JavaScript

Primeiros Passos

Um breve histórico

JavaScript é uma linguagem de programação ***interpretada*** com ***tipagem fraca e dinâmica***, orientada a objetos (baseada em ***protótipos***), de ***IO não bloqueante***, que nasceu para ser executada no front-end (browser). Através dela podemos ***manipular o DOM (Document Object Model),*** realizar chamadas ***assíncronas***, validar dados e executar lógica da aplicação.

Poucos anos depois de sua criação, a Netscape submeteu o JavaScript ao Ecma internacional, que passou a chamá-lo de ***ECMAScript***.

Em 2008 o Google lançou o ***Chrome V8***, a mais poderosa engine (de código aberto) da linguagem. O V8 subsidiou a tecnologia necessária para o ***NodeJS***, que hoje é um poderoso servidor JavaScript que roda no back-end.

Versão do JavaScript

EcmaScript 5 *vs* EcmaScript 6

Sintaxe Básica

Formas de declaração (exe1.html)

Declaração Interna:

```
<script type="text/javascript">
    // Código aqui...
</script>
```

Em projetos reais não é uma boa prática manter código JavaScript no mesmo arquivo do HTML.

Declaração Externa:

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

Dê preferência às referências externas. Separar a lógica da estrutura HTML é uma boa prática.
Observação: para fins didáticos, às vezes usaremos a declaração interna.

Precedência das tags <script> (exe2.html)

Assim como na estrutura HTML e CSS, a ordem na qual as tags <script> são definidas no documento HTML afeta no funcionamento do código JavaScript.

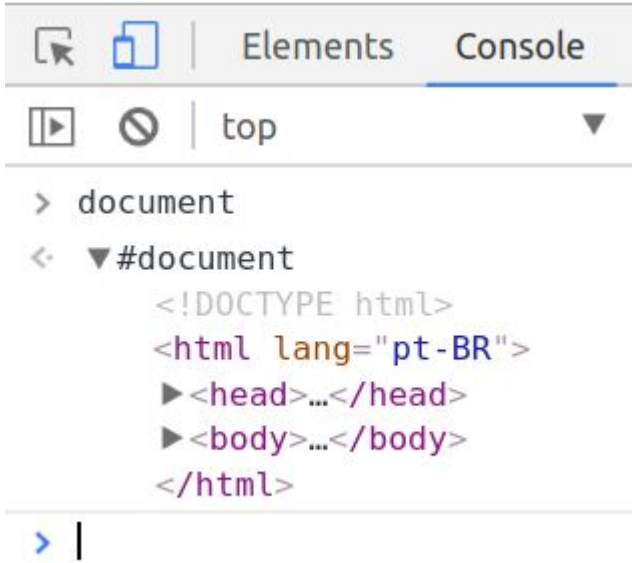
```
<script type="text/javascript" src="arquivo.js"></script>
<script type="text/javascript"> var b = 20; </script>
<script type="text/javascript" src="outro.js"></script>
```

Neste exemplo, o arquivo “outro.js” consegue obter as variáveis definidas nos demais scripts e exibi-las no console. Para imprimir dados no console, use:

```
console.log('Imprimir alguma coisa...');
```

Acesso ao DOM (exe3/index.html)

Abra o index.html do projeto TO-DO App (pasta exe3) no Google Chrome. Em seguida, abra a DevTools, digite a palavra *document* e clique Enter.



Um dos principais recursos do JavaScript é acessar e manipular o DOM. Isso é feito através do palavra reservada *document*

