Chương VII: HTML DOM

Khoa Công Nghệ Thông Tin & Truyền Thông
Đại học Cần Thơ
Giảng viên: Hà Duy An

Nội dung

1. HTML Document Object Model
2. Sự kiện
3. Styles with JavaScript
4. Scripting Forms

1. HTML DOCUMENT OBJECT MODEL

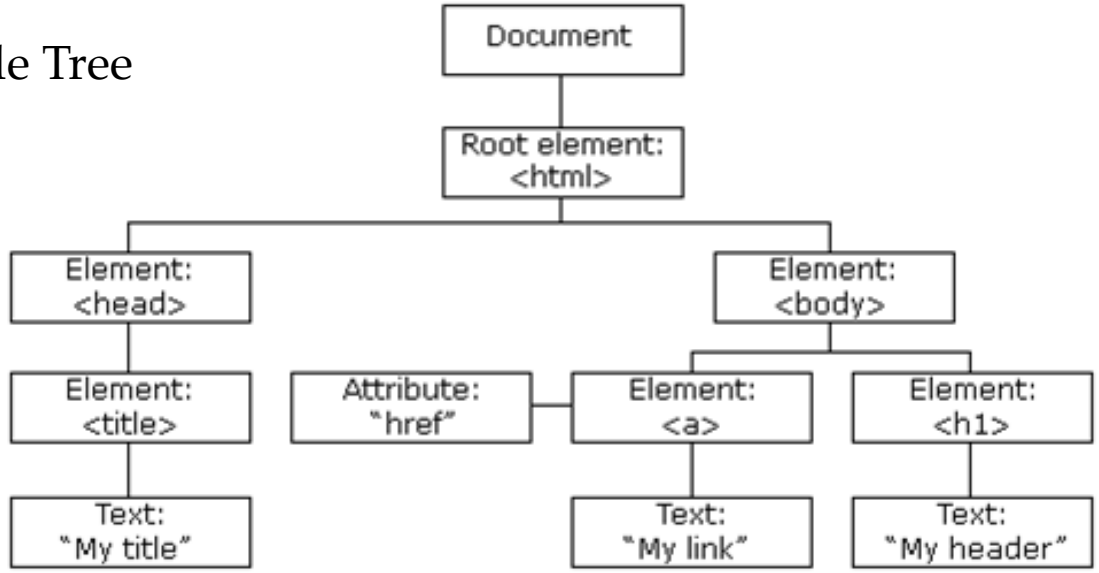
- Document Object Model (DOM) là một giao diện lập trình (API) cho các văn bản có cấu trúc như HTML hay XML
- DOM định nghĩa một tài liệu như là một cấu trúc cây phân cấp của các node, cho phép thêm, xóa, hay điều chỉnh từng thành phần riêng lẻ trên tài liệu đó
- DOM có tính độc lập về ngôn ngữ và nền tảng
- DOM là một chuẩn của W3C
- HTML Document Object Model (HTML DOM) hay mô hình đối tượng tài liệu HTML là việc áp dụng DOM vào các tài liệu HTML
- HTML DOM cung cấp một giao diện lập trình chuẩn cho HTML, định nghĩa cách thức truy cập và kiểm soát các tài liệu HTML

