

# PROGRAMAÇÃO WEB

Prof. Ms. Wilson Lourenço

wilson.slourenco@sp.senac.br

```
mirror_mod = modifier_ob.  
set mirror object to mirror.  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
selection at the end -add  
mirror_ob.select= 1  
mirror_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier  
mirror_ob.select = 0  
= bpy.context.selected_obj  
data.objects[one.name].se  
print("please select exactly  
  
--- OPERATOR CLASSES ---  
  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

# JAVASCRIPT

## Por que usar JavaScript?

- Na página de produto havíamos criado um `input range` para selecionar o tamanho da roupa.
- O problema é que não há feedback visual de qual valor está selecionado.
- Podemos então criar um outro elemento visual na página apenas para mostrar o valor atualmente selecionado no range.

- Mas que tag usar pra representar esse elemento cujo valor é resultado do valor escolhido no range?
- No HTML5, temos uma tag nova com valor semântico exato pra essa situação: o `<output>`.
- Essa tag representa a saída de algum cálculo ou valor simples obtido a partir de um ou mais campos de um formulário. Ele tem um atributo `for` que aponta de qual elemento saiu o seu valor.

`<output for="tamanho" name="valortamanho">42</output>`

- Visualmente, é como se fosse uma `<div>` simples. Depois vamos estilizar esse componente do jeito que quisermos com CSS.
- A grande sacada é o valor semântico da tag e o que ela representa.
- O valor em si está como 42 porque colocamos na mão, dentro da tag.
- O que precisamos é atualizar esse valor toda vez que o valor do input range mudar, ou seja, toda vez que o usuário arrastar o input range.

- O HTML vem pronto para o navegador com todo seu conteúdo e tags.
- Mudar o conteúdo de uma tag baseado numa ação do usuário (dentro do navegador) não é função do HTML.
- Pra isso, precisamos do JavaScript, JS para os íntimos.

# Características do JavaScript

- O JavaScript, como o próprio nome sugere, é uma linguagem de scripting.
- Uma linguagem de scripting é comumente definida como uma linguagem de programação que permite ao programador controlar uma ou mais aplicações de terceiros.
- No caso do JavaScript, podemos controlar alguns comportamentos dos navegadores através de trechos de código que são enviados na página HTML.

- Outra característica comum nas linguagens de scripting é que normalmente elas são linguagens interpretadas, ou seja, não dependem de compilação para serem executadas.
- Essa característica é presente no JavaScript: o código é interpretado e executado conforme é lido pelo navegador, linha a linha, assim como o HTML.



- O JavaScript também possui grande tolerância a erros, uma vez que conversões automáticas são realizadas durante operações.
- Como será visto no decorrer das explicações, nem sempre essas conversões resultam em algo esperado, o que pode ser fonte de muitos bugs, caso não conheçamos bem esse mecanismo.

- O script do programador é enviado com o HTML para o navegador, mas como o navegador saberá diferenciar o script de um código html?
- Para que essa diferenciação seja possível, é necessário envolver o script dentro da tag `<script>`.

# Console do Navegador

- Existem várias formas de executar códigos JavaScript em uma página. Uma delas é executar códigos no que chamamos de Console.
- A maioria dos navegadores desktop já vem com essa ferramenta instalada.
  - No Chrome, é possível chegar ao Console apertando F12 e em seguida acessar a aba "Console" ou por meio do atalho de teclado Control + Shift + C;
  - No Firefox, pelo atalho Control + Shift + K.

- O console faz parte de uma série de ferramentas embutidas nos navegadores especificamente para nós que estamos desenvolvendo um site. Essa série de ferramentas é o que chamamos de Developer Tools.

```
mirror_mod = modifier_ob.  
set mirror object to mirror  
mirror_mod.mirror_object =
```

```
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
selection at the end -add  
mirror_ob.select= 1  
mirror_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier  
mirror_ob.select = 0  
= bpy.context.selected_obj  
data.objects[one.name].se  
print("please select exactly
```

--- OPERATOR CLASSES ---