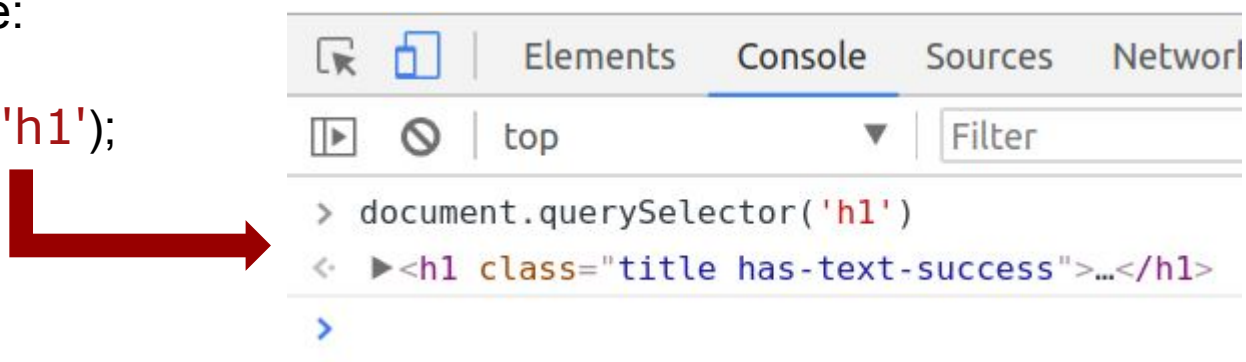
Acesso aos elementos do DOM (exe3/index.html)

Para obter os elementos do DOM, deve-se utilizar o comando a seguir, onde **XYZ** é uma string que recebe a tag, ou classe, ou o ID, com a mesma sintaxe já utilizada nos seletores CSS.

```
document.querySelector( XYZ );
```

Ainda no DevTools, digite:

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



Acesso aos elementos do DOM (exe3/index.html)

Vejamos outros exemplos do projeto TO-DO App:

document.querySelector('section');
document.querySelector('div.container');

```
> document.querySelector('section');
<-  ▶<section class="section">…</section>
> document.querySelector('div.container');
<-  ▶<div class="container">…</div>
```

Usando querySelector(), como faríamos para obter uma tag pelo ID?

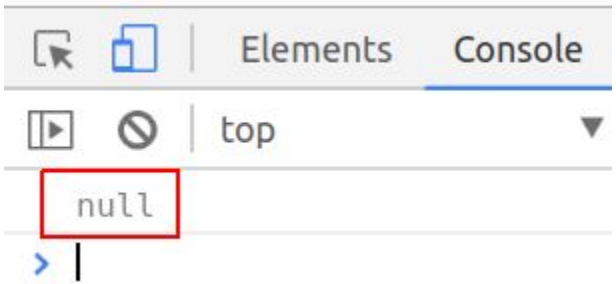
Acesso aos elementos do DOM (exe3/index.html)

Agora que sabemos como acessar o DOM, crie uma tag <script> no <head> do arquivo *exe3/index.html* e imprima no console o resultado da *querySelector()* buscando a <section>.

```
<script>
    console.log(document.querySelector('section'));
</script>
```

Qual o erro do código?

O que aconteceu?



Tag <script> no <head> ou no <body>?

O exemplo anterior não conseguiu imprimir a tag <section> no console porque a tag <script> foi definida no <head> da página. O JavaScript faz uma consulta no DOM para obter as tags através do *querySelector()*. No <head> a estrutura do HTML ainda não está pronta. Tente refazer o teste colocando o <script> exatamente antes do fechamento da tag <body>.

Agora funcionou!!!

Tag <script> no final do <body>!

Evite colocar as tags <script> no <head>, no início ou corpo do <body> (entre os trechos de HTML). Além do possíveis erros de acesso ao DOM, você pode encontrar problemas de performance para executar chamadas assíncronas ou de renderização da página no momento de download do JavaScript.

Sempre coloque suas tags <script> no fim do <body>, exatamente antes da tag de fechamento </body>.

Declaração de variáveis no JavaScript (exe4.html)

Veja a seguir um exemplo de declaração de variáveis no JavaScript.

