VIETNAM AUSTRALIA Vocational College

Slides 3.2: Xử lý sự kiện, kiểm tra dữ liệu hợp lệ với JS

Mentor: Nguyễn Bá Minh Đạo

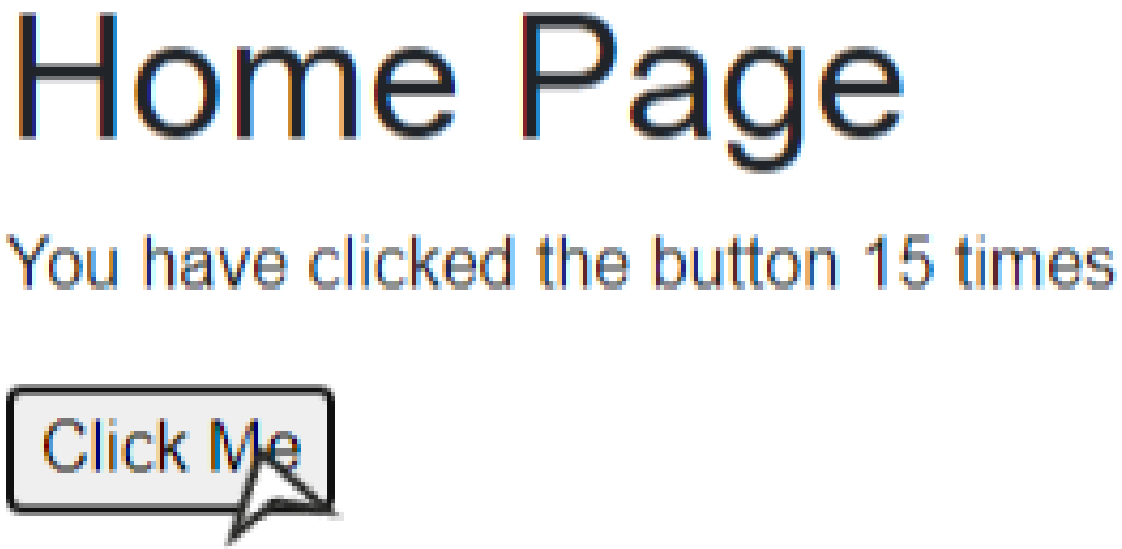
Nội dung

1. Tương tác sự kiện trong HTML với JavaScript
2. Thuộc tính sự kiện trong HTML với JavaScript
3. Gán các sự kiện trong HTML với JavaScript
4. Các sự kiện trong HTML với JavaScript
5. Lắng nghe sự kiện trong HTML với JavaScript
6. Thêm nhiều trình xử lý sự kiện vào cùng một HTML
7. Thêm một trình xử lý sự kiện vào đối tượng Window
8. Các cách truyền sự kiện trong HTML với JavaScript
9. Xóa sự kiện trong HTML với JavaScript
10. Kiểm tra dữ liệu hợp lệ với JavaScript

Tương tác sự kiện trong HTML với JavaScript

❑ **Tương tác với các sự kiện:**

- **HTML DOM** cho phép **JavaScript** tương tác với các sự kiện của **HTML**.
- Mã **JavaScript** có thể được thực thi khi xảy ra sự kiện trên **HTML**.
  - **Ví dụ**: Khi người dùng click vào một phần tử HTML.
- Để thực thi đoạn mã khi một người dùng click trên một phần tử HTML, thêm mã JavaScript vào thuộc tính sự kiện của HTML.
  - **Cú pháp**: onclick=JavaScript



Tương tác sự kiện trong HTML với JavaScript

- **Tương tác với các sự kiện:**
  - Một số sự kiện của HTML:
    - Khi người dùng click chuột
    - Khi một trang web đã tải
    - Khi một hình ảnh đã được tải
    - Khi con trỏ chuột di chuyển lướt qua (hover) một phần tử
    - Khi một trường nhập dữ liệu thay đổi
    - Khi một biểu mẫu được submit (form submitted)
    - Khi một người dùng đánh một phím

Tương tác sự kiện trong HTML với JavaScript

❑ Tương tác với các sự kiện:

- **Ví dụ**: Thay đổi nội dung của phần tử **<h1>** khi người dùng click vào phần tử đó.