```
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"
```

```
context):  
context.active_object is not
```

# SINTAXE BÁSICA DO JAVASCRIPT

# Operadores

- No JavaScript podemos somar, subtrair, multiplicar e dividir como em qualquer linguagem.
- Se você abrir o console do seu navegador, pode fazer alguns testes:

> 12 + 13

25

> 14 \* 3

42

> 10 - 4

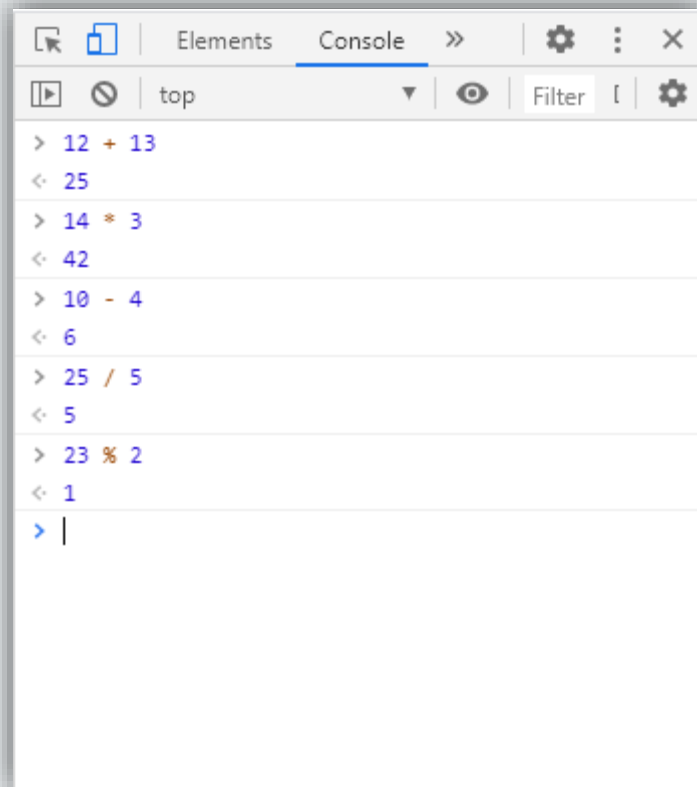
6

> 25 / 5

5

> 23 % 2

1

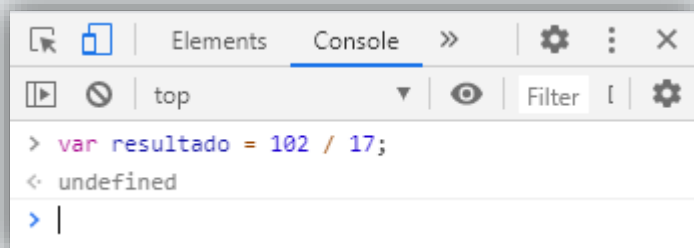


# Variáveis

- Para armazenarmos um valor para uso posterior, podemos criar uma variável:

> var resultado = 102 / 17;

undefined





- No exemplo acima, guardamos o resultado de  $102 / 17$  na variável `resultado`.
- O resultado de criar uma variável é sempre `undefined`.
- Para obter o valor que guardamos nela ou mudar o seu valor, podemos fazer o seguinte:

> resultado

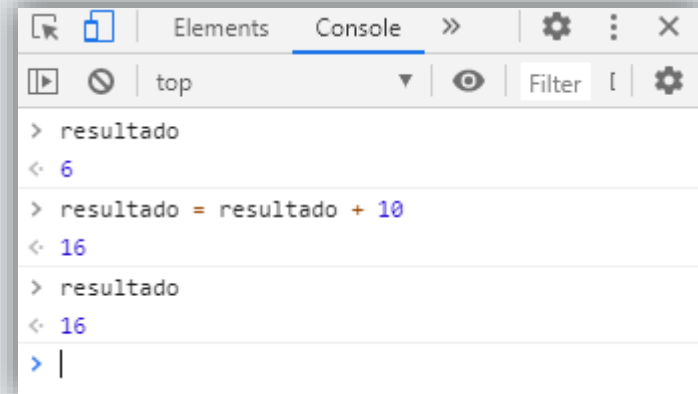
6

> resultado = resultado + 10

16

> resultado

16



- Também podemos alterar o valor de uma variável usando as operações básicas com uma sintaxe bem compacta:

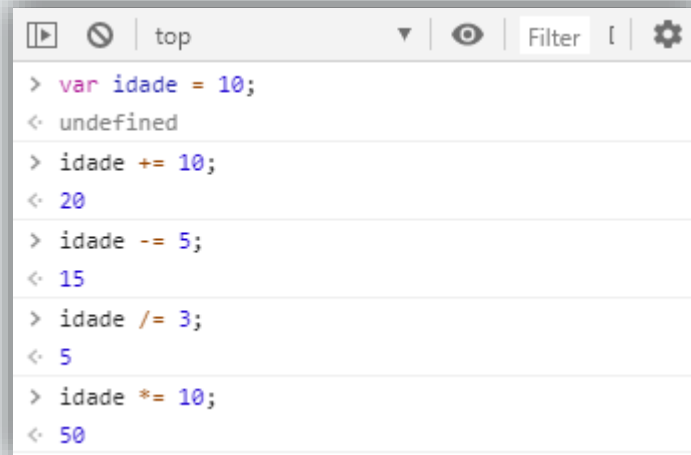
> var idade = 10; // undefined

> idade += 10; // idade vale 20

> idade -= 5; // idade vale 15

> idade /= 3; // idade vale 5

> idade \*= 10; // idade vale 50



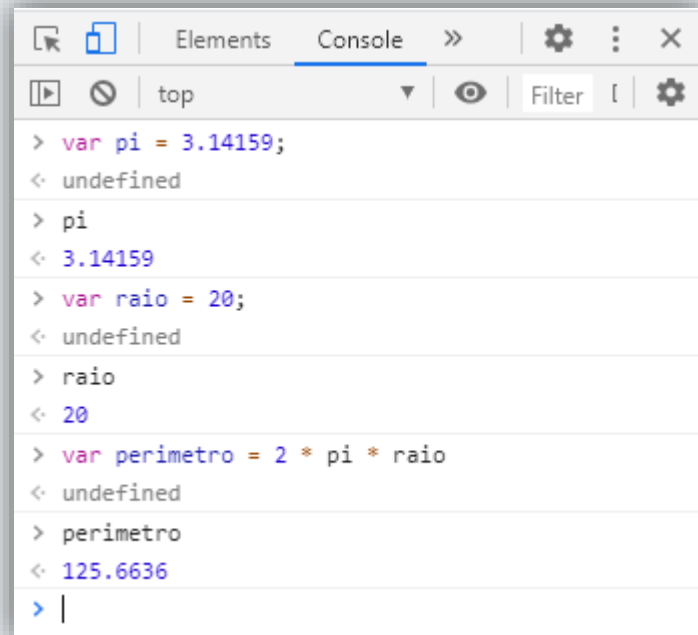
```
top
> var idade = 10;
< undefined
> idade += 10;
< 20
> idade -= 5;
< 15
> idade /= 3;
< 5
> idade *= 10;
< 50
```

# Tipos de Dados

- Não são apenas números que podemos salvar numa variável. O JavaScript tem outros tipos de dados:
  - Number
  - String

- **Number**

- Com esse tipo de dados é possível executar todas as operações que vimos anteriormente:
- `var pi = 3.14159;`
- `var raio = 20;`
- `var perimetro = 2 * pi * raio`



