



VIETNAM
AUSTRALIA
Vocational College

Slides 3.1 – Thay đổi nội dung HTML, định kiểu CSS với JS

Mentor: Nguyễn Bá Minh Đạo



Nội dung

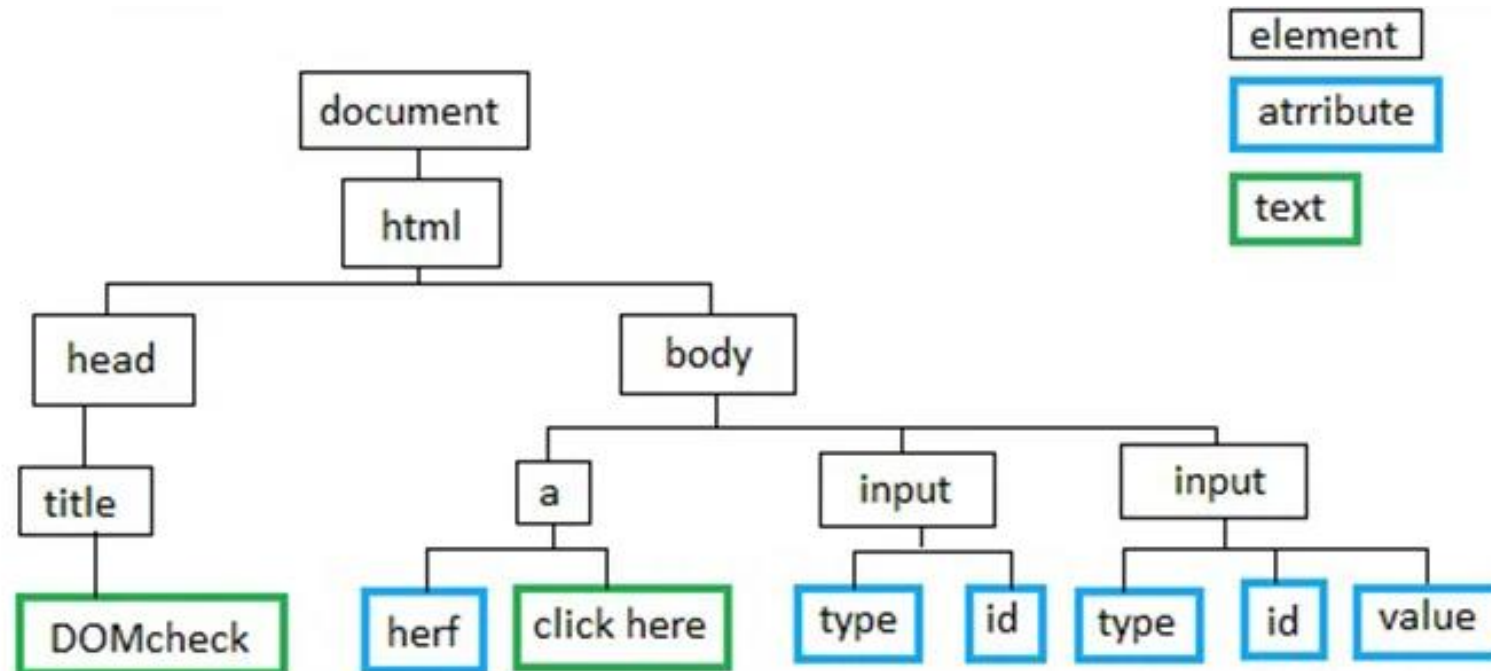
1. Tổng quan về HTML DOM
2. Đối tượng Document trong HTML DOM
3. Các phần tử Elements trong HTML DOM
4. Thay đổi nội dung HTML, định kiểu CSS với JS
5. Điều hướng HTML DOM với JavaScript
6. Tập hợp phần tử HTML DOM với JavaScript
7. Danh sách nút HTML DOM với JavaScript
8. Sự khác nhau giữa HTMLCollection và NodeList
9. Mô hình đối tượng BOM, Window của trình duyệt
10. Tóm tắt các phương thức phổ biến trong DOM



Tổng quan về HTML DOM

❑ Khái niệm về HTML DOM:

- ♦ Viết tắt của **D**ocument **O**bject **M**odel (DOM) – Mô hình đối tượng tài liệu.
- ♦ DOM là một tiêu chuẩn của W3C (World Wide Web Consortium).
- ♦ Khi một trang Web được tải lên, trình duyệt sẽ tạo ra một mô hình **DOM**.
- ♦ Mô hình **HTML DOM** được xây dựng bởi một cây phân tầng các đối tượng **Objects**.

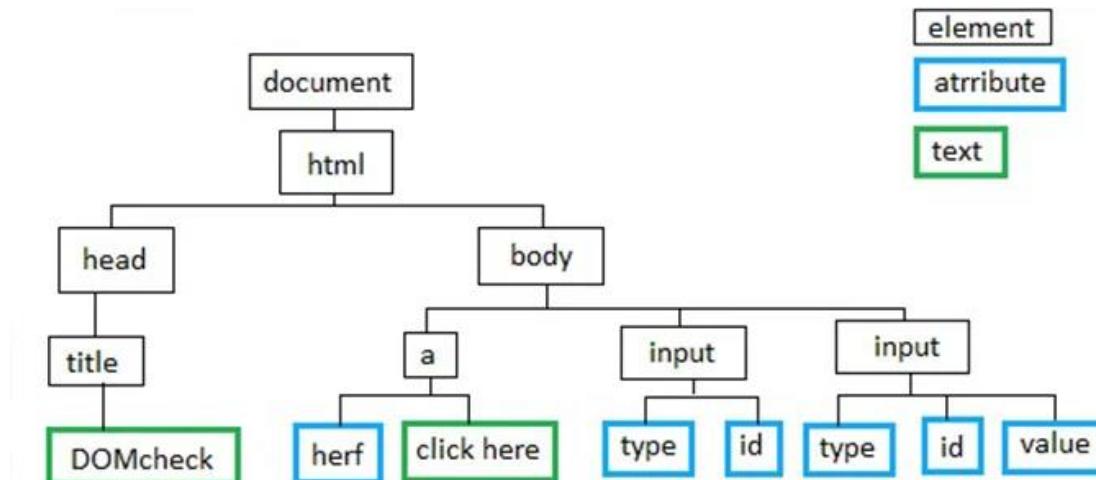




Tổng quan về HTML DOM

❑ Khái niệm về HTML DOM:

- ♦ Với DOM, JavaScript có thể **tạo ra các phần tử HTML động** (dynamic HTML) như sau:
 - JavaScript có thể **thay đổi các phần tử HTML** trong một trang Web.
 - JavaScript có thể **thay đổi các thuộc tính của phần tử HTML** trong 1 trang Web.
 - JavaScript có thể **thay đổi các khai báo định kiểu CSS** trong một trang Web.
 - JavaScript có thể **xóa bỏ các phần tử HTML và thuộc tính của phần tử HTML đang tồn tại** trong một trang Web.
 - JavaScript có thể **tạo mới các sự kiện trên các phần tử HTML** trong một trang Web.

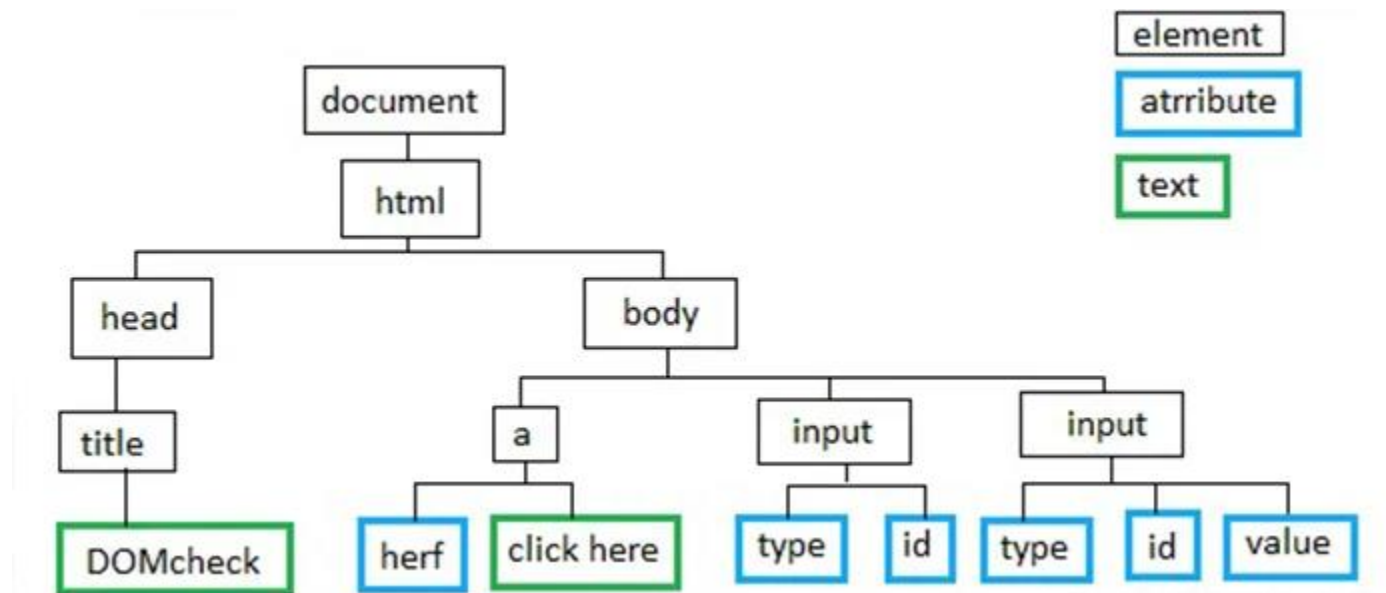




Đối tượng Document trong HTML DOM

❑ Khái niệm về đối tượng Document:

- ♦ Đối tượng **document** là **đối tượng gốc - cha (chủ sở hữu)** của tất cả các đối tượng khác trong trang **Web** của bạn.
 - Đối tượng tài liệu (**document**) đại diện cho trang Web của bạn.
 - Nếu bạn muốn truy cập bất kỳ phần tử nào trong trang HTML, bạn thường bắt đầu với việc truy xuất đối tượng **document**.





