Fundamentos de Programação Web

UNIDADE 03

JavaScript

Olá,

Continuando com o desenvolvimento ***front-ent***, o lado **cliente** da aplicação *web*, agora, trabalharemos com uma linguagem de programação que será processada no navegador, o ambiente que nos permite interagir com o cliente da aplicação *web*.

A linguagem de programação, em questão, é o **JavaScript**. Veja a seguir os pontos de destaque dessa linguagem.

- É uma linguagem de programação, pois provê todos os recursos de processamento, como variáveis, comandos condicionais (***if***/***else***), laços de repetição (***for***/***while***) e funções, por exemplo.
- O **JavaScript**, ou apenas **JS**, é uma linguagem de programação de uso geral, aplicada **principalmente** para **desenvolvimento *web***, mas que também pode ser utilizada para o desenvolvimento de *software* de forma geral.

- Inicialmente, o **JS** foi criado para adicionar **processamento no lado cliente da aplicação *web***. Contudo, essa poderosa linguagem de programação acabou expandindo suas aplicações. Atualmente, o lado servidor também pode ser desenvolvido com JS, com a solução **Node.JS**, que é um ambiente de execução JavaScript *server-side* – ou lado servidor. Não trabalharemos com esse aspecto da linguagem nesta disciplina, contudo é possível obter mais detalhes sobre o Node.JS em https://nodejs.org/pt-br/about/ .
- O **JavaScript** foi inventado por **Brendan Eich**, em 1995, e tornou-se um padrão **ECMA** (https://www.ecma-international.org/) em 1997. **ECMA-262** é o nome oficial do padrão. **ECMAScript** é o nome oficial da linguagem.

IMPORTANTE

- **JavaScript não é Java**. São linguagens completamente diferentes, tanto em conceito quanto em projeto.
- Nesta disciplina, abordaremos **programação lado servidor com PHP** (como veremos na próxima unidade), e não com **JavaScript**! 😉

HTML + CSS + JavaScript

- O **HTML estrutura o *layout* de um documento *web***, ou organiza os elementos visuais de um documento.
- O **CSS formata a exibição dos elementos HTML**, ou trata o estilo de apresentação do conteúdo do documento.
- O **JavaScript melhora a interatividade do cliente com a aplicação**, ou controla os conteúdos do documento para que eles sejam utilizados pelo usuário conforme definido pelas funcionalidades e requisitos da aplicação.

Fonte: A autora (2023).

Antes de estudar o material, vamos conhecer melhor que resultados podemos esperar ao trabalhar com **JavaScript**. Veja alguns exemplos apresentados nesta videoaula.

Primeiro exemplo com o JavaScript

Na videoaula **Trabalhando com o JavaScript**, vimos que podemos usar a janela de inspeção do navegador, como o Google Chrome, para executar comandos JavaScript. Contudo, o melhor local para nosso desenvolvimento JavaScript *front-end* é em um IDE (ambiente de desenvolvimento integrado), em arquivos HTML, ou arquivos próprios para o JavaScript, com a extensão .JS.

Então, vamos criar uma pasta na nossa máquina, por exemplo, na pasta **C:/Teste-JS**, que será editada com um IDE como o VS Code. Vamos executar a prática a seguir.

EXPERIMENTE

PRÁTICA: O meu primeiro HTML com JavaScript

Siga os passos indicados a seguir para ver o **JavaScript** funcionando.

1. Faça em uma pasta de trabalho, como a que mencionamos anteriormente, **C:/Teste-JS**.
2. Dentro dela, crie um documento **PrimeiroTesteJS.html**, com o código exemplificado a seguir. Após salvar o código, abra o arquivo no navegador.

Arquivo C:/Teste-JS/PrimeiroTesteJS.html

```
<!DOCTYPE html>
<html>
<body>

<h2>Meu primeiro Javascript</h2>

<button type="button" onclick = "document.getElementById('textoDataHora').
Clique em mim para apresentar a data e hora atuais.</button>

<p id="textoDataHora"></p> <!-- Elemento HTML parágrafo,   -->
                           <!-- inicialmente está vazio   -->
</body>                    <!-- Após clicar no botão, o JS -->
</html>                    <!--  muda innerHTML=data+hora  -->
```

Observe no código HTML:

- O elemento HTML `<button>` possui um **evento** onclick, que ocorre quando o usuário clica o botão.
- Neste momento, o **JavaScript** é **acionado**, e altera o elemento HTML com o `id="textoDataHora"`, que neste caso é um parágrafo, ou elemento HTML `<p>`. Dessa forma, o **HTML** `<p>` é alterado com o texto da **data** e **hora** atuais.
- A propriedade `innerHTML` define, ou retorna, o **conteúdo HTML** (ou HTML interno) de um elemento.
- O método **JavaScript** `Date()` retorna a **data** e a **hora** atuais.
- Como o elemento `<p>` está após o elemento `<button>`, o texto com a data e hora atuais aparece após o botão, como indicado na Figura deste exemplo.

