Desenvolvimento Web I

Aula 11 - JavaScript: Acessando Objetos - Parte 2

Apresentação

Nesta aula, continuaremos a trabalhar com objetos que são criados pelo navegador Web, ao carregar uma página HTML criada por você. Além disso, você vai aprender como criar dinamicamente elementos HTML (tags HTML) em uma página, mesmo após a página ter sido carregada, e ainda aprenderá a manipular e modificar os atributos dos elementos previamente definidos na página.

Bons estudos!

Vídeo 01 - Apresentação

Objetivos

- Manipular objetos da biblioteca de JavaScript.
- Manipular objetos do navegador e objetos de uso geral do JavaScript.

Biblioteca de objetos de JavaScript

Nesta aula, continuaremos a aprender sobre os objetos DOM HTML, além de compreender novos tipos de objetos, tais como **Objetos JavaScript** e **Objetos do Browser**. Cada um desses grupos de objetos serão apresentados com mais detalhes nas seções seguintes.

Objetos DOM HTML

Na aula passada, vimos exemplos e discutimos sobre os objetos **Link, Input Text** e **Form,** além de exemplos manipulando imagens. Nesta aula continuaremos estudando objetos DOM HTML.

Select e TextArea

O exemplo a seguir vai combinar o uso de dois novos elementos: **select** e **textarea**, além do **input text** que já foi visto em exemplos anteriores. A ideia desse exemplo é montar uma página contendo um campo de texto, um botão, uma lista e uma área de texto. Toda vez que o usuário clicar no botão, o conteúdo que está digitado no campo de texto vai ser adicionado como um item da lista.

Vídeo 02 - Objetos DOM

Além disso, toda vez que o usuário selecionar um item da lista, o conteúdo do item selecionado será acrescentado na área de texto. O código, apesar de mais complexo, contém somente conteúdo já tratado nos exemplos anteriores, mudando somente os elementos HTML que estão sendo manipulados.

O único trecho que deve ser realmente novo para você é o comando **document.createElement('option')**. Este permite criar novos elementos HTML de acordo com o nome passado como parâmetro. No caso do exemplo, o comando

serve para criar uma nova opção na lista de itens (tag **<option>** dentro da tag **<select>**).

```
<html>
<head>
   <script type="text/javascript">

      //Adiciona um novo option ao elemento select
      function adicionarItem(){
        var campoTexto=document.getElementById("item");
        var campoItens=document.getElementById("itens");

        var opcaoNova=document.createElement('option');
        opcaoNova.text = campoTexto.value;

        campoItens.add(opcaoNova);
      }

      //Concatena o texto do item selecionado na área de texto
      function adicionarTexto(){

        var campoAreaTexto=document.getElementById("areaTexto");
        var campoItens=document.getElementById("itens");

        var posicaoSelecionada = campoItens.selectedIndex;

        var novoTexto = campoItens[posicaoSelecionada].text;

        campoAreaTexto.value = campoAreaTexto.value + "\n" + novoTexto;

      }

   </script>
</head>
<body>
      <br/>
      <form>
         <input id="item" type="text" value="Digite algo!" />
         <input type="button" value="Adicionar Item" onClick="adicionarItem()"/>
         <br/>
         <select id="itens" onchange="adicionarTexto()">
           <option>Item Inicial</option>
         </select>
         <br/>
         <textarea id="areaTexto" rows="20" cols="30"></ textarea>
      </form>
   </body>
</html>
```

Table

O objeto Table representa uma tabela em HTML. Para cada tag <table> em um documento HTML, um objeto Table é criado pelo navegador Web. Como acontece no caso dos objetos que representam outras tags, esse objeto Table pode ser manipulado e alterado durante o acesso do usuário à página.

Com o uso de JavaScript é possível acessar o conteúdo de qualquer célula da tabela, sendo possível, inclusive, alterá-lo. Também é possível alterar a sua formatação: tamanho, bordas, espaçamentos etc. Além de tudo isso, é possível ainda acrescentar novas linhas e células a uma tabela já criada. Muita coisa pode ser feita em uma tabela através do uso de JavaScript, tudo vai depender do que for necessário à página. No quadro a seguir estão definidos os principais atributos do objeto Table.

