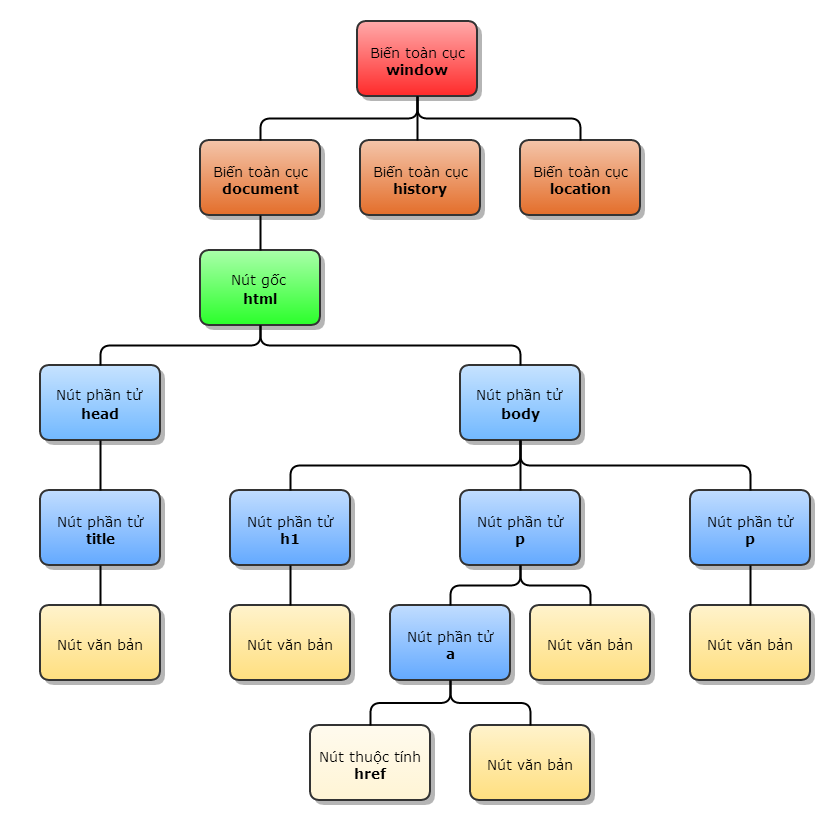
Hướng dẫn chi tiết

DOM là gì?

**DOM** là tên gọi tắt của (**D**ocument **O**bject **M**odel – tạm dịch *Mô hình Đối tượng Tài liệu*), là một chuẩn được định nghĩa bởi W3C dùng để truy xuất và thao tác trên các tài liệu có cấu trúc dạng HTML hay XML bằng các ngôn ngữ lập trình thông dịch (scripting language) như Javascript, PHP, Python. Trong phạm vi bài học này, mình sẽ dùng ngôn ngữ Javascript và HTML DOM

Đối với HTML DOM, mọi thành phần đều được xem là 1 nút (node), được biểu diễn trên 1 cây cấu trúc dạng cây gọi là DOM Tree. Các phần tử khác nhau sẽ được phân loại nút khác nhau nhưng quan trọng nhất là 3 loại: nút gốc (document node), nút phần tử (element node), nút văn bản (text node).

DOM – Xử lý các phần tử HTML trong Javascript

**Nút gốc**: chính là tài liệu HTML, thường được biểu diễn bởi thẻ <html>.

**Nút phần tử**: biểu diễn cho 1 phần tử HTML.

**Nút văn bản**: mỗi đoạn kí tự trong tài liệu HTML, bên trong 1 thẻ HTML đều là 1 nút văn bản. Đó có thể là tên trang web trong thẻ <title>, tên đề mục trong thẻ <h1>, hay một đoạn văn trong thẻ <p>.

Ngoài ra còn có **nút thuộc tính** (attribute node) và **nút chú thích** (comment node).

**Mong các bạn dành vài giây cho QUẢNG CÁO**

Nói chung đây là một quảng cáo về Hosting Azdigi để Góc Làm Web sẽ có một ít tiền để duy trì

Mình đang sử dụng và thấy nó nhanh, rẻ và dễ sử dụng. Các bạn dùng thử nhé.

Link đăng ký: [**https://my.azdigi.com/aff.php?aff=1612**](https://my.azdigi.com/aff.php?aff=1612)

Các thuộc tính của nút phần tử element

|  |  |
| --- | --- |
| **Thuộc tính** | **Ý nghĩa** |
| id | Định danh – là duy nhất cho mỗi phần tử nên thường được dùng để truy xuất DOM trực tiếp và nhanh chóng. |
| className | Tên lớp – Cũng dùng để truy xuất trực tiếp như id, nhưng 1 className có thể dùng cho nhiều phần tử. |
| tagName | Tên thẻ HTML. |
| innerHTML | Trả về mã HTML bên trong phần tử hiện tại. Đoạn mã HTML này là chuỗi kí tự chứa tất cả phần tử bên trong, bao gồm các nút phần tử và nút văn bản. |
| outerHTML | Trả về mã HTML của phần tử hiện tại. Nói cách khác, outerHTML = tagName + innerHTML. |
| textContent | Trả về 1 chuỗi kí tự chứa nội dung của tất cả nút văn bản bên trong phần tử hiện tại. |
| attributes | Tập các thuộc tính như id, name, class, href, title… |
| style | Tập các thiết lập định dạng của phần tử hiện tại. |
| value | Lấy giá trị của thành phần được chọn thành một biến. |