```
> var pi = 3.14159;
< undefined

> pi
< 3.14159

> var raio = 20;
< undefined

> raio
< 20

> var perimetro = 2 * pi * raio
< undefined

> perimetro
< 125.6636

> |
```

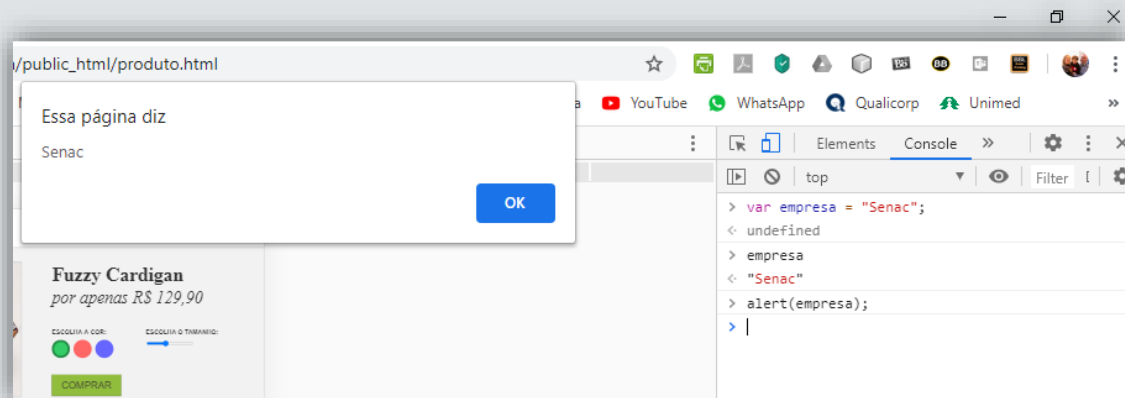
- # String

- Uma string em JavaScript é utilizada para armazenar trechos de texto:

`var empresa = "Senac";`

- Para exibirmos o valor da variável empresa fora do console, podemos executar o seguinte comando:

`alert(empresa);`



- O comando `alert` serve para criação de popups com algum conteúdo de texto que colocarmos dentro dos parênteses. O que acontece com o seguinte código?

```
var numero = 30;
```

```
alert(numero)
```

- O número 30 é exibido sem problemas dentro do popup. O que acontece é que qualquer variável pode ser usada no `alert`.
- O JavaScript não irá diferenciar o tipo de dados que está armazenado numa variável, e se necessário, tentará converter o dado para o tipo desejado.

- **Automatic semicolon insertion (ASI)**
  - É possível omitir o ponto e vírgula no final de cada declaração.
  - A omissão de ponto e vírgula funciona no JavaScript devido ao mecanismo chamado automatic semicolon insertion (ASI).



## A Tag `<script>`

- O console nos permite testar códigos diretamente no navegador.
- Porém, não podemos pedir aos usuários do site que sempre abram o console, copiem um código e coleem para ele ser executado.

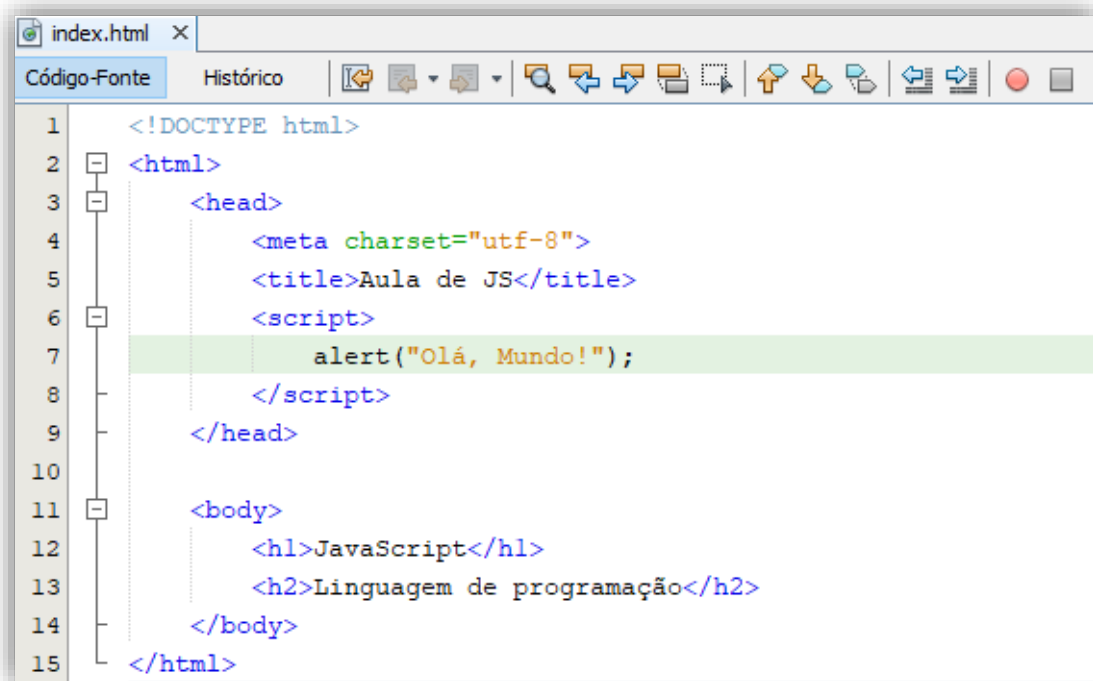
- Para inserirmos um código JavaScript em uma página, é necessário utilizar a tag `<script>` :