js_events_onclick_01.html | js_events_onclick_02.html

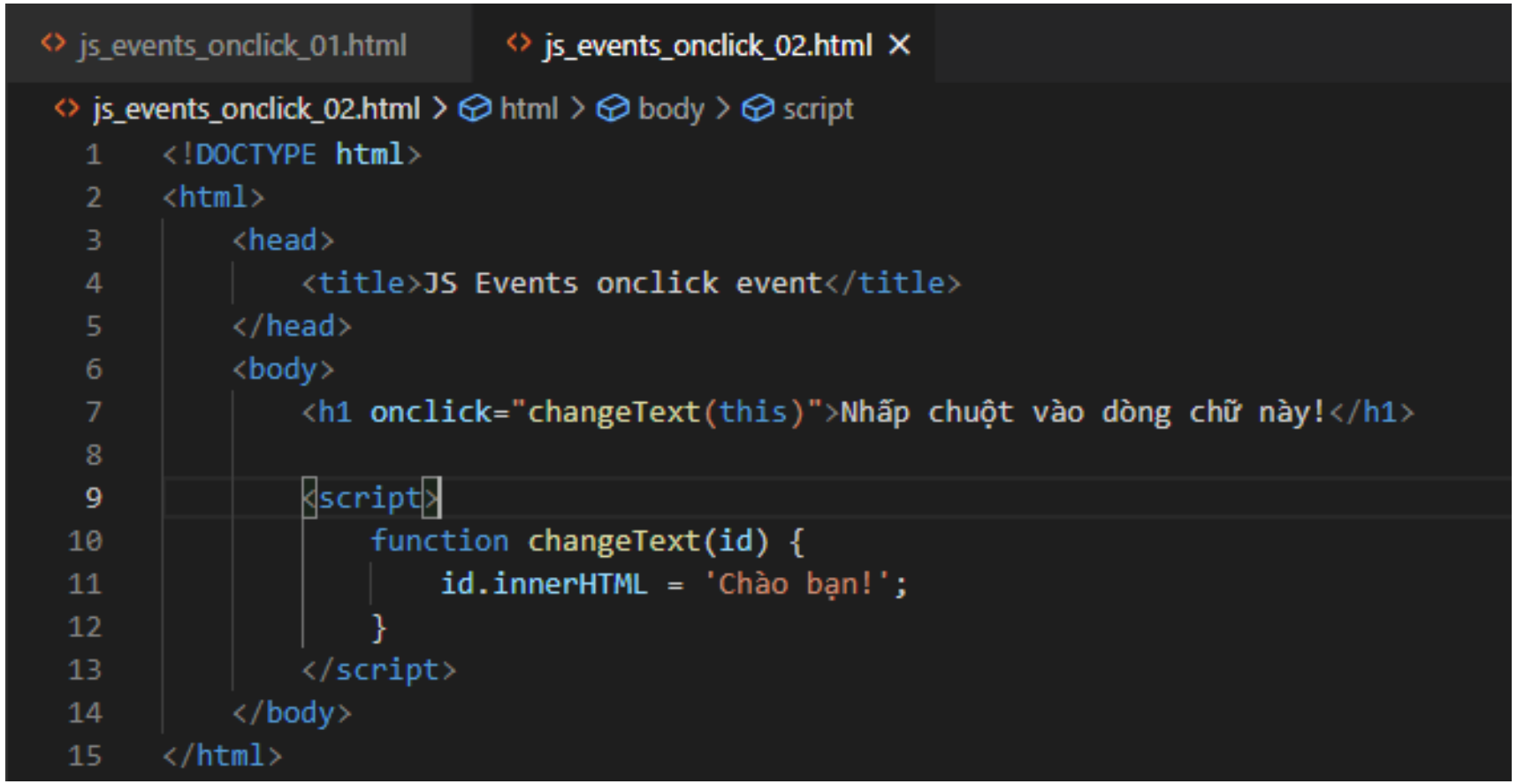
js_events_onclick_01.html > html

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <title>JS Events onclick event</title>
5       </head>
6       <body>
7           <h1 onclick="this.innerHTML='Chào bạn!'">Nhấp chuột vào dòng chữ này!</h1>
8       </body>
9   </html>
```

Tương tác sự kiện trong HTML với JavaScript

❑ Tương tác với các sự kiện:

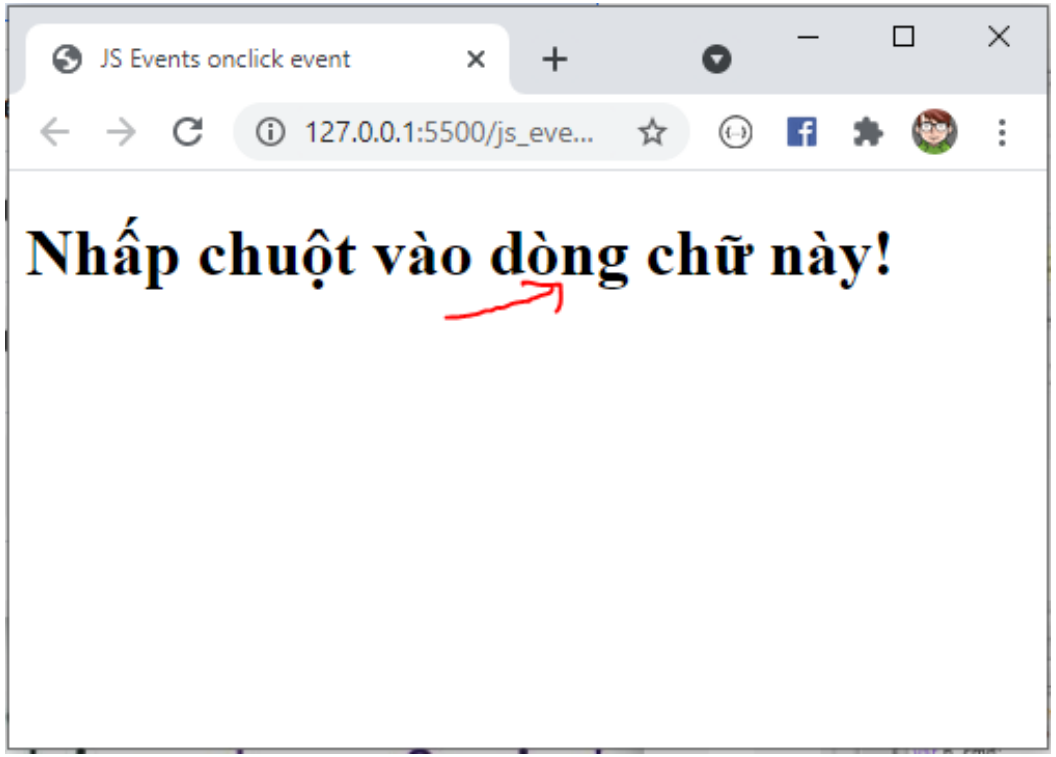
- **Ví dụ**: Thay đổi nội dung của phần tử **<h1>** khi người dùng click vào phần tử đó.

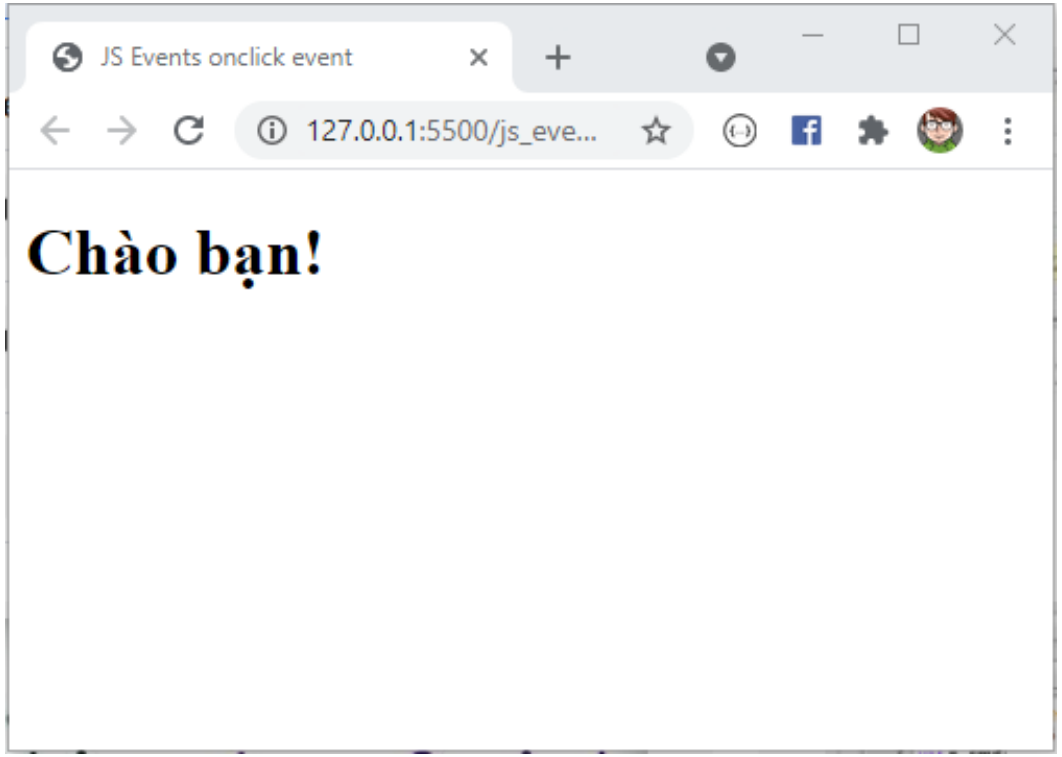


Tương tác sự kiện trong HTML với JavaScript

❑ Tương tác với các sự kiện:

- **Ví dụ**: Thay đổi nội dung của phần tử **<h1>** khi người dùng click vào phần tử đó.



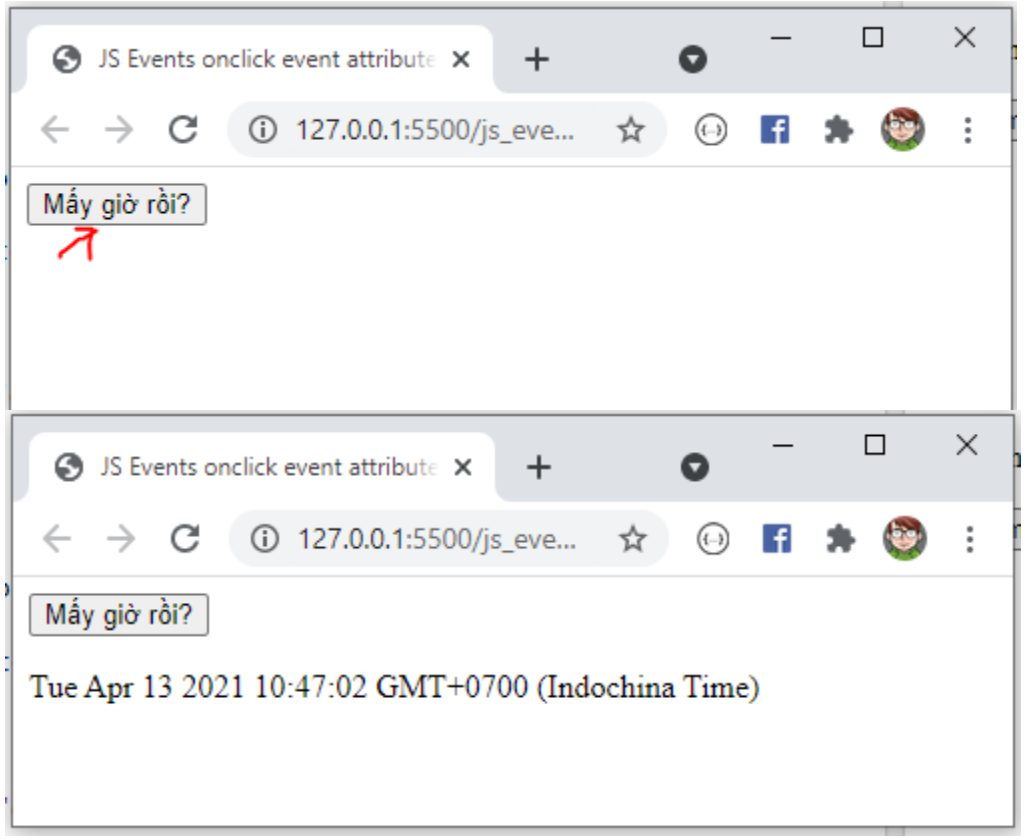


Thuộc tính sự kiện trong HTML với JavaScript

❑ Thuộc tính sự kiện HTML:

- Để gán sự kiện cho các phần tử HTML, ta có thể dùng các thuộc tính sự kiện.
- **Ví dụ**: Gán một sự kiện **onclick** cho phần tử nút bấm **button**

```
<!DOCTYPE html>
<html>
    <head>
        <title>JS Events onclick event attribute</title>
    </head>
    <body>
        <button onclick="displayDate()">Mấy giờ rồi?</button>

        <script>
            function displayDate() {
                document.getElementById("time").innerHTML = Date();
            }
        </script>

        <p id="time"></p>
    </body>
</html>
```

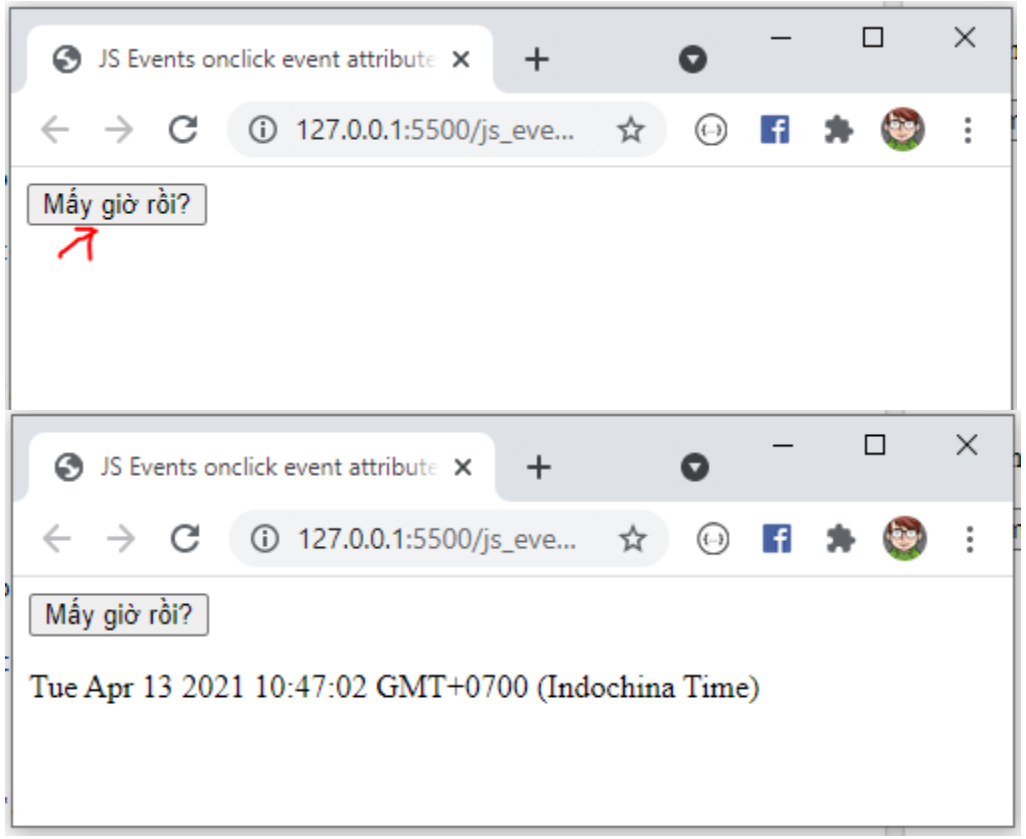


Gán các sự kiện trong HTML với JavaScript

❑ Gán các sự kiện sử dụng HTML DOM:

- HTML DOM cho phép bạn gán các sự kiện vào các phần tử HTML sử dụng JavaScript.
- **Ví dụ**: Gán một sự kiện **onclick** cho phần tử nút bấm **button**

```
<!DOCTYPE html>
<html>
    <head>
        <title>JS Events onclick event attribute</title>
    </head>
    <body>
        <button id="myBtn">Mấy giờ rồi?</button>

        <script>
            document.getElementById("myBtn").onclick= displayDate;

            function displayDate() {
                document.getElementById("time").innerHTML = Date();
            }
        </script>

