Blog Post

HTML5 - Histórico de Sessão e API Storage

mai 16, 2012 às 11:47 por DiegoBlanco

2 classificações

Descrição

HISTÓRICO DE SESSÃO E API STORAGE

Histórico de Sessão e API Storage

Um dos grandes desafios de usabilidade ao se construir aplicações web com a tecnologia atual é apresentar um modelo de navegação consistente para o usuário. Duas grandes lacunas nos impediam de fazê-lo:

Não havia uma forma simples de fazer com que as ações locais do usuário numa página fossem refletidas na próxima. Por exemplo, se o usuário abre e fecha itens em um menu em árvore e em seguida navega para a próxima página, era muito difícil fazer com que o menu aparece no mesmo estado na segunda página.
Não havia uma forma simples de fazer com que as ações do usuário numa página Ajax respondessem corretamente aos botões de controle de histórico do navegador (voltar e avançar).
HTML5 traz formas simples de solucionar os dois problemas.

Histórico de Sessão

Você provavelmente conhece o objeto history do navegador e seus métodos go, back e forward. Ele nos permite, via javascript, um controle básico do histórico de navegação. O mesmo controle que o usuário, voltar e avançar.

O objeto history foi vitaminado no HTML5 com dois novos métodos:

pushState(data,title[,url]): acrescenta uma entrada na lista de histórico.
replaceState(data,title[,url]): modifica a entrada atual na lista de histórico.
Com isso, você pode acrescentar itens à lista de histórico, associando dados ou mesmo uma URL a eles. Por exemplo, digamos que você tenha três elementos de conteúdo em sua página e um script que exiba um por vez de acordo com os cliques do usuário no menu:

```
function showContent(n){
// Escondemos todos os elementos de conteúdo
for(var i=1;i<4;i++)
document.getElementById('cont'+i).style.display='none'
// Exibimos o elemento escolhido
document.getElementById('cont'+n).style.display='block'
}
```

Vamos fazer com que nosso script acrescente uma linha de histórico ao selecionar um elemento:

```
function showPage(n){
// Escondemos todos os elementos de conteúdo
for(var i=1;i<4;i++)
document.getElementById('cont'+i).style.display='none'
// Exibimos o elemento escolhido
document.getElementById('cont'+n).style.display='block'
}
function showContent(n){
// Mostramos o conteúdo escolhido
showPage(n)
// Salvamos a página atual no histórico
history.pushState({page:n},'Conteúdo '+n)
}
```

Fazendo isso, cada vez que o usuário escolher um item no menu, o elemento será exibido e uma linha será acrescentada no histórico. O usuário poderá acessar normalmente esses itens de histórico usando o botão de voltar do navegador. Cada vez que ele usar o histórico, será disparado um evento popstate. Assim, para que nosso script esteja completo, basta tratar esse evento:

```
function showPage(n){
// Escondemos todos os elementos de conteúdo
for(var i=1;i<4;i++)
document.getElementById('cont'+i).style.display='none'
// Exibimos o elemento escolhido
document.getElementById('cont'+n).style.display='block'
}
function showContent(n){
// Mostramos o conteúdo escolhido
showPage(n)
// Salvamos a página atual no histórico
history.pushState({page:n},'Conteúdo '+n)
}
// Quando o usuário navegar no histórico, mostramos a página
relacionada:
window.onpopstate=function(e){
if(e.state)
showPage(e.page)
}
```

localStorage e sessionStorage

Até o HTML4, quando precisávamos armazenar dados no agente de usuário que persistissem entre as páginas, usávamos Cookies. Cookies nos permitiam armazenar o status de um menu javascript que precisava ser mantido entre as páginas, lembrar o nome do usuário, o histórico de operações realizadas por ele ou a última vez que ele visitou nosso site.

Com o aumento da complexidade das aplicações baseadas em web, duas grandes limitações dos Cookies nos incomodam:

Interface complexa: o código para armazenar Cookies envolve complexos cálculos com datas e controle do nome de domínio.
Limite de armazenamento: alguns agentes de usuário permitiam o armazenamento de no máximo 20 Cookies, com apenas 4KB cada.
HTML5 traz uma nova maneira de armazenar dados no client, a API Storage. Um objeto Storage possui os métodos:

getItem(key): obtém um valor armazenado no Storage
setItem(key,value):guarda um valor no Storage
removeItem(key): exclui um valor do Storage
clear(): limpa o Storage

Estão disponíveis dois objetos no escopo global (window): localStorage e sessionStorage.

O objeto localStorage armazena os dados no client sem expiração definida. Ou seja, se o usuário fechar o navegador e voltar ao site semanas depois, os dados estarão lá. O sessionStorage armazena os dados durante a sessão atual de navegação.

O código para armazenar um valor na Storage se parece com isso:

```
localStorage.
setItem('userChoice',33)
```

E quando você precisar desse valor, em outra página:

```
localStorage.getItem('userChoice')
```

Essa interface já é muito mais simples que a de Cookies. Mas pode ficar melhor. Você pode usar o Storage como um array. Por exemplo:

```
if(!sessionStorage['theme']){
sessionStorage['theme']='oldfurniture';
}
```

Não há como isso ser mais simples! Além disso, o espaço de armazenamento sugerido pela documentação é de 5MB para cada domínio, resolvendo, acredito que por mais uma década, o problema de espaço de armazenamento local.

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

---

marcas: Brazilian Portuguese, HTML 5, HTML5, Microsoft, Offline, Storage

Explicação

Comentários fechados

Os comentários foram fechados porque este conteúdo foi publicado há mais de 30 dias, porém, se você deseja continuar a conversa, crie um novo thread nos nossos Fóruns ou Entre em Contato Conosco e informe-nos.