PROGWEBI

PROF. MSC CLEYTON SLAVIERO
CSLAVIERO@GMAIL.COM

Aula 4

Formulários HTML; Javascript

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

Atividade ProgWeb I


```
#moustache {
position: bottom;
align: center; }
```

Texto 1
Lorem ipsum Lorem ipsum Lorem ipsum

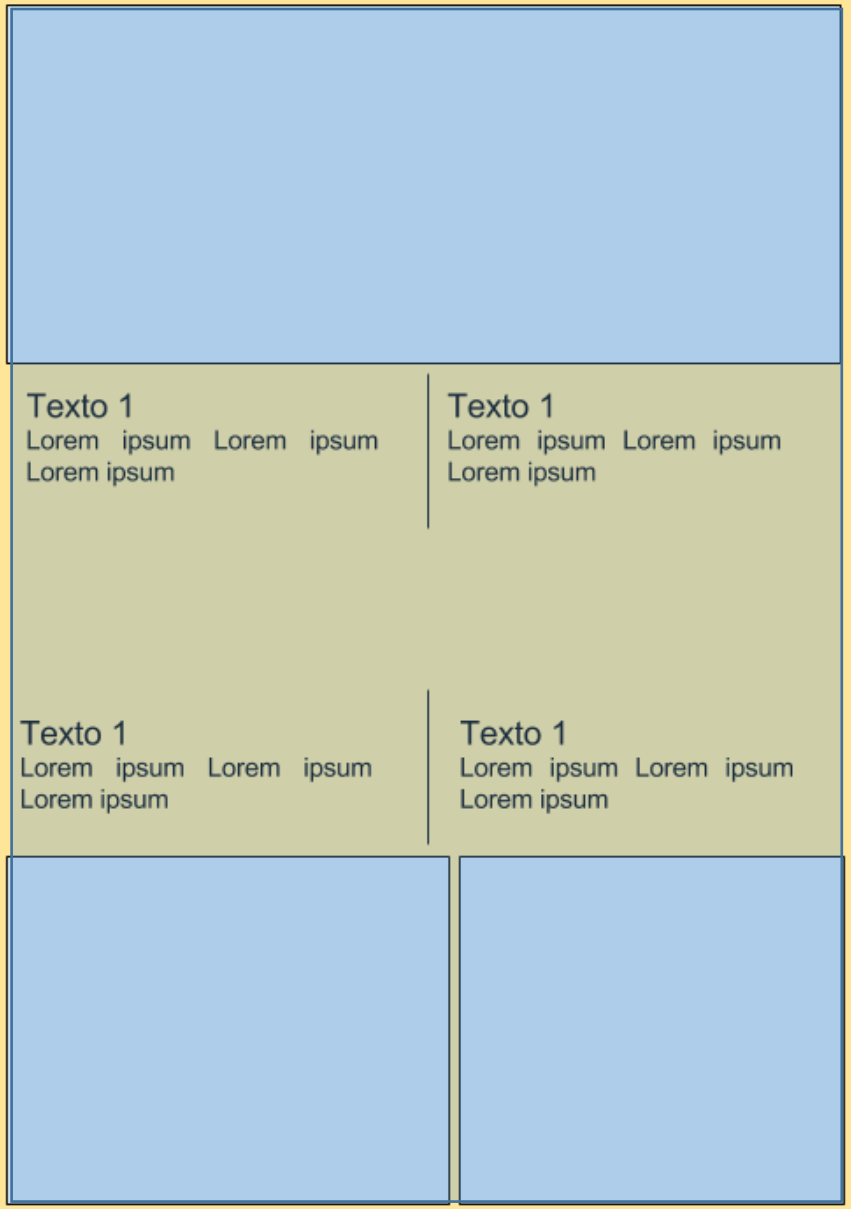
Texto 1
Lorem ipsum Lorem ipsum Lorem ipsum