Các phần tử Elements trong HTML DOM

❑ Tìm kiếm các phần tử HTML:

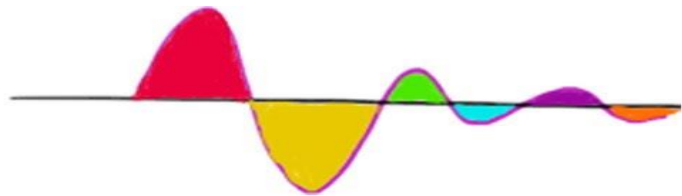
- ♦ Trong JavaScript, bạn muốn thao tác với các phần tử HTML.
- ♦ **Để thao tác**, trước tiên **ta** cần **phải tìm kiếm các phần tử HTML** ra **trước**. Chúng ta có một **vài cách để tìm kiếm** như sau:
 - Tìm các phần tử HTML thông qua **id**.
 - Tìm các phần tử HTML thông qua tên **tag**.
 - Tìm các phần tử HTML thông qua tên **class**.
 - Tìm các phần tử HTML thông qua các **bộ chọn CSS**.
 - Tìm các phần tử HTML thông qua các **collection objects của HTML**.



Các phần tử Elements trong HTML DOM

❑ Tìm kiếm các phần tử HTML:

- ♦ Tìm các phần tử HTML thông qua **id**.
 - Để tìm kiếm một phần tử HTML trong DOM đó là sử dụng **id** của phần tử đó.
 - Ví dụ: `var myElement = document.getElementById("intro");`
 - Nếu phần tử được tìm thấy, phương thức sẽ trả về phần tử đó dạng đối tượng (chứa trong biến **myElement**)
 - Nếu không tìm thấy phần tử đó, **myElement** sẽ chứa giá trị **null**.



JavaScript

getElementById()

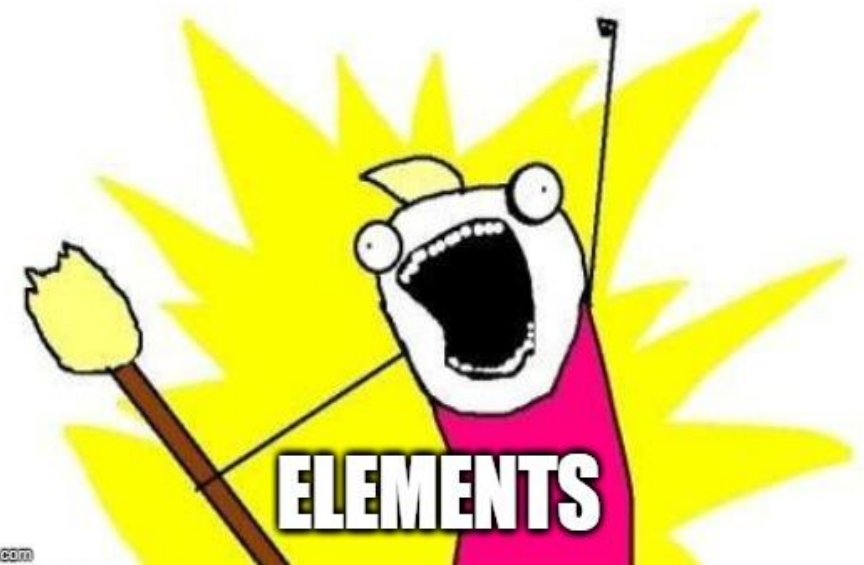


Các phần tử Elements trong HTML DOM

❑ Tìm kiếm các phần tử HTML:

- ♦ Tìm các phần tử HTML thông qua **tên class**.
 - Nếu bạn muốn tìm tất cả các phần tử HTML có cùng **tên class**, dùng phương thức **getElementsByClassName()**.
 - Ví dụ: `var myElement = document.getElementsByClassName("intro");`
 - Lưu ý: Cách tìm kiếm các phần tử thông qua tên class không làm việc được trên **Internet Explorer 8** và các phiên bản trước đó.

SELECT ALL THE





Các phần tử Elements trong HTML DOM

❑ Tìm kiếm các phần tử HTML:

- ♦ Tìm các phần tử HTML thông qua **các bộ chọn CSS**.
 - Nếu bạn muốn tìm tất cả các phần tử HTML khớp với **bộ chọn CSS** (id, tên class, kiểu, các thuộc tính, các giá trị của các thuộc tính, ...), dùng phương thức **querySelectorAll()**.
 - **Ví dụ:** Trả về danh sách tất cả các phần tử `<p>` với `class="intro"` ta làm như sau:
`var myElement = document.querySelectorAll ("intro");`
 - **Lưu ý:** Phương thức **querySelectorAll()** không làm việc được trên **Internet Explorer 8** và các phiên bản trước đó.

```
CSS Selectors
document.querySelector();
document.querySelectorAll();
```



Các phần tử Elements trong HTML DOM

❑ Tìm kiếm các phần tử HTML:

- ♦ Tìm các phần tử HTML thông qua **đối tượng tập hợp (collections)**.
 - **Ví dụ:** Tìm các phần tử **form** với **id="frm1"**, trong tập hợp các form và hiển thị giá trị của các phần tử.

The screenshot shows a web browser window and its source code. The browser window displays a form titled "Finding HTML Elements Using document.forms". The form has two text input fields: "First name:" with the value "Donald" and "Last name:" with the value "Duck". Below the inputs is a "Submit" button. A message below the form says "Click 'Try it' to display the value of each element in the form." and there is a "Try it" button. Below this, the values "Donald", "Duck", and "Submit" are displayed.

The source code on the left shows the HTML and JavaScript for this page. The HTML includes a form with id="frm1" and a button with onclick="myFunction()". The JavaScript defines a function myFunction() that iterates through the elements of the form and displays their values in the "demo" paragraph.

```
js_document_forms.html X
js_document_forms.html > html > body
1 <!DOCTYPE html>
2 <html>
3
4 <body>
5   <h2>Finding HTML Elements Using document.forms</h2>
6
7   <form id="frm1" action="/action_page.php">
8     First name: <input type="text" name="fname" value="Donald"><br>
9     Last name: <input type="text" name="lname" value="Duck"><br><br>
10    <input type="submit" value="Submit">
11  </form>
12
13  <p>Click "Try it" to display the value of each element in the form.</p>
14
15  <button onclick="myFunction()">Try it</button>
16
17  <p id="demo"></p>
18
19  <script>
20    function myFunction() {
21      var x = document.forms["frm1"];
22      var text = "";
23      var i;
24      for (i = 0; i < x.length; i++) {
25        text += x.elements[i].value + "<br>";
26      }
27      document.getElementById("demo").innerHTML = text;
28    }
29  </script>
30 </body>
31 </html>
```



Thay đổi nội dung HTML, định kiểu CSS với JS

❑ JavaScript có thể tạo ra nội dung động trong HTML:

♦ Trong JavaScript, lệnh **document.write()** có thể **được dùng để viết trực tiếp vào luồng xuất dữ liệu của HTML** (viết kết quả lên browser luôn).

• Lưu ý: Không dùng **document.write()** sau khi document đã được tải lên. Nó sẽ ghi đè hết tài liệu HTML trước đó.

The screenshot shows a code editor with a file named `js_document_write.html`. The code is as follows:

```
<?js_document_write.html X
<?js_document_write.html > html > body > script
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>JS Document Write</title>
5    </head>
6    <body>
7      <script>
8        document.write(Date());
9      </script>
10   </body>
11  </html>
```

Below the code editor, a web browser window is shown with the title "JS Document Write". The address bar displays the URL `127.0.0.1:5500/js_document_write.html`. The page content shows the date and time: "Wed Apr 07 2021 17:08:28 GMT+0700 (Indochina Time)".



Thay đổi nội dung HTML, định kiểu CSS với JS

❑ JavaScript có thể tạo ra nội dung động trong HTML:

- ♦ Để thay đổi nội dung của phần tử HTML, ta sử dụng thuộc tính **innerHTML**.

- Cú pháp:

document.getElementById(id).innerHTML = new HTML content;

- Ví dụ: Thay đổi nội dung của thẻ **<p>** với **id="greeting"** như đã làm thử.