```
<script>
```

```
    alert("Olá, Mundo!");
```

```
</script>
```

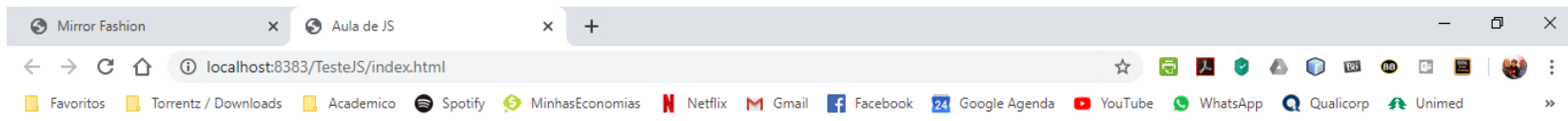
- A tag `<script>` pode ser declarada dentro da tag `<head>` assim como na tag `<body>`, mas devemos ficar atentos, porque o código é lido imediatamente dentro do navegador.
- Veja a consequência disso nos dois exemplos abaixo:



The image shows a web browser window with the title bar 'index.html'. The 'Código-Fonte' (Source Code) tab is active, displaying the HTML source code of a file named 'index.html'. The code is as follows:

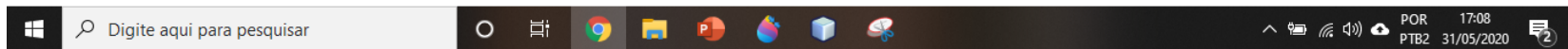
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Aula de JS</title>
6     <script>
7       alert("Olá, Mundo!");
8     </script>
9   </head>
10
11  <body>
12    <h1>JavaScript</h1>
13    <h2>Linguagem de programação</h2>
14  </body>
15 </html>
```

The line containing the JavaScript code, `alert("Olá, Mundo!");`, is highlighted in green. The browser's toolbar is visible at the top, showing various navigation and development tools.