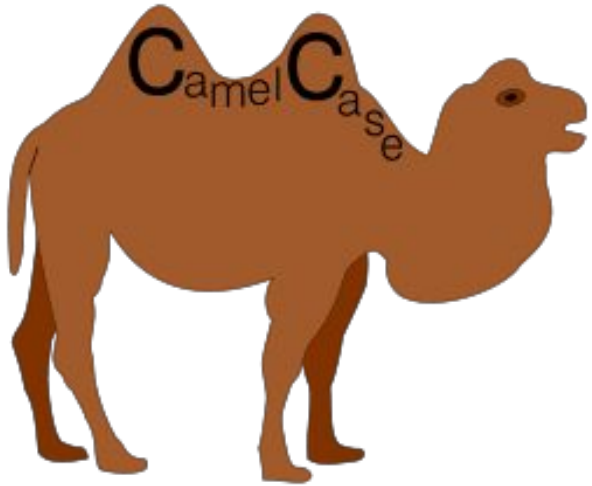
```
<script type="text/javascript">
    var valorInteiro = 200;
    var valorDecimal = 3.14;
    var valorBoolean = true;
    var valorTexto = 'Este é um texto!';
    var meuArray = [ 1 , 2 , 3 ];
</script>
```

Outros exemplos: https://www.w3schools.com/js/js_variables.asp

Declaração de variáveis no JavaScript (exe4.html)

Regras gerais sobre o nome das variáveis:

- Podem conter letras, dígitos, sublinhados e cifrões
- Devem começar com uma letra
- Podem começar com $ e _
- Diferenciam maiúsculas de minúsculas (case sensitive)
- Palavras reservadas não podem ser usadas (if, for …)

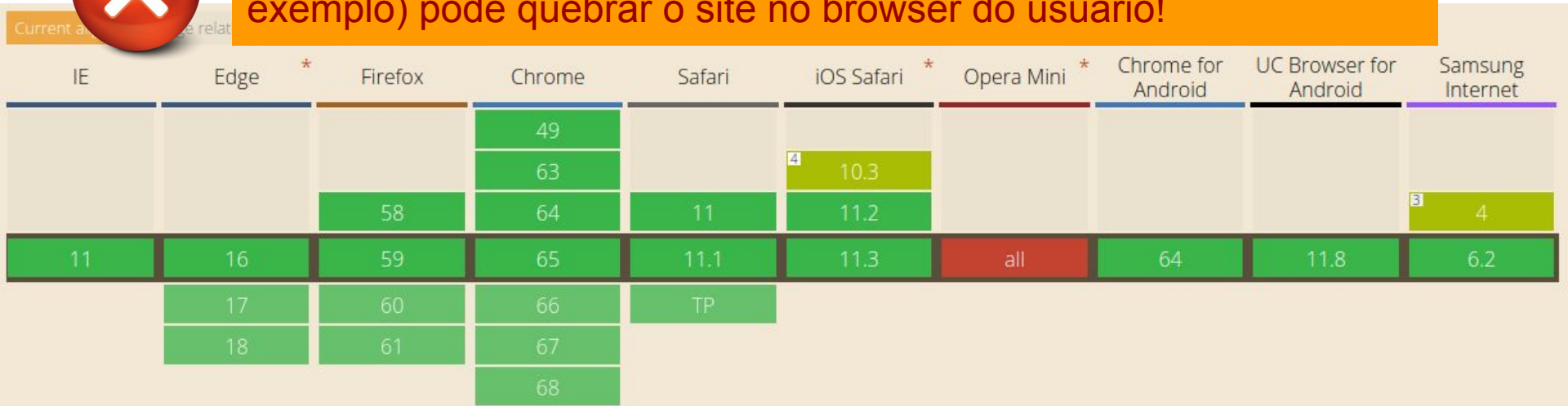


Uma boa prática é utilizar o padrão **lowerCamelCase** (ou **dromedaryCase**) para o nome das variáveis e funções, e o **CamelCase** para o nome das classes.

var , *let* ou *const* ??? (exe5.html)

O ECMAScript 6 introduziu as formas de declaração de variáveis *let* e *const*. O *let* é semelhante ao *var*, mas possui escopo de bloco (o *var* tem escopo de função) e exibe um erro no console caso seja declarada novamente. O *const* é utilizado para constantes, possui escopo de bloco e impede a alteração de seu valor.

Usar recursos do ECMAScript 6 sem um “transpiler” (babel, por exemplo) pode quebrar o site no browser do usuário!