The screenshot shows a code editor on the left and a web browser on the right. The code editor displays the following HTML and JavaScript code:

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html>
<html>
  <head>
    <title>JS Inner HTML</title>
  </head>
  <body>
    <p id="greeting">Xin chào FastTracker</p>
    <script>
      document.getElementById("greeting").innerHTML = "Xin chào VUSer";
    </script>
  </body>
</html>
```

The web browser on the right shows the title "JS Inner HTML" and the content "Xin chào VUSer", which is the result of the JavaScript code updating the innerHTML of the element with id="greeting".



Thay đổi nội dung HTML, định kiểu CSS với JS

❑ JavaScript có thể tạo ra nội dung động trong HTML:

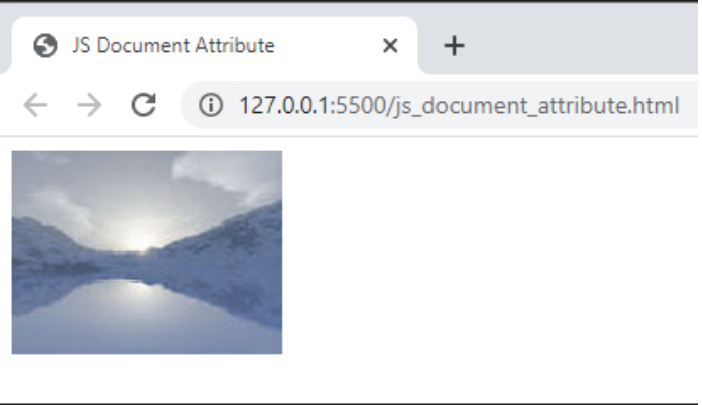
- ♦ Để thay đổi giá trị của thuộc tính của phần tử HTML, ta sử dụng **tên thuộc tính**.

- Cú pháp:

document.getElementById(id).[tên attribute] = new value;

- Ví dụ: Thay đổi nội dung của thuộc tính **src** của thẻ **** như sau:

```
js_document_attribute.html X
js_document_attribute.html > html > body > script
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>JS Document Attribute</title>
5   </head>
6   <body>
7     
8
9     <script>
10      document.getElementById("myImage").src = "https://www.w3schools.com/js/landscape.jpg";
11    </script>
12  </body>
13 </html>
```





Thay đổi nội dung HTML, định kiểu CSS với JS

❑ JavaScript có thể tạo ra nội dung động trong HTML:

♦ Để thay đổi kiểu dáng của phần tử HTML, ta sử dụng thuộc tính **style**.

- Cú pháp:

document.getElementById(id).style.[tên style] = new style value;

- Ví dụ 1: Thay đổi kiểu dáng của phần tử thẻ **<p>** như sau:

The screenshot shows a code editor on the left and a web browser on the right. The code editor displays the following HTML and JavaScript code:

```
js_document_style.html X
js_document_style.html > html > body > script
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>JS Document Style</title>
5   </head>
6   <body>
7     <p id="greeting">Hello VUSer</p>
8
9     <script>
10      document.getElementById("greeting").style.color = "blue";
11    </script>
12  </body>
13 </html>
```

The web browser on the right shows the title "JS Document Style" and the content "Hello VUSer" in blue text, demonstrating the effect of the JavaScript code.



Thay đổi nội dung HTML, định kiểu CSS với JS

❑ JavaScript có thể tạo ra nội dung động trong HTML:

- ♦ Để thay đổi kiểu dáng của phần tử HTML, ta sử dụng thuộc tính **style**.

- Cú pháp:

document.getElementById(id).style.[tên style] = new style value;

- Ví dụ 2: Thay đổi kiểu dáng của phần tử thẻ **<p>** như sau:

```
<> js_document_style_example_02.html X
<> js_document_style_example_02.html > html
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>JS Document Style</title>
5    </head>
6    <body>
7      <p id="greeting">Hello VUSer</p>
8
9      <input type="button" value="Ẩn chữ đi"
10         onclick="document.getElementById('greeting').style.visibility='hidden'">
11      <input type="button" value="Hiện chữ lại"
12         onclick="document.getElementById('greeting').style.visibility='visible'">
13    </body>
14  </html>
```



Thay đổi nội dung HTML, định kiểu CSS với JS

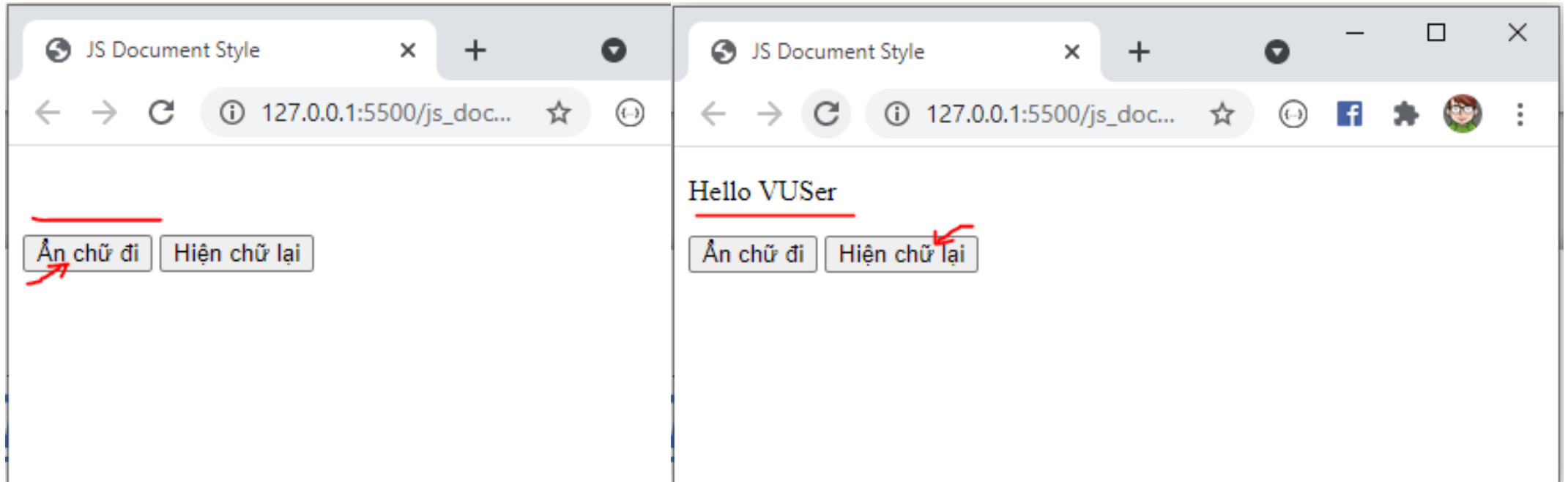
❑ JavaScript có thể tạo ra nội dung động trong HTML:

- ♦ Để thay đổi kiểu dáng của phần tử HTML, ta sử dụng thuộc tính **style**.

- Cú pháp:

`document.getElementById(id).style.[tên style] = new style value;`

- Ví dụ 2: Thay đổi kiểu dáng của phần tử thẻ **<p>** như sau:

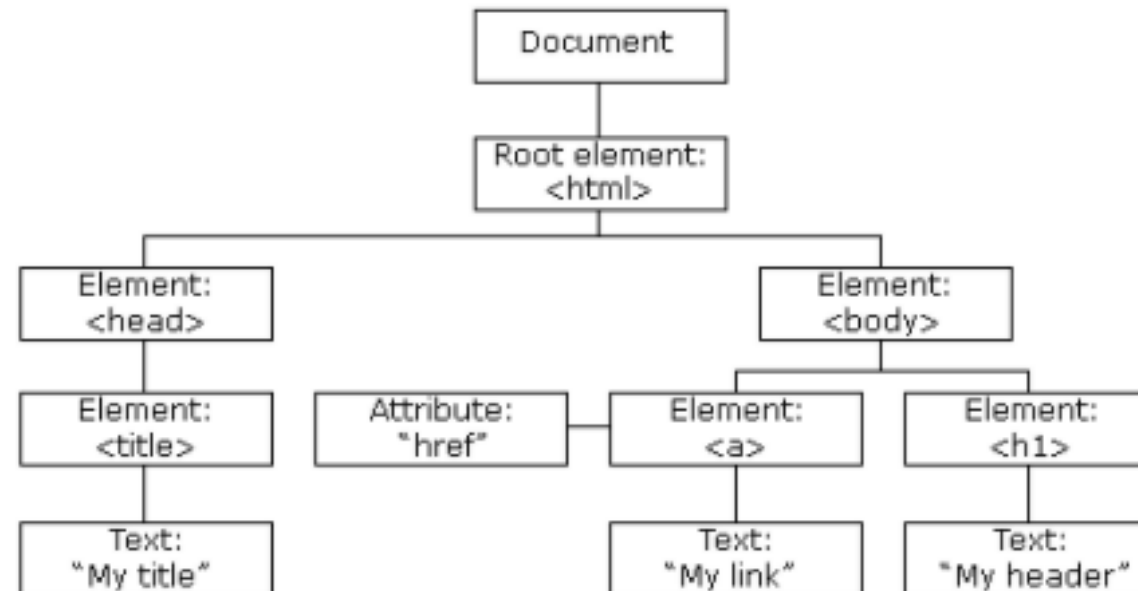




Điều hướng HTML DOM với JavaScript

❑ Các nút DOM (DOM Nodes):

- ♦ Theo tiêu chuẩn **W3C HTML DOM**, tất cả mọi thứ trong **HTML** được coi như là 1 nút:
 - Toàn bộ tài liệu HTML là một nút tài liệu. (thẻ **<html>** là một nút- **node**)
 - Mỗi phần tử HTML là một nút phần tử (**element node**).
 - Chữ trong các phần tử HTML là các nút chữ (**text nodes**).
 - Tất cả ghi chú đều là các nút ghi chú (**comment nodes**).





Điều hướng HTML DOM với JavaScript

❑ Quan hệ các nút (Node Relationships):

- ◆ Các nút trong cây các nút (**node tree**) có mối quan hệ thứ bậc với các nút còn lại.
- ◆ Một số khái niệm về cha (**parent**), con (**child**), và anh chị em ruột (**sibling**) thường được dùng để mô tả các mối quan hệ.
 - Trong một cây các nút (**node tree**), nút cao nhất (**top node**) được gọi là gốc (**root**) hoặc nút gốc (**root node**).
 - Mỗi nút đều có chính xác 1 nút cha, ngoại trừ nút gốc (không có nút cha).
 - Một nút có thể có nhiều nút con.
 - Anh chị em ruột (**siblings – brothers** hoặc **sisters**) là các nút có cùng nút cha.



Điều hướng HTML DOM với JavaScript

❑ Quan hệ các nút (Node Relationships):

- ♦ Từ đoạn mã HTML ở hình bên, ta có các quan hệ thứ bậc:

```
<html>
```

```
<head>
```

```
<title>DOM Tutorial</title>
```

```
</head>
```

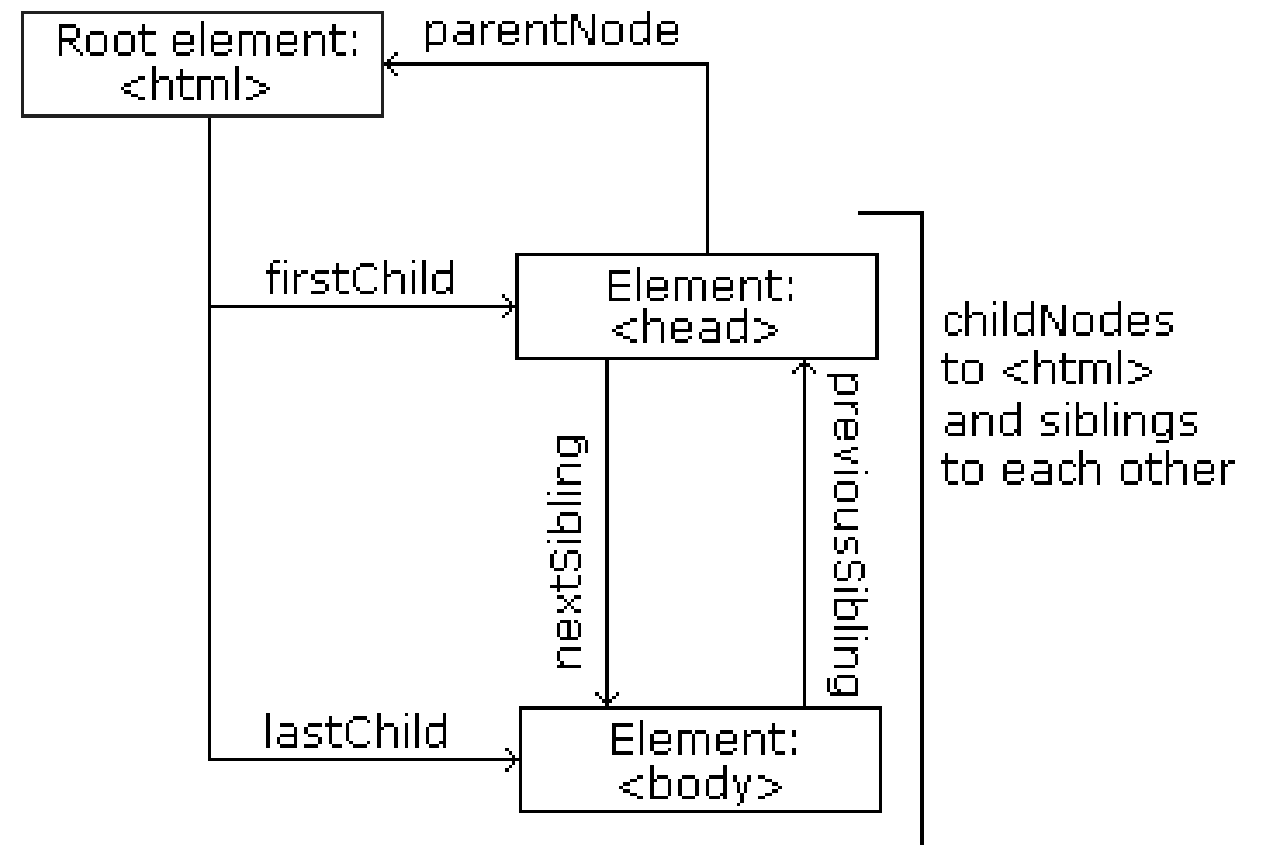
```
<body>
```

```
<h1>DOM Lesson one</h1>
```

```
<p>Hello world!</p>
```

```
</body>
```

```
</html>
```



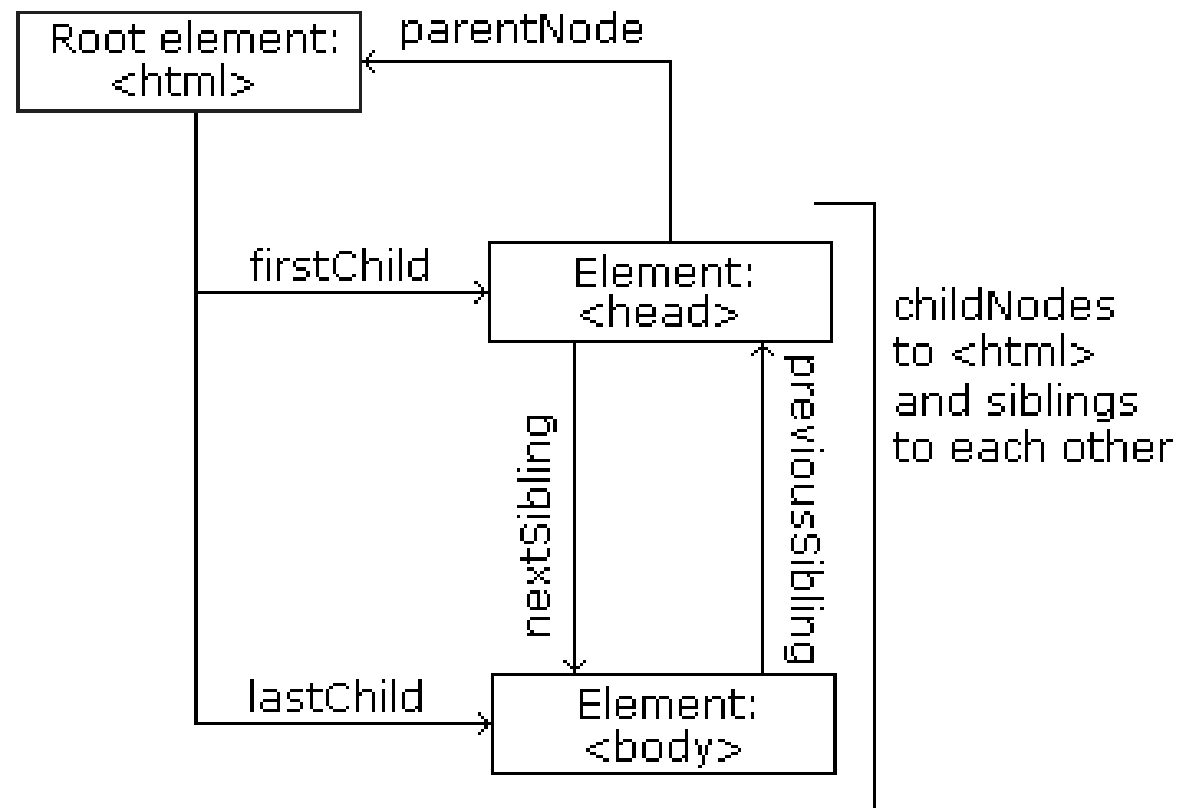


Điều hướng HTML DOM với JavaScript

❑ Điều hướng giữa các nút:

♦ Bạn có thể theo dõi các thuộc tính của nút để điều hướng (chuyển qua lại) giữa các nút với JavaScript:

- **parentNode**
- **childNodes[nodenum]**
- **firstChild**
- **lastChild**
- **nextSibling**
- **previousSibling**



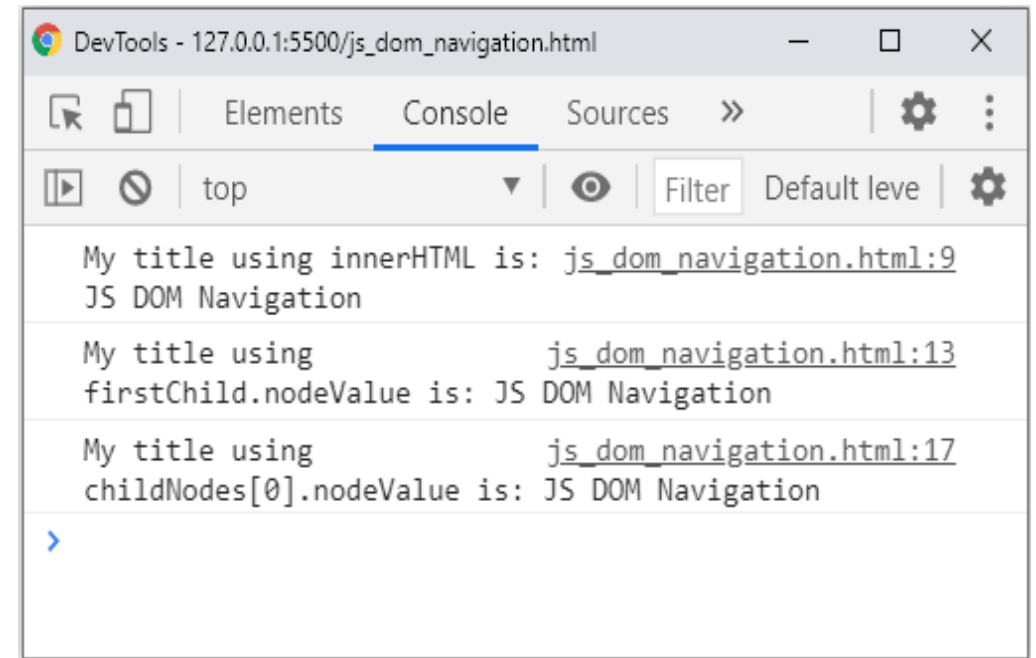


Điều hướng HTML DOM với JavaScript

❑ Các nút con và các giá trị của nút:

- ◆ Ví dụ: Lấy giá trị phần tử đầu tiên của thẻ **<title>**

```
js_dom_navigation.html X
js_dom_navigation.html > html > body > script
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title id="title">JS DOM Navigation</title>
5   </head>
6   <body>
7     <script>
8       var myTitle1 = document.getElementById("title").innerHTML;
9       console.log("My title using innerHTML is: " + myTitle1);
10
11       // Cách 1: Truy cập giá trị nút của nút con đầu tiên
12       var myTitle2 = document.getElementById("title").firstChild.nodeValue;
13       console.log("My title using firstChild.nodeValue is: " + myTitle2);
14
15       // Cách 2: Truy cập giá trị nút của nút con đầu tiên
16       var myTitle3 = document.getElementById("title").childNodes[0].nodeValue;
17       console.log("My title using childNodes[0].nodeValue is: " + myTitle3);
18     </script>
19   </body>
20 </html>
```





Tập hợp phần tử HTML DOM trong JavaScript

❑ Tạo mới phần tử nút HTML:

♦ Để thêm một phần tử mới vào phần tử cuối cùng của phần tử cha, trước tiên bạn phải tạo phần tử (nút phần tử), sau đó nối phần tử đó vào một phần tử hiện có.