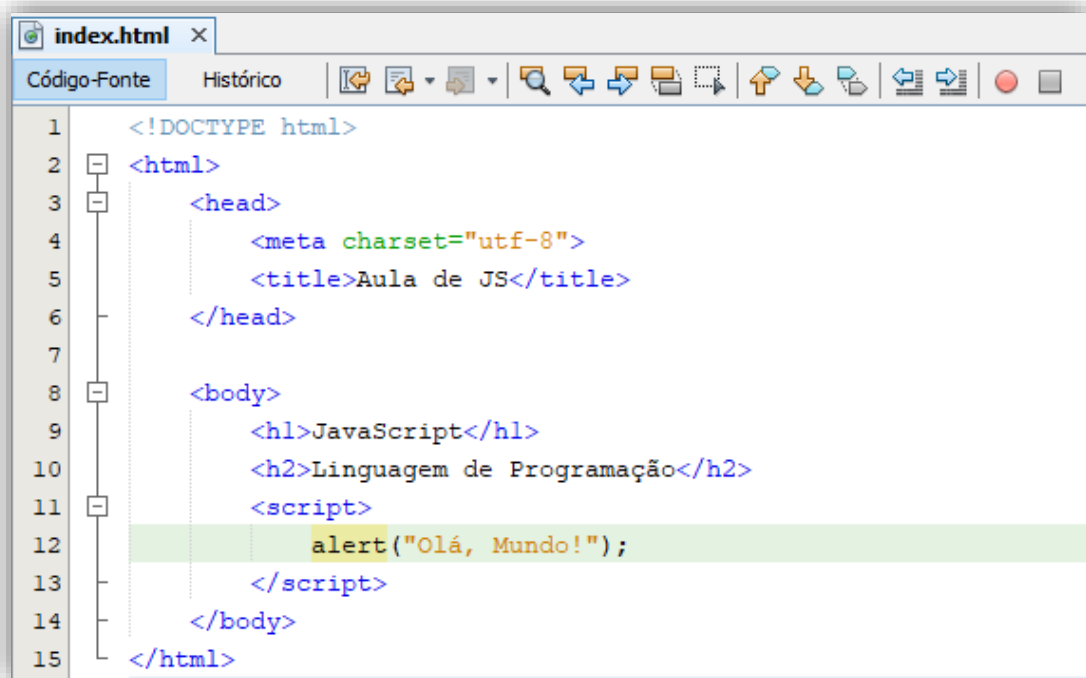


# JavaScript

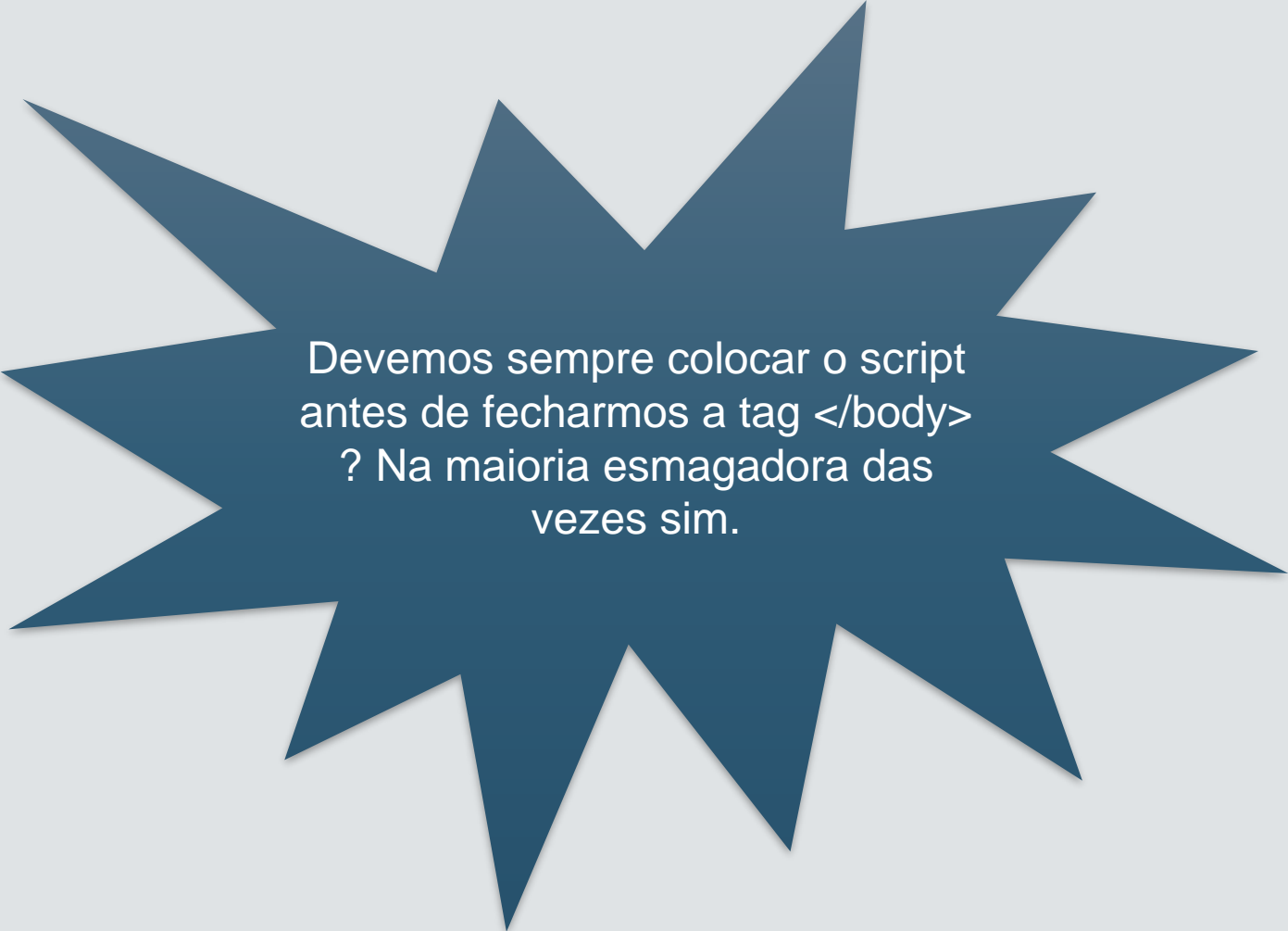
## Linguagem de programação



- Repare que, ao ser executado, o script trava o processamento da página.
- Imagine um script que demore um pouco mais para ser executado ou que exija alguma interação do usuário como uma confirmação.
- Não seria interessante carregar a página toda primeiro antes de sua execução por uma questão de performance e experiência para o usuário?
- Para fazer isso, basta removermos o script do `<head>` , colocando-o no final do `<body>`:



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Aula de JS</title>
6   </head>
7
8   <body>
9     <h1>JavaScript</h1>
10    <h2>Linguagem de Programação</h2>
11    <script>
12      alert("Olá, Mundo!");
13    </script>
14  </body>
15 </html>
```



Devemos sempre colocar o script  
antes de fecharmos a tag `</body>`  
? Na maioria esmagadora das  
vezes sim.



# JavaScript em Arquivo Externo

- Se o mesmo script for utilizado em outra página, como fazemos?
- Imagine ter que reescrever o script toda vez que ele for necessário.
- Para não acontecer isso, é possível importar scripts dentro da página utilizando também a tag `<script>`:

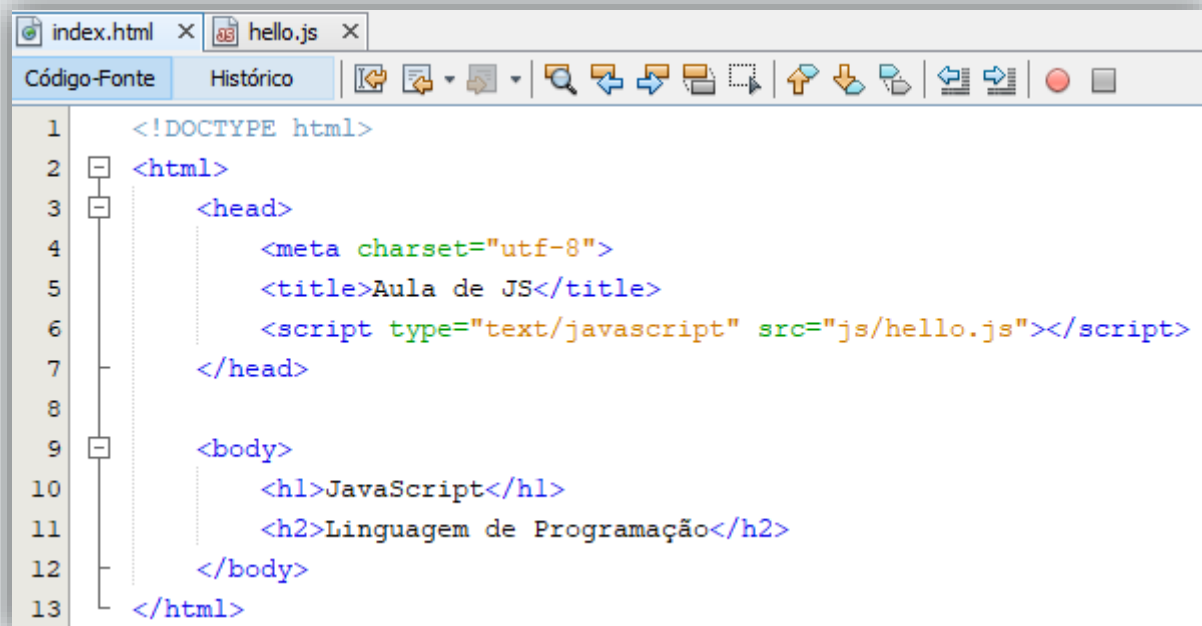
- No arquivo HTML

```
<script src="js/hello.js"></script> type="text/javascript"
```

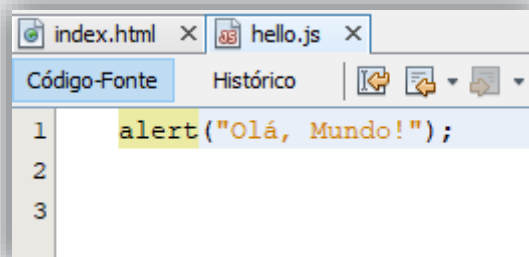
- Arquivo externo js/hello.js

```
alert("Olá, Mundo!");
```

- Com a separação do script em arquivo externo é possível reaproveitar alguma funcionalidade em mais de uma página.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Aula de JS</title>
6     <script type="text/javascript" src="js/hello.js"></script>
7   </head>
8
9   <body>
10    <h1>JavaScript</h1>
11    <h2>Linguagem de Programação</h2>
12  </body>
13 </html>
```



```
1 alert("Olá, Mundo!");
2
3
```

# Mensagens Secretas no Console

- É comum querermos dar uma olhada no valor de alguma variável ou resultado de alguma operação durante a execução do código.
- Nesses casos, poderíamos usar um alert.

- Porém, se esse conteúdo deveria somente ser mostrado para o desenvolvedor, o console do navegador pode ser utilizado no lugar do alert para imprimir essa mensagem:

```
var mensagem = "Olá mundo";  
console.log(mensagem);
```

## DOM: sua página no mundo JavaScript

- Para permitir alterações na página, ao carregar o HTML da página, os navegadores carregam em memória uma estrutura de dados que representa cada uma das nossas tags no JavaScript.
- Essa estrutura é chamada de DOM (Document Object Model). Essa estrutura pode ser acessada através da variável global `document`.

- **querySelector**

- Antes de sair alterando nossa página, precisamos em primeiro lugar acessar no JavaScript o elemento que queremos alterar.
- Como exemplo, vamos alterar o conteúdo de um título da página. Para acessar ele:

```
document.querySelector("h1");
```

- Esse comando usa os seletores CSS para encontrar os elementos na página.
- Usamos um seletor de nome de tag mas poderíamos ter usado outros:

`document.querySelector(".class")`

`document.querySelector("#id")`



## Elemento da página como variável

- Se você vai utilizar várias vezes um mesmo elemento da página, é possível salvar o resultado de qualquer `querySelector` numa variável:

```
var titulo = document.querySelector("h1");
```

