|  |  |
| --- | --- |
| **conHTML DOM Element** | |
| **Khi một trang web được load, trình duyệt sẽ tạo ra một Document Object Model của page.**  **HTML DOM là một tiêu chuẩn về cách lấy, thay đổi, thêm hoặc xoá các HTML elements**  **DOM - Document Object Model gồm 3 thành phần**  **+ Element**  **+ Attribute**  **+ Text** | |
| **Finding HTML ELEMENTS** | |
| **document.getElementById( “***id* **“)** | Tìm element theo element id |
| **document.getElementsByTagName(“** name **“)** | Tìm các element theo Tag Name |
| **document.getElementsByClassName(“** name **“)** | Tìm các element theo Class Name |
| Dựa vào 3 cách trên có thể tìm Element theo cách tìm trong css như sau:  <**div id = “id” >**  **<p> </p>** // lấy thẻ p nằm trong thẻ div có id = “id :  **</div>** **document.getElementById(“id”).getElementsByTagName(“p”)** | |
| **document.querySelector( “**css selector” **)** | Tìm element theo cách tìm element trong css |
| **document.querySelectorAll( “**css selector” **)** | Tìm các element theo cách tìm element trong css |
| **<div id = “id” >**  **<p> </p>** // lấy thẻ p nằm trong thẻ div có id = “id :  **</div>** **document.querySelectorAll(“ #id p ”);** | |
| **document.anchors** | Chọn các thẻ <a name= “ “> trong document |
| **document.body** | Chọn các thẻ <body> trong document |
| **document.documentElement** | Chọn tất cả các thẻ HTML page |
| **document.embers** | Chọn tất cả các thẻ <embed> |
| **document.forms** | Chọn tất cả các thẻ <form> |
| **document.head** | Chọn thẻ <head> |
| **document.images** | Chọn các thẻ <image> |
| **document.links** | Chọn các thẻ <a herf= > </a> |
| **document.scripts** | Chọn các thẻ <script> |
| **document.title** | Chọn thẻ <title> |
| Coi document là một Object chứa các Object bên trong, ta có thể lấy các thuộc tính và phương thức của Object để áp dụng:  vd: **document.form.length**; 🡪 xem có bao nhiêu element nằm trong *object* form trong *object* document.  **document.form[“ id “]** 🡪 truy cập vào thẻ form có key = id; | |
| **Changing HTML ELEMENTS** | |
| ***element.*innerHTML =** new html content | **Thay đổi nội dung bên trong thẻ** |
| ***element.*outerHTML =** new html content | **Ghi đè lên element** |
| ***element.*innerText =** new text node | **(innerText - trả về các text node nhìn thấy)** |
| ***element.*textContent =** new text node | **(textContent - trả về toàn bộ text node)** |
| ***element.*attribute =** new value | **Thay đổi giá trị thuộc tính của một thẻ** |
| <img id=”myImage” src=”smiley.gif” >  Thay đổi giá trị của attributre src như sau: **document.getElementById(“myImage”).src = “landscape.jpg”** | |
| ***element.*style.property =** new style | **Thay đổi style của một thẻ** |
| **document.getElementbyId(“xx”).style.color = “blue”;**  **- Hoặc dùng Object.assign( ) để gán nhiều giá trị cho *element.*style:**  **vd: Object.assign( *element.*style , {**  **width : ‘200px’,**  **height: ‘200px’,**  **color : ‘red’,**  **} );**  **// Object.assign dùng để gán giá trị cho Object** | |
| ***element.g*etAttribute( attribute)** | **Lấy giá trị thuộc tính** |
| ***element.*setAttribute( attribute, value)** | **Thay đổi giá trị thuộc tính của một thẻ** |
| <img id=”myImage” src=”smiley.gif” >  Thay đổi giá trị của attributre src như sau:  **document.getElementById(“myImage”).setAttribute( ”src”, “landscape.jpg” );** | |
| **Adding and Deleting Elements** | |
| **document.createElement( *element* )** | **Tạo một thẻ HTML** |
| **document.createTextNode( “text”)** | **Tạo một dòng text** |
| ***element*.removeChild( *element* )** | **Xoá element là con của *element*** |
| ***element*.appendChild( *element* )** | **Thêm element mới là last child của *element*** |
| ***element*.insertBefore( *element* , child)** | **Thêm element mới là anh của child của *element*** |
