

Theory 03

JavaScript Fundamentals

WEEK 01

Nội dung chính

❑ TỔNG QUAN VỀ JAVASCRIPT	03
❑ TỔNG QUAN DOM, ELEMENT	04
❑ QUAN HỆ TRONG HTML DOM	06
❑ HTML COLLECTION, NODE LIST	09
❑ XỬ LÝ HTML, CSS VỚI JS	10
❑ XỬ LÝ SỰ KIỆN VỚI JS	21
❑ LẮNG NGHE SỰ KIỆN VỚI JS	26
❑ XÁC THỰC DỮ LIỆU VỚI JS	30

AGENDA



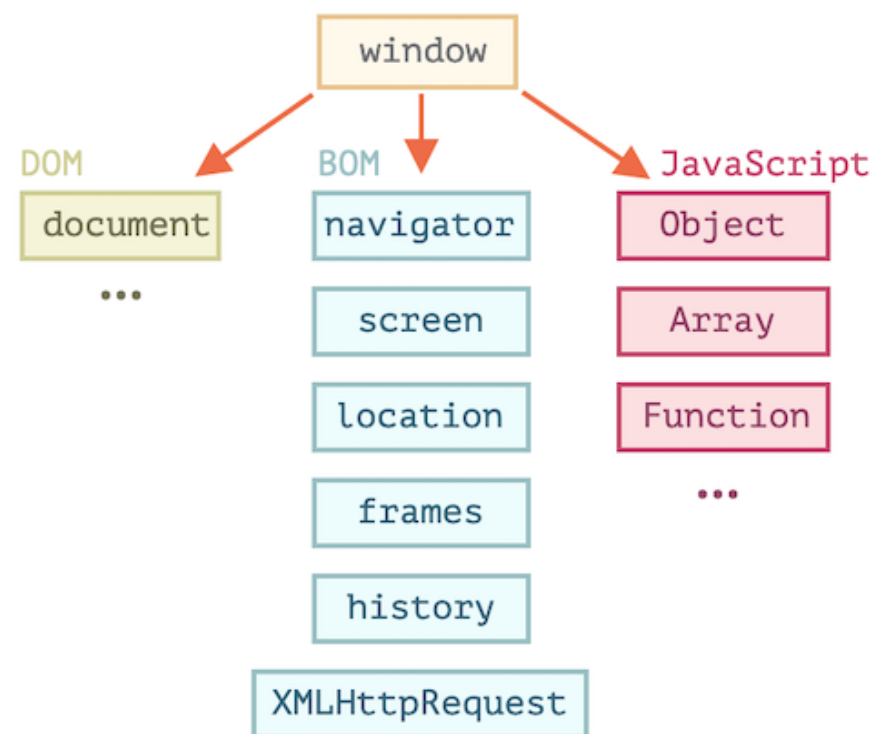
Tổng quan về JavaScript

- ❑ **JavaScript (JS)** là **ngôn ngữ lập trình** được sử dụng để:
 - Thay đổi nội dung, giao diện của trang Web
 - Tăng tính tương tác trực quan của trang Web
- ❑ Có 3 cách chính chèn mã JavaScript vào HTML như sau:
 - **Inline JavaScript**: Viết mã **Javascript** trực tiếp trên thẻ **HTML**
 - **Internal JavaScript**: Viết mã **JavaScript** bên trong cặp thẻ **<script>**
 - **External JavaScript**: Sử dụng file **.js** từ bên ngoài và nhúng vào trang **HTML**

Tổng quan về DOM, Element

❑ Đối tượng window:

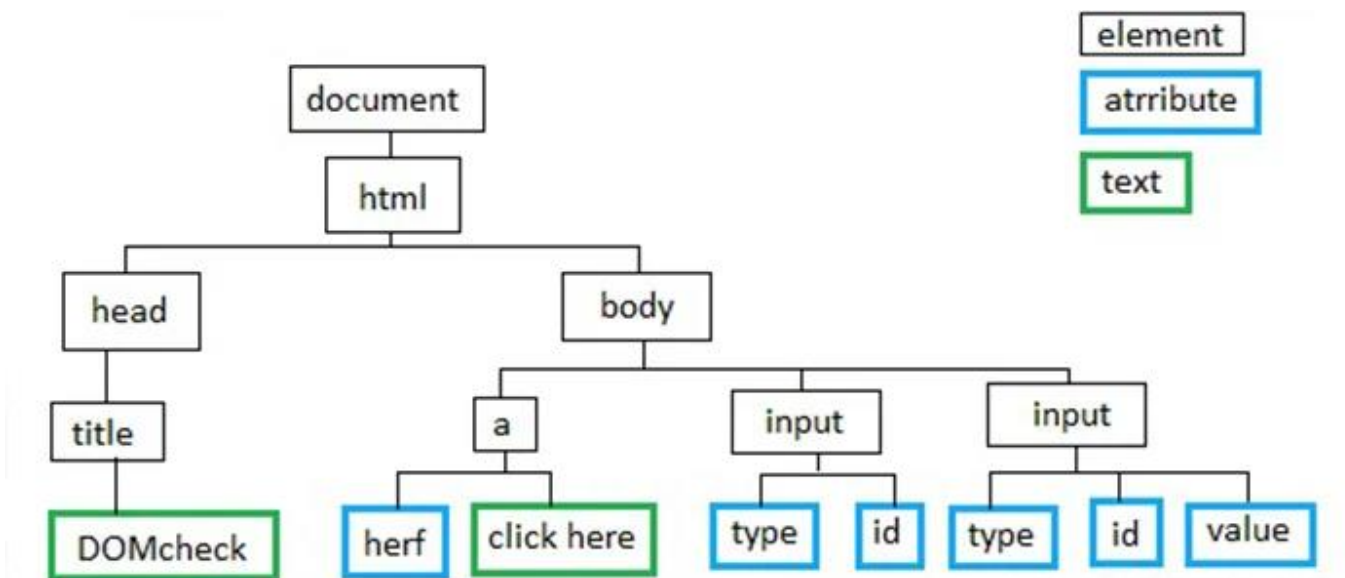
- Được hỗ trợ bởi tất cả các trình duyệt, đại diện cho cửa sổ trình duyệt (browser)
- Quản lý 3 đối tượng quan trọng trong trình duyệt: **DOM, BOM, JavaScript**
- Tất cả các **đối tượng, hàm và biến JavaScript** toàn cục sẽ tự động trở thành thành viên của đối tượng **window**.
- Ví dụ: `document.getElementById("header");`
=> Tương đương với:
`window.document.getElementById("header");`