#ParaTodosVerem

Figura 1: HTML com JavaScript. Fonte: A autora (2023).

IMPORTANTE

Posicionamento do JavaScript <script> ... </script>

Colocar *scripts* na parte inferior do elemento `<body>` geralmente permite que todos os elementos HTML da página já estejam disponíveis para serem acessados via JavaScript (na sequência da aula veremos como garantir que isso ocorra).

Eventos JavaScript em atributos HTML

Neste exemplo fizemos a definição do evento de click do botão diretamente em um atributo HTML chamado onClick. Apesar de ser a maneira mais fácil de realizar a chamada do evento, não é uma boa prática. Vamos utilizar esta abordagem nos primeiros exemplos, mas na sequência iremos verificar qual é a maneira recomendada de atribuição de eventos em JavaScript.

JavaScript – Visão geral

Uma forma rápida de reconhecer a sintaxe e semântica da linguagem JavaScript é identificando os elementos que podemos utilizar. Como, ao estudar esta linguagem, já conhecemos os fundamentos de programação (p. ex. utilização de variáveis, controle de fluxo com comando condicional ***if*/*else***, laços de repetição etc.), vamos direto a mais um exemplo com alguns destes componentes da programação na linguagem JavaScript.

EXPERIMENTE

PRÁTICA: Tratamento de entrada de dados com JavaScript

Siga os passos indicados a seguir para ver o **JavaScript** funcionando.

1. Faça em uma pasta de trabalho, como a que mencionamos anteriormente, **C:/Teste-JS**.
2. Dentro dela, crie um documento **Ola-JS.html**, com o código exemplificado a seguir. Após salvar o código, abra o arquivo no navegador.

```
</>
```

```
<!DOCTYPE html>
<html>
<body>
    <h2>Demonstração de JavaScript</h2>
    <p id="nom">Para conhecer você, clique no botão abaixo</p>
    <p id="num"></p>
    <br>
    <p id="tp_nom"></p>
    <p id="tp_num"></p>
    <button type="button" onclick="myFunction()">Ao acionar este botão,

    <script>
    // Declara uma função sem parâmetros de entrada.
    function myFunction() {

        // Declara variáveis "nome" e "num"
        // Em seguida atribui os valores digitados pelo usuário, no "pro
        let nom = prompt("Qual o seu nome? ");
        let num = prompt("Qual o seu número favorito? ");
```

Observe no código HTML:

- Escrevemos **comentários *inline*** com barras inclinadas duplas: //.
- **Blocos de comentários** são delimitados por /*... */.
- Definimos uma função sem parâmetros de entrada `function myFunction() { ... }`, que usa uma **palavra reservada** `function` para indicar o início do bloco da função.
- Verificamos que precisamos encerrar um comando com `;` ou delimitar um conjunto de comandos com `{ ... }`.
- Usamos a **palavra reservada** `let` para indicar a criação de uma variável.
- Concatenamos *strings* com **+**.
- Usamos o caractere reservado `"` dentro de uma *string* com ajuda da barra invertida `\"`, como no exemplo: `"A variável \"nom\" é uma "`. Nesse caso, queremos que as aspas duplas apareçam no texto, e não sejam confundidas com delimitadores de *string*.
- **Chamamos o código da função** com o **acionar** do elemento HTML `<button>`, indicando no seu evento `onclick="myFunction()"`.

#ParaTodosVerem

Figura 2: Entrada de dados com JavaScript. Fonte: A autora (2023).

Vamos detalhar mais alguns exemplos de JavaScript, para compreender mais sobre os seus recursos.

Para isso, primeiramente, precisamos conhecer alguns fundamentos do JavaScript, para então utilizá-lo.

1. Criando variáveis para o JavaScript

Palavras Reservadas para Variáveis

Em JavaScript, utilizamos *let* e *const* para declarar variáveis.

- ***let***: Utilizada para declarar variáveis que podem ter seu valor alterado posteriormente. O escopo de uma variável declarada com *let* é limitado ao bloco onde ela é definida.

```
</>
```

```
let meuNome; // Declara a variável
meuNome = 'Pedro'; // Atribui valor à variável
let idade = 27; // Declara e atribui valor à variável
```

- **const**: Utilizada para declarar constantes, ou seja, valores que não podem ser alterados após a atribuição inicial. Assim como *let*, *const* também tem escopo de bloco.

```
</>
```

```
const a = 5;
const b = 6; //valores de ‘a’ e ‘b’ não poderão ser alterados
let soma = a + b; // soma é 11, mas poderá ser alterada
```

- **let e const:** Essas palavras reservadas foram adicionadas ao padrão JavaScript em 2015 (ES6) e oferecem uma forma mais segura e robusta de declarar variáveis comparadas ao var.

