AJAX - Tương lai của ứng dụng Web

Bài 4: Các công nghệ trong AJAX - DOM - Document Object Model.

Document Object Model - DOM

Document Object Model (DOM) giúp phân tích một tài liệu (một trang web chẳng hạn) phục vụ cho cơ chế của JavaScript. Sử dụng DOM, cấu trúc của tài liệu có thể được phân rã theo cấu trúc cây và thao tác theo các nút. Đây là một khả năng đặc biệt hữu ích để viết một ứng dụng Ajax. Trong các ứng dụng web truyền thống, trình duyệt phải tái nạp các trang HTML theo một luồng từ server.

Trong một ứng dụng AJAX, sự thay đổi giao diện người dùng chủ yếu được tạo ra bởi DOM. Các thẻ HTML trong trang web được tổ chức theo cấu trúc cây. Gốc của cây là thẻ <HTML>, để biểu diễn tài liệu. Trong đó thẻ <BODY> biểu diễn phần thân của tài liệu, là gốc của phần hiển thị của tài liệu. Trong thân của tài liệu, có các bảng, paragraph, list, và các loại thẻ khác với các thẻ ở mức thấp hơn nữa.

Một biểu diễn theo mô hình DOM của một trang web là một cấu trúc cây, có các phần tử là các nút, rồi nó chứa các nút con trong nó, và cứ tiếp tục một cách đệ qui như thế. JavaScript làm việc với nút gốc của trang web hiện thời qua một biến toàn cục gọi là document, biến này là điểm bắt đầu của mọi thao tác trên DOM. Phần tử DOM đã được đặc tả bởi W3C. Mỗi phần tử DOM có một phần tử cha duy nhất, có hoặc không có các phần tử con, và có một số bất kỳ các thuộc tính, chúng được lưu trữ trong mảng móc nối.

Mối quan hệ giữa các phần tử DOM có thể được đối chiếu bởi danh sách các thành phần HTML. Mối quan hệ này là hai chiều. Sửa đổi mô hình DOM sẽ thay đổi cấu trúc HTML và dẫn đến thay đổi cách biểu diễn một trang web.

Bài 4 (tiếp): Các công nghệ trong AJAX - DOM - Làm việc với DOM bằng JavaScript.

Làm việc với DOM bằng JavaScript.

Trong một ứng dụng bất kỳ, nếu muốn thay đổi giao diện người dùng khi họ đang làm việc, thì phải cung cấp các phản hồi lại khi người dùng gửi các yêu cầu.

Để hiểu rõ cơ chế làm việc với DOM bằng JavaScript, chúng ta cùng xét một ví dụ về một trang HTML đơn giản.

Trích:

```
<html>
<head>
<link rel=’stylesheet’ type=’text/css’ href=’hello.css’ />
<script type=’text/javascript’ src=’hello.js’></script>
</head>
<body>
<p id=’hello’>hello</p>
<div id=’empty’></div>
</body>
```

Ta đã thêm vào các tham chiếu đến các file hello.css (dùng Cascading Style Sheet) và một file chứa mã nguồnJavaScript là hello.js. Ở đây cũng đồng thời khai báo một thẻ với một ID.

Còn đây là file hello.css chứa stylesheet để áp dụng cho các mục trong file HTML:

Trích:

```
.declared{
color: red;
font-family: arial;
font-weight: normal;
font-size: 16px;
}
.programmed{
color: blue;
font-family: helvetica;
font-weight: bold;
font-size: 10px;
}
```

Chúng ta định nghĩa hai style, để mô tả gốc của các nút DOM (tên của các style là tùy chọn). Các style này không dược dùng trong file HTML, nhưng chúng sẽ được áp dụng qua file JavaScript.

Trích:

```
window.onload=function(){
var hello=document.getElementById(’hello’);
hello.className=’declared’;
var empty=document.getElementById(’empty’);
addNode(empty,”reader of”);
```

```
addNode(empty,”Ajax in Action!”);
var children=empty.childNodes;
for (var i=0;i<children.length;i++){
children[i].className=’programmed’;
}
empty.style.border=’solid green 2px’;
empty.style.width=”200px”;
}
function addNode(el,text){
var childEl=document.createElement(”div”);
el.appendChild(childEl);
var txtNode=document.createTextNode(text);
childEl.appendChild(txtNode);
}
```

Hàm window.onload() sẽ được gọi khi trang web được nạp. Tại thời điểm này, cấu trúc cây DOM sẽ được thiết lập.

Trong bài 5 anh em ta sẽ học về: Tìm kiếm một DOM Node, Tạo DOM Node.

Bài 5: Các công nghệ trong AJAX - DOM - Tìm kiếm & Tạo DOM Node.

Tìm kiếm một DOM Node

Yêu cầu đầu tiên để làm việc trên DOM với JavaScript là đi tìm kiếm một phần tử để thay đổi. Trước hết cần bắt đầu tham chiếu qua nút gốc - root node, nút này thể hiện qua biến toàn cục document.