        <p id="time"></p>
    </body>
</html>
```



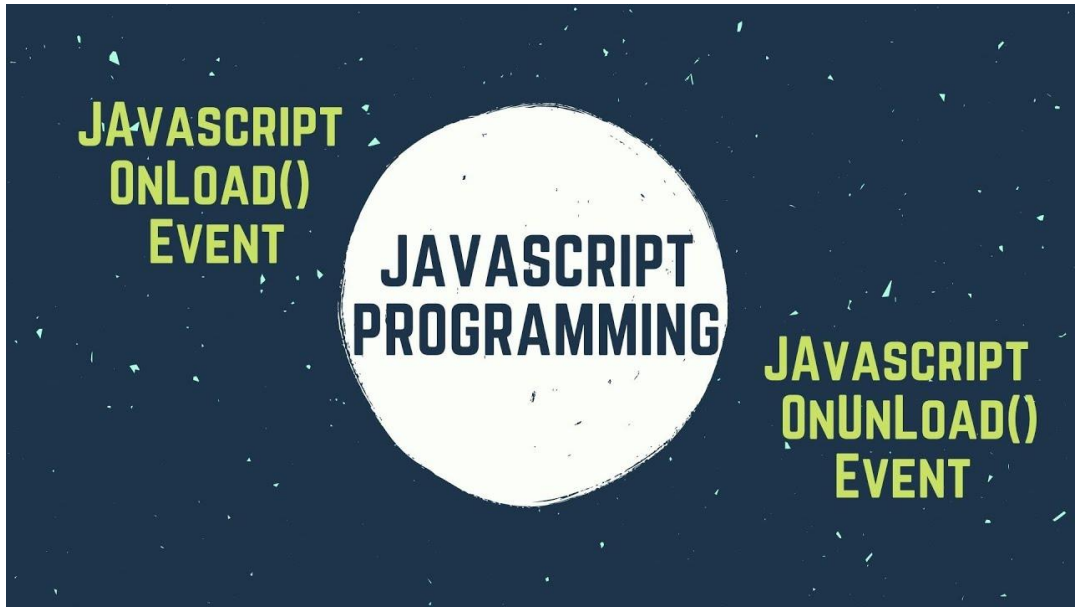
Các sự kiện trong HTML với JavaScript

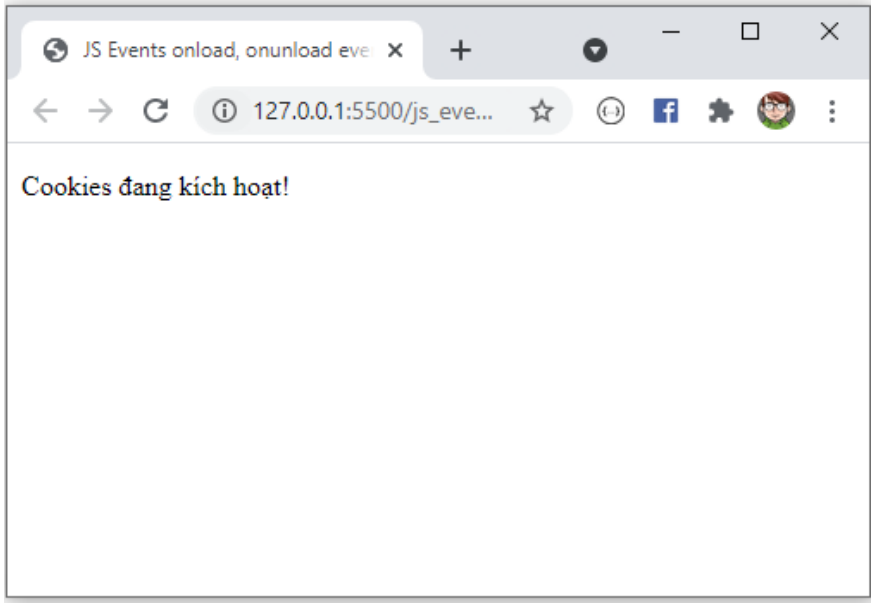
❑ Các sự kiện onload và onunload:

◆ Sự kiện **onload** và **onunload** được kích hoạt khi người dùng truy cập hoặc rời khỏi trang.

◆ Sự kiện **onload** có thể được sử dụng để kiểm tra loại trình duyệt và phiên bản trình duyệt của khách truy cập, đồng thời tải phiên bản phù hợp của trang web.

◆ Các sự kiện **onload** và **onunload** có thể được sử dụng để xử lý **cookie**





Các sự kiện trong HTML với JavaScript

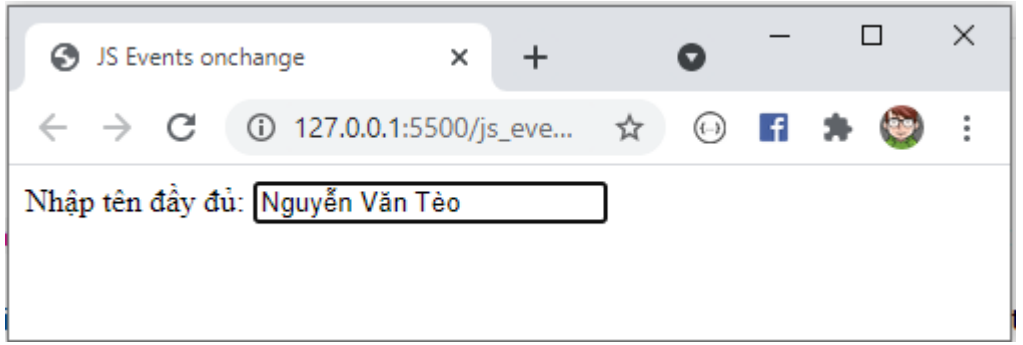
❑ Các sự kiện onload và onunload:

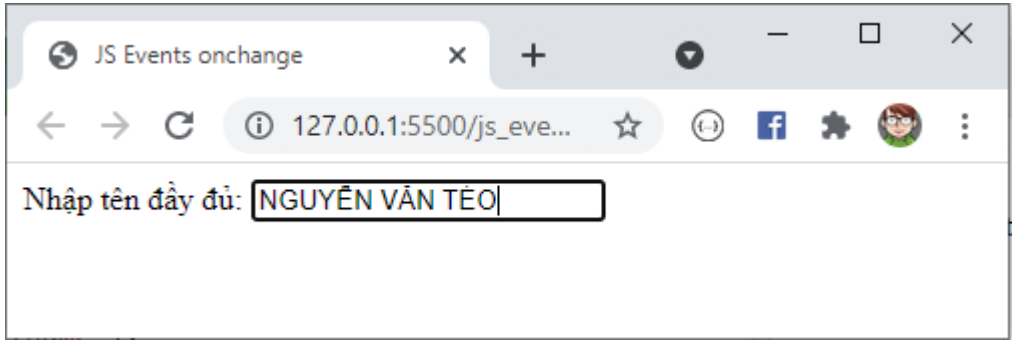
◆ Sự kiện **onchange** thường được sử dụng kết hợp để validate dữ liệu của các trường dữ liệu được nhập vào.

◆ **Ví dụ**: Sử dụng hàm **onchange** để gọi hàm **upperCase()** khi người dùng nhập nội dụng vào trường nhập dữ liệu.

js_events_onchange.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>JS Events onchange</title>

        <script>
            function myFunction() {
                var fullName = document.getElementById("fname");
                fullName.value = fullName.value.toUpperCase();
            }
        </script>
    </head>
    <body>
        Nhập tên đầy đủ: <input type="text" id="fname" onchange="myFunction()">
    </body>
</html>
```



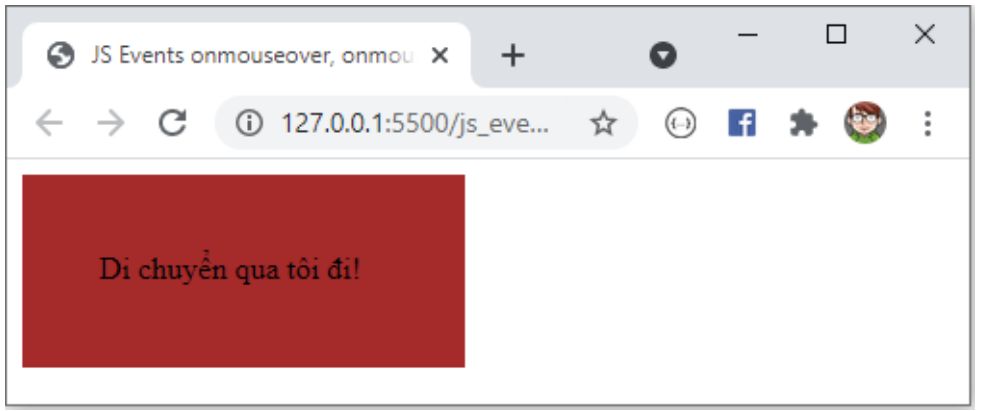


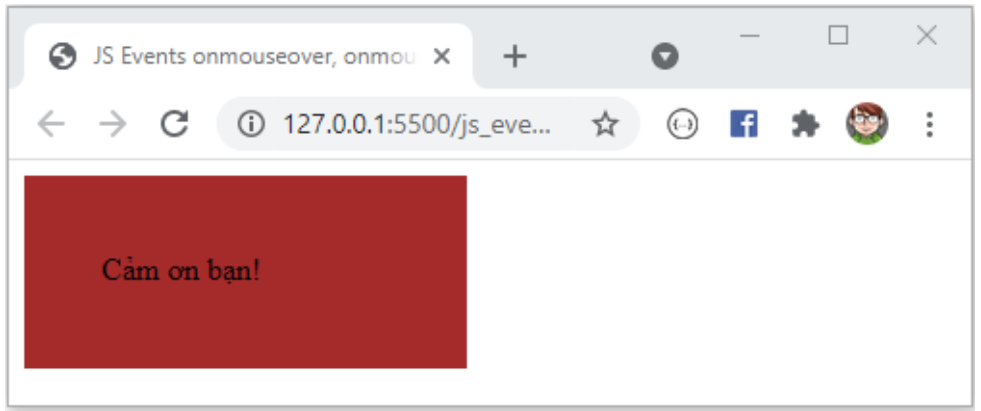
Các sự kiện trong HTML với JavaScript

❑ Các sự kiện onmouseover và onmouseout:

- Các sự kiện **onmouseover** và **onmouseout** có thể được sử dụng **để kích hoạt một chức năng khi người dùng di chuyển qua** hoặc **rời khỏi một phần tử HTML**.

```
<!DOCTYPE html>
<html>
    <head>
        <title>JS Events onmouseover, onmouseout</title>
    </head>
    <body>
        <div onmouseover="mOver(this)" onmouseout="mOut(this)"
            style="background-color: brown; width: 150px; height: 20px;
                padding: 40px;">Di chuyển qua tôi đi!</div>

            <script>
                function mOver(obj) {
                  obj.innerHTML = "Cảm ơn bạn!"
                }

                function mOut(obj) {
                  obj.innerHTML = "Di chuyển qua tôi đi!"
                }
            </script>
    </body>
</html>
```



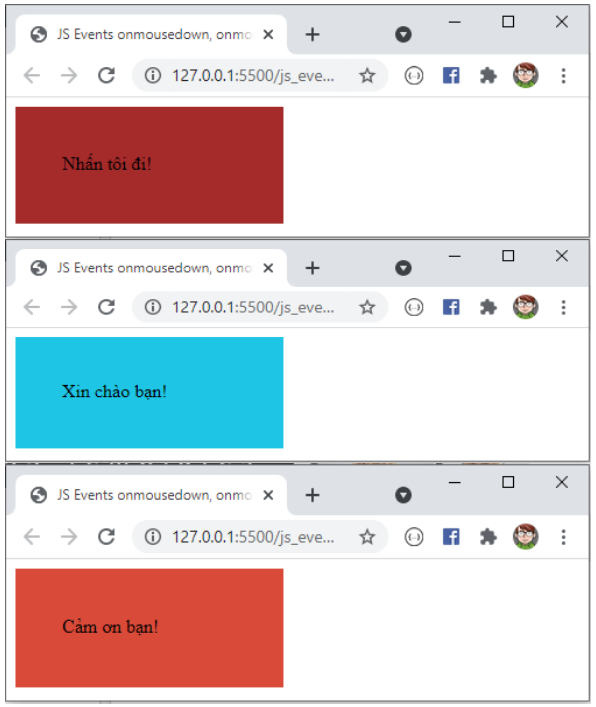


Các sự kiện trong HTML với JavaScript

❑ Các sự kiện onmousedown và onmouseup và onclick:

- **onmousedown**, **onmouseup** và **onclick** **đều là 1 phần của 1 cú nhấp chuột.**
  - Đầu tiên **khi một nút chuột được nhấp**, sự kiện **onmousedown** được kích hoạt.
  - Sau đó, **khi thả nút chuột**, sự kiện **onmouseup** được kích hoạt.
  - Cuối cùng, **khi nhấp chuột hoàn tất**, sự kiện **onclick** được kích hoạt.

```
<div onmousedown="mDown(this)" onmouseup="mUp(this)"
    style="background-color: brown; width: 150px;
        height: 20px; padding: 40px;">
    Nhấn tui đi!
</div>

<script>
    function mDown(obj) {
        obj.style.backgroundColor = "#1ec5e5";
        obj.innerHTML = "Xin chào bạn!";
    }

    function mUp(obj) {
        obj.style.backgroundColor = "#D94A38";
        obj.innerHTML = "Cảm ơn bạn!";
    }
</script>
```



Lắng nghe sự kiện trong HTML với JavaScript

❑ **Phương thức addEventListener():**

◆ Phương thức **addEventListener()** gắn một trình xử lý sự kiện vào phần tử được chỉ định.

◆ Phương thức **addEventListener()** gắn một trình xử lý sự kiện vào một phần tử mà không ghi đè các trình xử lý sự kiện hiện có.

◆ Bạn có thể thêm nhiều trình xử lý sự kiện vào một phần tử.

◆ Bạn có thể thêm nhiều trình xử lý sự kiện cùng loại vào một phần tử, ví dụ: hai sự kiện "nhấp chuột".

◆ Bạn có thể dễ dàng loại bỏ trình nghe sự kiện bằng cách sử dụng phương thức **removeEventListener()**.

