần thiết | - Các phần tử nội tuyến (**inline element**) là các phần tử không thể đặt trực tiếp bên trong phần tử body và được lồng trong các phần tử khối.  - Các phần tử nội tuyến không bắt đầu bằng một dòng mới  Các phần tử nội tuyến sẽ chỉ chiếm chiều rộng cần thiết. |

**7.2 Liệt kế các tag thuộc Block Elements và Inline Elements mà bạn biết**

* **Các tag thuộc Block Elements:**

<address>, <article>, <aside>, <blockquote>, <canvas>, <dd>, <div>, <dl>, <dt>, <fieldset>, <figcaption>, <figure>, <footer>, <form>, <h1>-<h6>, <header>, <hr>, <li>, <main>, <nav>, <noscript>, <ol>, <output>, <p>, <pre>, <section>, <table>, <tfoot>, <ul>, <video>

* **Các tag thuộc Inline Elements:**

<a><abbr> <acronym><b> <bdo><big><br> <button> <cite> <code><dfn><em><i><img><input><kbd><label><map><object><q><samp><script><select><small><span><strong><sub><sup><textarea><time><tt><var>

**8. Làm sao chèn được 1 video / audio vào trong trang web?**

- Để chèn 1 video/audio:

Ta dùng thẻ tag <video> </video>, tag <audio> </audio>

* Thẻ **<video>** là câu lệnh được sử dụng để chèn “trình xem phim” cho một website nào đó.
* Thẻ **<audio>** được dùng để tạo một "trình phát nhạc" cho trang web.

**Các thuộc tính của thẻ video/audio:** gồm có 9 thuộc tính

* + - **src**: xác định đường dẫn đến tập tin video mà bạn muốn phát
    - **controls**: xác định việc "thanh điều khiển của trình xem phim sẽ được hiển thị"
    - autoplay: Thiết lập hành động:  
      "sau khi trang web được tải xong, trình xem phim sẽ tự động phát video"
    - **loop**: Thiết lập hành động "video sẽ tự động được phát lặp lại sau mỗi lần kết thúc"
    - **muted**: Xác định việc trình xem phim sẽ mặc định được thiết lập ở chế độ "tắt tiếng"
    - **preload**: Xác định việc tập tin video có được tải cùng với lúc tải trang hay không
    - **poster**: Xác định một tập tin hình ảnh dùng để làm ảnh đại diện cho video trước khi phát
    - **width**: Xác định chiều rộng của trình xem phim
    - **height**: Xác định chiều cao của trình xem phim

Ví dụ:

* + thẻ **video:**

<video controls="controls">

    <source src="SampleVideo\_1280x720\_1mb.mp4" ">

</video>

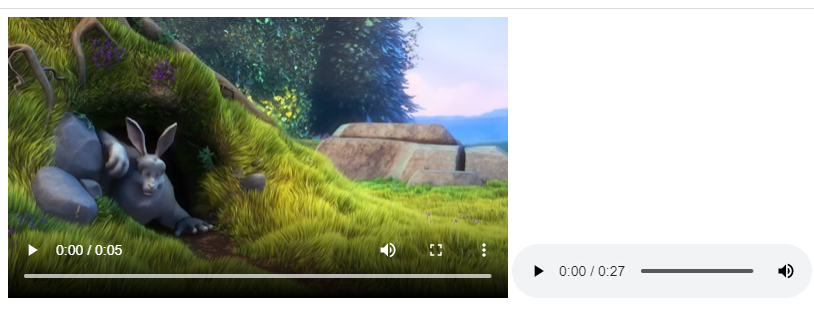
- thẻ **audio:**

<audio controls="controls">

     <source src="audio.mp3">

</audio>

**8.2 Tạo một trang HTML có chèn video và audio**

****