| ***element*.replaceChild( new *element* , old element)** | **Thay thế thẻ old element là child của element bằng new element** |
| **document.write( *text* )** | **Viết ra output HTML** |
| **Adding Events Handles** | |
| **element.onclick = function( ) { };** | **Thêm trình xử lý sự kiện** |
| Với thuộc tính **onclick** thì nó sẽ ghi đè nên ta phải thêm các function và trong một function --> dễ gây rối mắt. Nên nếu cần thực hiện nhiều việc ta nên dùng **addEventListener**. | |
| **element.addEventListener( “event”, function( ) { }; )** | **Thêm trình xử lý sự kiện – có thể thêm nhiều** |
| **document.getElementById(“myImage”).addEventListener(“click”, myFunction1 );**  **document.getElementById(“myImage”).addEventListener(“click”, myFunction2 );** | |
| **element.removeEventListener( “event”, function( ) { }; )** | **Xoá trình xử lý sự kiện** |
| **Event Bubbling or Event Capturing:** | **Sự kiện xảy ra liên tiếp** |
| **element.addEventListener( “event”, function( ) { }; , useCapture);**  **vd: ta có thẻ <div> bọc ngoài thẻ <p>:**  **<div id =”myDiv> <p id=”myP> </p> </div>**  - Tạo 2 trình xử lý sự kiện khi click vào thẻ p và thẻ div, khi click vào thẻ p trình xử lý sự kiện của thẻ div cũng chạy theo sau. vì mặc định **useCapture = false – bubbling propagation – Truyền bong bóng:**  **document.getElementById(“myDiv”).element.addEventListener( “click”, function( ) { }; , false);**  **document.getElementById(“myP”).element.addEventListener( “click”, function( ) { }; , false);**  - Để trình xử lý sự kiện của thẻ Div chạy trước thẻ P ta đổi **useCapture = true – capturing propagation:**  **document.getElementById(“myDiv”).element.addEventListener( “click”, function( ) { }; , true);**  **document.getElementById(“myP”).element.addEventListener( “click”, function( ) { }; , true);**  **\*\* Tóm lại: all useCapture = false ; từ trong ra ngoài**  **all useCapture = true; từ ngoài vào trong** | |
| **HTML DOM Navigation – Sơ đồ liên kết** | |
| - **Toàn bộ document là document node**  vd: document.documentElement  **- Tất cả thẻ HTML là element node**  vd: document.head  **- Chữ nằm trong thẻ HTML là text node**  vd: element.innerHTML  **- Tất cả thuộc tính HTML là attribute node**  vd: element.href  **- Tất cả comments là comment node** | |
| **Navigating Between Node** | Cách liên kết giữa các nod |
| * ***element*.parentNode - Tìm node là cha của node khác** * ***element* .childNodes[ nodenumber ] - Tìm node là con thứ [ number] của node khác** * ***element*.firstChild - Tìm node là con thứ nhất của node khác** * ***element*.lastChild - Tìm node là con cuối cùng của node khác** * ***element*.nextSibling - Tìm node là em ngay kế của node khác** * ***element*.previousSibling - Tìm node là anh ngay kế của node khác**   vd: **<title id = “demo” > Dom Tutorial </title>**  - **element node - <title>** có 1 thẻ con là **text node chứa text**, bản thân element node không chứa text  Ta có thể truy cập vào text node bằng **3** cách sau:  **document.getElementById(“demo”).innerHTML;**  **document.getElementById(“demo”).firstChild.nodeValue;**  **document.getElementById(“demo”).childNodes[0].nodeValue;** | |
| **.nodeValue** | **Lấy giá trị của node** |
| **.nodeValue xác định giá trị của node:**  **- nodeValue of element nodes is null**  **- nodeValue of text nodes is text**  **- nodeValue of attribute nodes is attribute value**  **<title id = “demo” > Dom Tutorial </title>**  document.getElementById(“demo”).nodeValue; == null  document.getElementById(“demo”).firstChild.nodeValue; == “Dom Tutorial” | |
| **.nodeName (*read-only*)** | **Lấy tên của node** |
