Blog Post

HTML5 - DOM e HTML5

mai 09, 2012 às 12:54 por DiegoBlanco

1 classificação 16 comentários

Descrição

DOM E HTML5

DOM e HTML5

O Modelo de Objetos do Documento (DOM, na sigla em inglês) é a interface entre a linguagem Javascript e os objetos do HTML. DOM é o método padrão para construção de aplicações ricas com Javascript e é amplamente conhecido e utilizado. Neste capítulo, supondo que você já conhece DOM para HTML 4 ou XHTML, vamos nos focar na diferença entre as versões anteriores do DOM e a do HTML 5.

Por quê DOM?

Os primeiros navegadores a incorporar um motor de Javascript tinham alert, prompt, document. write e mais meia dúzia de maneiras de se interagir com o usuário. E só. A idéia de acessar a árvore de objetos do HTML trouxe poder às interfaces com o usuário na web. A idéia era tão boa que os fabricantes de navegadores não puderam esperar até que tivéssemos uma especificação padrão que atendesse suas necessidades, e criaram cada um seu próprio método de resolver o problema. Isso resultou em anos e anos de incompatibilidade, em que era preciso escrever uma versão de seus scripts para cada navegador. Queremos, com certeza, evitar uma nova guerra de padrões. Por isso recomendamos a você, por mais sedutor que pareça utilizar um recurso proprietário Javascript, que se atenha ao DOM.

Vamos às diferenças

1. getElementsByClassName

Esse é um sonho antigo de todo desenvolvedor Javascript. Com HTML5 você pode fazer:

destaques = document.getElementsByClassName('destaque')

E isso retornará todos os elementos do HTML que possuem a classe "destaque".

2. innerHTML

Outro sonho antigo que se torna realidade. A propriedade innerHTML é uma idéia tão boa que todos os navegadores atuais já a suportam há muito tempo e todo desenvolvedor web sabe usá-la. Apesar disso, ela nunca havia sido descrita como um padrão.

Se porventura você nunca viu a propriedade innerHTML em ação (puxa, onde você estava nos úl-timos dez anos?) saiba que ela contém uma string, o conteúdo HTML da página. E você tem acesso de leitura e escrita a essa propriedade. Veja um exemplo de innerHTML:

```
function adicionaItem(nome){

    document.getElementById('lista').innreHTML += '<li>'+nome+'</ li>'

}
```

3. activeElement e hasFocus()

O documento HTML5 tem uma nova propriedade, activeElement, que contém o elemento que possui o foco no momento. O documento também possui o método hasFocus(), que retorna true se o documento contém o foco. Seu usuário pode estar trabalhando com múltiplas janelas, abas, frames, ou mesmo ter alternado para outro aplicativo deixando o navegador com sua aplicação Javascript rodando em segundo plano. O método hasFocus() é uma conveniente maneira de tratar ações que dependem do foco na aplicação atual.

Veja um exemplo de script dependente de foco:

```
<!DOCTYPE html>
<htmllang=”pt-BR”>
<head>
<metacharset=”UTF-8” />
<title>Notifier</title>
<script>
functionnotify(text){
document.getElementById(‘msg’).innerHTML+=’<p>’+text+’</p>’
titleFlick()
}
functiontitleFlick(){
  if(document.hasFocus()){
document.title=’Notifier’
  return
}
document.title=document.title==’Notifier’?’* Notifier’:’Notifier’
setTimeout(‘titleFlick()’,500)
}
</script>
</head>
<body>
<input type=”button” id=”notify” value=”Notify in 5 seconds”
onclick=”notify(‘Will notify in 5 seconds...’);setTimeout(‘notify(\’Event shoot!\’)’,5000)” />
<divid=”msg”></div>
</body>
</html>
```

4. getSelection()

Os objetos document e window possuem um método getSelection(), que retorna a seleção atual, um objeto da classe Selection. A seleção tem, entre outros, os seguintes métodos e propriedades:

anchorNode

  O elemento que contém o início da seleção

focusNode

O elemento que contém o final da seleção

selectAllChildern(parentNode)

Seleciona todos os filhos de parentNode

deleteFromDocument()

Remove a seleção do documento

rangeCount

A quantidade de intervalos na seleção

getRangeAt(index)