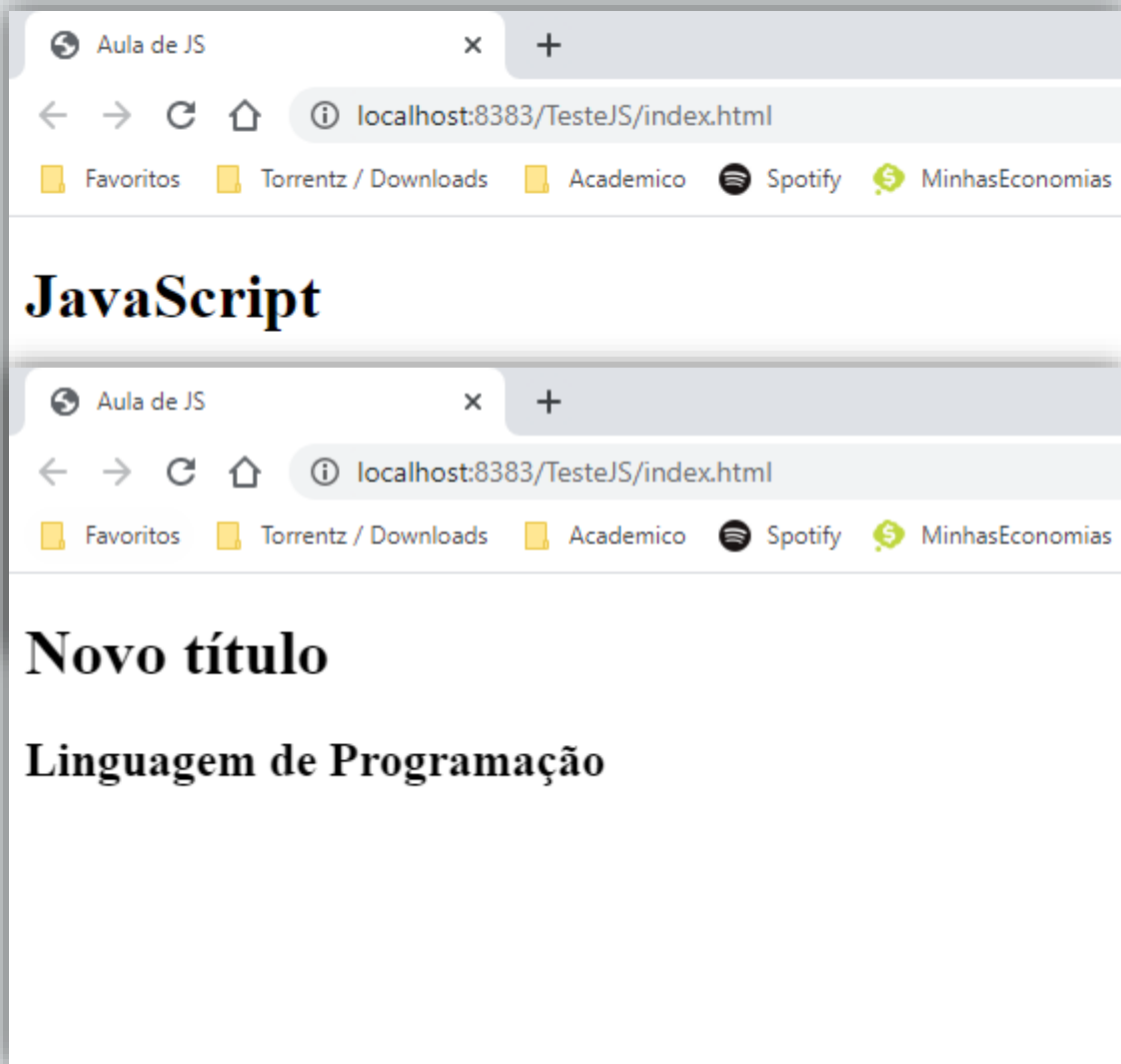
- Executando no console, você vai perceber que o elemento correspondente é selecionado. Podemos então manipular seu conteúdo. Você pode ver o conteúdo textual dele com:

> titulo.textContent

- Essa propriedade, inclusive, pode receber valores e ser alterada:

>titulo.textContent = "Novo título"

```
Elements Console >>
top
> var titulo = document.querySelector("h1");
< undefined
> titulo.textContent
< "JavaScript"
> titulo.textContent = "Novo título"
< "Novo título"
> |
```



## querySelectorAll

- As vezes você precisa selecionar vários elementos na página. Várias tags com a classe `.cartao` por exemplo.
- Se o retorno esperado é mais de um elemento, usamos `querySelectorAll` que devolve uma lista de elementos (array).

`document.querySelectorAll(".cartao")`

- Podemos então acessar elementos nessa lista através da posição dele (começando em zero) e usando o colchetes:

// primeiro cartão

```
document.querySelectorAll(".cartao")[0]
```

# Alterações no DOM

- Ao alterarmos os elementos da página, o navegador sincroniza as mudanças e alteram a aplicação em tempo real.

# Funções e os Eventos do DOM

- Apesar de ser interessante a possibilidade de alterar o documento todo por meio do JavaScript, é muito comum que as alterações sejam feitas quando o usuário executa alguma ação, como por exemplo, mudar o conteúdo de um botão ao clicar nele e não quando a página carrega.

- Porém, por padrão, qualquer código colocado no `<script>`, como fizemos anteriormente, é executado assim que o navegador lê ele.
- Para guardarmos um código para ser executado em algum outro momento, por exemplo, quando o usuário clicar num botão, é necessário utilizar alguns recursos do JavaScript no navegador.



- Primeiro vamos criar uma função:

```
function mostraAlerta() {  
    alert("Funciona!");  
}
```

- Ao criarmos uma função, simplesmente guardamos o que estiver dentro da função, e esse código guardado só será executado quando chamarmos a função, como no seguinte exemplo:

```
mostraAlerta()
```

- Para chamar a função `mostraAlerta` só precisamos utilizar o nome da função e logo depois abrir e fechar parênteses.
- Agora, para que essa nossa função seja chamada quando o usuário clicar no botão da nossa página, precisamos do seguinte código:

```
function mostraAlerta() {  
    alert("Funciona!");  
}
```

// obtendo um elemento através de um seletor de ID

```
var botao =  
document.querySelector("#botaoEnviar");
```

```
botao.onclick = mostraAlerta;
```

- Note que primeiramente foi necessário selecionar o botão e depois definir no onclick que o que vai ser executado é a função mostraAlerta.
- Essa receita será sempre a mesma para qualquer código que tenha que ser executado após alguma ação do usuário em algum elemento.
- O que mudará sempre é qual elemento você está selecionando, a qual evento você está reagindo e qual função será executada.