Mỗi nút trong DOM là một nút con (hoặc nút con cấp hai, ba…) của document, nhưng cứ đi dần vào cây DOM, sẽ thấy một tài liệu phức tạp được biểu diễn bởi DOM, và việc tìm kiếm là rất khó khăn.

Vì thế có các cách sau để tìm kiếm một nút nhanh chóng hơn. Mỗi phần tử HTML có một thuộc tính ID, ví dụ như,

Trích:

```
<p id=’hello’>
```

hay

Trích:

```
<div id=’empty’></div>
```

Mỗi một nút DOM có thể có một ID gán cho nó, và ID này có thể được dùng để tham chiếu tới nút qua hàm :

Trích:

```
var hello=document.getElementById(’hello’);
```

Trong một số trường hợp, cần duyệt qua cấu trúc cây từng bước một, mỗi nút DOM có một nút cha và nhiều nút con. Chúng có thể được truy cập bởi các thuộc tính parentNode và childNodes, thuộc tính parentNode trả về một đối tượng DOM node khác, trong khi childNodes trả về một mảng javascript:

Trích:

```
var children=empty.childNodes;
for (var i=0;i<children.length;i++){
…
}
```

Một cách khác để tìm kiếm là dựa trên loại thẻ HTML, dùng phương thức getElementsByTagName(). Ví dụ, document.getElementsByTagName(”UL”) sẽ trả về chuỗi tất cả các thẻ <UL> trong tài liệu.

Tạo DOM Node

Trong nhiều trường hợp cần tạo các nút mới và thêm nó vào tài liệu. JavaScript cung cấp một số phương thức để làm điều đó. Các phương thức chuẩn để tạo nút mới là document.createElement() và document.createTextNode(), phương thức createElement() có thể được dùng để tạo ra bất kỳ phần tử HTML nào, tham số là kiểu của loại thẻ HTML;
var childEl=document.createElement(”div”);
createTextNode() tạo một nút thể hiện qua một đoạn text, thường được tìm thấy trong các thẻ về heading, div, paragraph, và list item.
var txtNode=document.createTextNode(”some text”);

Chuẩn DOM coi các text node tách rời khỏi biểu diễn HTML. Chúng không có các stye để áp đặt cho trực tiếp và vì thế chúng yêu cầu ít bộ nhớ hơn.
Một nút khi được tạo ra phải được gắn vào tài liệu trước khi hiển thị trên trình duyệt, phương thức appendChild() được dùng để thực hiện điều này el.appendChild(childEl);
Ba phương thức createElement(), createTextNode(), và appendChild() cho phép thực hiện hầu hết các thao tác để thêm một nút vào tài liệu.

Một tool (editor) mã nguồn mở, rất cool, xem chi tiết tại:

Bài 6: Các công nghệ trong AJAX - DOM - Phần cuối.

Thêm style vào tài liệu

DOM cũng cung cấp các phương thức để chỉnh sửa style của các phần tử và tạo định dạng lại cho cấu trúc đã được định nghĩa trong stylesheet.
Mỗi phần tử trong trang web có thể có nhiều giao diện trực quan có thể được áp đặt qua DOM, như vị trí, chiều dài và rộng, mầu sắc, canh lề và đường viền. Các trình duyệt ngày nay đều cung cấp các ràng buộc JavaScript cho phép thay đổi giao diện mức thấp và áp đặt các định dạng một cách nhất quán và dễ dàng với các lớp CSS.
Thuộc tính className
Ví dụ đoạn code sau sẽ áp đặt các quy tắc biểu diễn của lớp declared cho một nút:

Trích:

```
hello.className=’declared’;
```

với hello tham chiếu tới một nút DOM.
Thuộc tính style
Ví dụ, đoạn mã sau bổ sung các thuộc tính style cho nút empty:

Trích:

```
empty.style.border=”solid green 2px”;
empty.style.width=”200px”;
```

Sử dụng thuộc tính innnerHTML

Các phần tử DOM của các trình duyệt web đều hỗ trợ một thuộc tính gọi là innerHTML, cho phép các nội dung kiểu string tùy ý được gán cho các phần tử theo thuộc tính này, như trong đoạn mã sau:

Trích:

```
function addListItemUsingInnerHTML(el,text){
el.innerHTML+=”<div class=’programmed’>”+text+”</div>”;
}
```

Trên đây ta vừa xét một cách sơ lược về JavaScript, CSS, và DOM. Chúng được tập hợp trong một công nghệ gọi là Dynamic HTML (DHTML), và có thể thấy Ajax sử dụng rất nhiều kỹ thuật DHTML.

Như vậy, các bạn đã nắm được 2 công nghệ dùng trong AJAX: CSS và DOM. Trong các phần tiếp theo chúng ta cùng nhau thảo luận về: XML và việc truyền dữ liệu bất đồng bộ (Giới thiệu về XML và XSLT, XMLHttpRequest) và có lẽ cũng nên nói qua một chút về JS (JavaScript).