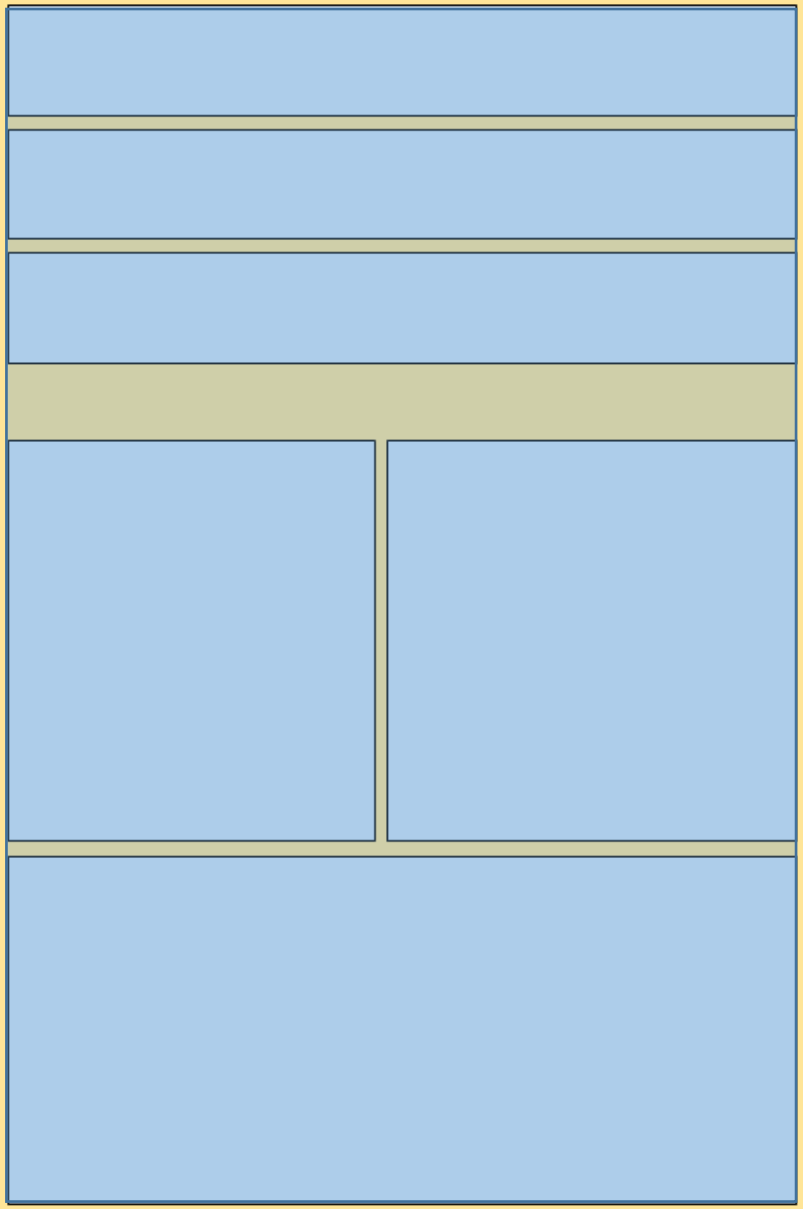
Texto 1
Lorem ipsum Lorem ipsum Lorem ipsum

Texto 1
Lorem ipsum Lorem ipsum Lorem ipsum

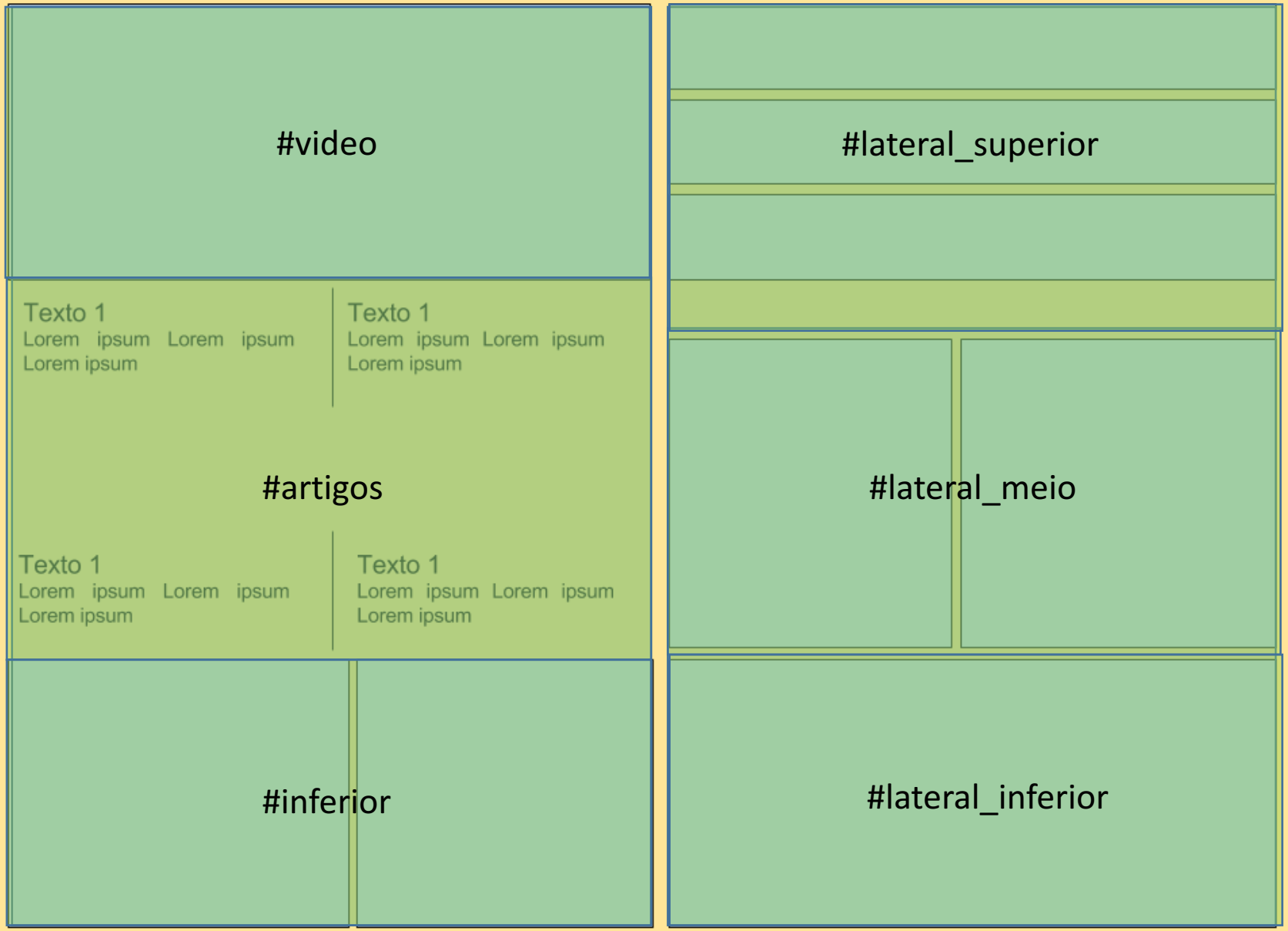
#esquerda

#direita





Universidade Federal de Mato Grosso
Campus Rondonópolis


#video
Texto 1
Lorem ipsum Lorem ipsum
Lorem ipsum
Texto 1
Lorem ipsum Lorem ipsum
Lorem ipsum
#artigos
Texto 1
Lorem ipsum Lorem ipsum
Lorem ipsum
Texto 1
Lorem ipsum Lorem ipsum
Lorem ipsum
#inferior

#lateral_superior
#lateral_meio
#lateral_inferior

Formulários e HTML

- A especificação da linguagem HTML inclui uma série de elementos destinados a construção de formulários para páginas e sistemas web.
- Esse elementos são utilizados, principalmente, para o envio de dados/informações/arquivos/etc por parte dos usuários.
- Tornando a utilizando 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).
    - GET: Envia dados junto com o endereço
    - POST: envia dados no cabeçalho da requisição

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:
- input [text | password | radio | checkbox | reset | submit | hidden]
  - Atributos: value, disabled, size,maxlength
- Textarea
  - Atributos: rows e cols
- select
  - optgroup
  - option
- button
- Label
  - Atributo for combinando com id do input!
- 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

Universidade Federal
de Mato Grosso
Campus Rondonópolis

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 Andresseen, 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 Manufacters 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";
```

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
}
```


Universidade Federal
de Mato Grosso
Campus Rondonópolis

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”);
```


Universidade Federal de Mato Grosso
Campus Rondonópolis

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

Condicionais

- Condicionais utilizam o formato

```
if (condição) { ... }[else { ...}]
```

Exemplo:

```
if (valor1 > valor2) {
    valor = valor1;
} else if (valor1<=valor2){
    valor = valor2;
}
```

Exercícios (sala)

- Escreva um código que mostre os números ímpares entre 1 e 10.
- Escreva um código que calcule a soma de 1 até 100. (obs: a resposta é 5050)
- Crie um Array igual ao abaixo e mostre apenas os nomes das pessoas que tenham 4 letras.
  - var pessoas = ["João", "José", "Maria", "Sebastião", "Antônio"];
-

Manipulando HTML com Javascript

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 contem 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 e encontra-se 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";
```


Universidade Federal de Mato Grosso
Campus Rondonópolis

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>
```

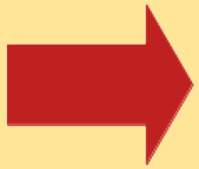
ou ainda:

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

Coleções de objetos

- Em todos os casos apresentados, as variáveis *paragrafos* e *itens* são arrays e <u>não podem</u> ser "acessadas diretamente", como fizemos antes:

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



ERRADO!

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

CORRETO!

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
- ***getQuerySelector*** *('seletor')*: Retorna o **primeiro** elemento correspondente ao seletor CSS passado como parâmetro.
- ***getQuerySelectorAll*** *('seletor')* : Retorna **um array** com todos os elemento correspondente ao seletor CSS passado como parâmetro.

Adicionando objetos HTML ao DOM

- É possível também criar elementos HTML
- Uma forma é adicionar diretamente o código em HTML do elemento, usando a propriedade `innerHTML`
- **Outra forma é utilizando**

```
document.createElement(”elemento”);
```

- **Posteriormente, você precisa adicionar o elemento ao DOM, usando** `appendChild(objeto)` **no elemento que quiser adicionar**
  - **Outra forma é usando**
    `elementoPai.insertBefore(novo_elemento,elementoDeReferencia)`

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

Eventos

- 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

Manipulação de eventos

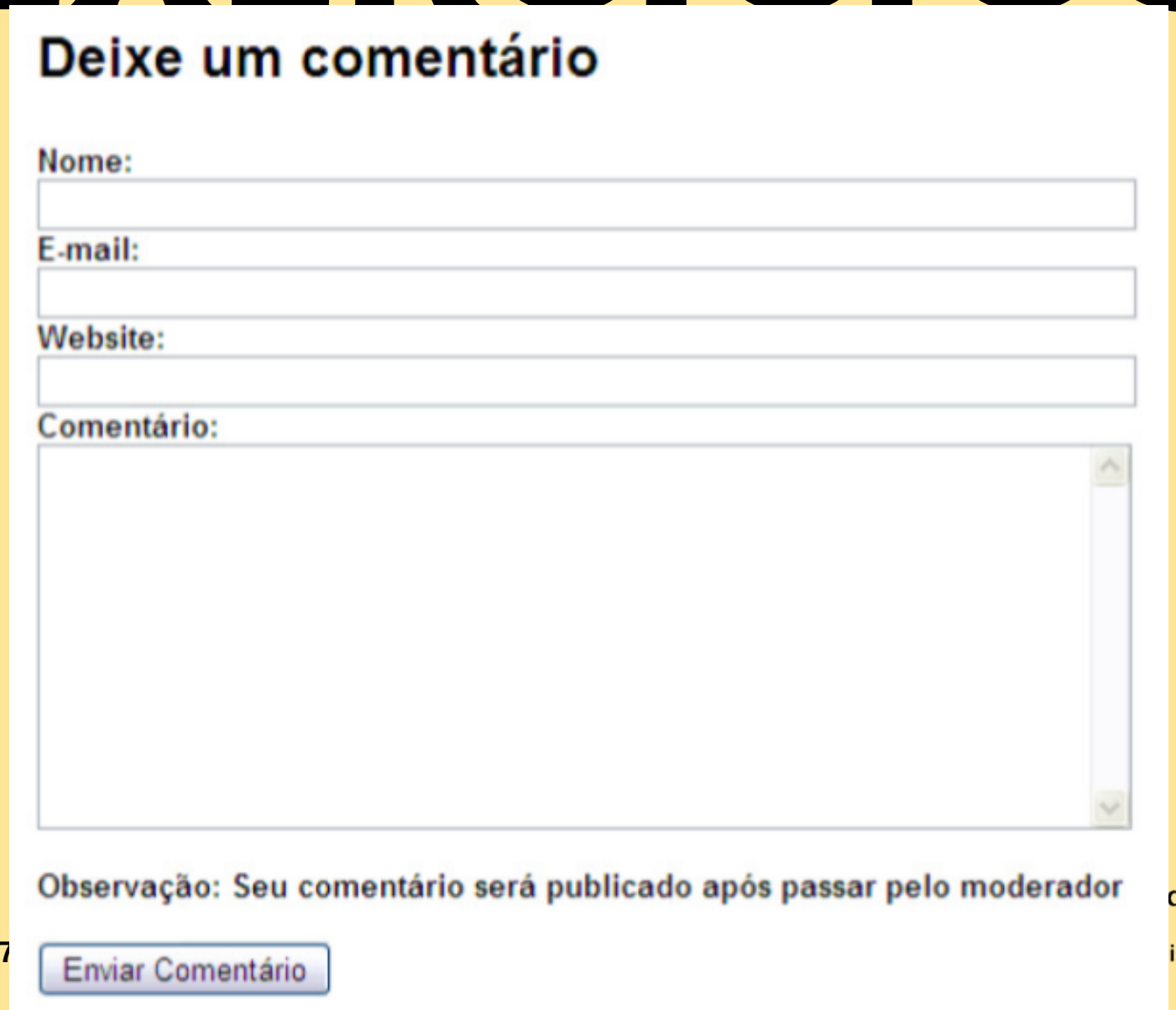
- Para removermos eventos utilizamos o método removeEventListener exemplo:

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

Importante: não funciona para funções anônimas

Exercícios

- Recrie este formulário



Exercícios
JS – DOM

- Em Javascript
  1. Faça um algoritmo que receba um objeto com um conjunto de informações (nome, endereço, telefone, etc) e exiba na tela as informações, formatadas (estilizadas)
     1. Pegue as informações de um formulário e exiba acima dele (ou em um alert!)
  2. Faça um algoritmo que, ao posicionar (hover) o mouse em um elemento, preencha ele com alguma cor aleatória.

  3. Faça um algoritmo que formate dados numéricos para valores em real

     Dica: use replace() e toFixed(), se necessário

Exercícios

- 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