Các phương thức xử lý các nút phần tử

|  |  |
| --- | --- |
| **Phương thức** | **Ý nghĩa** |
| getElementById(id) | Tham chiếu đến 1 nút duy nhất có thuộc tính id giống với id cần tìm. |
| getElementsByTagName(tagname) | Tham chiếu đến tất cả các nút có thuộc tính tagName giống với tên thẻ cần tìm, hay hiểu đơn giản hơn là tìm tất cả các phần tử DOM mang thẻ HTML cùng loại. Nếu muốn truy xuất đến toàn bộ thẻ trong tài liệu HTML thì hãy sử dụng document.getElementsByTagName(‘\*’). |
| getElementsByName(name) | Tham chiếu đến tất cả các nút có thuộc tính name cần tìm. |
| getAttribute(attributeName) | Lấy giá trị của thuộc tính. |
| setAttribute(attributeName, value) | Sửa giá trị của thuộc tính. |
| appendChild(node) | Thêm 1 nút con vào nút hiện tại. |
| removeChild(node) | Xóa 1 nút con khỏi nút hiện tại. |

Truy xuất các phần tử

Có 2 phương pháp truy xuất các phần tử trong DOM là truy xuất gián tiếp (dựa theo vị trí của phần tử so với nút gốc để truy xuất), và truy xuất trực tiếp (dựa theo các định danh như id, class, tag name … để truy xuất).

Mình sẽ dùng phương pháp trực tiếp, vì nó chính xác vì dễ dàng hơn.

Truy xuất và lấy giá trị, thuộc tính

Muốn lấy giá trị, chúng ta phải xác định được phần tử html. Các bạn xem ví dụ sau nhé:

<input type="text" id="input\_1" value="1" gia\_tri="2">

<div class="mt-3" id="kq\_input\_1">ABC</div>

Chúng ta sẽ lấy **value** và **thuộc tính tự định nghĩa là gia\_tri** của **input** và xuất ra div bên dưới

<script type="text/javascript">

var input\_1 = document.getElementById("input\_1")

var value\_input\_1 = input\_1.value

console.log(value\_input\_1)

var gia\_tri\_input\_1 = input\_1.getAttribute("gia\_tri")

console.log(gia\_tri\_input\_1)

var div\_input\_1 = document.getElementById("kq\_input\_1")

div\_input\_1.innerHTML = value\_input\_1

</script>

**Hướng dẫn:**

+ Dùng phương thức**getElementById(“input\_1”)** để xác định phần tử cần xử lý

+ Lấy dữ liệu của thuộc tính value bằng cách truy xuất thuộc tính value của phần tử

+ Lấy dữ liệu của thuộc tính tự định nghĩa gia\_tri bằng phương thức **getAttribute(“gia\_tri)**

+ Lấy dữ liệu trong phần tử khối div bằng cách truy xuất thuộc tính innerHTML. Ngược lại, để gán giá trị bên trong khối div bằng cách gán thuộc tính innerHTML.

Thêm thuộc tính cho phần tử

Vẫn với đoạn code bên trên, chúng ta sẽ thêm thuộc tính readonly cho input

<button class="btn btn-outline-danger" onclick="readOnly()">Set Read only</button>

<script type="text/javascript">

function readOnly() {

var input\_1 = document.getElementById("input\_1")

input\_1.setAttribute("readonly", true)

}

</script>

+ Dùng sự kiện onclick để chạy hàm js. Về phần sự kiện – event, mình sẽ nói rõ trong bài học sau.

+ Sử dụng phương thức getElementById() để xác định phần tử.

+ Sử dụng phương thức **setAttribute(“tên thuộc tính”, gái trị)** để thêm thuộc tính.

Thêm xóa hoặc thay thế phần tử

Chúng ta sẽ thêm một phần tử vào bên trong một phần tử khác trong ví dụ sau:

Khối sẽ được thêm phần tử vào bên trong

<div id="them\_vao\_day"></div>

Thêm thẻ h1

<script type="text/javascript">

var them\_vao\_day = document.getElementById("them\_vao\_day")

var h1 = document.createElement("h1")

h1.innerHTML = "Thẻ h1 vừa thêm vào"

them\_vao\_day.appendChild(h1)

</script>

+ Xác định phần tử sẽ thêm vào bằng phương thức getElementById()

+ Khởi tạo phần tử con được thêm vào bằng **phương thức createElement()**

+ Thêm giá trị cho phần tử bằng cách**gán thuộc tính innerHTML.**

+ Dùng **phương thức appendChild()** để thêm phần tử được khởi tạo.

Truy xuất và thay đổi thuộc tính CSS của phần tử, thẻ html

Chúng ta có khối div màu xanh như sau:

<div id="div\_css" style="margin: 50px; width: 100px; height: 50px; background-color: red"></div>

Chúng ta sẽ truy xuất là xem thuộc tính css là margin, và chỉnh sửa đổi màu cho khối này.

<script type="text/javascript">

var div\_css = document.getElementById("div\_css")

var css = div\_css.style.margin

console.log(css)

div\_css.style.backgroundColor = "blue"

</script>

+ Xác định phần từ.

+ Truy xuất thuộc tính css của khối div bằng cách truy xuất thuộc tính style của phần tử vừa xác định.

+ Thay đổi thuộc tính background-color bằng cách **gán lại thuộc tính style.backgroundColor** cho phần tử.

Lưu ý: Các thuộc tính css như background-color, margin-top … sẽ được viết lại bằng cách **bỏ dấu gạch ngang** và **viết hoa chữ cái đầu mỗi từ** ( trừ từ đầu tiên).

Đó là tất cả nội dung của bài học. Các bạn xem video nếu cần hướng dẫn chi tiết nhé.

Co