```
<body>
  <div id="div1">
    <p id="p1">Đây là một đoạn văn bản.</p>
    <p id="p2">Đây là một đoạn văn bản khác.</p>
  </div>

  <script>
    var para = document.createElement("p");
    var node = document.createTextNode("Đây là một đoạn văn mới.");
    para.appendChild(node);
    var element = document.getElementById("div1");
    element.appendChild(para);
  </script>
</body>
```



Tập hợp phần tử HTML DOM trong JavaScript

❑ Tạo mới phần tử nút HTML:

◆ Để thêm một phần tử mới vào phần tử cuối cùng của phần tử cha, trước tiên bạn phải tạo phần tử (nút phần tử), sau đó nối phần tử đó vào một phần tử hiện có.

The screenshot shows a web browser window on the left and the Chrome DevTools 'Elements' panel on the right. The browser window displays three paragraphs of text: 'Đây là một đoạn văn bản.', 'Đây là một đoạn văn bản khác.', and 'Đây là một đoạn văn mới.' The DevTools 'Elements' panel shows the DOM tree. The root is `<!DOCTYPE html>`, followed by `<html>`, `<head>...</head>`, and `<body>...</body>`. Inside the body, there is a `<div id="div1">` element. This div contains three `<p>` elements: `<p id="p1">Đây là một đoạn văn bản.</p>`, `<p id="p2">Đây là một đoạn văn bản khác.</p>`, and `<p>Đây là một đoạn văn mới.</p>`. The third paragraph is highlighted in blue, indicating it is the selected element in the DOM tree.



Tập hợp phần tử HTML DOM trong JavaScript

❑ Tạo mới phần tử nút HTML:

♦ Giải thích ví dụ trên:

- **var** para = **document.createElement**("p"); // Tạo một phần tử **<p>** mới

- **var** node = **document.createTextNode**("Đây là đoạn văn bản mới.");

// Để tạo nội dung chữ cho phần tử **<p>**, bạn phải tạo một nút chữ trước.

// Dòng mã này dùng để tạo một nút chữ.

- para.**appendChild**(node); // Sau đó, bạn phải nối nút chữ vào phần tử **<p>**

- **var** element = **document.getElementById**("div1");

// Cuối cùng, bạn phải nối phần tử mới với một phần tử đang tồn tại.

// Dòng code này để tìm một phần tử đang tồn tại.

- element.**appendChild**(para);

// Dòng code này để nối phần tử mới với một phần tử đang tồn tại vừa tìm ra.



Danh sách nút HTML DOM với JavaScript

❑ Tạo mới phần tử nút HTML:

♦ Để thêm một phần tử mới vào trước phần tử bất kỳ của phần tử cha, ta có thể dùng phương thức **insertBefore()** như ví dụ sau.

```
<body>
  <div id="div1">
    <p id="p1">Đây là một đoạn văn bản.</p>
    <p id="p2">Đây là một đoạn văn bản khác.</p>
  </div>

  <script>
    var para = document.createElement("p");
    var node = document.createTextNode("Đây là một đoạn văn mới.");
    para.appendChild(node);

    var element = document.getElementById("div1");
    var child = document.getElementById("p1");
    element.insertBefore(para, child);
  </script>
</body>
```



Danh sách nút HTML DOM với JavaScript

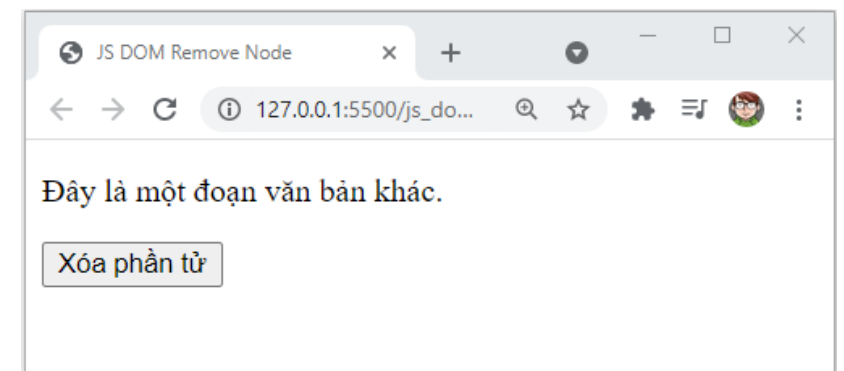
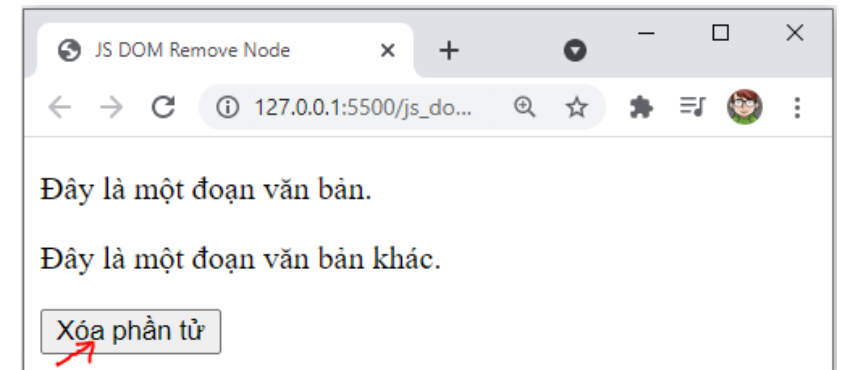
❑ Xóa phần tử nút HTML:

♦ Để xóa một phần tử đang tồn tại trên một trang Web, ta có thể dùng phương thức **remove()** như ví dụ sau.

```
<body>
  <div>
    <p id="p1">Đây là một đoạn văn bản.</p>
    <p id="p2">Đây là một đoạn văn bản khác.</p>
  </div>

  <button onclick="myFunction()">Xóa phần tử</button>

  <script>
    function myFunction() {
      var element = document.getElementById("p1");
      element.remove();
    }
  </script>
</body>
```





Danh sách nút HTML DOM với JavaScript

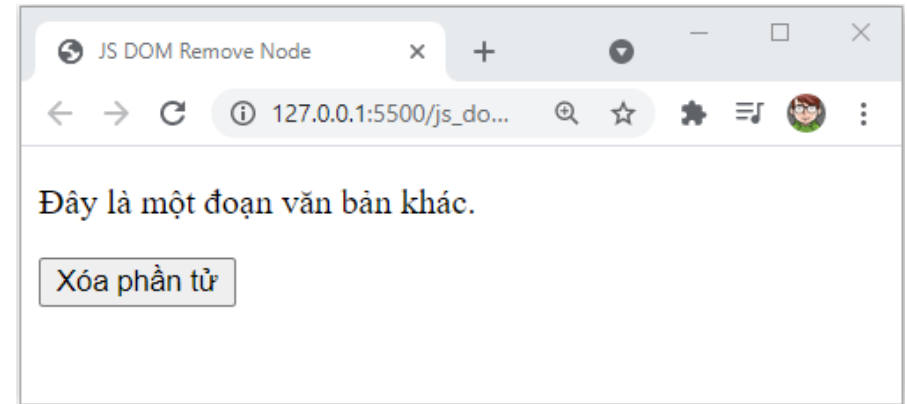
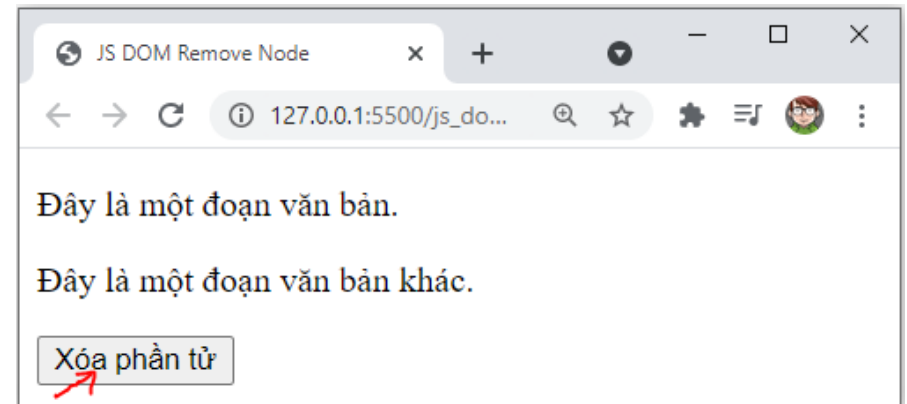
❑ Xóa phần tử nút HTML:

♦ Ở một số trình duyệt cũ, để xóa phần tử nút, bạn có thể tìm nút cha và dùng phương thức **removeChild()** để xóa phần tử con như ví dụ sau.