| **.nodeName xác định tên của node:**  **- nodeName of element nodes is Tag name** ( Trả về tag name dạng UpperCase)  **- nodeName of text nodes is #text**  **- nodeName of attribute nodes is attribute name**  **- nodeName of document node is #document**    **<title id = “demo” > Dom Tutorial </title>**  document.getElementById(“demo”).nodeName; == TITLE  document.getElementById(“demo”).firstChild.nodeName; == #TEXT | |
| **.nodeType (*read-only)*** | **Loại của node** |
| Trả loại Number xác định loại của node:  **1 Element node**  **2 Attributr node**  **3 Text node**  **8 Comment node** vd: <!-- this is comment -->  **9 Document node** vd: The HTML document  **10 Document type node** vd: <!Doctype html> | |
| **Creating new HTML Elements ( node )** | **Tạo một node mới** |
| Để thêm một element mới vào HTML DOM ta phải tạo một element ( element node) , sau đó thêm nó vào một element đã có trước.  **B1: tạo một element node mới:**  **const newElement = document.creatElement( “p” );** // tạo một element node **<p></p>**  **B2: tạo một text node mới:**  **const newText = document.creatTextNode( “this is text” );** // tạo một text node “this is text”  **B3: Thêm newText vào trong newElement:**  **newElement.appendChild(newText)** // tạo một element node **<p>**this is text**</p>**  **B4: Thêm newElement vào những element có sẵn.** | |
| ***element*.insertBefore( node , child)** | **Thêm node trước child của element** |
| **<div id=”div1”>**  **<p id=”p1” > </p>**  **<p id=”p2” > </p>**  **</div>**  Để thêm **newElement** vào vị trí nằm giữa 2 thẻ p:  **const element = document.getElementById(“div1”);**  **const child = document.getElementById(“p2”);**  **element.insertBefore(newElement, child); --> <div id=”div1”>**  **<p id=”p1” > </p>**  **<p> this is text </p>**  **<p id=”p2” > </p>**  **</div>** | |
| ***element*.replaceChild( new *element* , child)** | **Thay thế child của element** |
| vd: element.replaceChilde( newElement, child);  🡺 **<p id=”p2” > </p> trở thành <p> this is text </p>** | |
| **HTML DOM Collections** | |
| Phương thức **getElementsByTagName( )** sẽ trả về những HTMLCollection Object như một dạng Array,  Ta có thể truy cập vào các element bằng các index number hoặc id, hoặc name,  vd: **<p> </p>**  **<p id=”keyId” > </p>**  **<p> </p>**  **myCollection = document.getElementByTagName(“p”);** // trả về một collection các thẻ <p>  **Để truy cập vào element node <p> </p> ta dùng: myCollection[1]; hoặc myCollection[“keyId”];**  **ta có thuộc tính length: myCollection.length; == 3** | |
| **HTML DOM Node Lists** | |
| Phương thức **getElementsByClassName( ) VÀ querySelectorAll( )** sẽ trả về những HTMLCollection Object như một dạng Array,  Ta có thể truy cập vào các element bằng các index number.  \*\* Điểm khác nhau giữa HTML Collections và HTML Node Lists:  HTML Collection items có thể truy cập bằng name , id hoặc index number.  HTML Node List items chỉ có thể truy cập bằng index number. | |
| **Classlist Property** | |
| ***elementNode*.classlist;** | Trả về Object là class của node |
| ***elementNode*.classlist.length;** | Trả về độ dài là của Classlist |
| ***elementNode*.classlist.value;** | Trả về giá trị của Classlist |
| ***elementNode*.classlist.add(‘** className1’ **, ‘** className2 **‘);** | Thêm class vào Classlist |
| ***elementNode*.classlist.contain(‘** className’**);** | Kiểm tra class có trong Classlist không |
| ***elementNode*.classlist.remove(‘** className’**);** | Xoá class khỏi Classlist |
| ***elementNode*.classlist.toogle(‘** className’**);** | Kiểm tra nếu có thì xoá class khỏi Classlist còn nếu không có thì add class vào. |
| **DOM EVENTS** | |
|  |  |
| **PreventDefault and StopPropagation** | |
| ***event*.preventDefault** | Ngăn chặn hành vi mặc định |
|  | |
| ***event*.stopPropagation** | Ngăn chặn việc nổi bọt |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |