# Programação II

Javascript

# O que é JavaScript

- Não é Java!
- Constitui um dos pilares do desenvolvimento Web
  - HTML: estrutura, marcação do conteúdo
  - CSS: layout, formatação, aparência
  - JavaScript: comportamentos dinâmicos executados no cliente
    - Criar, remover, exibir e ocultar elementos;
    - Modificar dinamicamente valores de atributos e propriedades CSS;
    - Responder a eventos;
    - Etc.
- Linguagem interpretada pelo navegador.
  - Entretanto, começa também a ser bastante utilizada do lado do servidor através de ambientes como o node.js.

# Inserindo JavaScript (internamente)

- Para inserir um código JavaScript em uma página, é necessário utilizar a tag <script>:

```
<script>
alert("Olá, Mundo!");
</script>
```

- Você pode se deparar com exemplos iniciando com `<script type="text/javascript">` mas o atributo `type` não é mais necessário, já que Javascript é a linguagem de script padrão da Web.
- Podemos inserir diversos scripts na página.

# Inserindo JavaScript (externamente)

- Utiliza-se a mesma tag `<script>`, mas com o atributo `src` apontando para o caminho do arquivo em questão (extensão .js).

```
<script src="js/meuarquivo.js"></script>
```

  O arquivo externo (*.js) não deve conter a tag `<script>`.

- Este método possui a vantagem de manter separado o HTML e o Javascript, facilitando a manutenção e possibilitando que o carregamento seja mais rápido, devido ao cache dos arquivos *.js.

# Onde inserir a tag <script>?

- A tag `<script>` pode ser inserida no `<head>` ou no `<body>`, sendo a primeira opção a mais comum.
- Dependendo do código, o melhor local é antes do fechamento da tag `<body>`.
  - O navegador, ao encontrar uma tag `<script>`, precisa executar o código especificado dentro da tag (ou do arquivo externo referenciado pelo atributo 'src'), bloqueando assim a renderização do restante da página (o que impacta na velocidade de carregamento da mesma).
  - Além disso, se existem elementos abaixo da tag `<script>` que são manipulados por esse código, é necessário adicionar eventos indicando que a página já foi carregada completamente, caso contrário, o código não funcionará corretamente.

# Saídas

- Um script pode exibir dados de saída das seguintes formas:
  - Dentro de um elemento qualquer da página, usando a propriedade innerHTML:

    ```
    document.getElementById("demo").innerHTML = 5 + 6;
    ```

  - Diretamente na saída HTML (para efeito de testes), usando document.write():

    ```
    document.write(5 + 6);
    ```

  - Em uma caixa de diálogo, usando window.alert() ou simplesmente alert():

    ```
    window.alert(5 + 6);
    ```

  - Escrevendo no console do navegador com console.log():

    ```
    console.log(5 + 6);
    ```

    - Obs: o console encontra-se nas ferramentas do desenvolvedor do navegador (F12)

# Tipos de Dados

- String:

```
var empresa = "Teste";
empresa.length; // tamanho da string
```

- Boolean

```
var b1 = true;
b1 = false;
```

- Number:

```
var vinte = 20;
var pi = 3.14159;
```

- Funções de conversão de string para number:

```
var inteiro = parseInt("10");
var float = parseFloat("10.22");
```

# Funções

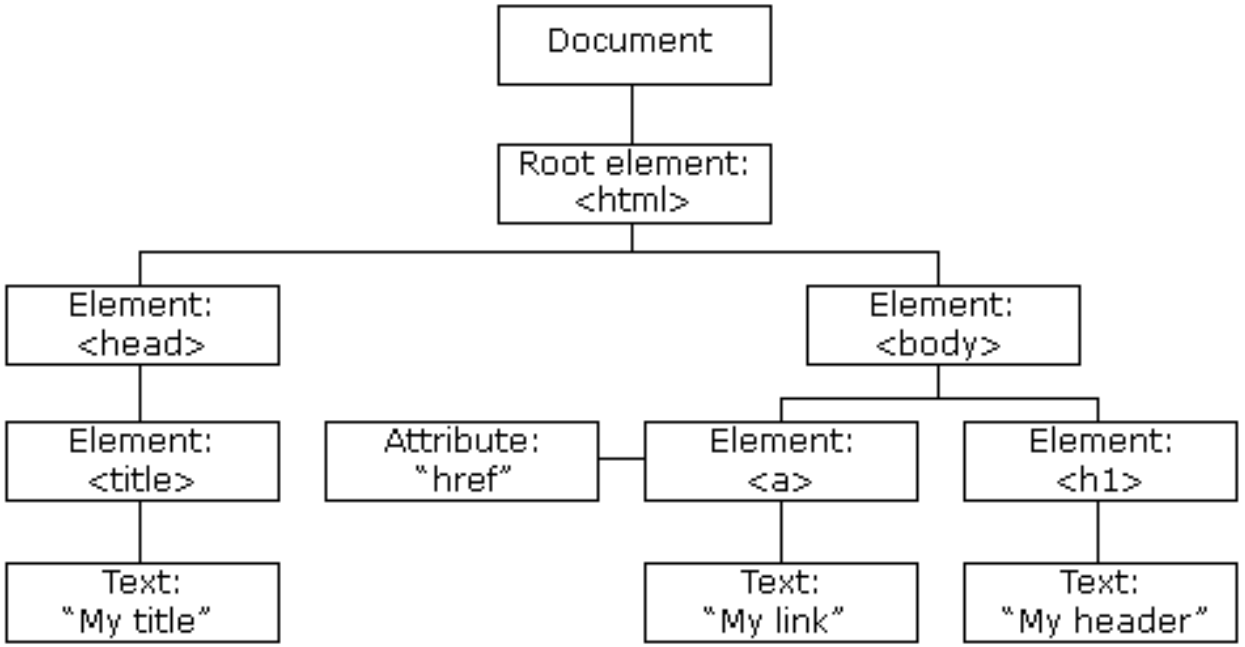
- A sintaxe para a escrita e chamada de funções é mostrada abaixo:

```
function somaDoisNumeros(numero1, numero2){
    alert(numero1 + numero2); // exibe a soma;
    return numero1 + numero2; // retorna a soma;
}

somaDoisNumeros(45, 35); // exibe 80
```

# HTML DOM

- Quando o navegador visita uma página web, ele constrói uma coleção de objetos que representam os elementos da página e suas relações hierárquicas (árvore).
- Esta coleção é chamada de DOM (*Document Object Model*). O DOM define um padrão para acessar, modificar, adicionar ou remover elementos em documentos HTML, bem como as propriedades, métodos e eventos para cada elemento.




*"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

# HTML DOM

- Podemos utilizar Javascript para manipular o DOM, realizando ações como:
  - Modificar elementos HTML na página;
  - Modificar os atributos dos elementos na página;
  - Modificar os estilos CSS dos elementos na página;
  - Remover elementos e atributos;
  - Adicionar novos elementos e atributos;
  - Reagir aos eventos relacionados aos elementos da página.

# A API DOM

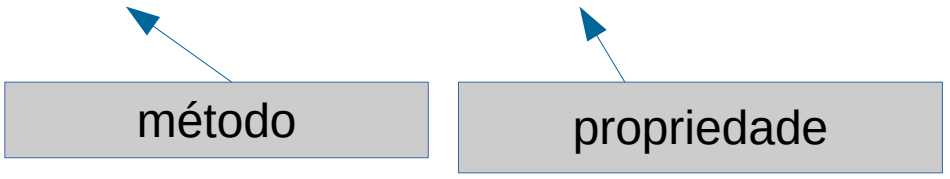
- No DOM, todos os elementos HTML são definidos como objetos.
- A API (*Aplication Programming Interface*) é formada por propriedades e métodos:
  - Propriedade: valor que pode ser obtido ou alterado;
  - Método: ação que pode ser feita (ex: adicionar ou remover elemento)

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```



# Encontrando elementos HTML

- O objeto **document** representa toda a página e é o “dono” de todos os demais objetos dentro dela.
- Com base no objeto **document**, podemos acessar os demais através dos métodos abaixo:

| Método | Descrição |
|---|---|
| `document.getElementById("id")` | Localiza um elemento pelo id * |
| `document.getElementsByTagName("tagName")` | Localiza um ou mais elemento(s) pelo nome da tag ** |
| `document.getElementsByClassName("className")` | Localiza um ou mais elementos pelo nome da classe ** |
| `document.querySelectorAll("css selector")` | Localiza um ou mais elementos que se enquadram no seletor css indicado ** |

* retorna uma referência direta ao elemento com o id especificado;
** retornam um array numérico de referências aos elementos selecionados.

# Exemplo

```
<p>parágrafo 1</p>
<p>parágrafo 2</p>
<p>parágrafo 3</p>
<p class="teste">parágrafo 4</p>
<p class="teste">parágrafo 5</p>
<div id="exemplo">uma div de exemplo</div>

<script>
x = document.getElementById("exemplo"); // retorna um único elemento
console.log(x.innerHTML);

y = document.getElementsByTagName("p"); // retorna um array com 5 elementos
for(i = 0; i < y.length; i++){
  alert("alterando o parágrafo "+ (i+1));
  y[i].innerHTML += ' - alterado';
}

z = document.getElementsByClassName("teste"); // retorna um array com 2 elementos
for(i = 0; i < z.length; i++){
  alert("alterando o "+ (i+1)+ "º parágrafo com a classe teste");
  z[i].innerHTML += ' - novamente';
}
</script>
```

# Alterando elementos HTML

- Alterando o conteúdo HTML:
  - Sintaxe: document.getElementById(id).**innerHTML** = new HTML;

```
document.getElementById("demo").innerHTML = "<b>Hello JavaScript</b>";
```

- Alterando o valor de um atributo:
  - Sintaxe: document.getElementById(id).**attribute** = new value;

```
document.getElementById("myImage").src = "picture.gif";
```

- Alterando uma propriedade CSS:
  - Sintaxe: document.getElementById(id).**style.property** = new style;

```
document.getElementById("demo").style.fontSize = "25px";
document.getElementById("demo").style.display = "none"; // ocultar
document.getElementById("demo").style.display = "block"; // exibir
```

# Eventos

- Eventos são situações provocadas pelo usuário ou pelo próprio navegador, às quais a página pode reagir através de Javascript.
  - A finalização do carregamento de uma página;
  - Uma alteração em um campo de formulário;
  - Um clique, duplo clique ou movimentação do mouse sobre um elemento;
  - Etc.
- Podemos associar *event handlers* (manipuladores de eventos) diretamente aos elementos HTML:

```
<button onclick="document.getElementById('demo').innerHTML='Você
clicou!'">Clique aqui</button>
```

Código ou chamada a uma função

# Principais eventos

- `onAbort`: quando se aborta o carregamento de uma imagem
- `onBlur`: quando o elemento perde o foco
- `onChange`: quando o valor do campo é alterado
- `onClick`: quando o elemento recebe um clique
- `onError`: quando um erro ocorrer enquanto uma imagem ou janela é carregada
- `onFocus`: ativado quando o elemento recebe o foco
- `onLoad`: quando o elemento é carregado
- `onMouseOver`: quando o mouse está sobre o elemento
- `onMouseOut`:quando o mouse sai do elemento
- `onSubmit`: quando o form é submetido
- `onUnload`: quando a página é descarregada

Consulte a lista completa:

https://www.w3schools.com/jsref/dom_obj_event.asp

# Funções e Eventos

- Códigos HTML e Javascript são executados automaticamente quando a página é carregada. Mas podemos usar funções para executar um código Javascript somente quando um evento ocorrer:

```
<html>
<head>
<script>
function displayDate(){
  document.getElementById("exemplo").innerHTML = Date();
}
</script>
</head>
<body>
<p id="exemplo"></p>
<button id="botao" onclick="displayDate()">Ver data</button>
</body>
</html>
```

# Eventos

- Ou podemos associar os *event handlers* através do DOM, mantendo a ideia da separação entre a marcação HTML e a programação:

```
<html>
<head>
</head>
<body>
<p id="exemplo"></p>
<button id="botao">Ver data</button>
<script>
document.getElementById("botao").onclick = displayDate;
function displayDate(){
  document.getElementById("exemplo").innerHTML = Date();
}
</script>
</body>
</html>
```

# Event listener

- *Listener* é um objeto que é notificado quando um evento ocorre.
- O método **`addEventListener()`** vincula um *event handler* a um elemento, sem sobrescrever outros eventos já vinculados.

```
element.addEventListener("click", function(){ alert("Hello World!"); });
```

- ou

```
element.addEventListener("click", myFunction);

function myFunction() {
    alert ("Hello World!");
}
```

- Podemos adicionar diversos *handlers* para cada elemento, inclusive de mesmo tipo (dois manipuladores para *click*, por exemplo).
- Podemos remover um *event listener* utilizando o método **`removeEventListener()`**.

# Event listener

```
<p>Adicionando vários listeners ao mesmo botão:</p>

<button id="myBtn">Clique aqui</button>

<p id="demo"></p>

<script>
var x = document.getElementById("myBtn");
x.addEventListener("mouseover", myFunction);
x.addEventListener("click", mySecondFunction);
x.addEventListener("mouseout", myThirdFunction);

function myFunction() {
    document.getElementById("demo").innerHTML += "Passou o mouse!<br>";
}

function mySecondFunction() {
    document.getElementById("demo").innerHTML += "Clicou!<br>";
}

function myThirdFunction() {
    document.getElementById("demo").innerHTML += "Mouse saiu!<br>";
}
</script>
```

# Validação de Formulários

```
<script>
function validateForm(){
   var x = document.getElementById("fname").value;
  // var x = document.forms["myForm"]["fname"].value; // mesmo efeito que a linha anterior
   if (x == null || x == "" || x.indexOf(" ") == -1){
      alert("O nome completo deve ser preenchido");
      return false;
   }
}
</script>

<form name="myForm" action="" onsubmit="return validateForm()" method="post">
<label>Nome completo: <input type="text" name="fname" id="fname"></label>
<br>
<input type="submit" value="Enviar">
</form>
```