```
<body>
  <div id="div1">
    <p id="p1">Đây là một đoạn văn bản.</p>
    <p id="p2">Đây là một đoạn văn bản khác.</p>
  </div>

  <button onclick="myFunction()">Xóa phần tử</button>

  <script>
    function myFunction() {
      var parent = document.getElementById("div1");
      var child = document.getElementById("p1");
      parent.removeChild(child);
    }
  </script>
</body>
```





Danh sách nút HTML DOM với JavaScript

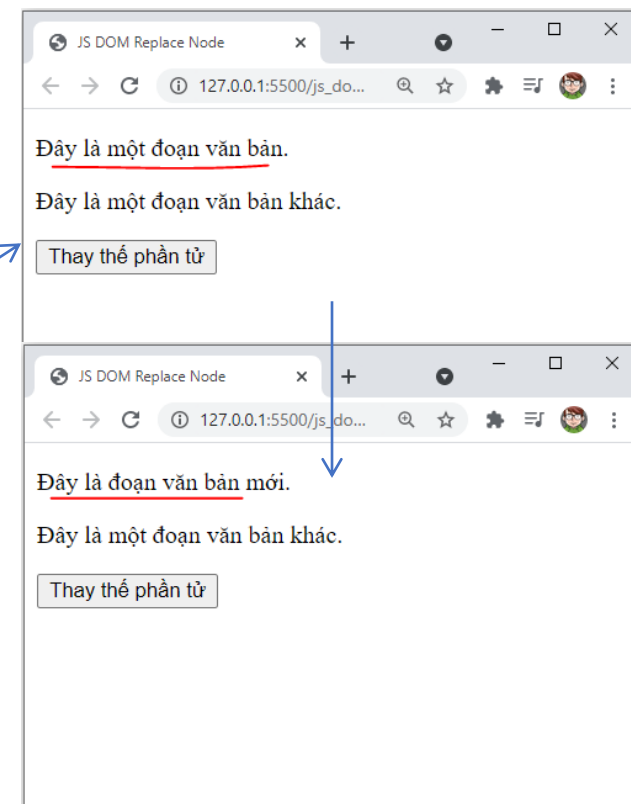
❑ Thay thế phần tử nút HTML:

◆ Để thay thế một phần tử đang tồn tại trên một trang Web, ta có thể dùng phương thức **replaceChild()** như ví dụ sau.

```
<body>
  <div id="div1">
    <p id="p1">Đây là một đoạn văn bản.</p>
    <p id="p2">Đây là một đoạn văn bản khác.</p>
  </div>

  <button onclick="myFunction()">Thay thế phần tử</button>

  <script>
    function myFunction() {
      var parent = document.getElementById("div1");
      var child = document.getElementById("p1");
      var para = document.createElement("p");
      var node = document.createTextNode("Đây là đoạn văn bản mới.");
      para.appendChild(node);
      parent.replaceChild(para, child);
    }
  </script>
</body>
```





Danh sách nút HTML DOM với JavaScript

❑ Đối tượng HTMLCollection:

- ♦ Phương thức **getElementsByTagName()** trả về một đối tượng **HTMLCollection**.
- ♦ Đối tượng **HTMLCollection** là một danh sách (tập hợp) các phần tử HTML **giống như một mảng**.
- ♦ Đoạn mã sau đây chọn tất cả các phần tử **<p>** trong một **document**:
 - Các phần tử trong **Collection** có thể được truy cập bằng chỉ mục.
 - Để truy cập phần tử **<p>** thứ hai, bạn có thể viết: **y = x[1];**
 - **Lưu ý:** Chỉ mục bắt đầu từ 0.

```
<body>
  <p>Xin chào VUSer!</p>
  <p>Xin chào FastTracker!</p>
  <p id="demo"></p>

  <script>
    var myCollection = document.getElementsByTagName("p");
    document.getElementById("demo").innerHTML =
      "Nội dung innerHTML của đoạn văn bản thứ 2 là: " +
      myCollection[1].innerHTML;
  </script>
</body>
```



Danh sách nút HTML DOM với JavaScript

❑ Số lượng phần tử của HTMLCollection:

- ◆ Thuộc tính **length** xác định số lượng phần tử trong **HTMLCollection**.
- ◆ Đoạn mã hàm myFunction() sau đếm số lượng các phần tử **<p>** trong **document**.
- ◆ Thuộc tính **length** rất hữu ích khi bạn muốn lặp qua các phần tử trong một tập hợp.

```
<body>
  <p>Xin chào VUSer!</p>
  <p>Xin chào FastTracker!</p>
  <p id="demo"></p>

  <script>
    var myCollection = document.getElementsByTagName("p");
    document.getElementById("demo").innerHTML =
      "document này chứa " + myCollection.length + " đoạn văn bản.";
  </script>
</body>
```

```
<body>
  <p>Xin chào VUSer!</p>
  <p>Xin chào FastTracker!</p>
  <p>Nhấp vào nút để thay đổi màu của tất cả các phần tử p.</p>

  <button onclick="myFunction()">Bấm đi, tui tô màu đỏ cho coi!</button>

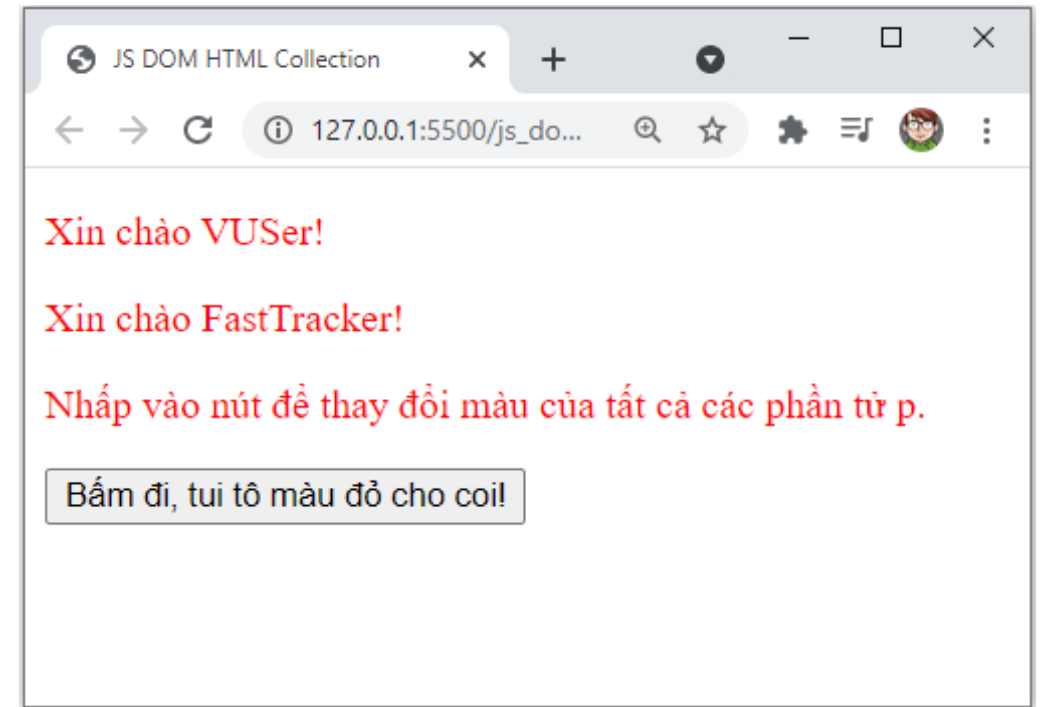
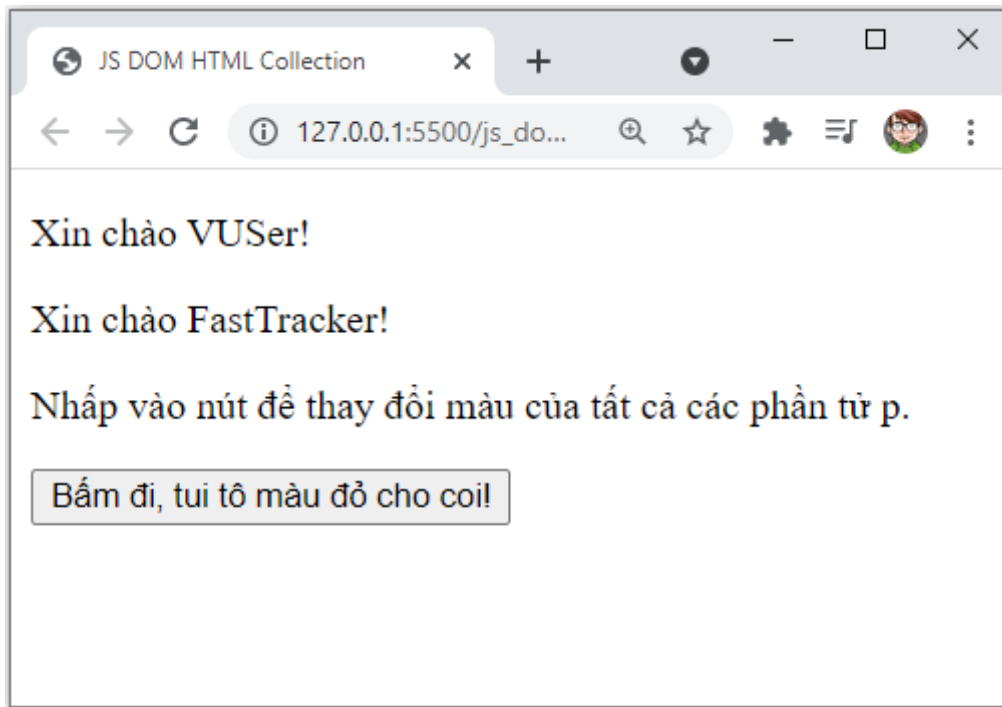
  <script>
    function myFunction() {
      var myCollection = document.getElementsByTagName("p");
      var i;
      for (i = 0; i < myCollection.length; i++) {
        myCollection[i].style.color = "red";
      }
    }
  </script>
</body>
```



Danh sách nút HTML DOM với JavaScript

❑ Số lượng phần tử của HTMLCollection:

♦ Thuộc tính **length** rất hữu ích khi bạn muốn lặp qua các phần tử trong một tập hợp:



♦ Lưu ý: **HTMLCollection** có thể trông giống mảng, nhưng nó không phải là mảng.



Danh sách nút HTML DOM với JavaScript

❑ Đối tượng NodeList:

- ◆ Đối tượng **NodeList** là một danh sách (tập hợp) các nút được trích xuất từ một **document**.
- ◆ Một đối tượng **NodeList** gần giống như một đối tượng **HTMLCollection**.
- ◆ Một số trình duyệt (cũ hơn) trả về đối tượng **NodeList** thay vì **HTMLCollection** cho các phương thức như **getElementsByClassName()**.
- ◆ Tất cả các trình duyệt trả về một đối tượng **NodeList** cho thuộc tính **childNodes**.
- ◆ Hầu hết các trình duyệt trả về một đối tượng **NodeList** cho phương thức **querySelectorAll()**.



Danh sách nút HTML DOM với JavaScript

❑ Đối tượng NodeList:

- ◆ Đoạn mã sau chọn tất cả các nút **<p>** trong một document.
- ◆ Thuộc tính **length** xác định số lượng phần tử trong NodeList.
- ◆ Đoạn mã sau đây đếm số lượng các phần tử **<p>** trong một document.