Retorna o intervalo na posição index

addRange(range)

Adiciona um intervalo à seleção

removeRange(range)

Remove um intervalo da seleção

Intervalos de seleção

Você deve ter notado acima que uma seleção é um conjunto de intervalos, da classe Range. Cada intervalo possui, entre outros, os seguintes métodos e propriedades:

deleteContent()

Remove o conteúdo do intervalo

setStart(parent,offset)

Seta o início do intervalo para o caractere na posição offset dentro do elemento DOM parent

setEnd(parent,offset)

Seta o final do intervalo para o caractere na posição offset dentro do elemento DOM parent

Tanto os objetos Selection quanto os objetos Range retornam o texto da seleção quando convertidos para strings.

document.head

O objeto document já possuía uma propriedade body, uma maneira conveniente de acessar o elemento body do HTML. Agora ele ganhou uma propriedade head, maneira também muito conveniente de acessar o elemento head.

Selector API

A Selector API não é novidade do HTML5, é anterior a ele. Mas como ainda é desconhecida de parte dos desenvolvedores, convém dizer que ela existe, e que continua funcionando no HTML5. Com a selector API você pode usar seletores CSS para encontrar elementos DOM.

A Selector API expõe duas funções em cada um dos elementos DOM: querySelector e querySelectorAll. Ambas recebem como argumento uma string com um seletor CSS. A consulta é sempre feita na subtree do elemento DOM a partir do qual a chamada foi disparada. A querySelector retorna o primeiro elemento que satisfaz o seletor, ou null caso não haja nenhum. A querySelectorAll retorna a lista de elementos que satisfazem o seletor.

Veja, neste exemplo, um script para tabelas zebradas com Selector API:

```
<!DOCTYPE html>
<htmllang=”pt-BR”>
<head>
<metacharset=”UTF-8” />
<title>Zebra</title>
<style>
.zebraon{background:silver}
</style>
<script>
window.onload=function(){
  varzebrar=document.querySelectorAll(‘.zebra tbody tr’)
  for(vari=0;i<zebrar.length;i+=2)
   zebrar[i].className=’zebraon’
}
</script>
</head>
<body>
<tableclass=”zebra”>
<thead><tr>
<th>Vendedor</th> <th>Total</th>
</tr></thead>
<tbody><tr>
<td>Manoel</td> <td>12.300,00</td>
</tr><tr>
<td>Joaquim</td> <td>21.300,00</td>
</tr><tr>
<td>Maria</td> <td>13.200,00</td>
</tr><tr>
<td>Marta</td> <td>32.100,00</td>
</tr><tr>
<td>Antonio</td> <td>23.100,00</td>
</tr><tr>
<td>Pedro</td> <td>31.200,00</td>
</tr></tbody>
</table>
</body>
</html>
```

Características especiais de DomNodeList

As listas de elementos retornadas pelos métodos do DOM não são Arrays comuns, são objetos DomNodeList, o que significa que, entre outros métodos especiais, você pode usar list[0]ou list(0)para obter um elemento da lista. Também pode usar list["name"]ou list("name") para obter um objeto por seu nome. Duas adições interessantes do HTML5 ao usar este último método:

O objeto é buscado pelos atributos name ou id.
Uma lista de campos de formulário com o mesmo valor no atributo name (uma lista de radio buttons, por exemplo) será retornada caso mais de um objeto seja encontrado. Essa lista contém um atributo especial, value, muito conveniente. Ele contém o valor do

radio marcado e, ao ser setado, marca o radio correspondente.

Datasets

Você pode atribuir dados arbitrários a um elemento HTML qualquer, prefixando seus atributos com "data-". Por exemplo:

```
<img src="computador1.jpg" alt="Novo GNexius Forethinker" id="c1" data-processor="35Ghz" data-memory="4
```

Você pode acessar esses valores via Javascript, através do atributo dataset, assim:

```
var img=document.getElementById('c1')

proc=img.dataset.processor
```

As propriedades de dataset têm permissão de leitura e escrita.

Este treinamento faz parte do Microsoft Virtual Academy: http://www.microsoftvirtualacademy.com

---

marcas: Brazilian Portuguese, DOM, HTML 5, HTML5, Microsoft

Episódios relacionados

HTML5 - Novos Eventos DOM