◆ Cú pháp: `element.addEventListener(event, function, useCapture);`

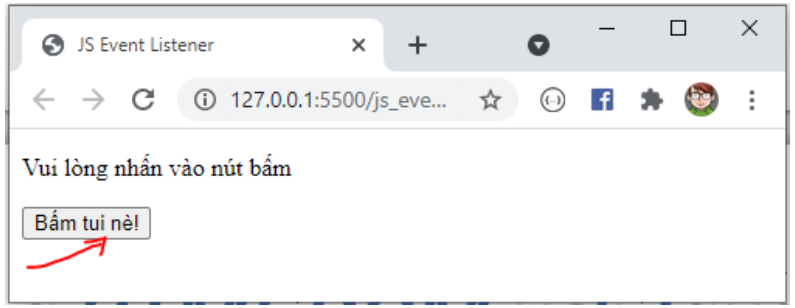
Lắng nghe sự kiện trong HTML với JavaScript

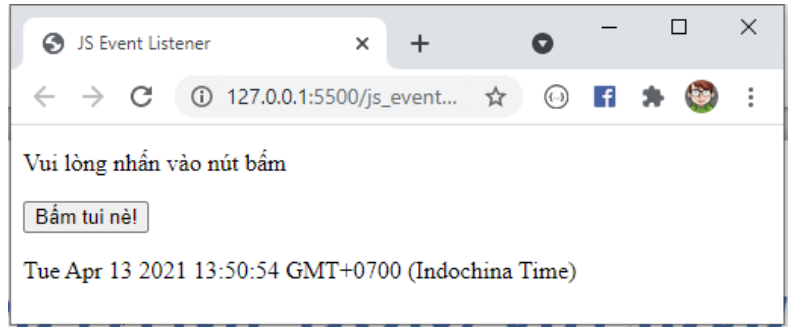
❑ Phương thức addEventListener():

- **Ví dụ**: Thêm một phương thức lắng nghe sự kiện khi người dùng nhấn vào nút bấm.

js_event_listener_01.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>JS Event Listener</title>
    </head>
    <body>
        <p>Vui lòng nhấn vào nút bấm</p>

        <button id="myBtn">Bấm tui nè!</button>

        <p id="result"></p>

        <script>
            document.getElementById("myBtn").addEventListener("click", displayDate);

            function displayDate() {
                document.getElementById("result").innerHTML = Date();
            }
        </script>
    </body>
</html>
```



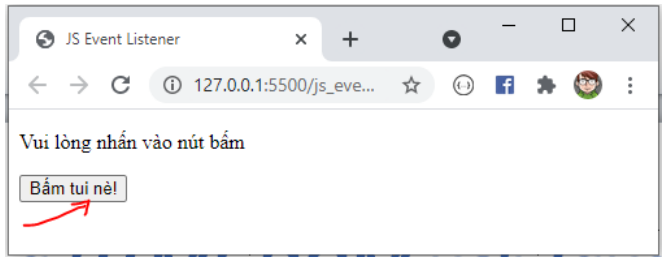


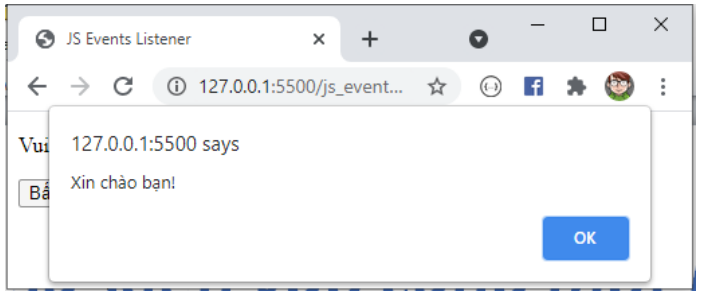
Thêm nhiều trình xử lý sự kiện vào cùng một HTML

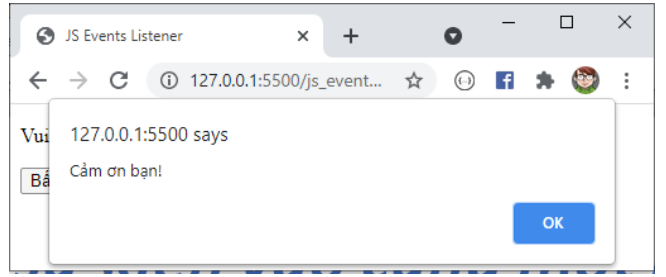
❑ Phương thức addEventListener():

◆ Phương thức **addEventListener()** cho phép bạn thêm rất nhiều **sự kiện giống nhau** vào một phần tử mà không ghi đè các sự kiện đang có trước đó.

```
<!DOCTYPE html>
<html>
    <head>
        <title>JS Events Listener</title>
    </head>
    <body>
        <p>Vui lòng nhấn vào nút bấm</p>

        <button id="myBtn">Bấm tui nè!</button>

        <script>
            var myButton = document.getElementById("myBtn");
            myButton.addEventListener("click", myFunction);
            myButton.addEventListener("click", someOtherFunction);

            function myFunction() {
                alert("Xin chào bạn!");
            }

            function someOtherFunction() {
                alert("Cảm ơn bạn!");
            }
        </script>
    </body>
</html>
```







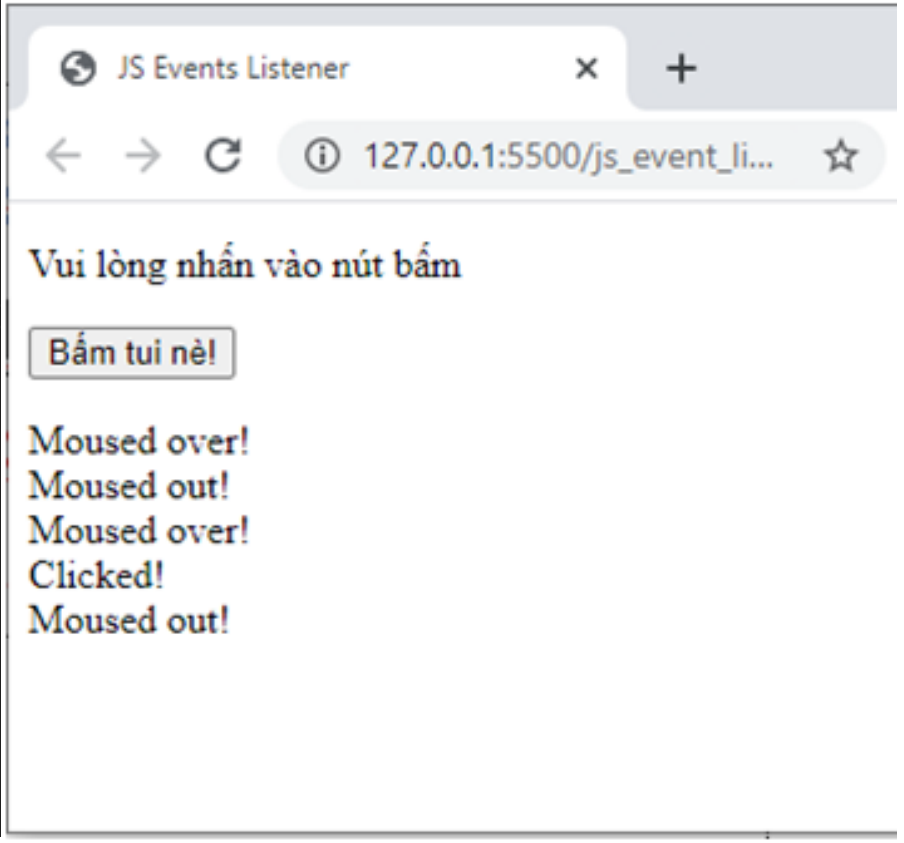
Thêm nhiều trình xử lý sự kiện vào cùng một HTML

❑ Phương thức addEventListener():

◆ Phương thức **addEventListener()** cho phép bạn thêm rất nhiều **sự kiện khác nhau** vào một phần tử mà không ghi đè các sự kiện đang có trước đó.

js_event_listener_03.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>JS Events Listener</title>
    </head>
    <body>
        <p>Vui lòng nhấn vào nút bấm</p>

        <button id="myBtn">Bấm tui nè!</button>

        <p id="result"></p>

        <script>
            var myButton = document.getElementById("myBtn");
            myButton.addEventListener("mouseover", myFunction);
            myButton.addEventListener("click", mySecondFunction);
            myButton.addEventListener("mouseout", myThirdFunction);

            function myFunction() {
                document.getElementById("result").innerHTML += "Moused over!<br>";
            }

            function mySecondFunction() {
                document.getElementById("result").innerHTML += "Clicked!<br>";
            }

            function myThirdFunction() {
                document.getElementById("result").innerHTML += "Moused out!<br>";
            }
        </script>
    </body>
</html>
```



Thêm một trình xử lý sự kiện vào đối tượng Window

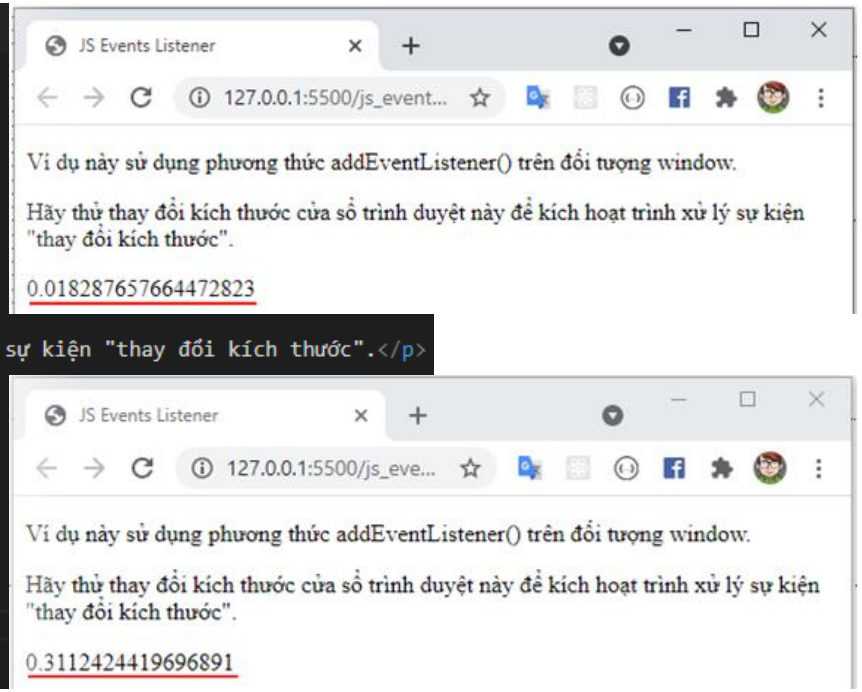
❑ Phương thức addEventListener():

◆ Phương thức **addEventListener()** cho phép bạn thêm trình xử lý sự kiện trên bất kỳ đối tượng HTML DOM nào như các **phần tử HTML**, **tài liệu HTML**, đối tượng cửa sổ (**Window**) hoặc các đối tượng khác hỗ trợ sự kiện, như đối tượng **xmlHttpRequest**.

◆ Thêm trình xử lý sự kiện sẽ kích hoạt khi người dùng thay đổi kích thước cửa sổ:

js_event_listener_04.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>JS Events Listener</title>
    </head>
    <body>
        <p>Ví dụ này sử dụng phương thức addEventListener() trên đối tượng window.</p>

        <p>Hãy thử thay đổi kích thước cửa sổ trình duyệt này để kích hoạt trình xử lý sự kiện "thay đổi kích thước".</p>

        <p id="demo"></p>

        <script>
            window.addEventListener("resize", function(){
            document.getElementById("demo").innerHTML = Math.random();
        });
        </script>
    </body>
</html>
```

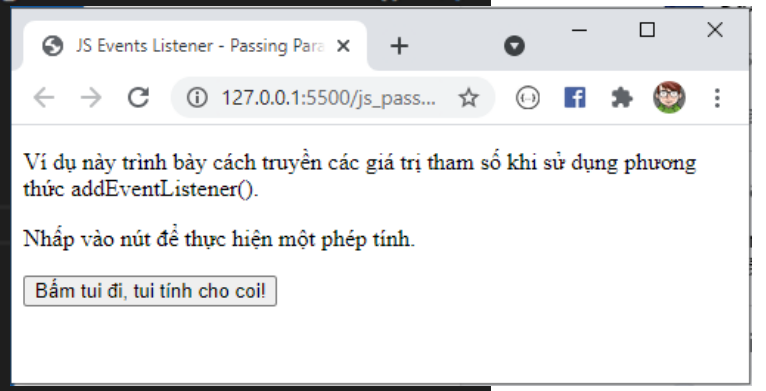


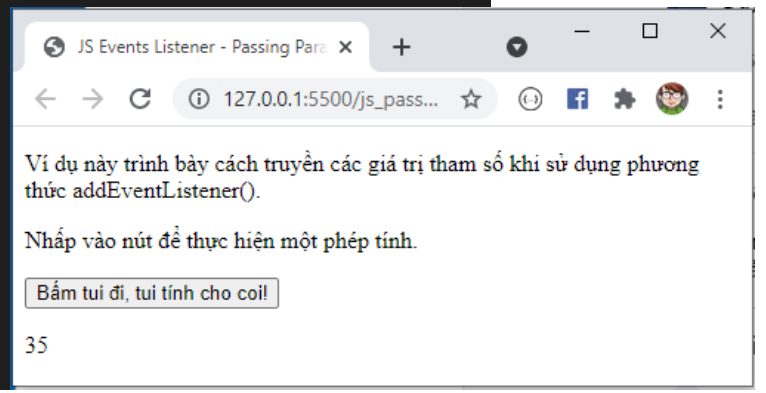
Thêm một trình xử lý sự kiện vào đối tượng Window

❑ Truyền tham số:

◆ Khi chuyển các giá trị tham số, hãy sử dụng "hàm ẩn danh" để gọi hàm được chỉ định với các tham số.

```
<p>Ví dụ này trình bày cách truyền các giá trị tham số khi sử dụng phương thức addEventListener().</p>

<p>Nhấp vào nút để thực hiện một phép tính.</p>

<button id="myBtn">Bấm tui đi, tui tính cho coi!</button>

<p id="demo"></p>

<script>
var p1 = 5;
var p2 = 7;

document.getElementById("myBtn").addEventListener("click", function() {
  myFunction(p1, p2);
});

function myFunction(a, b) {
  var result = a * b;
  document.getElementById("demo").innerHTML = result;
}
</script>
```

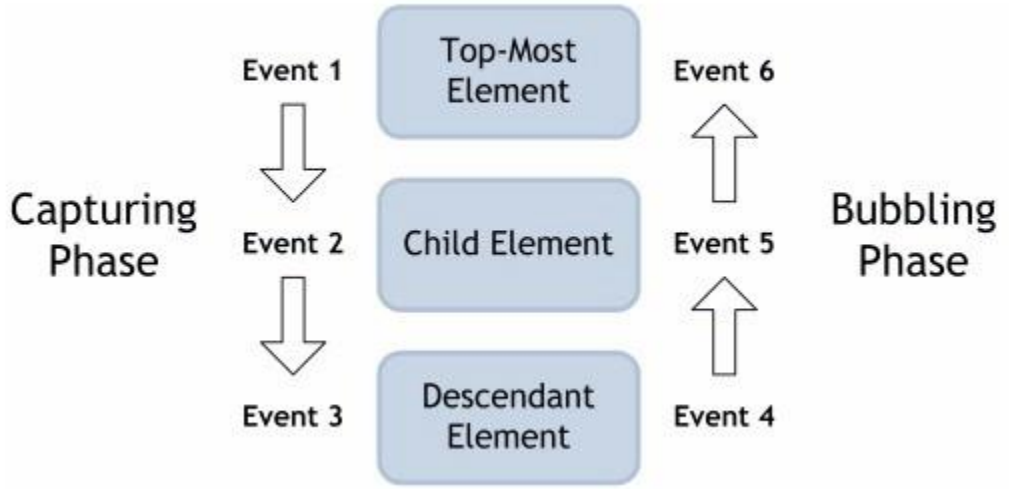




Các cách truyền sự kiện trong HTML với JavaScript

❑ **Bong bóng sự kiện & ghi lại sự kiện:**

- Có hai cách lan truyền sự kiện trong HTML DOM:
  - Bong bóng sự kiện (**Event Bubbling**).
  - Chụp ảnh sự kiện (**Event Capturing**).
- **Câu hỏi**: lan truyền sự kiện là một cách xác định thứ tự phần tử khi một sự kiện xảy ra. Nếu bạn có phần tử **<p>** bên trong phần tử ** ** và người dùng nhấp vào phần tử **<p>**, thì sự kiện "nhấp chuột" của phần tử nào sẽ được xử lý đầu tiên?



Các cách truyền sự kiện trong HTML với JavaScript

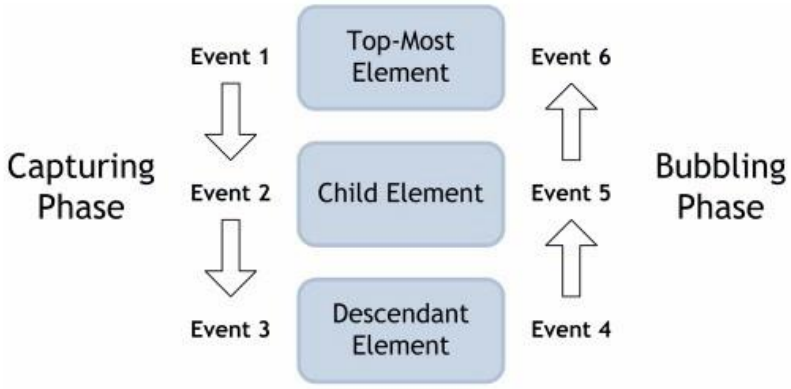
❑ **Bong bóng sự kiện & ghi lại sự kiện:**

◆ Có hai cách lan truyền sự kiện trong HTML DOM:

- Bong bóng sự kiện (**Event Bubbling**).
- Chụp ảnh sự kiện (**Event Capturing**).

◆ Trong cách bong bóng sự kiện, sự kiện của phần tử bên trong nhất được xử lý trước rồi đến phần bên ngoài:

- Sự kiện nhấp chuột của phần tử **<p>** được xử lý đầu tiên
- Sau đó là sự kiện nhấp chuột của phần tử ** **.



Các cách truyền sự kiện trong HTML với JavaScript

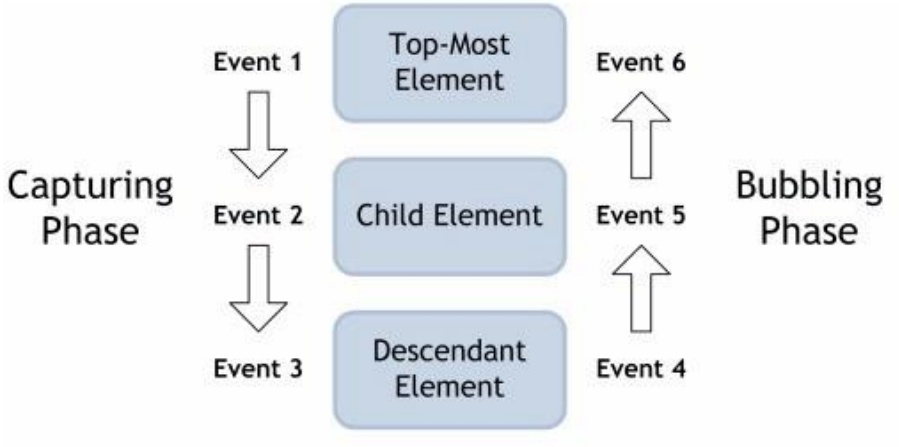
❑ **Bong bóng sự kiện & ghi lại sự kiện:**

◆ Có hai cách lan truyền sự kiện trong HTML DOM:

- Bong bóng sự kiện (**Event Bubbling**).
- Chụp ảnh sự kiện (**Event Capturing**).

◆ Trong ghi lại sự kiện, sự kiện của phần tử bên ngoài nhất được xử lý đầu tiên và sau đó sự kiện nhấp chuột của phần tử bên trong:

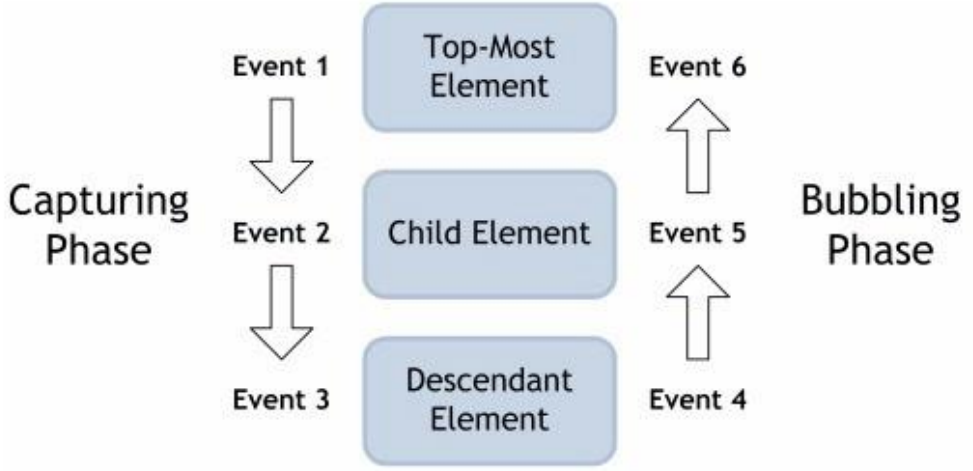
- ** ** sẽ được xử lý đầu tiên.
- Sau đó là sự kiện nhấp chuột của phần tử **<p>**.



Các cách truyền sự kiện trong HTML với JavaScript

❑ Bong bóng sự kiện & ghi lại sự kiện:

- ◆ Có hai cách lan truyền sự kiện trong HTML DOM:
  - Bong bóng sự kiện (**Event Bubbling**).
  - Chụp ảnh sự kiện (**Event Capturing**).
- ◆ Cú pháp: **addEventListener**(*event*, *function*, *useCapture*);
- ◆ Giá trị mặc định là **false**, giá trị này sẽ sử dụng lan truyền bong bóng, khi giá trị được đặt thành **true**, sự kiện sẽ sử dụng sự lan truyền ghi lại.



Các cách truyền sự kiện trong HTML với JavaScript

❑ Bong bóng sự kiện & ghi lại sự kiện:

◆ **Ví dụ**: Giá trị mặc định là **false**, giá trị này sẽ sử dụng lan truyền bong bóng, khi giá trị được đặt thành **true**, sự kiện sẽ sử dụng sự lan truyền ghi lại.

```
<head>
    <title>JS Events Listener - Bubbling, Capturing</title>
    <style>
        #myDiv1,
        #myDiv2 {
            background-color: coral;
            padding: 50px;
        }

        #myP1,
        #myP2 {
            background-color: white;
            font-size: 20px;
            border: 1px solid;
            padding: 20px;
        }
    </style>
    <meta content="text/html; charset=utf-8" http-equiv="Content-Type">
</head>
```

```
<div id="myDiv1">
    <h2>Bubbling:</h2>
    <p id="myP1">Bấm tui nè!</p>
</div><br>

<div id="myDiv2">
    <h2>Capturing:</h2>
    <p id="myP2">Bấm tui nè!</p>
</div>
```

Các cách truyền sự kiện trong HTML với JavaScript

❑ Bong bóng sự kiện & ghi lại sự kiện:

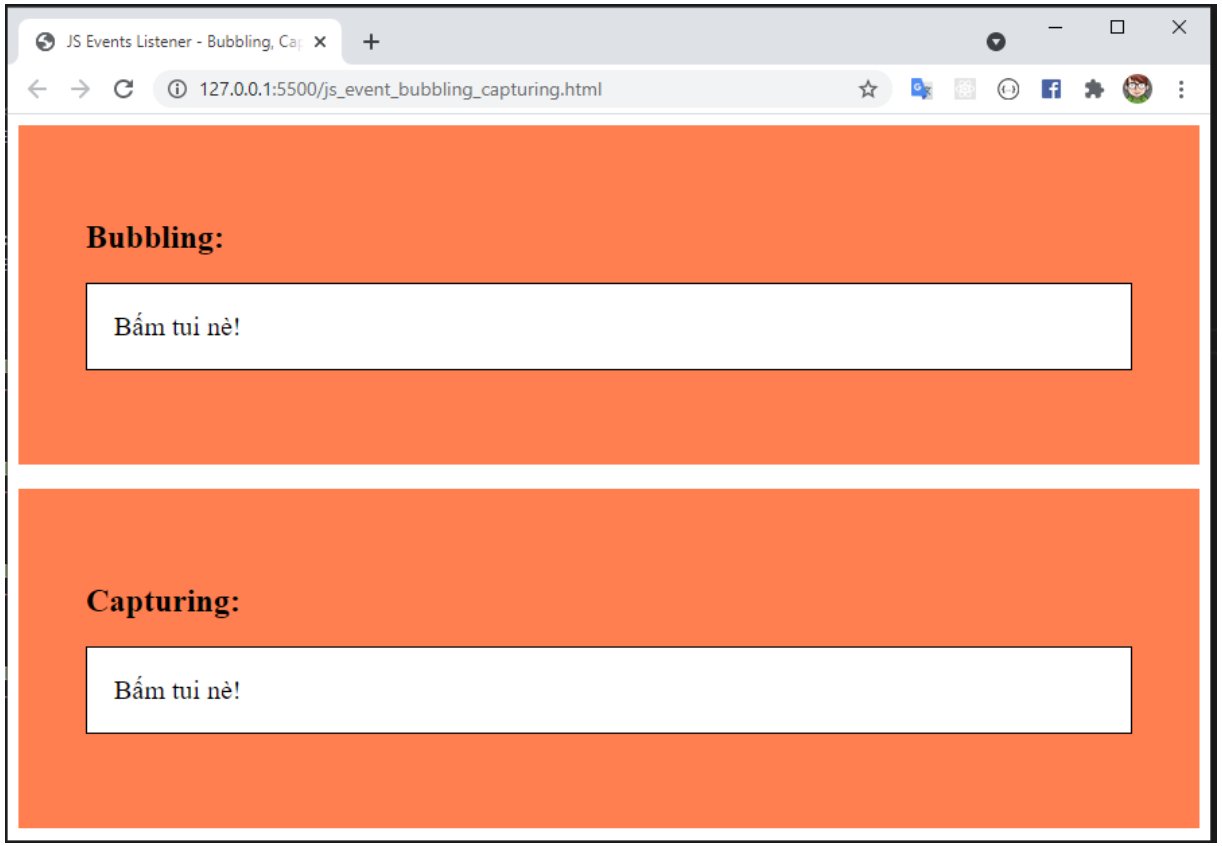
◆ **Ví dụ**: Giá trị mặc định là **false**, giá trị này sẽ sử dụng lan truyền bong bóng, khi giá trị được đặt thành **true**, sự kiện sẽ sử dụng sự lan truyền ghi lại.

```
<script>
    document.getElementById("myP1").addEventListener("click", function () {
        alert("Bạn đã nhấn vào phần tử màu trắng!");
    }, false);

    document.getElementById("myDiv1").addEventListener("click", function () {
        alert("Bạn đã nhấn vào phần tử màu cam!");
    }, false);

    document.getElementById("myP2").addEventListener("click", function () {
        alert("Bạn đã nhấn vào phần tử màu trắng!");
    }, true);

    document.getElementById("myDiv2").addEventListener("click", function () {
        alert("Bạn đã nhấn vào phần tử màu cam!");
    }, true);
</script>
```

Các cách truyền sự kiện trong HTML với JavaScript

❑ Bong bóng sự kiện & ghi lại sự kiện:

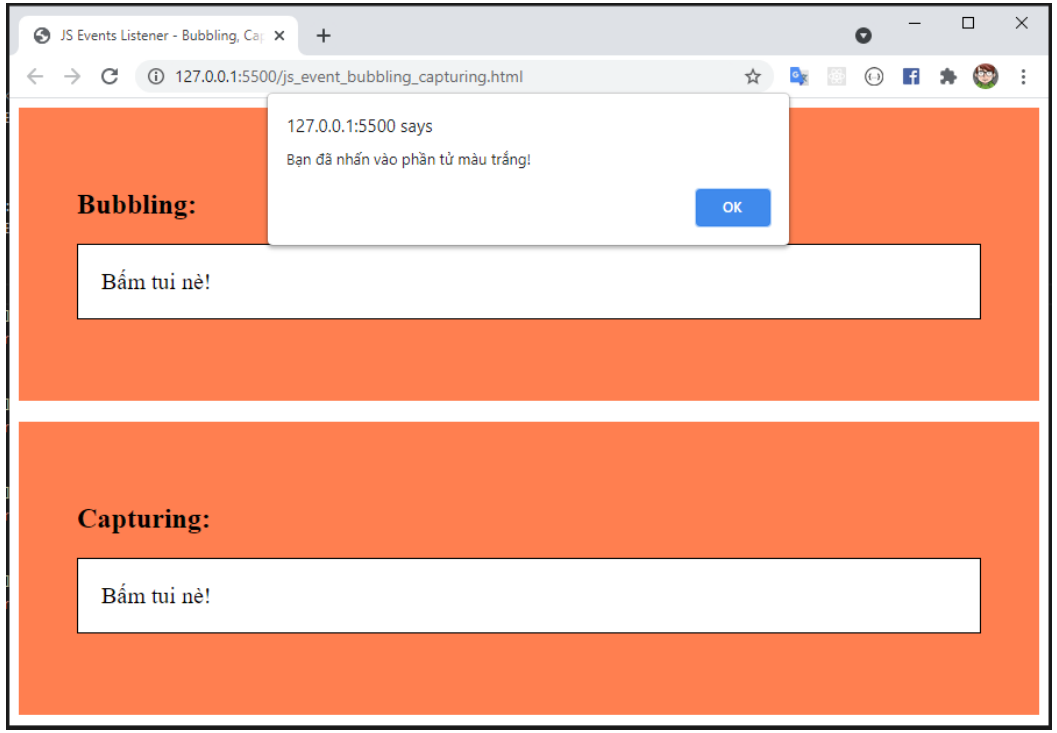
◆ **Ví dụ**: Giá trị mặc định là **false**, giá trị này sẽ sử dụng lan truyền bong bóng, khi giá trị được đặt thành **true**, sự kiện sẽ sử dụng sự lan truyền ghi lại.

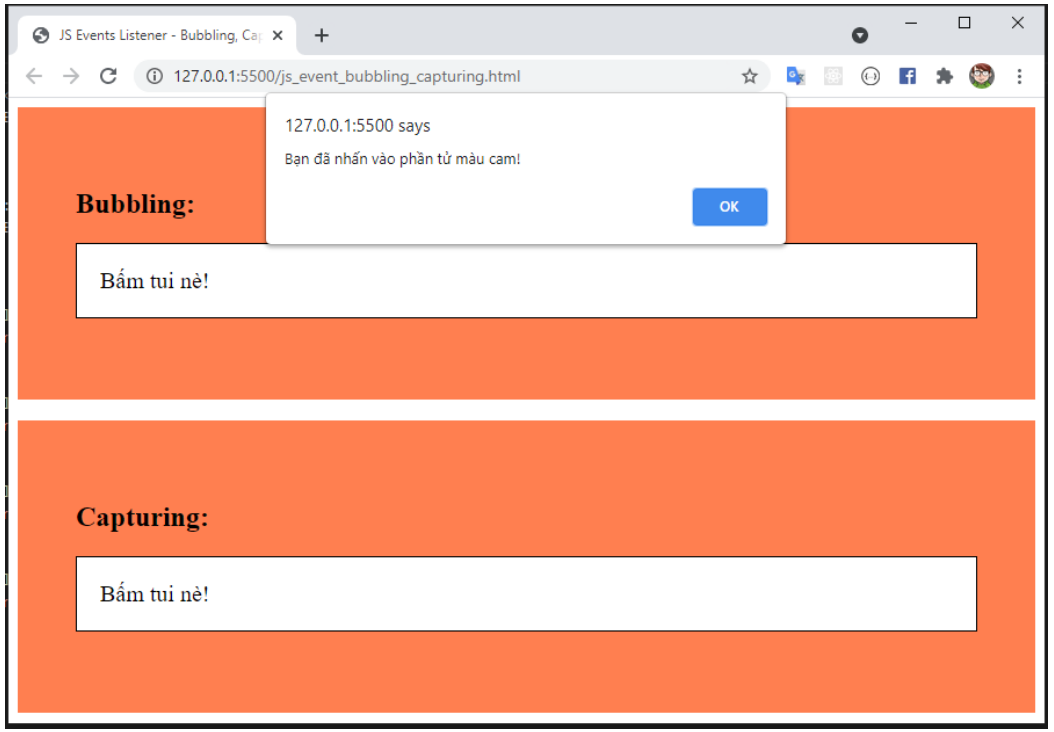


Các cách truyền sự kiện trong HTML với JavaScript

❑ Bong bóng sự kiện & ghi lại sự kiện:

◆ **Ví dụ**: Giá trị mặc định là **false**, giá trị này sẽ sử dụng lan truyền bong bóng, khi giá trị được đặt thành **true**, sự kiện sẽ sử dụng sự lan truyền ghi lại. (**Buddling**)

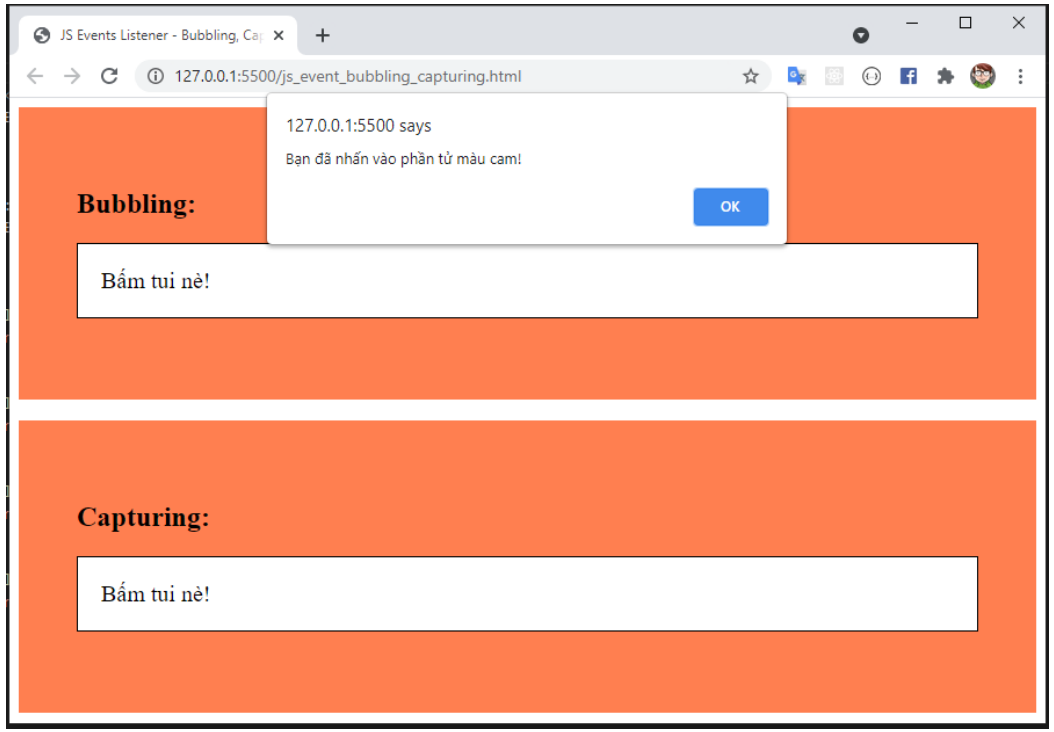


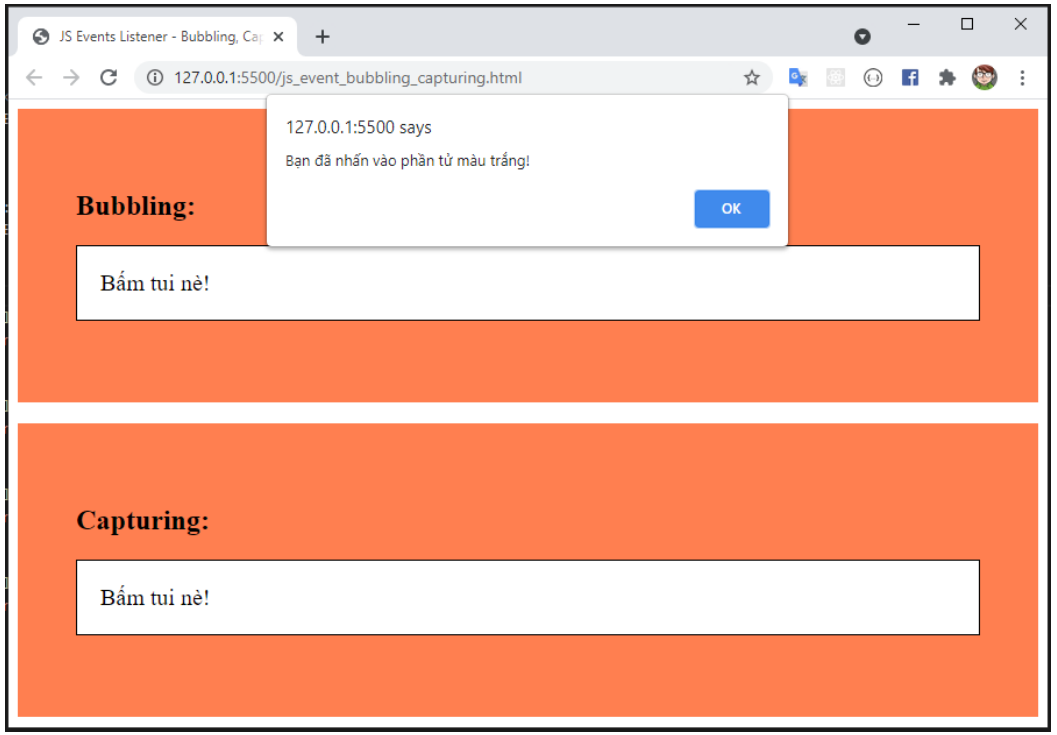


Các cách truyền sự kiện trong HTML với JavaScript

❑ Bong bóng sự kiện & ghi lại sự kiện:

◆ **Ví dụ**: Giá trị mặc định là **false**, giá trị này sẽ sử dụng lan truyền bong bóng, khi giá trị được đặt thành **true**, sự kiện sẽ sử dụng sự lan truyền ghi lại. (**Capturing**)





Xóa sự kiện trong HTML với JavaScript

❑ Phương thức xóa sự kiện - removeEventListener():

◆ Phương thức **removeEventListener()** xóa các trình xử lý sự kiện đã được đính kèm với phương thức **addEventListener()**

```
<head>
    <title>JS Remove Event Listener</title>
    <style>
        #myDIV {
          background-color: coral;
          border: 1px solid;
          padding: 50px;
          color: white;
          font-size: 20px;
        }
    </style>
</head>
```

```
<h2>JavaScript removeEventListener()</h2>

<div id="myDIV">
  <p>Phần tử div này có một trình xử lý sự kiện onmousemove
    mỗi khi bạn di chuyển chuột vào bên trong trường màu cam này.</p>
  <p>Nhấp vào nút để loại bỏ trình xử lý sự kiện của div.</p>
  <button onclick="removeHandler()" id="myBtn">Xóa</button>
</div>

<p id="demo"></p>
```

Xóa sự kiện trong HTML với JavaScript

❑ Phương thức xóa sự kiện - removeEventListener():

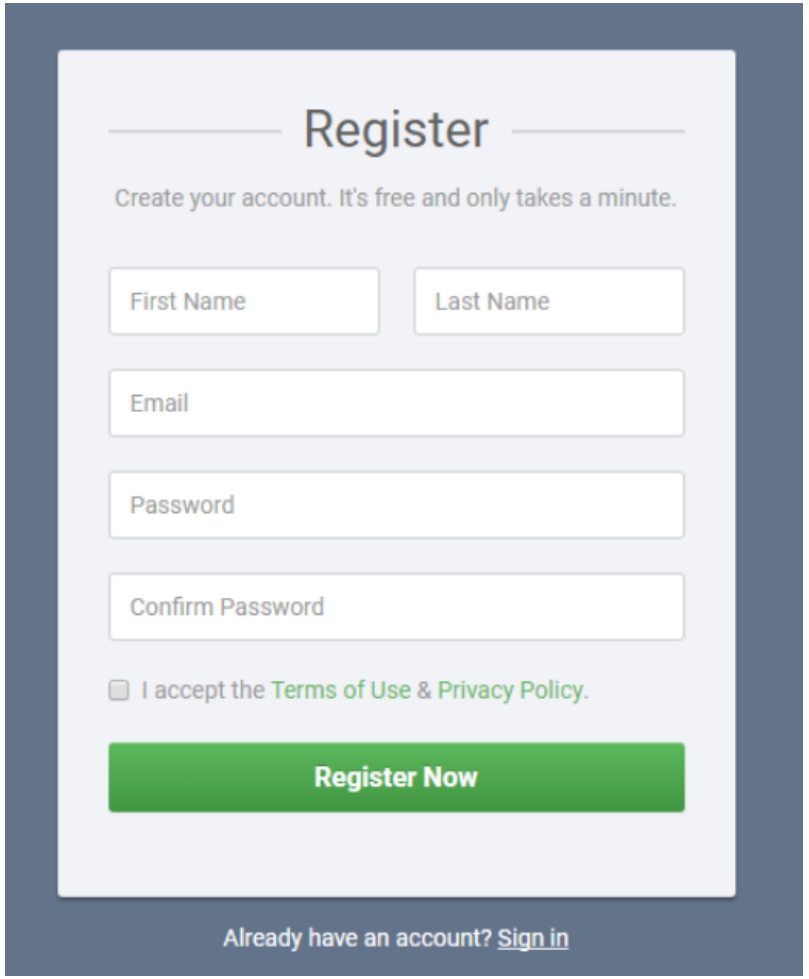
◆ Phương thức **removeEventListener()** xóa các trình xử lý sự kiện đã được đính kèm với phương thức **addEventListener()**

```
<script>
    document.getElementById("myDIV").addEventListener("mousemove", myFunction);

    function myFunction() {
        document.getElementById("demo").innerHTML = Math.random();
    }

    function removeHandler() {
        document.getElementById("myDIV").removeEventListener("mousemove", myFunction);
    }
</script>
```

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ **Giới thiệu về kiểm lỗi dữ liệu:**

- Bạn **đã thường xuyên gặp một website mà ở đó người dùng nhập các thông tin vào** một **biểu mẫu** (form) **trước khi gửi tới máy chủ**. Chẳng hạn **biểu mẫu đăng ký tài khoản**.
- **Các thông tin** mà **người dùng nhập vào biểu mẫu cần** phải **được xác thực** (validate) **để đảm bảo sự hợp lý của dữ liệu**.
- Một **vài ví dụ về kiểm lỗi dữ liệu**:
  - Kiểm tra đảm bảo **dữ liệu không rỗng**.
  - Kiểm tra **định dạng email**.
  - Kiểm tra **định dạng số điện thoại**.
  - ...



Kiểm tra dữ liệu hợp lệ với JavaScript

❑ **Các cách kiểm lỗi dữ liệu trên Form:**

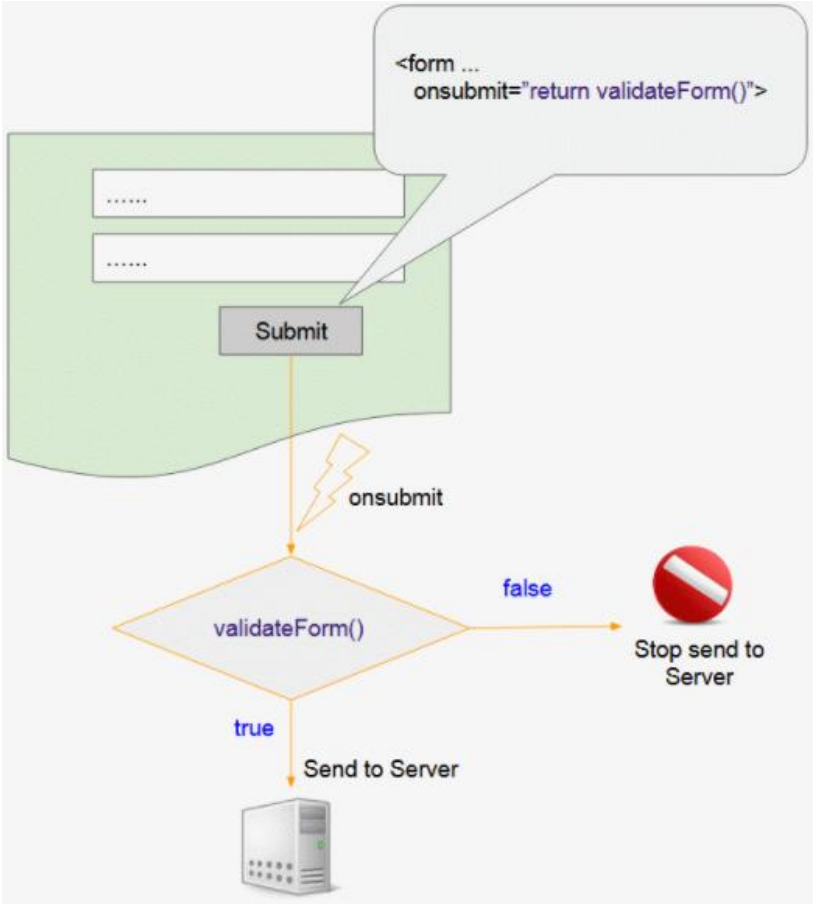
◆ Về **cơ bản có 3 cách để kiểm tra lỗi dữ liệu**:

▪ **Dữ liệu** của **form** sẽ **được gửi tới server** (máy chủ), và **việc kiểm tra** (**validate**) **sẽ được thực hiện tại phía máy chủ**.

▪ **Dữ liệu** của **form** sẽ **được kiểm tra tại phía client bằng cách sử dụng Javascript**, điều này giúp **server** không phải làm việc quá nhiều, và tăng hiệu năng cho ứng dụng.

▪ Sử dụng cả 2 phương thức trên để xác thực **form**.

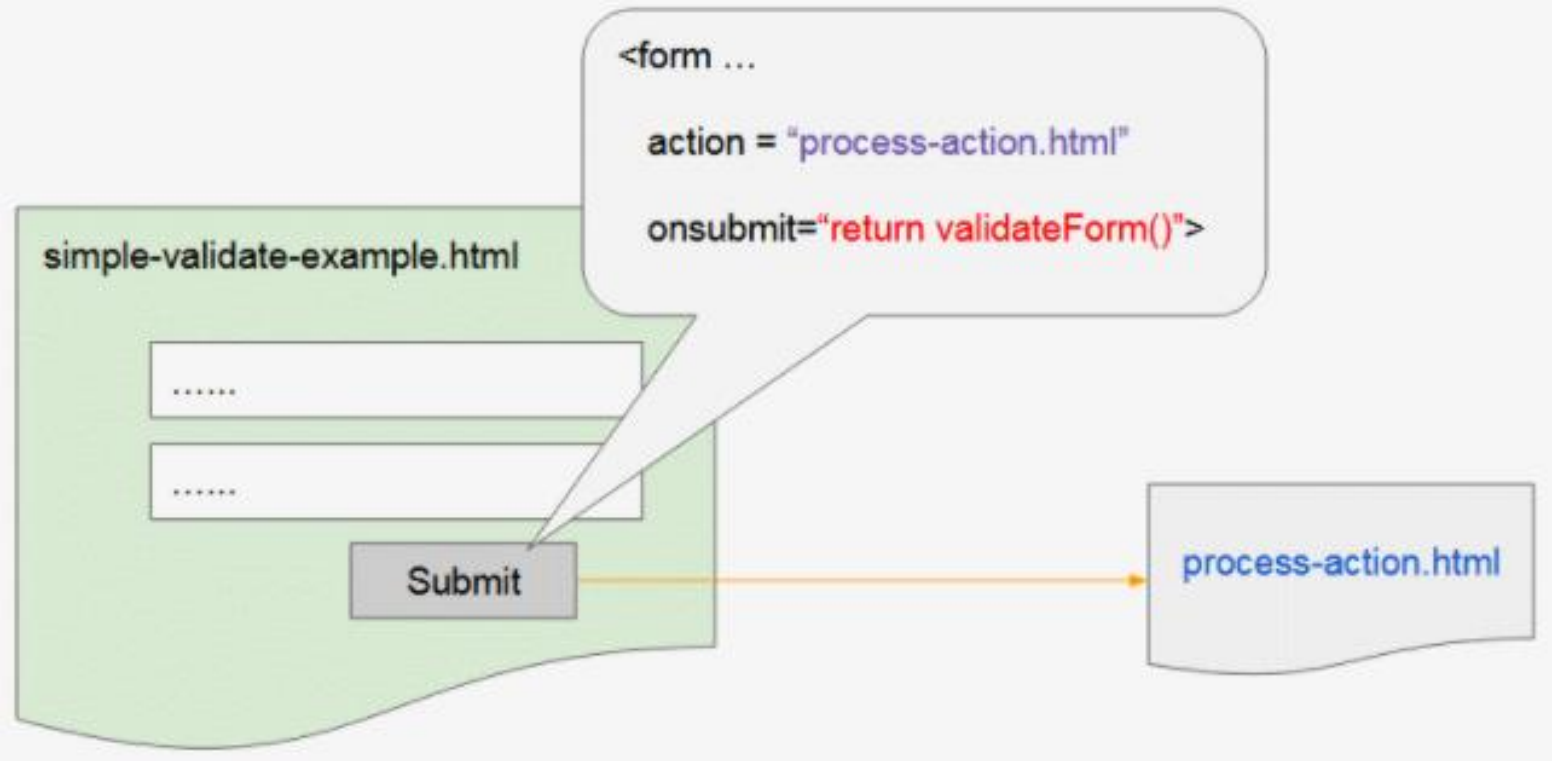
=> Bạn đang học về giao diện, chúng ta sẽ tìm hiểu việc sử dụng JS để kiểm lỗi dữ liệu Form.



Kiểm tra dữ liệu hợp lệ với JavaScript

❑ **Lưu ý khi kiểm lỗi dữ liệu trên Form:**

◆ Thuộc tính (attribute) **action** của **<form>** được sử dụng để chỉ định trang mà dữ liệu sẽ được gửi đến, hay nói cách khác đây chính là trang sẽ xử lý dữ liệu được gửi đến từ **<form>** của trang hiện tại.



Kiểm tra dữ liệu hợp lệ với JavaScript

- **Ví dụ kiểm lỗi dữ liệu trên Form:**
  - **Tạo** một **form đăng nhập** bao **gồm 2 trường dữ liệu**: **Username** và **Password**.
  - Có **3 thuộc tính form** cần lưu ý:
    - **method**="GET" -> loại phương thức sẽ được xử lý khi submit form này.
    - **action**="process-action.html" -> chỉ định trang mà dữ liệu sẽ được gửi đến.
    - **onsubmit**="return validateForm()" -> hàm xử lý JS được gọi khi submit form.