Atributos	Descrição
cell[]	Retorna um vetor com todas as células de uma tabela (tag<td>)
rows[]	Retorna um vetor contendo todas as linhas de uma tabela (tag<tr>)
border	Representa a propriedade border da tabela.
caption	Representa a legenda da tabela.
cellPadding	Representa o atributo cellPadding definido na tag <table>.
cellSpacing	Representa o atributo cellSpacing definido na tag <table>.
width	Representa a largura da tabela.

Quadro 1 - Atributos do objeto Table
Fonte: Autor

- TableRow – Representa uma linha da tabela (tag <tr>);
- TableCell – Representa uma célula da tabela (tag <td>).

O exemplo mostrado a seguir demonstra a manipulação de tabelas em JavaScript. Nesse exemplo, o usuário deve digitar seu nome e CPF em dois campos textos e depois clicar no botão de inserir registro.

Ao fazer isso, uma nova linha na tabela é criada contendo duas células, cada uma com os valores digitados nos campos de texto. Como pode ser visto, a operação insertRow(), que é invocada no objeto Table (linha 13), cria um novo objeto do tipo TableRow (tag <tr>), adiciona-o após a posição definida e retorna-o. Nesse nosso exemplo, indicamos que a nova linha seja inserida após a última, ou seja, no índice igual à quantidade de linhas já existentes (length) do vetor rows do objeto Table. A nova linha é retornada e armazenada em uma variável chamada de novaLinha. Esse novo objeto é usado para criar duas células (linhas 16 e 17), através do método insertCell(). Por fim, o conteúdo da célula é modificado através do acesso aos seus atributos innerHTML (linhas 20 e 21).

```
<html>
  <head>
    <script type="text/javascript">
      function inserirRegistro(){
          var tabela = document.getElementById('registros');

          var nome = document.getElementById('inputNome').value;
          var cpf = document.getElementById('inputCPF').value;
          // Recupera a quantidade de linhas atuais
          var linhasTotais = tabela.rows.length;

          //adiciona a nova linha no fim da tabela
          var novaLinha = tabela.insertRow(linhasTotais);

          //adiciona duas células à linha criada
          var novaCelulaNome=novaLinha.insertCell(0);
          var novaCelulaCPF=novaLinha.insertCell(1);

          //define o conteúdo das células criadas
          novaCelulaNome.innerHTML=nome;
          novaCelulaCPF.innerHTML=cpf;
      }
    </script>
  </head>
  <body>
    <table id="registros" border="1">
      <tr>
        <th>Nome</th>
        <th>CPF</th>
      </tr>
      <tr>
        <td>Maria Bonita</td>
        <td>029.221.222-22</td>
      </tr>
    </table>
    <br />
    Nome: <input type="text" id="inputNome" /> <br />
    CPF: <input type="text" id="inputCPF" /> <br />
    <input type="button" onclick="inserirRegistro()" value="Inserir registro">
  </body>
</html>
```

Atividade 01

1. Rode os exemplos mostrados até agora nesta aula. Reporte se você encontrou alguma dificuldade. Você pode fazer uso do site w3schools (http://www.w3schools.com/js/default.asp) para testar os códigos através da opção “Try it yourself”.

2. Crie uma página HTML contendo um formulário com os seguintes campos textos: nome da cidade, população, estado e país. Ao se clicar em um botão de inserir, deve-se adicionar essas informações no final de uma tabela previamente criada na página.

3. Altere o exemplo anterior para criar uma funcionalidade de editar: ao se clicar no nome de uma cidade listada na tabela, os dados dela serão levados ao formulário para edição por parte do usuário. Ao se clicar no botão do formulário, as alterações feitas pelo usuário devem ser levadas de volta para a tabela, na linha que estava sendo editada. Dica: use o vetor rows para acessar as linhas já existentes.