# Quais Eventos Existem?

- Existem diversos eventos que podem ser utilizados em diversos elementos para que a interação do usuário dispare alguma função:
  - `oninput`: quando um elemento input tem seu valor modificado
  - `onclick`: quando ocorre um click com o mouse
  - `ondblclick`: quando ocorre dois clicks com o mouse

- onmousemove: quando mexe o mouse
- onmousedown: quando aperta o botão do mouse
- onmouseup: quando solta o botão do mouse (útil com os dois acima para gerenciar drag'n'drop)
- onkeypress: quando pressionar e soltar uma tecla
- onkeydown: quando pressionar uma tecla
- onkeyup: quando soltar uma tecla

- onblur: quando um elemento perde foco
- onfocus: quando um elemento ganha foco
- onchange: quando um input, select ou textarea tem seu valor alterado
- onload: quando a página é carregada
- onunload: quando a página é fechada
- onsubmit: disparado antes de submeter o formulário (útil para realizar validações)

## Exercícios: Mostrando o Tamanho do Produto

- Na página produto.html, vamos mexer no fieldset do tamanho do nosso produto.
- Iremos acrescentar no input range a propriedade oninput, que chama uma função JS e costuma ser "realtime", mas é menos compatível do que usar onchange (por exemplo, no IE não funciona).

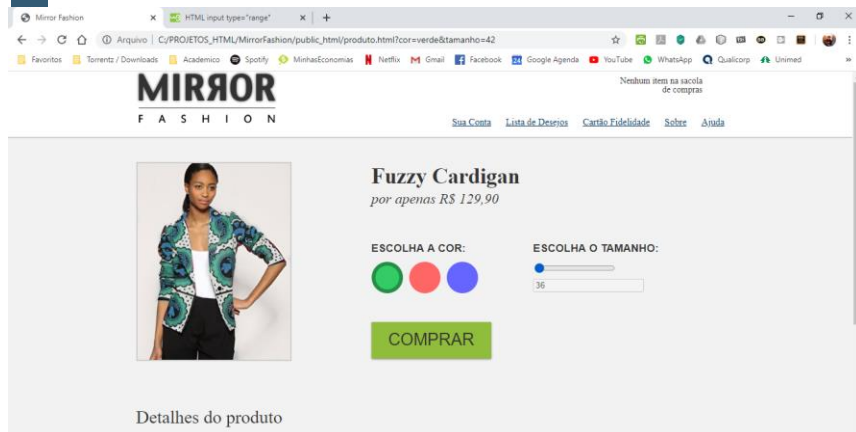


- Iremos acrescentar também a propriedade `onchange`, que funciona em mais browsers, mas não é em todos que atualiza em realtime.
- Por isso, usaremos as duas simultaneamente.
- Colocaremos, ainda, um `input type text` para exibir o valor.

```
<fieldset class="tamanho">  
  <legend>Escolha o tamanho:</legend>  
  <input type="range" min="36" max="46" value="42" step="2" name="tamanho" id="tamanho"  
    oninput="mostraTamanho.value=value" onchange="mostraTamanho.value=value">  
  <input type="text" id="mostraTamanho" value="42" disabled>  
</fieldset>
```

Valor mostrado inicialmente, que está de acordo com o value padrão no range

Propriedade usada para o usuário não conseguir editar o text



- Agora vamos estilizar nosso text para ele ficar com o design mais agradável.
- No arquivo produto.css, **FORA** das media queries existentes, incluir:

```
191 input[type=text]{  
192     height: 44px;  
193     width: 44px;  
194     text-align: center;  
195     border: 3px solid #666;  
196     border-radius: 50%;  
197     box-shadow: 1px 1px 3px #333;  
198     color: red;  
199     font-weight: bold;  
200     font-size: 16px;  
201 }
```

**MIRROR**  
F A S H I O N

Nenhum item na sacola  
de compras

[Sua Conta](#) [Lista de Desejos](#) [Cartão Fidelidade](#) [Sobre](#) [Ajuda](#)



## Fuzzy Cardigan

por apenas R\$ 129,90

ESCOLHA A COR:

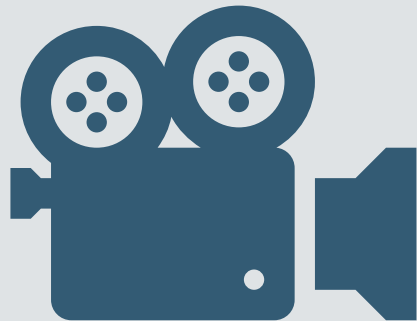


ESCOLHA O TAMANHO:



COMPRAR

Detalhes do produto



## Vídeo

- JAVASCRIPT EM 6 MINUTOS: Tudo Que Você Precisa saber!
- <https://www.youtube.com/watch?v=Fu6p9TidKZc>



## Quizz - HTML, CSS e JavaScript

- <https://quizizz.com/join/>
- No usuário, colocar seu RA

# Dicas para Estudo



Seja “CURIOSO”:

Procure revisar o que foi estudado.

Pesquise as referências bibliográficas.



Seja “ANTENADO”:

Leia a próxima aula.



Seja  
“COLABORATIVO”:

Traga assuntos relevantes para a sala de aula.

Participe da aula.

Proponha discussões relevantes sobre o conteúdo.



Prof. Ms. Wilson Lourenço





**Dúvidas?  
Não mais..**