```
<h2>Enter your Username and Password</h2>

<div style="border:1px solid #ddd; padding: 5px;">
    <form method="GET" action="process-action.html" onsubmit="return validateForm()">
        Username: <input type="text" name="username" id="username" />
        <br><br> Password: <input type="password" name="password" id="password" />
        <br><br>
        <button type="submit">Submit</button>
    </form>
</div>
```

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ **Ví dụ kiểm lỗi dữ liệu trên Form:**

◆ **Hiện thực** hàm **validateForm**() đơn giản **để kiểm tra dữ liệu hợp lệ ở 2 trường dữ liệu**: **Username** và **Password**.

```
<script type="text/javascript">
    function validateForm() {
        var u = document.getElementById("username").value;
        var p = document.getElementById("password").value;

        if (u == "") {
            alert("Please enter your Username");
            return false;
        }
        if (p == "") {
            alert("Please enter you Password");
            return false;
        }

        alert("All datas are valid!, send it to the server!");

        return true;
    }
</script>
```

❑ **Ví dụ kiểm lỗi dữ liệu trên Form:**

◆ **Tạo** một **trang process-action.html** để **làm trang đích đến khi validate form thành công**.

```
<!DOCTYPE html>
<html>

<head>
    <title>Process Action</title>
</head>

<body>
    <h3>Process Action Page</h3>

    OK, I got data!

    <br/><br/>

    <a href="javascript:history.back();">[Go Back]</a>
</body>

</html>
```

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ Ví dụ kiểm lỗi dữ liệu trên Form:

◆ Bạn **có thể truy cập** vào **dữ liệu** của **một trường** (field) **thông qua id** hoặc **name** của **trường** (field) dữ liệu đó bằng cách đặt các thuộc tính **id**, **name** vào input như sau:

```
Username: <input type="text" name="username" id="username" />
<br><br> Password: <input type="password" name="password" id="password" />
```

◆ Và **truy xuất trường dữ liệu** trên **thông qua id** như sau:

```
var field = document.getElementById("field");
var value = field.value;
```

◆ Và **truy xuất trường dữ liệu** trên **thông qua name** như sau:

```
var myForm = document.forms["myForm"];
var uname = myForm["username"].value;
var pass = myForm["password"].value;
```

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ Ví dụ kiểm lỗi dữ liệu trên Form:

◆ **Khi** người dùng **nhập dữ liệu không chính xác trên một trường** của **form**, bạn **nên thông báo cho người dùng** và **đồng thời focus** vào **trường đó**.

```
<body>

    <h2>Enter your Username and Password</h2>

    <div style="border:1px solid #ddd; padding: 5px;">

        <form name="myForm" method="GET"
                action="process-action.html"
                onsubmit="return validateForm()">
            Username: <input type="text" name="username" />
            <br><br> Password: <input type="password" name="password" />
            <br><br>
            <button type="submit">Submit</button>
        </form>

    </div>

</body>
```

```
<script type="text/javascript">
    function validateForm() {
        // Get form via form name:
        var myForm = document.forms["myForm"];

        var u = myForm["username"].value;
        var p = myForm["password"].value;

        if (u == "") {
            alert("Please enter your Username");
            myForm["username"].focus(); // Focus
            return false;
        }
        if (p == "") {
            alert("Please enter you Password");
            myForm["password"].focus(); // Focus
            return false;
        }

        alert("All datas are valid!, send it to the server!");

        return true;
    }
</script>
```

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ **Ví dụ kiểm lỗi dữ liệu trên Form:**

◆ **Thêm** thuộc tính **required** vào một trường của **form** để nói với trình duyệt rằng **trường này là bắt buộc**, **trình duyệt** sẽ **tự động kiểm tra** và **thông báo** cho **người dùng nếu người dùng không nhập** vào **trường đó**.

```
<h2>Required attribute</h2>

<div style="border:1px solid #ddd; padding: 5px;">

    <form name="myForm" action="process-action.html"
          onsubmit="return validateForm()">
        Your Name: <input type="text" name="fullName" value="" required/>
        <br/><br/>
        <button type="submit">Submit</button>
    </form>

</div>
```

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ Ví dụ kiểm lỗi dữ liệu trên Form:

◆ Một vài loại phần tử **<input>** mới được giới thiệu trong **HTML 5**, chẳng hạn ***color, date, datetime-local, email, month, number, range, search, tel, time, url, week***,.

◆ Các phần tử này có các thuộc tính (attribute) đặc biệt giúp trình duyệt biết cách để **validate** dữ liệu của nó một cách tự động.

Thuộc tính	Mô tả
Disable	Chỉ định rõ ràng input này sẽ bị vô hiệu hóa (disable)
Max	Chỉ định giá trị lớn nhất của phần tử input này
Min	Chỉ định giá trị nhỏ nhất của phần tử input này
Pattern	Chỉ định pattern của phần tử input này
Required	Chỉ định rằng trường đầu vào là bắt buộc. Người dùng phải nhập dữ liệu.
Type	Chỉ định kiểu của phần tử input

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ Ví dụ kiểm lỗi dữ liệu trên Form:

◆ Một phần tử **<input type="number">** với các thuộc tính **min, max**, trình duyệt sẽ thông báo cho người dùng nếu họ nhập vào một con số nằm ngoài phạm vi cho phép.

```
<h2>Attribute min, max</h2>

<div style="border:1px solid #ddd; padding: 5px;">

    <form name="myForm" action="process-action.html">
        Enter your score (0-100):
        <input type="number" name="score" min="0" max="100" />
        <br/><br/>
        <button type="submit">Submit</button>
    </form>

</div>
```

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ Ví dụ kiểm lỗi dữ liệu trên Form:

- Yêu cầu người dùng nhập vào một mã số quốc gia có 2 ký tự.

```
<h2>Attribute: pattern</h2>

<div style="border:1px solid #ddd; padding: 5px;">

    <form name="myForm" action="process-action.html">
        Country code:
        <input type="text" name="countryCode" pattern="[A-Za-z]{2}"
                title="Two letter country code" />
        <br/><br/>
        <button type="submit">Submit</button>
    </form>

</div>
```

Kiểm tra dữ liệu hợp lệ với JavaScript

Ví dụ kiểm lỗi dữ liệu trên Form:

- Yêu cầu người dùng nhập vào mật khẩu có ít nhất 8 ký tự.

```
<h2>Attribute: pattern</h2>

<div style="border:1px solid #ddd; padding: 5px;">

    <form name="myForm" action="process-action.html">
        Password:
        <input type="password" name="password" pattern=".{8,}"
                title="8 or more characters" />
        <br/><br/>
        <button type="submit">Submit</button>
    </form>

</div>
```

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ Ví dụ kiểm lỗi dữ liệu trên Form:

◆ Yêu cầu người dùng nhập vào một mật khẩu mạnh, có ít nhất 8 ký tự, có ít nhất một chữ hoa (uppercase), và có ít nhất một chữ thường (lowercase).

```
<h2>Attribute: pattern</h2>
Password must contain 8 or more characters that are of at least one number,
and one uppercase and lowercase letter:
<br/><br/>

<div style="border:1px solid #ddd; padding: 5px;">

    <form name="myForm" action="process-action.html">
        Password:
        <input type="password" name="password"
                pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"
                title="Invalid password!" />
        <br/><br/>
        <button type="submit">Submit</button>
    </form>

</div>
```

Kiểm tra dữ liệu hợp lệ với JavaScript

❑ Ví dụ kiểm lỗi dữ liệu trên Form:

◆ Yêu cầu người dùng nhập vào địa chỉ **email**, sử dụng thuộc tính **pattern** để đảm bảo người dùng nhập vào một **email** đúng định dạng.

```
<h2>Attribute: pattern</h2>

<div style="border:1px solid #ddd; padding: 5px;">

    <form name="myForm" action="process-action.html">
        Email:
        <input type="password" name="password"
                pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$"
                title="Invalid password!" />
        <br/><br/>
        <button type="submit">Submit</button>
    </form>

</div>
```

Tổng kết nội dung bài học

- ❑ Tương tác sự kiện trong HTML với JavaScript
- ❑ Thuộc tính sự kiện trong HTML với JavaScript
- ❑ Gán các sự kiện trong HTML với JavaScript
- ❑ Các sự kiện trong HTML với JavaScript
- ❑ Lắng nghe sự kiện trong HTML với JavaScript
- ❑ Thêm nhiều trình xử lý sự kiện vào cùng một HTML
- ❑ Thêm một trình xử lý sự kiện vào đối tượng Window
- ❑ Các cách truyền sự kiện trong HTML với JavaScript
- ❑ Xóa sự kiện trong HTML với JavaScript
- ❑ Kiểm tra dữ liệu hợp lệ với JavaScript