Objetos JavaScript

Fazem parte deste grupo alguns objetos de uso geral, os quais geralmente também estão presentes na maioria das linguagens de programação, ou seja, que independem de elementos HTML. Esses objetos são:

- Array – É usado para armazenar múltiplos valores em uma única variável.

- Boolean – É usado para manipular e fazer conversões entre tipos booleanos e outros.

- Date – É usado para se trabalhar com datas e horas.

- Math – Possui diversas funções matemáticas.

- Number – É usado para manipular e fazer conversões entre tipos numéricos e outros.

- String – É usado para se manipular textos.
- RegExp – É usado para manipular expressões regulares em textos (casamento de padrão e funções de "search-and-replace").

Por questões de tempo, iremos abordar neste curso somente os objetos Math, Date e String, pois são os mais comumente utilizados em scripts. Para saber mais detalhes sobre os outros, é só acessar: http://www.w3schools.com/jsref/.

Date

O objeto Date é usado quando é necessário trabalhar com datas e horas. Ele representa uma data, incluindo informação de ano, mês, dia, hora, minuto, segundo e milésimo de segundo. Além disso, ele oferece diversas facilidades para manipulação de valores que representam datas, além de ter diversos métodos que permitem a conversão de e para outros tipos de dados (como string e inteiro). Um objeto Date pode ser criado de quatro maneiras diferentes, a saber:

- Criar uma data a partir da informação do relógio do computador onde o script está sendo executado (data e hora atuais).

```
var d = new Date();
```

- Criar uma data a partir de um número inteiro que representa a quantidade de milésimos de segundos desde 1º de janeiro de 1970(mesmo padrão usado no Java).

```
var d = new Date(milisegundos);
```

- Criar uma data a partir da conversão de uma string formatada no padrão ISO-8601 ou em padrão simplificado: YYYY-MM-DDTHH:mm:ss.sssZ, YYYY-MM-DD, YYYY-MM, etc.

```
var d = new Date(dataString);
```

- Criar uma data considerando um determinado ano, mês, dia, hora, minuto, segundo e milésimo de segundo.

var d = new Date(ano, mes, dia, hora, minuto, segundo, milisegundo);

O trecho de código abaixo exemplifica como apresentar na página a hora atual da máquina onde a página é carregada.

```
<html>
  <body>
   <script type="text/javascript">
      var d=new Date();
      document.write(d);
   </script>
  </body>
</html>
```

Além dos construtores, existem várias funções interessantes para manipular datas e horas. No Quadro 2, você pode encontrar os principais métodos existentes no objeto Date.

Método	Descrição
getDate()	Retorna o dia do mês(de 1-31)
getDay()	Retorna o dia da semana(de 0-6)
getFullYear()	Retorna o ano(4 dígitos)
getHours()	Retorna o hora(de 0-23)
getMilliseconds()	Retorna o milésimo de segundo(de 0-999)
getMinutes()	Retorna os minutos(de 0-59)
getMonth()	Retorna o mês(de 0-11)
getSeconds()	Retorna os segundos(from 0-59)
getTime()	Retorna a qauntidade de milésimos de segundo desde 1 de Janeiro de 1970
parse()	Converte uma string (representando uma data) em um objeto Date
setDate()	Altera o dia do mês
setFullYear()	Altera o ano
setHours()	Altera a hora
setMilliseconds()	Altera o milésimo de segundo
setMinutes()	Altera os minutos
setMonth()	Altera os mês
setSeconds()	Altera os segundos
toDateString()	Converte a data em um string legível
toLocaleString()	Converte a data em um string, considerando as convenções do idioma onde a pagina foi carregada

