



Programação em Ambiente Web I

Aula 4 – Formulários HTML; Javascript

Prof. Msc Cleyton Slaviero

`cslaviero@gmail.com`

Agradecimentos ao Prof. Jivago Medeiros (UFMT-Cba)
Fabiano Taguchi (UFMT-Roo)

Formulários e HTML

- A especificação da linguagem HTML inclui uma série de elementos destinados a construção de formulários para páginas e sistemas web.
- Esses elementos são utilizados, principalmente, para o envio de dados/informações/arquivos/etc por parte dos usuários.
- Tornando a utilização de formulários imprescindível em sistemas web.

Tag <form>

- A tag <form> é utilizada para descrever e delimitar a existência de um formulário no documento HTML. Sintaxe básica:

<form>

...

</form>

- Importante: permite-se vários formulários por página, porém, não é permitido aninhá-los. Geralmente, considera-se apenas o primeiro.

Atributos

- Principais atributos:
- **id**: String com identificação única do elemento
- **action**: URI que define para “onde” será enviada (submetida) as informações do formulário
- **method**: Define se os dados serão enviados incorporados ao corpo do formulário (POST) ou à URL (GET).

Mais informações e atributos sobre formulários:

<https://developer.mozilla.org/docs/Web/HTML/Element/form>

Componentes de um formulário

- Elementos que são tradicionalmente inseridos entre as tags `<form>...</form>` para permitir que o usuário entre com dados, a citar:
 - `input` [text | password | radio | checkbox | reset | submit | hidden]
 - `textarea`
 - `select`
 - `optgroup`
 - `option`
 - `button`
 - `label`
 - `fieldset`
 - `legend`

Importante: Assim como nos demais elementos HTML, também é possível aplicar aos elementos de formulário uma grande gama de estilos utilizando CSS

Javascript

A yellow square containing the letters 'JS' in a bold, black, sans-serif font, representing the JavaScript logo.

Javascript

- JavaScript (JS) é uma linguagem de programação interpretada, baseada em *script* e comumente utilizada em navegadores web (*client-side*).
- É uma linguagem multi-paradigma, adotando diferentes paradigmas, entre esses, alguns aspectos de orientação a objetos. É fracamente tipada com tipagem dinâmica.
- Proposta inicialmente para o navegador Netscape em 1995 e adotada parcialmente pelo Internet Explorer em 1996.

História

- 1994 – Lançamento do (Mosaic) Netscape
 - Era necessária uma linguagem que auxiliasse desenvolvedores
- Nome original: **Mocha**
 - Criada em 10 dias (!) por Brendan Eich, em Maio/95
 - Escolhido por Marc Andreessen, fundador da NetScape
- Setembro/95
 - Livescript
- Dez/95
 - Javascript (jogada de marketing!)
 - Lançada junt com o Netscape 2.0

História

- 1996
 - Jscript (Microsoft) – versão do Javascript para IE
 - Problemas de compatibilidade
- 96-97
 - ECMA (*European Computer Manufacturers Association*)
 - Padrão ECMAScript
- 98 - ECMA2 , 99 – ECMA3
- 2005
 - Brendan Eich e Mozilla se juntaram a Ecma novamente
 - Douglas Crackford - Ajax
- Jun/2016
 - ECMAScript 7



Inserindo no navegador

- Os scripts JavaScript são inseridos nos documentos HTML utilizando a tag `<script>`
- Um script JS pode ser definido diretamente entre a tag, ex:
 - `<script> alert('Olá'); </script>`
- Ou ser oriundo de um arquivo externo, usualmente com a extensão .js, ex:
 - `<script src="meu-script.js"></script>`

Variáveis

- Não precisamos dizer o tipo

```
var a = 123;
```

- E a tipagem é **dinâmica** (podemos alterar o tipo da variável em tempo de execução):

```
a = "agora sou uma string";
```

Funções

- Usa-se a palavra reservada `function` e não é necessário “tipar” o retorno e os parâmetros

```
function soma(a,b) {  
    var c = a+b;  
    return c;  
}
```

Funções

- Podemos definir uma função "*dentro de uma variável*", e depois instanciá-la:

```
var A = function () {  
    console.log("Sou uma classe?");  
}
```

```
b = new A ();
```

```
c = new A ();
```

Prototype

- Como vimos, JS é uma linguagem multi-paradigma cuja a orientação a objetos implementada é baseada em **prototipação**. Isso significa que podemos fazer alterações nas "classes" durante a execução e todas as instâncias recebem essas alterações, exemplo:

```
A.prototype.ola = function () {  
    alert("Olá Mundo!");  
}
```

```
c.ola();
```

Arrays

- Usualmente, declaramos arrays em JS de duas formas:

```
var arr = ["a", "b", "c", "d"];
```

ou

```
var arr = new Array("a", "b", "c", "d");
```

Arrays

- Os valores do array podem ser acessados utilizando:

`arr[0], arr[1],...`

OU

`arr.1, arr.2, ...`

- Sendo "uma variável um array", podemos utilizar métodos como:

`push(), pop(), shift(), splice()`, etc

- E propriedades como `.length`

Outros métodos e propriedades:

http://www.w3schools.com/js/js_array_methods.asp



Universidade Federal
de Mato Grosso
Campus Rondonópolis

Objetos

- Objetos podem possuir métodos e propriedades
- A forma recomendada de declarar objetos é:
`var obj = {nome : “João”, idade : “21”, curso : “SI”};`
- Os valores de um objeto pode ser acessados utilizando:
`obj.nome, obj.idade, obj.curso`
ou
`obj['nome'], obj['idade'], obj['curso']`
- Objetos aceitam que as “propriedades sejam funções” (métodos)

Objetos

```
var aluno = {  
    nome : "João",  
    idade : "21",  
    curso : "SI",  
    correr : function () {  
        if (this.idade <= 30) {  
            console.log(obj.nome+" Corre muito!");  
        }  
        else {  
            console.log(obj.nome+" Corre pouco!");  
        }  
    }  
};
```



Manipulação do DOM

- O que é o **DOM** ?
 - *Document Object Model*
 - É a árvore dos elementos (objetos) renderizados (exibidos) pelo navegador.
 - Após o carregamento de um documento HTML o navegador gera o objeto *document* que contém todos os elementos da página.
 - A linguagem JavaScript fornece um conjunto de métodos para a manipulação do DOM.

document.getElementById()

- O método `getElementById()` é utilizado para recuperarmos um elemento no document (DOM) pelo seu atributo id

```
<p id='paragrafo'>...</p>
```

```
document.getElementById('paragrafo');
```

- Era um dos principais métodos para manipulação do DOM antes do surgimento e popularização das bibliotecas JavaScript.

`document.getElementById()`

- Após recuperarmos um elemento, podemos fazer “qualquer coisa” com ele, por exemplo:
- Alterar estilo
- Acessar/atualizar atributos
- Remover do DOM
- ...

document.getElementById()

- Exemplo:

```
var elem = document.getElementById('paragrafo');
```



A partir desse instante, a variável **elem** faz referência ao elemento com id “paragrafo” que se encontra no DOM do documento HTML.

Assim, utilizando o elem é possível, por exemplo, retornar, ou alterar o texto do elemento ou mesmo retornar ou alterar a borda do elemento.



```
elem.textContent;  
elem.style.backgroundColor="#006600";  
;
```



Manipulação de atributos

- A linguagem JavaScript também fornece métodos para a manipulação de atributos dos elementos, entre eles:

`getAttribute("atributo")`

que retorna o valor do "atributo", por exemplo:

`elem.getAttribute("id");`

`elem.getAttribute("type");`

`elem.getAttribute("placeholder");`

- Alguns atributos devem ser acessados diretamente, sem utilizar o método `getAttribute`, exemplo:

`elem.value;` ou ainda

`elem.value = "novo valor";`



Manipulação de atributos

- Outros métodos importantes na manipulação de atributos:

```
setAttribute(“atributo”, “valor”);
```

Altera o **valor** do **atributo**, caso o elemento ainda não possua o **atributo**, o **atributo** é inserido com o **valor**.

```
removeAttribute(“atributo”);
```

Remove do elemento o **atributo**.



Coleções de objetos

- Conforme vimos, o método `document.getElementById()` faz referência a um único objeto.
- Porém, também é possível manusear coleções de objetos do DOM

`getElementsByTagName("tag")`

- Propriedade `.children`
 - Retorna os nós filhos de um elemento no DOM

Coleções de objetos

- Seja o seguinte HTML:

```
<p id="paragrafo-1">Lorem ipsum dolor sit amet</p>  
<p class="paragrafo">consectetur adipiscing elit</p>  
<p class="paragrafo">Nam fringilla felis et efficitur</p>  
<p id="paragrafo-2">Nunc lobortis in eros sed</p>
```

Poderíamos utilizar o seguinte JavaScript para referenciar todos esses elementos:

```
var paragrafos =  
    document.getElementsByTagName("p");
```

Coleções de objetos

- Seja o seguinte HTML:

```
<ul id="lista">  
  <li>Lorem ipsum dolor sit amet</li>  
  <li>consectetur adipiscing elit</li>  
  <li>Nam fringilla felis et efficitur</li>  
  <li>Nunc lobortis in eros sed</li>  
</ul>
```


ou ainda:

```
var itens = document.getElementById("lista").children;
```


Coleções de objetos

- Em todos os casos apresentados, as variáveis *paragrafos* e *itens* são **arrays** e não podem ser “acessadas diretamente”, como fizemos antes:

```
paragrafos.textContent;  
itens.style.backgroundColor="#006600";
```

 **ERRADO!**

```
paragrafos[0].textContent;  
itens[2].style.backgroundColor="#006600";
```

 **CORRETO!**

Laços e iterações

- Como manipulamos arrays?
- Javascript possui vários tipos de laços de repetição
 - Mais usados
 - `for`
 - `do...while`
 - `while`
 - `for...in`