```
<body>
  <p>Xin chào VUSer!</p>
  <p>Xin chào FastTracker!</p>
  <p id="demo"></p>

  <script>
    var myNodeList = document.querySelectorAll("p");
    document.getElementById("demo").innerHTML =
      "Nội dung innerHTML của đoạn văn bản thứ 2 là: " +
      myNodeList[1].innerHTML;
  </script>
</body>
```

```
<body>
  <p>Xin chào VUSer!</p>
  <p>Xin chào FastTracker!</p>
  <p id="demo"></p>

  <script>
    var myNodeList = document.querySelectorAll("p");
    document.getElementById("demo").innerHTML =
      "document này chứa " + myNodeList.length + " đoạn văn bản.";
  </script>
</body>
```

```
<body>
  <p>Xin chào VUSer!</p>
  <p>Xin chào FastTracker!</p>
  <p>Nhấp vào nút để thay đổi màu của tất cả các phần tử p.</p>

  <button onclick="myFunction()">Bấm đi, tui tô màu đỏ cho coi!</button>

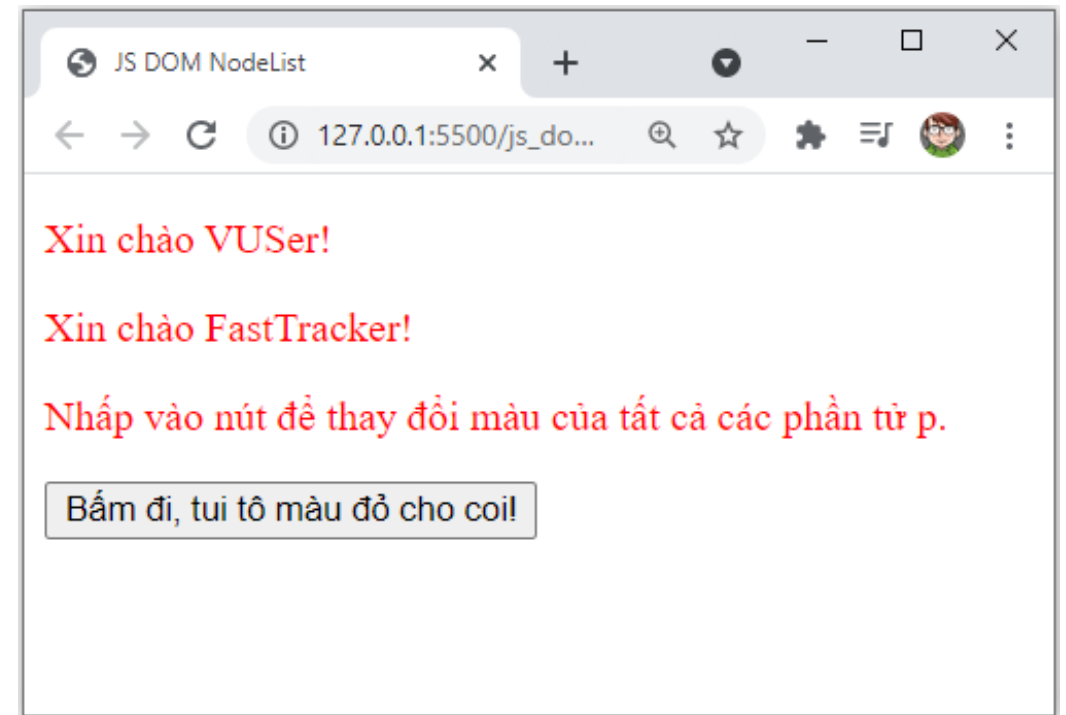
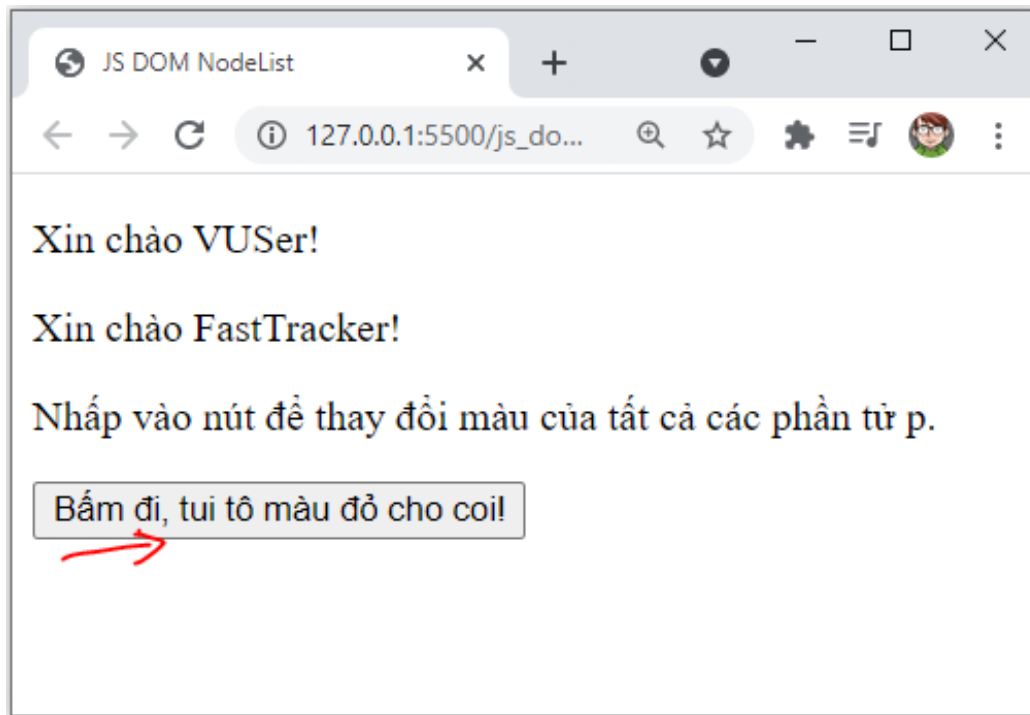
  <script>
    function myFunction() {
      var myNodeList = document.querySelectorAll("p");
      var i;
      for (i = 0; i < myNodeList.length; i++) {
        myNodeList[i].style.color = "red";
      }
    }
  </script>
</body>
```



Danh sách nút HTML DOM với JavaScript

❑ Số lượng phần tử của NodeList:

♦ Thuộc tính **length** rất hữu ích khi bạn muốn lặp qua các phần tử trong một tập hợp các phần tử HTML:



♦ Lưu ý: **NodeList** có thể trông giống mảng, nhưng nó không phải là mảng.



Sự khác nhau giữa HTMLCollection và NodeList

❑ HTMLCollection vs NodeList:

- ♦ **HTMLCollection** là một tập hợp **các phần tử HTML**.
- ♦ **NodeList** là một tập hợp **các nút phần tử HTML**.
- ♦ Các mục **HTMLCollection** có thể được **truy cập bằng tên, id hoặc số chỉ mục** của chúng.

```
<ul id="myList">
  <!-- List items -->
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
  <li>List item 4</li>
  <li>List item 5</li>
</ul>
```

Whitespace: `<ul id="myList"> List item`

No whitespace: `<ul id="myList">List item`

```
► HTMLCollection(10) [li, li, li, li, li, li, li, li, li, li] test.html:24
NodeList(23) [text, comment, text, li, text, li, text, li, text, li, text, li, text, li, text, li, text, li, text, li, text, li] test.html:25
```



Sự khác nhau giữa HTMLCollection và NodeList

❑ HTMLCollection vs NodeList:

- ♦ Các mục **NodeList** chỉ có thể được **truy cập bằng số chỉ mục** của chúng.
- ♦ Chỉ đối tượng **NodeList** mới có thể chứa các **nút thuộc tính** và **nút văn bản**.