Quadro 2 - Métodos do objeto Date
Fonte: Fonte: Autor

O trecho de código a seguir é referente à implementação de um relógio digital, usando as funções do objeto Date. Como pode ser visto, a função calcularHora() recupera a data e a hora atual a partir do construtor padrão do objeto Date e depois

guarda em variáveis separadas as informações de hora, minuto e segundo (linhas 6 a 13).

```
1  <html>
2    <head>
3      <script type="text/javascript">
4        function calcularHora(){
5          var hoje = new Date();
6          var hora = hoje.getHours();
7          var minuto = hoje.getMinutes();
8          var segundo = hoje.getSeconds();
9
10         // adiciona 0 na frente do número caso o número seja <10
11         minuto=formata(minuto);
12         segundo=formata(segundo);
13
14         var horario = hora+":"+minuto+":"+segundo;
15
16         var relogio = document.getElementById('relogio');
17
18         relogio.innerHTML = horario;
19
20         setTimeout('calcularHora()',500);
21       }
22
23       function formata(i){
24         if (i<10){
25           i="0" + i;
26         }
27         return i;
28       }
29     </script>
30   </head>
31   <body onload="calcularHora()">
32     <div id="relogio"></div>
33   </body>
34 </html>
```

O minuto e segundo são formatados através do uso da função auxiliar **formata()**, que tem como objetivo acrescentar um zero na frente, caso o número só tenha um dígito (menor que 10). Depois, é criada uma variável chamada **horario**, que guarda a hora atual no formato: HORA:MINUTO:SEGUNDO (linha 15).

Por fim, o elemento HTML com id igual a "**relogio**" (linha 34) é recuperado, através do uso da função **document.getElementById()**, e o seu conteúdo é alterado para apresentar a hora atual no formato legível (linhas 17 a 19).

A última linha da função é uma chamada ao método **setTimeout()** (linha 21), que funciona como um agendamento de uma chamada de função. Em outras palavras, essa linha faz com que a função **calcularHora()** indicada como parâmetro seja chamada novamente daqui a 500 milésimos de segundos.

Math

O objeto **Math** oferece diversas funções matemáticas bem comuns, tais como as de seno, cosseno, máximo, mínimo, logaritmo, etc. Além disso, ele também possui algumas constantes muito usadas em operações matemáticas, tais como PI, E (Número de Euler), Raiz quadrada de 2, etc. O Quadro 3 apresenta mais detalhes das funções existentes.

Vídeo 03 - Objetos Math e Date

Método	Descrição
abs(x)	Retornar o valor absoluto de x
cos(x)	Retorna o cosseno de x
max(x,y,z,...,n)	Retorna o maior dos números
min(x,y,z,...,n)	Retorna o menor dos números
pow(x,y)	Retorna o valor x elevado a y
random()	Retorna um valor aleatório entre 0 e 1
round(x)	Arredondamento do número
sin(x)	Retorna o seno de x
sqrt(x)	Retorna a raiz quadrada de x
tan(x)	Retorna a tangente de x

Quadro 3 - Métodos do objeto Math
Fonte: Autor

O trecho de código a seguir exemplifica o uso de algumas funções matemáticas:

```
<html>
  <body>
    <script type="text/javascript">
      document.write(Math.round(0.60) + "<br />");
      document.write(Math.random() + "<br />");
      document.write(Math.abs(-8) + "<br />");
      document.write(Math.max(4,5,6,67,1) + "<br />");
      document.write(Math.sqrt(4.60));
    </script>
  </body>
</html>
```

String

Esses objetos são usados para manipular e armazenar textos. Eles oferecem diversos métodos relacionados à manipulação de textos. O Quadro 4 apresenta os principais métodos que podem ser usados em strings.

Método	Descrição
chartAt()	Retorna o caracter de uma determinada posição
concat()	Concatena strings
indexOf()	Retona o índice da primeira ocorrência de um texto nesta string
lastIndexOf()	Retona o índice da última ocorrência de um texto nesta string
replace()	Procura e substitui todas as ocorrências de uma palavra por outra palavra nesta string
split()	Quebra um string em vários strings de acordo com um separador
substring()	Extrai uma parte da string
toLowerCase()	Converte tudo para minúsculo
toUpperCase()	Converte tudo para maiúsculo

Quadro 4 - Métodos do objeto string. **Fonte:** Autor

Seguem abaixo dois trechos de código e seus respectivos resultados:

Trecho 1:

```
<script type="text/javascript">
    var str="Hello world!";
    document.write(str.indexOf("d") + "<br />");
    document.write(str.indexOf("WORLD") + "<br />");
    document.write(str.indexOf("world"));
</script>
```

Resultado Trecho 1:

```
10
-1
6
```

Trecho 2:

```
<script type="text/javascript">
    var str="Hello world!";
    document.write(str.substring(3)+"<br />");
    document.write(str.substring(3,7));
</script>
```

Resultado Trecho 2:

```
lo world!
lo w
```

Atividade 02

1. Rode os exemplos mostrados sobre os Objetos JavaScript. Reporte se você encontrou alguma dificuldade.
2. Crie uma página HTML com um formulário contendo os seguintes campos: texto 1, índice e letra. Em seguida, faça com que, após o usuário digitar o texto 1, um índice, e clicar em um botão do formulário, o valor do campo letra seja igual a letra do texto 1, cujo índice foi indicado no campo índice.

Objetos do Browser

Neste grupo, estão os objetos que representam o ambiente onde a página Web está executando, em particular o navegador. Esses objetos

Vídeo 04 - Objetos Browser

- Window – Representa uma janela aberta no navegador.
- Navigator – Representa o próprio navegador.
- History – Contém as URLs visitadas pelo usuário.
- Location – Contém informações relacionadas ao endereço sendo acessado.

Dica:

Neste curso, iremos abordar somente os três mais usados, que são **Window, Navigator** e **History**. Mais detalhes sobre os outros podem ser acessados em: <http://www.w3schools.com/jsref/>.

Window

O objeto **Window** representa uma janela do navegador aberta. No entanto, se o documento contém frames (**<frame>** ou **<iframe>**), o navegador cria um objeto **Window** para a página HTML e um adicional para cada frame da página. O quadro 5 apresenta diversos métodos que oferecem funcionalidades de manipulação da janela.

Método	Descrição
alert()	Abre uma janela de alerta com um botão de OK
blur()	Tira o foco da janela atual
close()	Fecha a janela atual
confirm()	Mostra uma caixa de diálogo com um botão OK e Cancela
createPopup()	Cria uma janela pop-up
focus()	Traz o foco para a janela atual
moveBy()	Move a janela em relação ao local atual dela
moveTo()	Move a janela para local específico
open()	Abre uma nova janela do navegador
print()	Imprime o conteúdo da página
prompt()	Abre uma janela para que o usuário digite algo
resizeBy()	Redimensiona a janela em relação ao seu tamanho atual
resizeTo()	Redimensiona a janela para uma altura e largura específica

Quadro 5 - Métodos do objeto Window
Fonte: Autor

O trecho de código abaixo demonstra o uso do método **confirm()**. Neste exemplo, é aberta uma caixa de diálogo com dois botões (OK e Cancela) e, dependendo da opção que o usuário clicar, o texto acima do botão (**<h2>**) é alterado.

```
<html>
   <head>
      <script type="text/javascript">
         function show_confirm(){
             var r = window.confirm("Pressione um botão!");

             var opcao = document.getElementById("opcao");

             if (r==true)
             {
               opcao.innerHTML = "Você pressionou OK!";
             }
             else
             {
               opcao.innerHTML = "Você pressionou Cancela!";
             }
         }
      </script>
   </head>
   <body>
      <h2 id="opcao"><h2>
      <input type="button" onclick="show_confirm()" value="Abre Caixa de Diálogo" />
   </body>
</html>
```

Navigator

Esse objeto representa o navegador propriamente dito e, dessa forma, permite acesso a informações sobre ele, como, por exemplo, o nome do navegador, a versão etc. O Navigator contém basicamente quatro atributos, conforme mostrado na quadro 6.