Tổng quan về DOM, Element

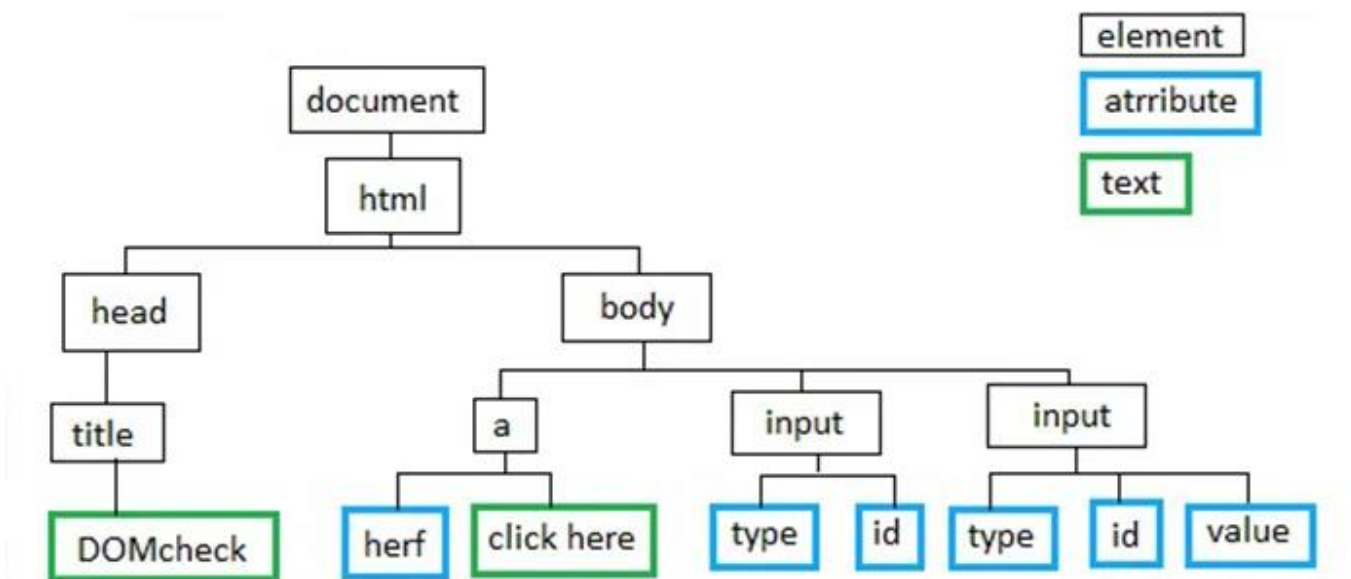
□ HTML DOM, Element:

- HTML DOM (**D**ocument **O**bject **M**odel), còn gọi là đối tượng **document**
- Document là đối tượng đại diện cho một trang HTML được tạo ra khi trình duyệt tải lên, được xây dựng bởi một cây phân tầng các đối tượng **Element**