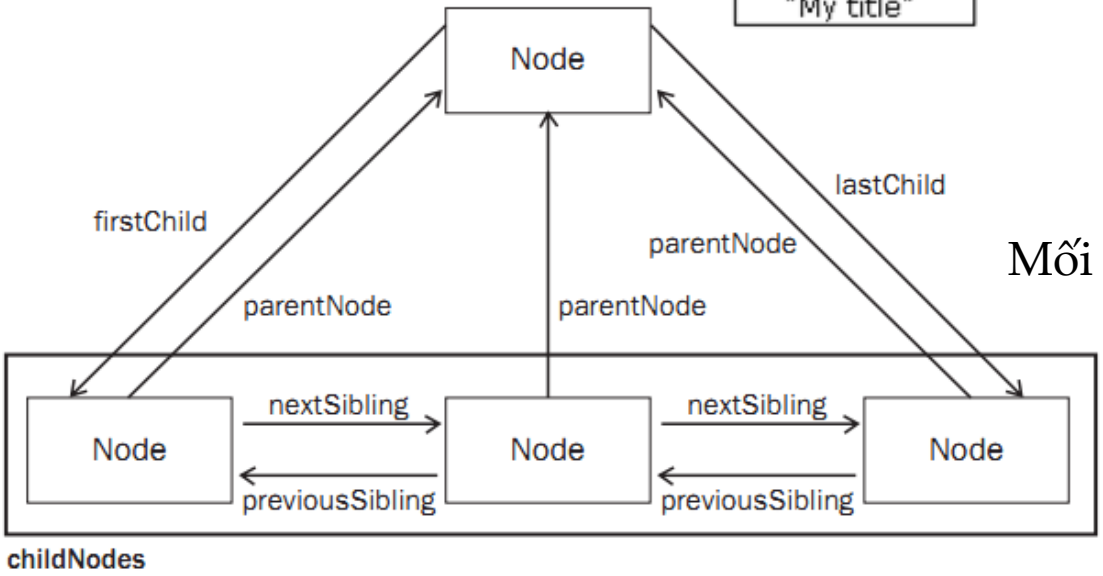
HTML DOM Nodes

- Theo DOM các tài liệu HTML có thể được trình bày như là các node phân cấp
- Có các loại node (node type) khác nhau, mỗi loại được sử dụng để trình bày hay đánh dấu một thông tin khác nhau trên tài liệu
- Mỗi loại node có các đặc điểm, dữ liệu, phương thức khác nhau, và có các mối quan hệ với các node khác
- Mọi thứ trên trang html đều là các node:
  - Toàn bộ trang: document node
  - Phần tử <html>: document element
  - Một phần tử bất kỳ: element node
  - Các văn bản: text node
  - Thuộc tính: attribute node
  - DOCTYPE declaration: document type node
  - Chú thích: comment node

HTML DOM Node Tree

HTML DOM Node Tree





Mối quan hệ giữa các node với nhau

HTML DOM Properties and Methods

- Các node có thể được truy cập bằng các ngôn ngữ lập trình như JavaScript
- Giao diện lập trình của HTML DOM định nghĩa các thuộc tính và phương thức chuẩn cho các node object
- **HTML DOM Properties**:
  - x.innerHTML - the text value of x
  - x.nodeName - the name of x
  - x.nodeValue - the value of x
  - x.parentNode - the parent node of x
  - x.childNodes - the child nodes of x
  - x.attributes - the attributes nodes of x
- **HTML DOM Methods**
  - x.getElementById(*id*) - get the element with a specified id
  - x.getElementsByTagName(*name*) - get all elements with a specified tag name
  - x.appendChild(*node*) - insert a child node to x
  - x.removeChild(*node*) - remove a child node from x

HTML DOM Properties

- Thuộc tính innerHTML: lấy về hay thay đổi nội dung của một thành phần html

```
<html>
<body>

<p id="intro">Hello World!</p>

<script
type="text/javascript">
txt=document.getElementById("i
ntro").innerHTML;
document.write("<p>The text
from the intro paragraph: " +
txt + "</p>");
</script>

</body>
</html>
```

Hello World!

The text from the intro paragraph: Hello World!

HTML DOM Properties (tt)

- Thuộc tính nodeName: trả về tên của một node
  - Là một read-only properties
  - nodeName của một element node chính là tên thẻ
  - nodeName của một attribute node là tên của thuộc tính
  - nodeName của một text node là "#text"
  - nodeName của document node là "#document"
- Thuộc tính nodeValue: xác định giá trị của một node
  - nodeValue của một element nodes là undefined
  - nodeValue của một text node là giá trị text của nó
  - nodeValue của một thuộc tính là giá trị của thuộc tính đó

HTML DOM Properties (tt)

- Có tất cả 12 loại node khác nhau, tất cả điều thừa kế tử kiểu cơ bản là Node:
  - Node.ELEMENT_NODE (1)
  - Node.ATTRIBUTE_NODE (2)
  - Node.TEXT_NODE (3)
  - Node.CDATA_SECTION_NODE (4)
  - Node.ENTITY_REFERENCE_NODE (5)
  - Node.ENTITY_NODE (6)
  - Node.PROCESSING_INSTRUCTION_NODE (7)
  - Node.COMMENT_NODE (8)
  - Node.DOCUMENT_NODE (9)
  - Node.DOCUMENT_TYPE_NODE (10)
  - Node.DOCUMENT_FRAGMENT_NODE (11)
  - Node.NOTATION_NODE (12)
- Tất cả các node đều có thuộc tính nodeType => trả về kiểu của node