Método	Descrição
appName	Retorna o nome do navegador
appVersion	Retorna o versão do navegador
cookieEnabled	Determina se os cookies estão ou não habilitados
platform	Retorna a plataforma (sistema operacional) onde o navegador está executando

Quadro 6 - Atributos do objeto Navigator
Fonte: Autor

O trecho de código abaixo exemplifica o uso desses atributos.

```
1  <html>
2     <body>
3     <script type="text/javascript">
4          document.write("Browser: " + navigator.appName);
5          document.write("<br /><br />");
6          document.write("Versão do Browser: " + navigator.appVersion);
7          document.write("<br /><br />");
8          document.write("Cookies habilitados: " + navigator.cookieEnabled);
9          document.write("<br /><br />");
10         document.write("Plataforma: " + navigator.platform);
11    </script>
12    </body>
13 </html>
```

History

O objeto history armazena as URLs visitadas pelo usuário desde quando o navegador foi aberto e, dessa forma, permite navegar (ir e voltar) pelas páginas do histórico. Na verdade o objeto history tem uma lista de URLs visitadas por cada Aba (Tab) do navegador, ou seja, cada Aba tem sua lista de visitas e portanto tem um funcionamento distinto dos botões Voltar e Avançar, que funciona somente no histórico de navegação daquela Aba. Páginas visitadas antes de uma nova Aba ser aberta não são utilizadas pelo objeto history. Esse objeto possui basicamente três métodos, são eles:

- back() – Volta para a página anterior do histórico.
- forward() – Vai para a página seguinte do histórico.
- go() – Vai para uma página qualquer do histórico. Recebe como parâmetro o número do item do histórico, sendo go(-1) igual a back() e go(1) igual a forward().

Abaixo, segue um exemplo usando os três métodos:

```
<html>
  <head>
    <script type="text/javascript">
      function goBack2(){
        window.history.go(-2)
      }
      function goBack(){
        window.history.back()
      }
      function goForward(){
        window.history.forward()
      }
    </script>
  </head>
  <body>
    <input type="button" value="Voltar 2 página" onclick="goBack2()" />
    <input type="button" value="Voltar 1 página" onclick="goBack()" />
    <input type="button" value="Prosseguir 1 página" onclick="goForward()" />
  </body>
</html>
```

Chegamos ao fim da nossa aula! Concluímos nossos estudos sobre JavaScript, bem como Servlets e JSP. Que tal agruparmos todo esse conhecimento adquirido em um projeto bacana? Até a próxima aula!

Atividade 03

---

1. Rode os exemplos mostrados sobre os Objetos JavaScript. Reporte se você encontrou alguma dificuldade.
2. Crie uma página HTML com dois botões: um para abrir a própria página em um popup e outro com um botão para fechar a janela atual.
3. Crie uma página HTML com um formulário que pede dados X e Y para movimentação da janela e uma indicação se os dados são a nova posição atual ou se são dados para deslocamento (valores negativos indica mover janela para cima e/ou para esquerda). Ao preencher esses dados e clicar em um botão de movimentar, a janela deve ser reposicionada de acordo com o indicado pelo usuário.

Resumo

Nesta aula você aprendeu como manipular objetos JavaScript, que são criados pelo navegador, ao carregar sua página Web. A partir dos exemplos mostrados, você consegue ter uma noção do poder que JavaScript dá ao programador para tornar a página mais interativa e atrativa para o usuário.

Autoavaliação

1. Indique quais objetos DOM HTML você aprendeu hoje e como você pode manipulá-los, quais recursos estão disponíveis, etc.
2. Indique quais objetos JavaScript de uso geral você aprendeu hoje e como você pode manipulá-los, quais recursos estão disponíveis, etc.
3. Indique quais objetos do navegador você aprendeu hoje e como você pode manipulá-los, quais recursos estão disponíveis, etc.

Referências

JAVASCRIPT and HTML DOM reference. Disponível em: <http://www.w3schools.com/jsref/>. Acesso em: 17 set. 2012.

W3C. Disponível em: <http://www.w3.org/>. Acesso em: 17 set. 2012.

W3SCHOOL. Disponível: <http://www.w3schools.com/>. Acesso em: 17 set. 2012.