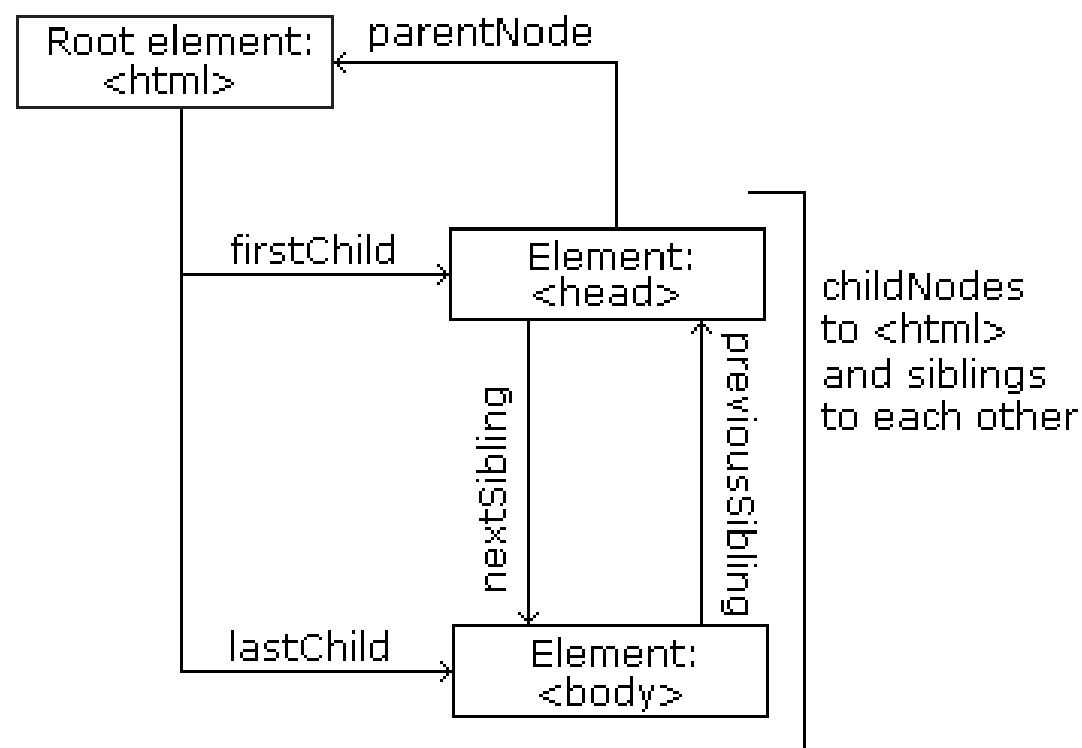
Quan hệ trong HTML DOM

- ❑ Theo tiêu chuẩn **W3C**, tất cả mọi thứ trong **HTML** coi như 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**)



Quan hệ trong HTML DOM

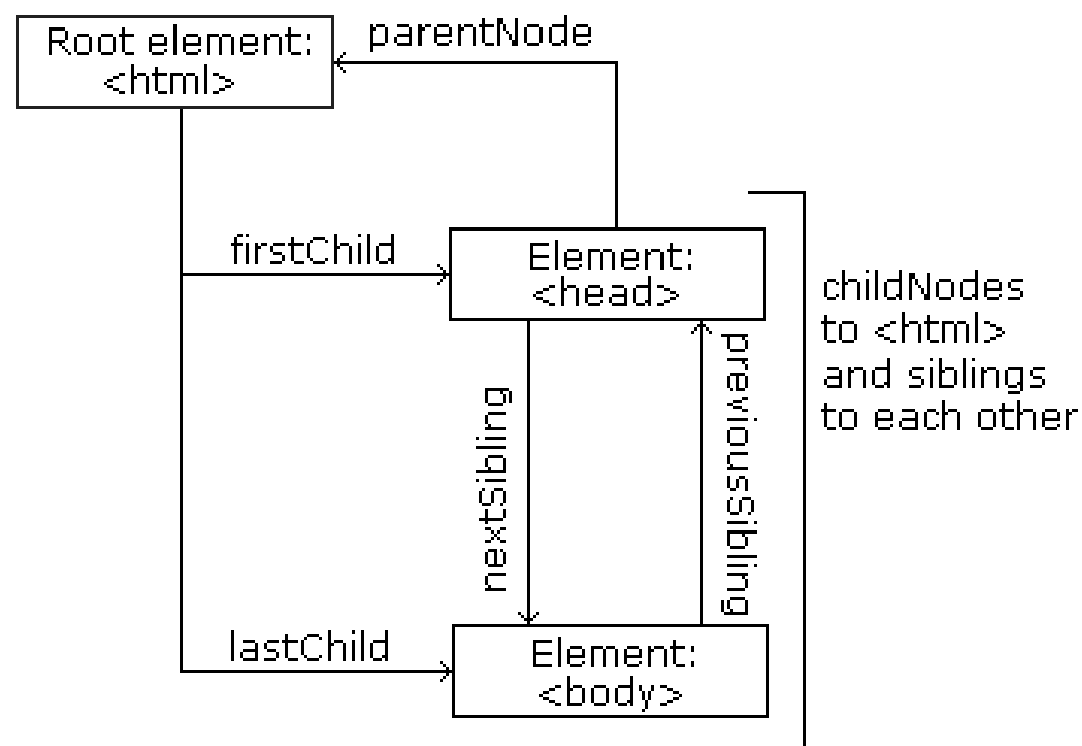
- ❑ Các nút trong cây (**node tree**) có mối quan hệ cha (**parent**), con (**child**), và anh em (**sibling**) với các nút còn lại:
 - Trong **node tree**, nút cao nhất (**top node**) được gọi là nút gốc (**root node**)
 - Mỗi nút đều có chính xác 1 nút cha, ngoại trừ nút gốc (không có nút cha)
 - Một nút có thể có nhiều nút con
 - Các nút anh em (**siblings**) là các nút có cùng nút cha



Quan hệ trong HTML DOM

❑ Hiểu quan hệ giữa các nút để điều hướng giữa các nút với JavaScript:

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

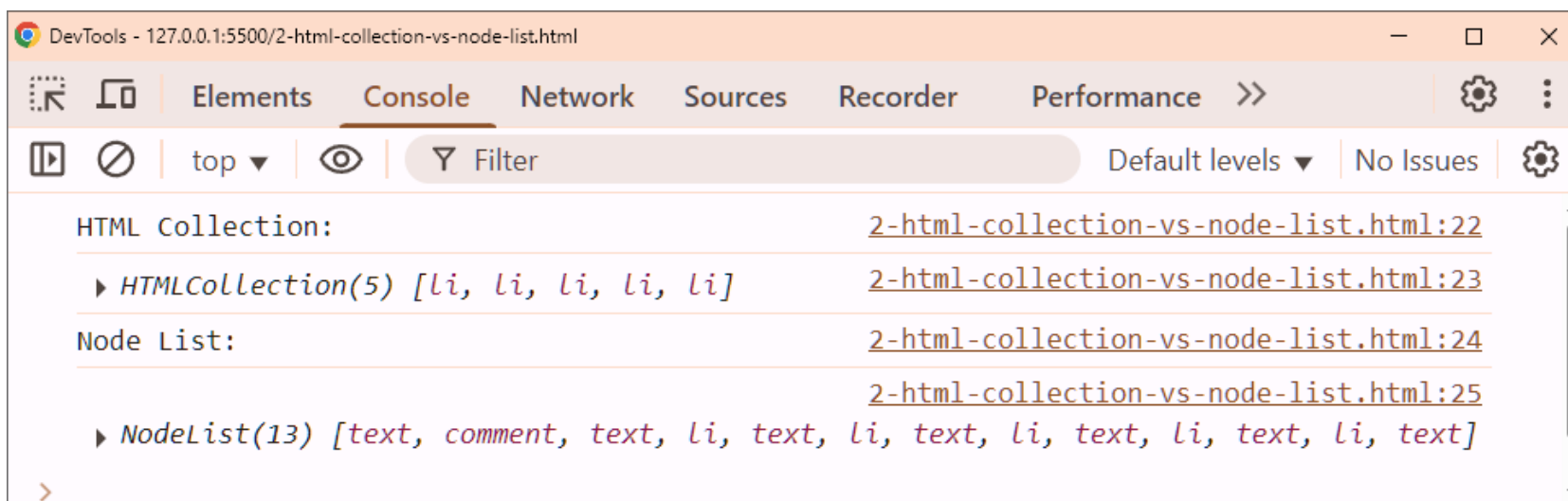


HTML Collection, Node List

□ HTML Collection và Node List:

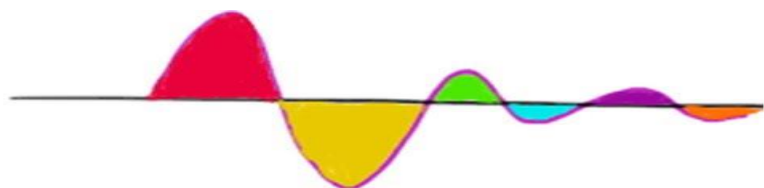
○ **HTML Collection** là một **danh sách các phần tử HTML**, mỗi **phần tử** có thể được **truy cập** bằng **tên thẻ**, **id** hoặc **chỉ mục**.

○ **Node List** là một **danh sách các nút (node)** trích xuất từ **document**, mỗi **nút** chỉ có thể được **truy cập** bằng **chỉ mục**, có thể chứa các **nút thuộc tính**, **nút văn bản**



Xử lý HTML, CSS với JS

- ❑ **Tìm nút HTML:** Sử dụng **id** của phần tử đó với cú pháp:
`let 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**. Ngược lại, **myElement** sẽ chứa giá trị **null**.

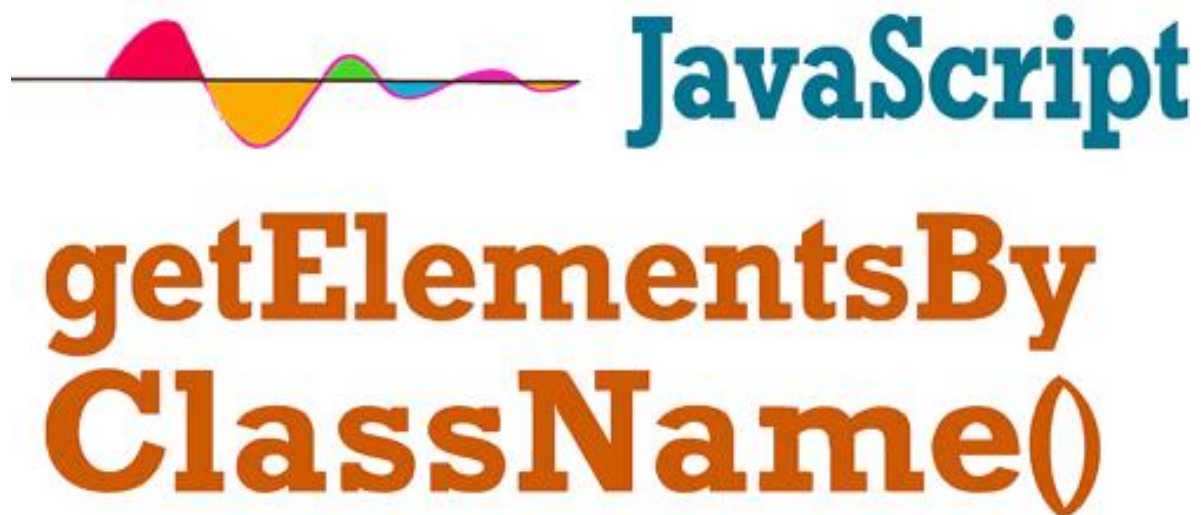


JavaScript

getElementById()

Xử lý HTML, CSS với JS

- ❑ **Tìm nút HTML:** Sử dụng **tên class** của phần tử đó với cú pháp:
`let 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

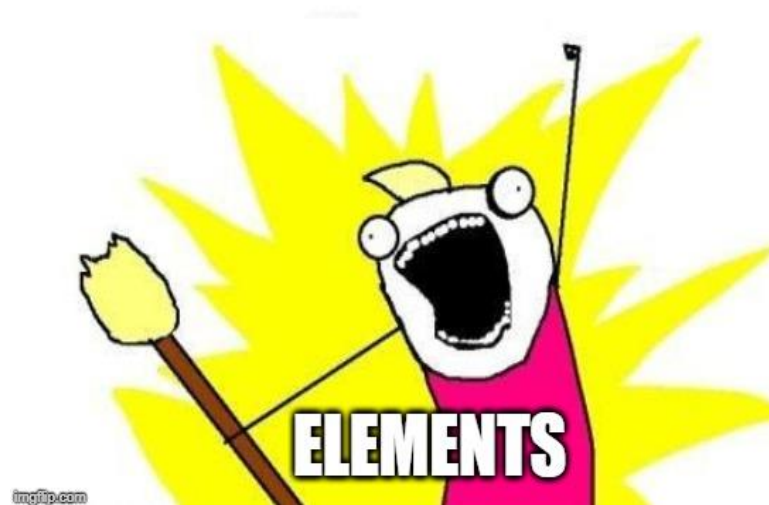


Xử lý HTML, CSS với JS

- ❑ **Tìm tất cả các nút HTML:** Sử dụng **tên bộ chọn CSS** (id, class, kiểu, thuộc tính, các giá trị thuộc tính,...) của phần tử đó với cú pháp:

```
let myElement = document.querySelector("intro");
```

SELECT ALL THE



Xử lý HTML, CSS với JS

- ❑ **Tạo mới nút HTML:** Để thêm một nút mới vào nút cuối cùng trong các nút con của nút cha hiện tại, đầu tiên ta phải tạo nút (node) HTML, sau đó nối nút mới vào cuối cùng trong các nút con của nút cha bằng **appendChild()**

```
<div id="firstDiv">
  <p id="p1">This is first paragraph</p>
  <p id="p2">This is second paragraph</p>
</div>

<script>
  let pElement = document.createElement("p");
  let pTextNode = document.createTextNode("This is a new paragraph");
  pElement.appendChild(pTextNode);
  let firstDiv = document.getElementById("firstDiv");
  firstDiv.appendChild(pElement);
</script>
```

Xử lý HTML, CSS với JS

- ❑ **Tạo mới nút HTML:** Để thêm một nút mới vào trước nút bất kỳ trong danh sách nút con của nút cha, ta có thể dùng phương thức **insertBefore()**

```
<div id="firstDiv">
  <p id="p1">This is first paragraph</p>
  <p id="p2">This is second paragraph</p>
</div>
<script>
  let pElement = document.createElement("p");
  let pTextNode = document.createTextNode("This is a new paragraph");
  pElement.append(pTextNode);

  let firstDiv = document.getElementById("firstDiv");
  let firstChild = document.getElementById("p1");
  firstDiv.insertBefore(pElement, firstChild);
</script>
```

Xử lý HTML, CSS với JS

- ❑ **Xóa 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

```
<div id="firstDiv">
  <p id="p1">This is first paragraph</p>
  <p id="p2">This is second paragraph</p>
</div>
<button onclick="removeFirstParagraph()">Remove first paragraph</button>

<script>
  function removeFirstParagraph() {
    let firstParagraph = document.getElementById("p1");
    firstParagraph.remove();
  }
</script>
```

Xử lý HTML, CSS với JS

- ❑ **Xóa 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

```
<div id="firstDiv">
  <p id="p1">This is first paragraph</p>
  <p id="p2">This is second paragraph</p>
</div>
<button onclick="removeFirstParagraph()">Remove first paragraph</button>

<script>
  function removeFirstParagraph() {
    let firstDiv = document.getElementById("firstDiv");
    let firstParagraph = document.getElementById("p1");
    firstDiv.removeChild(firstParagraph);
  }
</script>
```


Xử lý HTML, CSS với JS

- ❑ **Thay thế 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

```
<div id="firstDiv">
  <p id="p1">This is first paragraph</p>
  <p id="p2">This is second paragraph</p>
</div>
<button onclick="replaceFirstParagraph()">Replace first paragraph</button>

<script>
  function replaceFirstParagraph() {
    let firstDiv = document.getElementById("firstDiv");
    let firstParagraph = document.getElementById("p1");
    let newParagraph = document.createElement("p");
    let newTextNode = document.createTextNode("This is a new paragraph");
    newParagraph.appendChild(newTextNode);
    firstDiv.replaceChild(newParagraph, firstParagraph);
  }
</script>
```

Xử lý HTML, CSS với JS

❑ Thay đổi nội dung trong HTML:

○ **Cách 1:** lệnh `document.write()` có thể được dùng để viết trực tiếp lên trang HTML (cần kiểm soát kỹ lưỡng khi xài)

○ **Cách 2:** dùng thuộc tính `innerHTML` để thay đổi nội dung của phần tử HTML

Cú pháp: `document.getElementById(id).innerHTML = new HTML content;`

```
<p id="greeting">Hello Like Lioners</p>
<button onclick="changeGreeting()">Change Greeting</button>

<script>
  function changeGreeting() {
    document.getElementById("greeting").innerHTML = "Good morning Like Lioners";
  }
</script>
```

Xử lý HTML, CSS với JS

❑ Thay đổi giá trị thuộc tính của HTML:

- Cú pháp: **document.getElementById(id).**[tên attribute] = new value;
- Ví dụ: Thay đổi giá trị của thuộc tính src của thẻ **** như sau:

```

<br>
<button onclick="changeImage()">Change Image</button>

<script>
  function changeImage() {
    let newImageUrl = "https://www.w3schools.com/js/landscape.jpg";
    document.getElementById("myImage").src = newImageUrl;
  }
</script>
```

Xử lý HTML, CSS với JS

❑ Thay đổi giá trị CSS của HTML bằng thuộc tính style:

- Cú pháp: **document.getElementById(id).style.[tên style]** = new style value;
- Ví dụ: Thay đổi kiểu dáng của phần tử thẻ **<p>** như sau:

```
<p id="greeting">Hello Like Lioners</p>
<button onclick="hiddenText()">Hidden Text</button>
<button onclick="displayText()">Display Text</button>

<script>
  function hiddenText() {
    let currentText = document.getElementById("greeting");
    currentText.style.visibility = "hidden";
  }

  function displayText() {
    let currentText = document.getElementById("greeting");
    currentText.style.visibility = "visible";
  }
</script>
```

Xử lý sự kiện với JS

❑ Tương tác sự kiện trong HTML với JS:

- **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**
- **Cú pháp:** `onclick=JavaScript`
- **Ví dụ:** Khi người dùng click vào một phần tử HTML

Home Page

You have clicked the button 15 times



Xử lý sự kiện với JS

❑ **Một số sự kiện của HTML mà JS có thể được truyền vào để tương tác:**

- Khi người dùng click chuột
- 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

Xử lý sự kiện với JS

❑ Sự kiện **onchange**:

○ 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 dung vào trường nhập dữ liệu

```
<label for="fullName">Enter your full name: </label>
<input type="text" id="fullName" onchange="getUpperCaseValue()">

<script>
  function getUpperCaseValue() {
    let fullName = document.getElementById("fullName");
    fullName.value = fullName.value.toUpperCase();
  }
</script>
```

Xử lý sự kiện với JS

❑ 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

```
<div id="mySpace" class="my-space" onmouseover="mOver(this)" onmouseout="mOut(this)">
|   Hover Me
|
|</div>

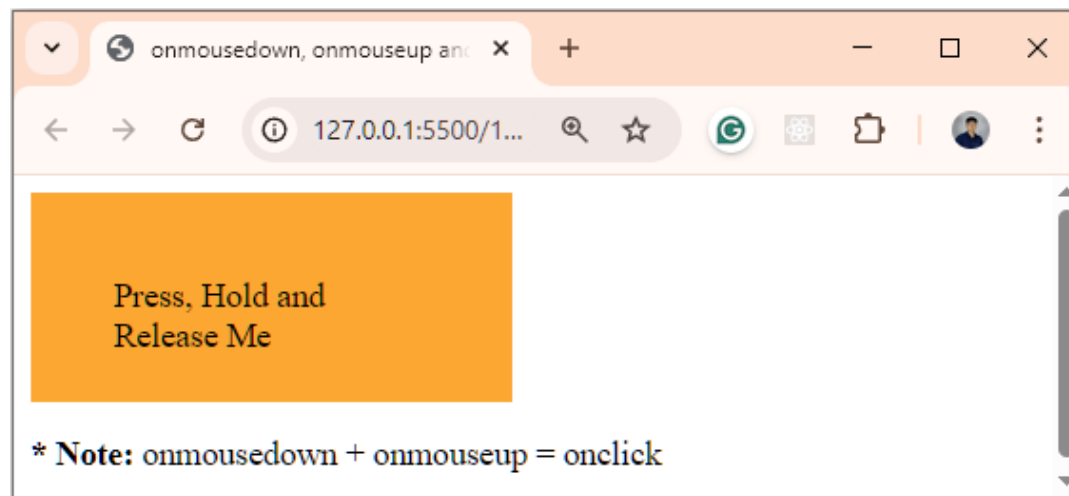
<script>
|   function mOver(currentTag) {
|       currentTag.innerHTML = "JavaScript is fun!";
|   }

|   function mOut(currentTag) {
|       currentTag.innerHTML = "Hover Me"
|   }
|</script>
```


Xử lý sự kiện với JS

❑ Sự kiện onmousedown, onmouseup và onclick:

- **onmousedown**, **onmouseup** và **onclick** đều là 1 phần của 1 cú nhấp chuột
- Đầu tiên **khi nút chuột được nhấn**, 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



Lắng nghe sự kiện với JS

❑ Phương thức `addEventListener()`:

- `addEventListener()` gắn một trình xử lý sự kiện vào phần tử được chỉ định, và 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, kể cả các sự kiện cùng loại. Ví dụ: 2 sự kiện nhấp chuột.
- Sau đó, **khi thả nút chuột**, sự kiện `onmouseup` được kích hoạt
- `removeEventListener()` giúp loại bỏ trình lắng nghe sự kiện hiện đang có
- **Cú pháp:** `element.addEventListener(event, function, useCapture);`

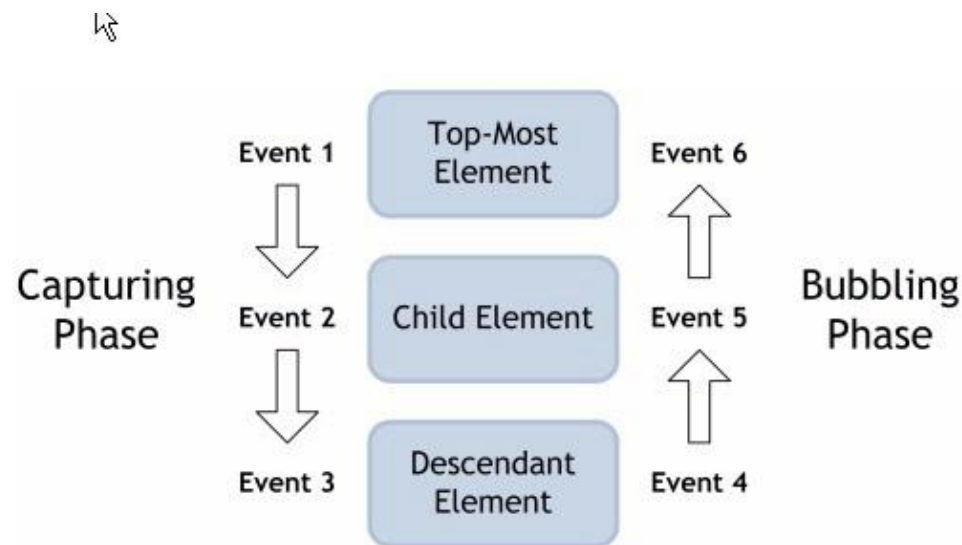
Lắng nghe sự kiện với JS

❑ Thứ tự truyền nhiều sự kiện:

○ Có 2 cách truyền nhiều sự kiện trong HTML DOM:

➤ **Event Bubbling:** các sự kiện của phần tử bên trong cùng sẽ được xử lý trước rồi tiếp tới các sự kiện của các phần tử bên ngoài được xử lý => **trong ra ngoài**

➤ **Event Capturing:** các sự kiện của phần tử bên ngoài cùng sẽ được xử lý trước rồi tiếp tới các sự kiện của các phần tử bên trong được xử lý => **ngoài vào trong**



Lắng nghe sự kiện với JS

❑ Thử tự truyền nhiều sự kiện:

- Cú pháp: **addEventListener**(*event*, *function*, *useCapture*);

- Trong đó:

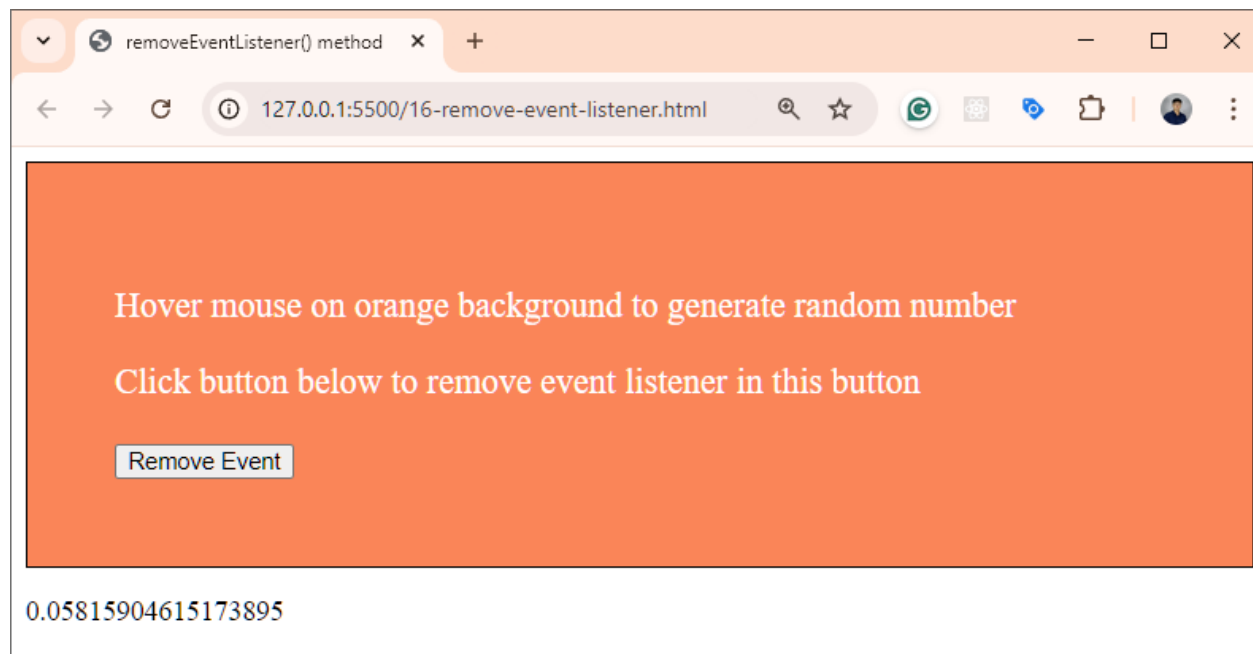
- **useCapture: false**, là giá trị mặc định -> giá trị này quy định thử tự truyền nhiều sự kiện theo dạng **Event Bubbling** (trong ra ngoài)

- **useCapture: true** -> giá trị này quy định thử tự truyền nhiều sự kiện theo dạng **Capturing Bubbling** (ngoài vào trong)

Lắng nghe sự kiện với JS

❑ Xóa sự kiện trên HTML DOM:

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



Xác thực dữ liệu với JS

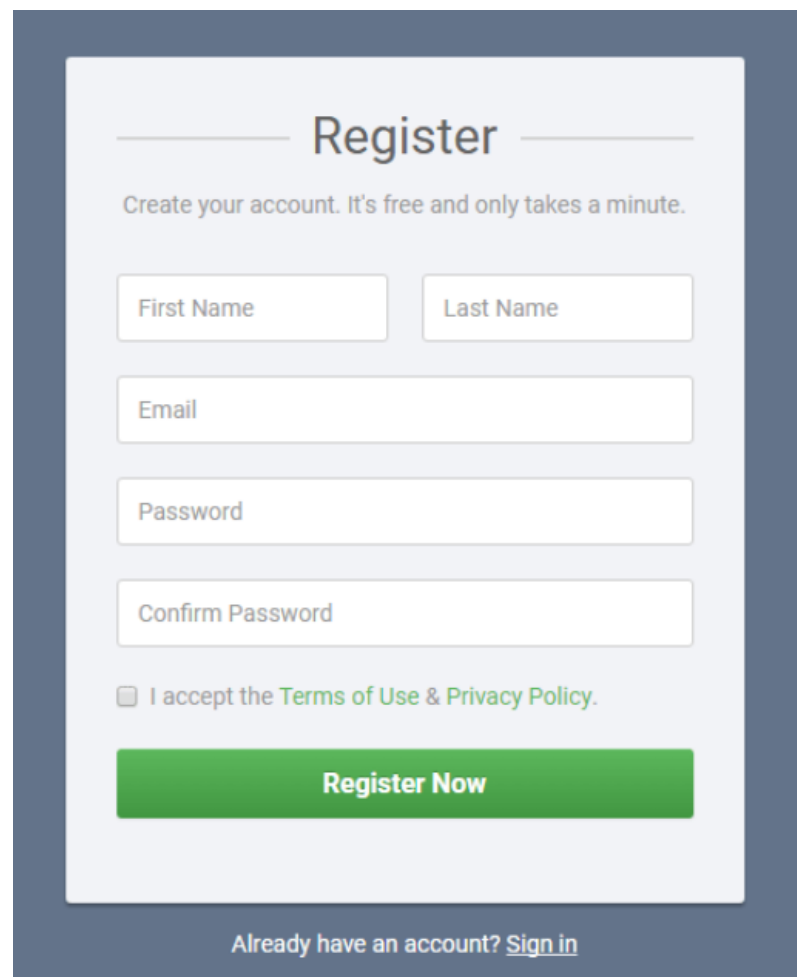
❑ Xác thực dữ liệu có lỗi hay không:

○ Bạn **thường gặp** các **website** mà **người dùng nhập các thông tin vào một 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 lệ của dữ liệu**

○ Một vài ví dụ về kiểm lỗi dữ liệu:

- Kiểm tra đảm bảo **dữ liệu không bị rỗng**
- Kiểm tra **định dạng email, số điện thoại**
- ...



Register

Create your account. It's free and only takes a minute.

First Name Last Name

Email

Password

Confirm Password

☐ I accept the [Terms of Use & Privacy Policy](#).

Register Now

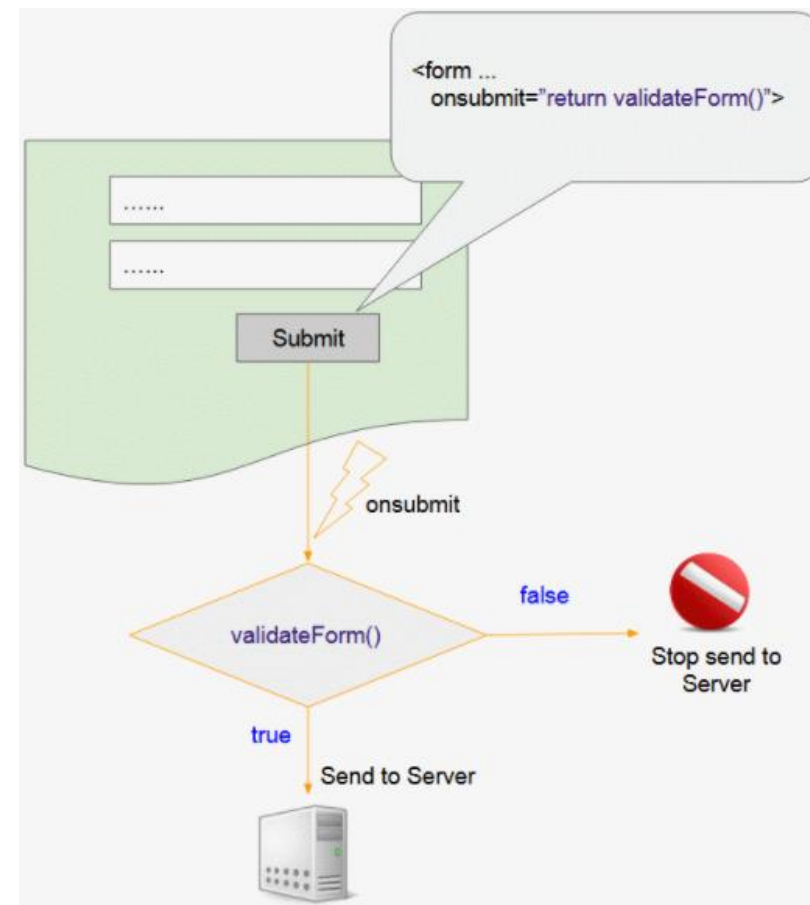
Already have an account? [Sign in](#)

Xác thực dữ liệu với JS

❑ Cách xác thực dữ liệu có lỗi hay không:

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

○ Thuộc tính **action** của **<form>** được sử dụng để **chỉ định trang mà dữ liệu sẽ được gửi đến**, vì vậy chúng ta sẽ **gọi hàm xử lý xác thực dữ liệu ở thuộc tính action** để **xác thực chắc chắn dữ liệu hợp lệ trước khi gửi lên server**.



Xác thực dữ liệu với JS

❑ Lưu ý thêm một số loại **type** mới trong HTML5 của phần tử **<input>**:

○ Ví dụ: **color, date, datetime-local, email, month, number, range, search, tel, time, url, week,...** => các phần tử mới 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

Tóm tắt bài học

- ☐ Tổng quan về JavaScript
- ☐ Tổng quan DOM, Element
- ☐ Quan hệ trong HTML DOM
- ☐ HTML Collection, Node List
- ☐ Thao tác HTML, CSS với JS
- ☐ Thao tác sự kiện HTML với JS
- ☐ Lắng nghe sự kiện HTML với JS
- ☐ Xác thực dữ liệu với JS