```
<ul id="myList">
  <!-- List items -->
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
  <li>List item 4</li>
  <li>List item 5</li>
</ul>
```

Whitespace: `<ul id="myList"> List item`

No whitespace: `<ul id="myList">List item`

► `HTMLCollection(10) [li, li, li, li, li, li, li, li, li, li]` `test.html:24`

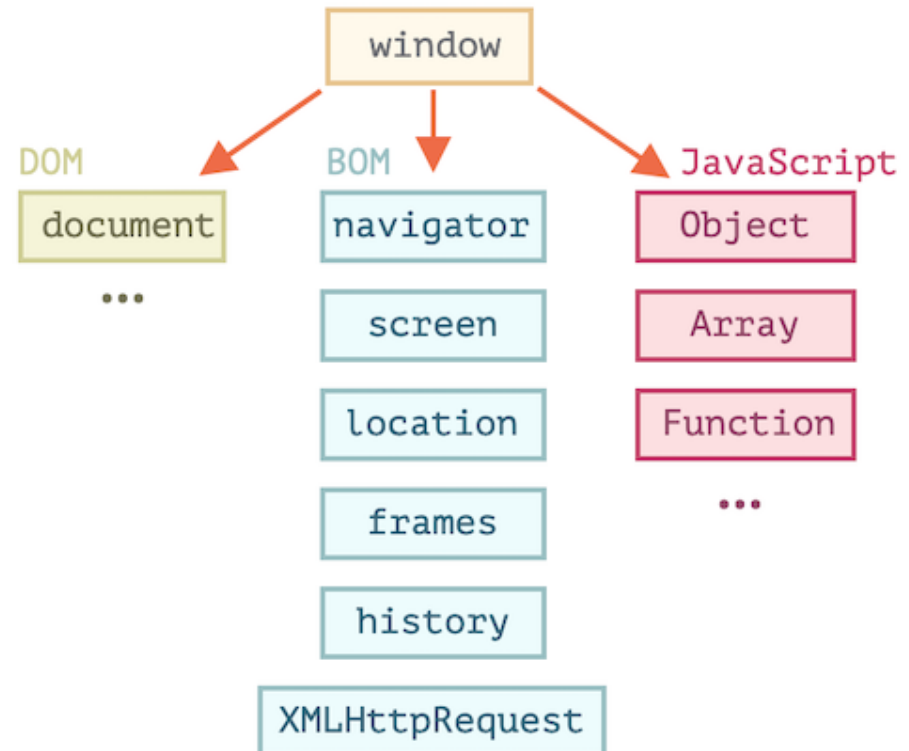
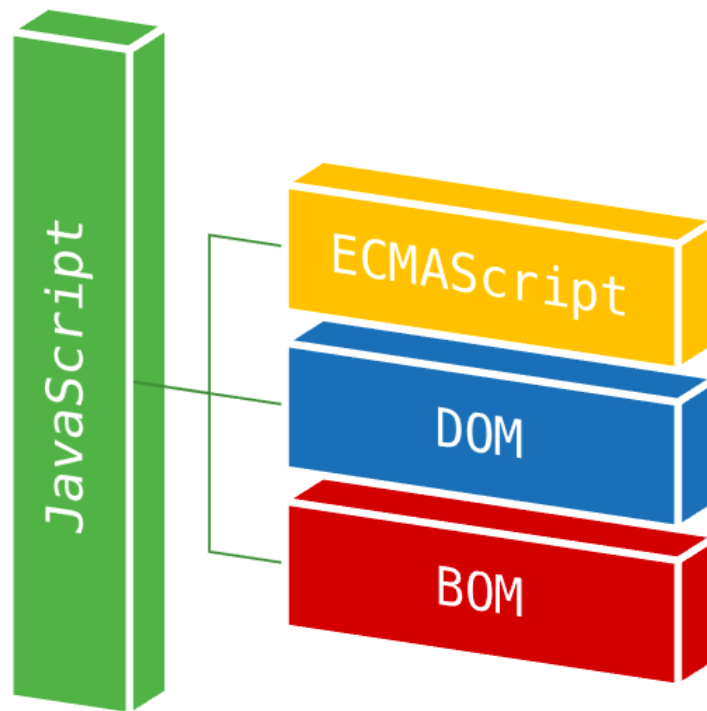
► `NodeList(23) [text, comment, text, li, text, li, text, li, text, li, text, li, text, li, text, li, text, li, text, li, text]` `test.html:25`



Mô hình đối tượng BOM của trình duyệt

❑ BOM là gì?

♦ Mô hình đối tượng trình duyệt (BOM) cho phép JavaScript "nói chuyện với" trình duyệt thông qua một số đối tượng phổ biến như: navigator, screen, location,...





Mô hình đối tượng Window của trình duyệt

❑ Đối tượng window:

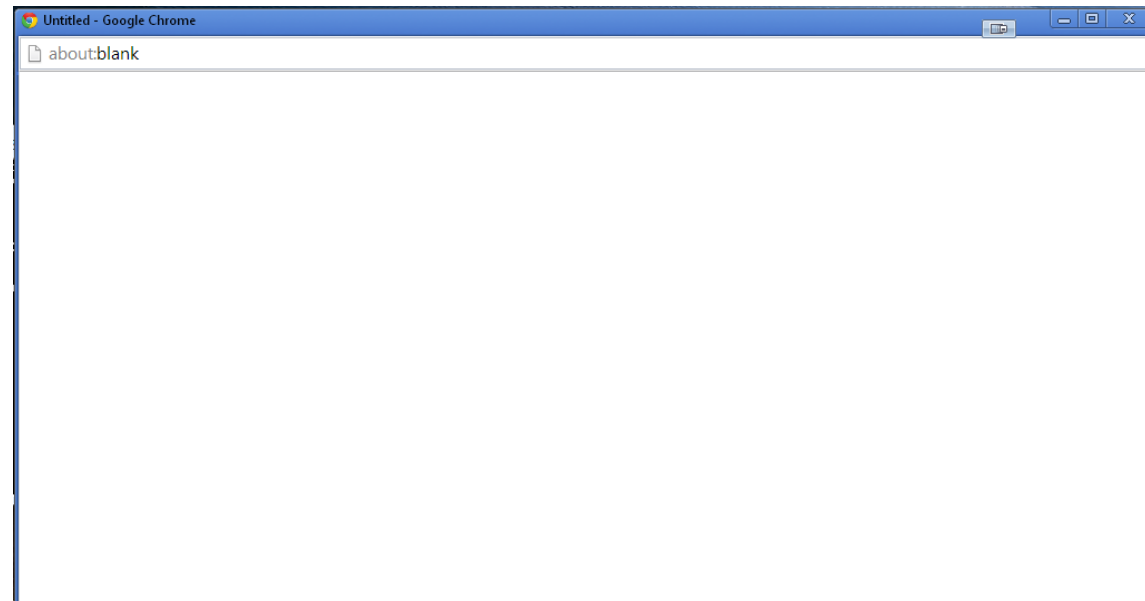
- ◆ Đối tượng **window** được hỗ trợ bởi tất cả các trình duyệt. Nó đại diện cho cửa sổ của trình duyệt.
- ◆ Tất cả các **đối tượng**, **hàm** và **biến JavaScript toàn cục** sẽ tự động **trở thành thành viên của đối tượng window**.
- ◆ Các biến toàn cục là thuộc tính của đối tượng **window**.
- ◆ Các hàm toàn cục là các phương thức của đối tượng **window**.
- ◆ Ngay cả đối tượng **document** (của **HTML DOM**) cũng là một thuộc tính của đối tượng **window**.
 - **window.document.getElementById**("header");
 - Tương tự với: **document.getElementById**("header");



Mô hình đối tượng Window của trình duyệt

❑ Các phương thức của window:

- ♦ Một số phương thức khác của đối tượng **window**.
 - **window.open()**: Mở một cửa sổ mới.
 - **window.close()**: Đóng cửa sổ hiện tại.
 - **window.moveTo()**: Di chuyển cửa sổ hiện tại.
 - **window.resizeTo()**: Điều chỉnh lại kích thước của sổ hiện tại.





Tóm tắt các phương thức phổ biến trong DOM

❑ Một số thao tác thường dùng với đối tượng document:

◆ Tìm kiếm các phần tử HTML:

Phương thức	Mô tả
<code>document.getElementById(id)</code>	Tìm kiếm phần tử thông qua id của phần tử đó.
<code>document.getElementsByTagName(name)</code>	Tìm kiếm các phần tử thông qua tên các phần tử đó.
<code>document.getElementsByClassName(name)</code>	Tìm kiếm các phần tử thông qua tên class các phần tử đó.

◆ Thay đổi các phần tử HTML:

Phương thức	Mô tả
<code>element.innerHTML = new html content</code>	Thay đổi nội dung của một phần tử HTML.
<code>element.attribute = new value</code>	Thay đổi giá trị thuộc tính của một phần tử HTML.
<code>element.style.property = new style</code>	Thay đổi phong cách (style) của một phần tử HTML.
Thuộc tính	Mô tả
<code>element.setAttribute(attribute, value)</code>	Thay đổi giá trị thuộc tính của một phần tử HTML.



Tóm tắt các phương thức phổ biến trong DOM

❑ Một số thao tác thường dùng với đối tượng document:

◆ Thêm và xóa các phần tử:

Phương thức	Mô tả
<code>document.createElement (element)</code>	Tạo một phần tử HTML.
<code>document.removeChild(element)</code>	Xóa một phần tử HTML.
<code>document.appendChild(element)</code>	Thêm một phần tử HTML.
<code>document.replaceChild(new, old)</code>	Thay thế một phần tử HTML.
<code>document.write(text)</code>	Viết vào trong luồng xuất dữ liệu của HTML. (viết lên giao diện browser)

◆ Thêm các xử lý sự kiện:

Phương thức	Mô tả
<code>document.getElementById (id).onclick = function(){code}</code>	Thêm mã xử lý sự kiện vào sự kiện onclick.



Tổng kết nội dung bài học

- ☐ Tổng quan về HTML DOM
- ☐ Đối tượng Document trong HTML DOM
- ☐ Các phần tử Elements trong HTML DOM
- ☐ Thay đổi nội dung HTML, định kiểu CSS với JS
- ☐ Điều hướng HTML DOM với JavaScript
- ☐ Tập hợp phần tử HTML DOM với JavaScript
- ☐ Danh sách nút HTML DOM với JavaScript
- ☐ Sự khác nhau giữa HTMLCollection và NodeList
- ☐ Mô hình đối tượng BOM, Window của trình duyệt
- ☐ Tóm tắt các phương thức phổ biến trong DOM