Exercício (exe3/index.html)

- Crie um arquivo chamado “script.js” na pasta “js”
- Altere o index.html para apontar para esse script
- No “script.js”, obtenha as tags <input> do nome do usuário e o <button> logar. **Sugestão:** use ID
- Guarde ambas tags em variáveis
- Por fim, escreva as variáveis no console do navegador

Funções e Eventos do HTML

Funções no JavaScript (exe6.html)

As funções são a alma do JavaScript! Vamos começar pelo básico: como declará-las. Elas são muito úteis para reaproveitar código. Imagine uma situação hipotética que você precise imprimir várias tags no console. Ao invés de repetir o mesmo código sempre, você poderia criar a seguinte função:

```
function imprimirNoConsole(nomeDaTag) {
    var h1Tag = document.querySelector(nomeDaTag);
    console.log(h1Tag);
}
```

Eventos do HTML (exe7.html)

O DOM do HTML é fortemente centrado em eventos. Eventos são **ações** que ocorrem com os elementos HTML. Por exemplo, um botão *"é clicado"*, um campo de entrada de texto *"é digitado"*, entre outros.

Através do JavaScript nós somos capazes de **escutar** esses eventos através dos **Listeners** ("ouvintes"), conforme a seguir. Neste caso, *executarLogin* é uma função JavaScript.

```
botaoLogar.addEventListener('click', executarLogin);
```

Breve referência sobre os eventos (exe7.html)

click	Um evento de mouse que ocorre quando alguma tag é clicada.
dblclick	Quando uma tag recebe um duplo clique.
input	Quando o valor de um elemento é alterado. Exemplo: quando uma checkbox de “concordo com os termos” for acionada, pode-se habilitar um botão do formulário.
load	Quando um elemento é carregado. Exemplo: você pode realizar uma ação quando uma imagem é carregada.
submit	Quando um formulário é enviado. Exemplo: útil para capturar a submissão de um formulário e validar os valores dos campos.

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

Eventos com funções anônimas (exe8.html)

Uma função anônima, além de não ter nome, geralmente é utilizada como parâmetro para outras funções. Um exemplo clássico é a passagem de uma função anônima para os *Listeners* de eventos.

```
tagBotao.addEventListener('click',function () {
    console.log('Eu fui clicado...');
});
```

Uma função (anônima ou nomeada) passada como parâmetro para outra função é conhecida como **callback**.
Lembre-se dessa palavra! Ela será muito utilizada durante o curso.

Obtendo o valor do <input> (exe9.html)

Agora que sabemos como tratar dos eventos, fica fácil obter o valor dos <input>’s para tratamento posterior no JavaScript. Isso pode ser feito pela propriedade ***value***.

```
var tagInput = document.querySelector('#texto');
var valorInput = tagInput.value;
console.log(valorInput);
```

Continuação do exercício (exe3/index.html)

Altere o arquivo “script.js” para:

- Tratar do evento “click” do <button> de logar
- Quando o botão logar for clicado, deve-se obter o valor do <input> de nome e guardá-lo em uma variável
- Por fim, exiba o nome digitado no console do browser

Manipulando o texto das tags

Obtendo o valor das tags (exe10.html)

Assim como a propriedade ***value*** pode ser usada para obter o valor de um <input>, podemos utilizar a propriedade ***textContent*** para obter o valor de uma tag. Veja o exemplo a seguir, onde estamos pegando o texto do <h1>.

```
var tagH1 = document.querySelector('h1');
console.log(tagH1.textContent);
```

Alterando os valores (exe11.html)

Tanto a propriedade ***input***, quanto a propriedade **textContent**, podem ser utilizadas para alterar o valor das tags.

```
var tagH5 = document.querySelector('h5');
tagH5.textContent = 'Texto alterado pelo JS';

var tagInput = document.querySelector('input');
tagInput.value = 'Valor alterado pelo JS';
```

Continuação do exercício (exe3/index.html)

Altere o arquivo “script.js” para:

- Obter a tag da “notificação de sucesso”
- Quando o botão logar for clicado, deve-se alterar o texto da notificação para “Seja bem-vindo XYZ!”, onde XYZ é o valor digitado no <input>
- **Dica:** use o sinal de + para concatenar strings.

Obrigado!