Laço for

- `for ([expressaoInicial]; [condicao]; [incremento])
 declaracao`

- Exemplo

```
for (var i=0;i<itens.length;i++) {  
    ...  
}
```

Laço do...while

- Repete até que a condição especificada seja falsa

Exemplo:

```
var i = 0;  
do  
    ...  
    i = i+1;  
while (i<itens.length) ;
```

Laço while

- Repete o laço enquanto condição for verdadeira

```
n = 0 ;  
x = 0 ;  
while (n < 3)  
{  
  
    n++ ;  
    x += n ;  
  
}
```



Laço for...in

- Executa iterações a partir de uma variável específica

```
for (variavel in objeto)
{
    declaracoes
}
```

Coleções de objetos

- Ou ainda, podemos iterar sobre o array

```
for (i=0;i<paragrafos.length;i++) {  
    console.log(paragrafos[i].textContent);  
}
```

OU

```
for (i=0;i<itens.length;i++) {  
  
console.log(itens[i].style.backgroundColor="#006600");  
}
```

Coleções de objetos

- Outros métodos para nos referenciarmos a coleções de elementos do **DOM**:
- ***getElementsByClassName*** ('className') : Retorna um array de objetos de acordo com o nome da classe passada como parâmetro
- ***querySelector*** ('selector') : Retorna o **primeiro** elemento correspondente ao seletor CSS passado como parâmetro.
- ***querySelectorAll*** ('selector') : Retorna um array com todos os elemento correspondente ao seletor CSS passado como parâmetro.



Eventos

- A linguagem Javascript fornece um vasto conjunto de recursos para trabalharmos com **eventos**:
- Os eventos geralmente estão relacionados a **interações do usuário**:
 - Ao clicar em um objeto, ao pressionar uma tecla do teclado, ao passar o mouse sobre um objeto, entre outros.
- Ou ainda **relacionados ao documento HTML**:
 - Ao carregar página, ao sair da página, ao redimensionar a página, entre outros.

Eventos

- Alguns exemplos de eventos:
 - *onclick*: Disparado quando há um **click do mouse**
 - *onload*: Disparado quando a **página é carregada**
 - *onkeydown*: Disparado quando uma **tecla** do teclado é **pressionada**.
 - *onmouseover*: Disparado quando o **ponteiro do mouse passa sobre** um objeto.
 - *onfocus*: Disparado quando um **campo** de formulário **recebe o foco** (por exemplo o cursor é colocado em um campo de texto)
 - *onfocusout*: Disparado quando um **campo perde o foco**.

Eventos

- Alguns exemplos de eventos:
 - *oninput* / *onchange*: Disparado quando há uma **entrada / alteração de dados em um campo** de formulário.
 - *onsubmit*: Disparado quando um **formulário é submetido** (enviado).
- Lista de eventos Javascript
 - http://www.w3schools.com/jsref/dom_obj_event.asp
 - <https://developer.mozilla.org/en-US/docs/Web/Events>

- Podemos manipular eventos diretamente nos elementos HTML por meio de atributos:

```
<span id="span-1" onclick="alert('Fui clicado!');">  
    Lorem ipsum dolor sit amet  
</span>
```

```
<span id="span-2" onmouseover="alert('passaram o  
mouse!');">  
    Nunc lobortis in eros sed  
</span>
```

Manipulação de eventos

- Ou ainda, diretamente no JavaScript:

```
<script>
    document.getElementById("span-1")
        .addEventListener("click",function() { alert('fui clicado!'); });
</script>
```

```
<script>
    document.getElementById("span-2")
        .addEventListener("mouseover",function(){alert('passaram o mouse!');});
</script>
```


Manipulação de eventos

- Observe que:

Utiliza-se uma função anônima

`document.getElementById("span-1")
 .addEventListener("click",function() { alert('fui clicado!'); }));`

O prefixo *on* é descartado

- Para removermos eventos utilizamos o método `removeEventListener` exemplo:

```
document.getElementById("span-1")  
    .removeEventListener("click", funcaoClick);
```



Importante: não funciona para funções anônimas

Exercícios

- Recrie este formulário

Deixe um comentário

Nome:

E-mail:

Website:

Comentário:

Observação: Seu comentário será publicado após passar pelo moderador

Enviar Comentário

Atividade

- Em sua página HTML das primeiras aulas, crie um formulário HTML com as seguintes informações
 - Nome (máximo 50 caracteres, 50 caracteres aparecendo)
 - Sobrenome (max 100 caracteres, 100 caracteres aparecendo)
 - Nome de usuário (max 20 caracteres)
 - Senha
 - Data de nascimento
 - Endereço
 - Bairro
 - Cidade
 - Estado
 - Botão **Enviar**
- Todos os campos devem ser devidamente identificados e organizados semanticamente
- Use CSS para alterar a seu gosto o formulário
- O formulário deve ter um botão que leva para a mesma página, usando o método GET e que verifica se a senha possui mais de 8 caracteres