HTML DOM Properties (tt)

- **VD:**

```
<p id="intro">Hello World!</p>

<script type="text/javascript">
x=document.getElementById("intro");
document.write("nodeType of x: "+x.
nodeType+"<br/>nodeName of x: "+x.nodeName
+"<br/>nodeValue of x: "+x.firstChild.
nodeValue);
</script>
```

Truy xuất node object

- *node*.getElementById(*"id"*) : trả về một đối tượng html có định danh (id) tương ứng

  VD:

```
<p id="intro">Hello World!</p>
<p>This example demonstrates the <b>getElementById</b>
 method!</p>

<script type="text/javascript">
x=document.getElementById("intro");
document.write("<p>The text from the intro paragraph:
" + x.innerHTML + "</p>");
</script>
```

Truy xuất node object (tt)

- *node*.getElementsByTagName("*tagname*") : trả về một NodeList object chứa các node có tên thẻ là *tagname*
- *node*.getElementsByName("nameValue") : trả về một NodeList object chứa các node có thuộc tính name với giá trị là "nameValue"
- Thông qua các mối quan hệ:
  - X.childNodes: là một NodeList object, chứa các node con của node X
  - X.previousSibling, X.nextSibling: trỏ đến các node anh em của node X
  - X.firstChild, X.lastChild: trỏ đến node con đầu tiên, và node con cuối cùng của node X
  - X.parentNode: trỏ đến node cha của node X

Truy xuất node object (tt)

VD:

```
<form>
<fieldset>
<legend> Which color do you prefer? </legend>
<input type="radio" value="red" name="color" id="colorRed">
<label for="colorRed"> Red </label>
<input type="radio" value="green" name="color" id="colorGreen">
<label for="colorGreen"> Green </label>
<input type="radio" value="blue" name="color" id="colorBlue">
<label for="colorBlue"> Blue </label>
 </fieldset>
</form>
 <hr/>
<script type="text/javascript">
// retrieve all elements in the document
var allElement=document.getElementsByTagName("*");
// retrieve all legend element in the document
var legendElement=document.getElementsByTagName("legend");
// retrieve all element have name is color
var getElementWithName=document.getElementsByName("color");
document.write("Number of element in this document: "+ allElement.
length +"<br/>");
</script>
```

Truy xuất node object (tt)

VD:

```
<p id="pa"><span>span 1</span><span id="sp2">span
2</span><span>span 3</span></p>

<script type="text/javascript">
x=document.getElementById("pa");
y=document.getElementById("sp2");

document.write("x.firstChild.firstChild.nodeValue
= "+x.firstChild.firstChild.nodeValue);
document.write("<br/>
y.parentNode.lastChild.firstChild.nodeValue = "+x.
childNodes.item(0).nextSibling.firstChild.
nodeValue);
document.write("<br/>
y.parentNode.lastChild.firstChild.nodeValue = "+y.
parentNode.lastChild.firstChild.nodeValue);
```

Thêm, xóa các phần tử HTML

- Tạo một phần tử html:
  - document.createElement("*tagName*"): hàm trả về một đối tượng html có kiểu tương ứng với *tagName*
- Thêm một phần tử html:
  - X.appendChild(*newNode*): thêm vào một node con cho node X, node mới thêm vào sẽ trở thành lastChild của X.
  - X.insertBefore(*newNode*, *childNodeOfX*): thêm vào một node con cho node X, node mới được thêm vào sẽ nằm trước node *childNodeOfX*
- Thay thế một phần tử này bằng một phần tử khác:
  - X.replaceChild(*newNode*, *childNodeOfX*): *newNode* sẽ thay thế *cho childNodeOfX* trong danh sách các con của node X
- Xóa một phần tử html:
  - removeChild(*childNodeOfX*): xóa một node con của node X

Thêm, xóa các phần tử HTML (tt)

VD:

```
<table border="1" id="t1"><tr><td>Cell 1</td></tr></table>
<button onclick="addCell()">Add Cell</button>
<button onclick="removeCell()">Remove Cell</button>
<script type="text/javascript">
var n=1;
tb=document.getElementById("t1");
function addCell()
{
    var c=document.createElement("td");
    c.innerHTML="Cell "+n;
    tb.firstChild.firstChild.appendChild(c);
    n++;
}
function removeCell()
{
    tb.firstChild.firstChild.removeChild(tb.firstChild.firstChild.
lastChild);
    n--;
}
```

Attributes

- X.getAttribute("*attributeName*"): trả về giá trị của thuộc tính *attributeName* của phần tử X, trả về null nếu thuộc tính không tồn tại
- X.setAttribute("*attributeName*", "*attributeValue*"): gán giá trị cho thuộc tính *attributeName* với giá trị là *attributeValue* của phần tử X
- X.removeAttribute("*attributeName*"): xóa bỏ một thuộc tính của phần tử X
- Tất cả các thuộc tính của một thành phần HTML đều được định nghĩa là thuộc tính của các đối tượng HTML trong mô hình HTML DOM (VD: X.id, X.className, X.onClick,…)
- Các hàm getAttribute, setAttribute không được hỗ trợ giống nhau trên tất cả các trình duyệt => hạn chế sử dụng, chỉ nên sử dụng nó cho các thuộc tính tự định nghĩa

Attributes (tt)

```
<html>
<head>
<style type="text/css">
.redBold{ color:red; font-weight:bold;}
.blueItalic{ color:blue; font-style:italic;}
</style>
<script type="text/javascript">
function changeClass(obj){obj.setAttribute("class"
,"blueItalic")}
function backToPreviousClass(obj){obj.setAttribute
("class","redBold")}
</script>
</head>
<body>
<p onmouseover="changeClass(this)" class="redBold"
 onMouseOut="backToPreviousClass(this)">Move mouse
 over to change style for this paragraph</p>
</body>
</html>
```

HTML Collection Object

- HTMLCollection Object giống như một mảng các đối tượng dùng để lưu trữ một danh sách các node theo thứ tự, và có thể truy xuất theo vị trí như một mảng, hay thông qua phương thức item("position"), tuy nhiên nó không phải là một thể hiện của kiểu dữ liệu mảng
- Một số HTML Collection đặt biệt của đối tượng document: cho phép truy xuất đến một số thành phần html nhanh hơn
  - document.anchors: chứa tất cả các phần tử <a> có thuộc tính name trong trang html
  - document.forms : chứa tất cả các phần tử <form> trong trang html. Giống như: document .getElementsByTagName("form")
  - document.images: chứa tất cả các phần tử <img> trong trang html. Giống như: document .getElementsByTagName("img")
  - document.links : chứa tất cả các phần tử <a> có thuộc tính href trong trang html

HTML Collection Object (tt)

VD:

```
<img src="bgdesert.jpg" />
<img src="klematis2.jpg" /><br />
<button id="btChange">Change Image</button>
<script type="text/javascript">
    document.getElementById("btChange").onclick
=function(){
        var img=document.images;
        for(var i=0;i<img.length;i++)
        {
            var pattern=new RegExp(
"bgdesert.jpg");
            if(pattern.test(img[i].src))
            {   img[i].src="img_tree.png";}
        }
    }

</script>
```

2. SỰ KIỆN

- Với JavaScript ta có thể tạo ra các nội dung động cho một trang web bằng cách viết các script xử lý nội dung của trang web được kích hoạt bằng các sự kiện
- Các sự kiện được định nghĩa sẵn: http://w3schools.com/jsref/dom_obj_event.asp
- Cách viết script để xử lý một sự kiện trên trang web:
  - Dựa vào các thuộc tính là các sự kiện của các thẻ: attributeEvent= " *javaScript code/fucntion* "
  - Dựa vào các thuộc tính là các sự kiện của một đối tượng html: htmlObj.event=*functionName* **hay**

    htmlObj.event=function() {*javaScript code*}

2. SỰ KIỆN (tt)

VD:

```
<html>
<head>
<script type="text/javascript">
var d=document;
function loadScript()
{
    var pa=document.getElementById("pa");
    alert("Contents of this webpage:\n"+pa.innerHTML);
}
</script>
</head>
<body onLoad="loadScript()">
<p id="pa">This is a paragraph!</p>
</body>
</html>
```

2. SỰ KIỆN (tt)

- **Timing Events:** dùng để định thời gian thực thi một đoạn mã lệnh nào đó
- **Phương thức:**
  - let t = setTimeout(*function, milliseconds*): thực thi lệnh một hàm vào một khoản thời gian trong tương lai
    - Hủy sự kiện Timing: clearTimeout(*setTimeout_variable*)
  - let t = setInterval(*function, milliseconds*) : thực thi lệnh một hàm định kỳ lặp lại
    - Hủy sự kiện Timing: clearInterval(*setInterval_variable*)

VD:
```
function timeMsg()
{
  var t=setTimeout("alertMsg()",3000);
}
function alertMsg()
{
  alert("Hello");
}
```

3. STYLES WITH JAVASCRIPT

- **Load external css with JavaScript:** tạo một thành phần <link> sau đó thêm nó vào thành phần <head>

```
<script type="text/javascript">
function loadStyles(url)
{
    var linkObj=document.createElement("link");
    linkObj.type="text/css";
    linkObj.rel="stylesheet" ;
    linkObj.href=url;
    document.head.appendChild(linkObj);
}
</script>
<button onclick="loadStyles('style.css')">
Change Background</button>
```

3. STYLES WITH JAVASCRIPT (tt)

- **Thêm các luật css vào thẻ <style>**

```
<script type="text/javascript">
function change(cssText)
{
    var style = document.createElement("style");
    style.type = "text/css";
    style.innerHTML=cssText;
    document.head.appendChild(style);
}
</script>
<button onclick=
"change('body{background:red;}')">Change red
Backgound</button>
```

3. STYLES WITH JAVASCRIPT (tt)

- **Sử dụng đối tượng style**: có thể dùng đối tượng style với các thuộc tính của nó để định dạng cho một thành phần html
  - Cú pháp: *htmlObj*.style.*property*="*value*"
  - Các thuộc tính của đối tượng style: http://w3schools.com/jsref/dom_obj_style.asp

VD:

```
<html>
<head>
</head>
<body>
<h1 id="h1" onclick=
"document.getElementById('h1').style.color='red'">
Click Me!</h1>
</body>
</html>
```

3. STYLES WITH JAVASCRIPT (tt)

- **Sử dụng thuộc tính className:**

```
<html>
<head>
<title>Untitled Document</title>
<style type="text/css">
.red{color:red;}
.green{color:green;}
.blue{color:blue;}
</style>
<script type="text/javascript">
function changeColor(radioBox)
{
    document.getElementById("pa").className=radioBox.id;
}
</script>
</head>
```

3. STYLES WITH JAVASCRIPT (tt)

- **Sử dụng thuộc tính className (tt):**

```
<body>
<form>
<fieldset>
<legend>Let choose the color of text for paragraph below:</legend>
<input type="radio" name="color" id="red" onclick="changeColor(this)" />
<label for="red" class="red">Red</label>
<input type="radio" name="color" id="green" onclick="changeColor(this)" />
<label for="green" class="green">Green</label>
<input type="radio" name="color" id="blue" onclick="changeColor(this)" />
<label for="blue" class="blue">Blue</label>
</fieldset>
</form>
<p id="pa">This is some text. This is some text. This is some text. This
is some text. This is some text. This is some text.</p>
</body>
</html>
```

3. STYLES WITH JAVASCRIPT (tt)

- **Sử dụng thuộc tính `classList` với các phương thức sau:**
  - `add()`
  - `remove()`
  - `contains()`
  - `toggle()`

```
<html>
<head>
    <style>
        .highlight {
            background-color: yellow;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h1 id="myElement">Hello World!</h1>
    <button onclick="addClass()">Add Class</button>
    <button onclick="removeClass()">Remove Class</button>
    <button onclick="toggleClass()">Toggle Class</button>

    <script>
        function addClass() {
            const element = document.getElementById('myElement');
            element.classList.add('highlight');
        }

        function removeClass() {
            const element = document.getElementById('myElement');
            element.classList.remove('highlight');
        }

        function toggleClass() {
            const element = document.getElementById('myElement');
            element.classList.toggle('highlight');
        }
    </script>
</body>
</html>
```

4. SCRIPTING FORMS

```
<script type="text/javascript">
function validateForm()
{
var x=document.forms["myForm"]["fname"].value;
if (x==null || x=="")
  {
    alert("First name must be filled out");
    return false;
  }
}
</script>
<form name="myForm" action="demo_form.asp"
onsubmit="return validateForm()" method="post">
First name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

Question ?


THANK YOU !