Notas sobre *var*

Embora ainda suportado, var não é recomendado em código moderno por causa de seu comportamento de escopo de função, o que pode levar a bugs difíceis de rastrear. Preferimos *let* e *const* por sua clareza e segurança.

```
</>
```

```
var variavelGlobal = 'Var tem escopo de função ou global';
```

Compatibilidade de Navegadores

A maioria dos navegadores modernos suporta *let* e *const*, portanto iremos adotá-los em detrimento ao *var*. Utilizar *let* e *const* melhora a legibilidade, manutenção e segurança do código, alinhando-o com as melhores práticas recomendadas no desenvolvimento moderno em JavaScript.

SAIBA MAIS

- https://www.w3schools.com/js/js_variables.asp
- https://www.w3schools.com/js/js_let.asp
- https://www.w3schools.com/js/js_const.asp
- https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/Variables

2. Números e operadores

Utilizando Números

- Experimente declarar os comandos a seguir no console da **janela de inspeção** do navegador:

```
</>
```

```
let A = 10; // Declara variável inteira
let B = 7.89 ;   // Declara variável com ponto flutuante
typeof A;        // Retorna o tipo de dados da variável = number
typeof B; // Retorna o tipo de dados da variável = number
```

- Valores numéricos são tratados como ***number***, o que facilita o tratamento de valores numéricos.

Com valores numéricos podemos realizar as operações aritméticas:

Operador	Descrição	Tente na janela de inspeção	Resultado
+	Soma	3 + 4	7
-	Subtração	8 - 3	5
*	Multiplicação	2 * 7	14
**	Exponenciação	3 ** 2	9

/	Divisão	8 / 2	4
%	Resto da divisão inteira	9 % 2	1
++	Adiciona 1	x = 3; x--; x	2
--	Subtrai 1	x = 3; x++; x	4

Com valores numéricos podemos realizar as operações de atribuição:

Operador	Descrição	Tente na janela de inspeção	Resultado
+=	Igual a **x = x + y**	x = 3; x += 4; x	7
-=	Igual a **x = x - y**	x = 3; x -= 4; x	-1
*=	Igual a **x = x * y**	x = 3; x *= 4; x	12
/=	Igual a **x = x / y**	x = 30; x /= 2; x	15
%=	Igual a **x = x % y**	x = 30; x %= 4; x	2
**=	Igual a **x = x ** y**	x = 3; x **= 2; x	9

Com valores numéricos podemos realizar as operações de comparação:

Operador	Descrição	Tente na janela de inspeção	Resultado
===	Igual	5 === 2 + 3	true
!==	Diferente	5 !== 2 + 3	false
<	Menor	5 < 10	true
>	Maior	5 > 10	false
<=	Menor ou igual	5 <= 5 + 3	true
>=	Maior ou igual	5 >= 13	false

SAIBA MAIS

- https://www.w3schools.com/js/js_operators.asp
- https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/Math

3. Condicional no JavaScript

Condicional

- Permite tomada de decisão. A estrutura mais simples está apresentada a seguir:

```
</>
```

```
1 if (condição) {
2    // bloco a ser executado se a "condição" for VERDADE
3 }
```

- Porém, é possível encadear várias tomadas de decisão:

```
</>
```

```
1 if (condição1) {
2   // bloco a ser executado se a "condição1" for VERDADE
3 } else if (condição2) {
4  // bloco a ser executado se a "condição1" for FALSA e a "condição2" for V
5 } else {
6  // bloco a ser executado se a "condição1" for FALSA e a "condição2" for F
7 }
```

SAIBA MAIS

- https://www.w3schools.com/js/js_if_else.asp
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/if...else

4. Switch no JavaScript

Switch

- Poderosa estrutura para tomada de decisão que permite selecionar um ou vários blocos de comandos para serem executados. É uma alternativa mais estruturada para encadeamentos de `if/else`:

</>

```
switch(expression) {
  case x:
    // bloco de código executa se "expression" é igual a x
    break;
  case y:
    // bloco de código executa se "expression" é igual a y
    break;
  default:
    // bloco de código executa se "expression" é diferente de x e diferent
}
```

SAIBA MAIS

- https://www.w3schools.com/js/js_switch.asp
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/switch

5. Laço de repetição no JavaScript

Laço de repetição (*loop*)

- A estrutura mais simples do loop do tipo for está apresentada a seguir:

</>

```
// Início: i=0; Finaliza: i = 5; A cada repetição: i++
let text = " ";
for (let i = 0; i < 5; i++) {
   text += "O número é " + i + "<br>";
 }
```

- Também, temos a alternativa para *loop* do tipo *while*, cuja estrutura básica está apresentada a seguir:

SAIBA MAIS

- https://www.w3schools.com/js/js_loop_for.asp
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/for
- https://www.w3schools.com/js/js_loop_while.asp
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/while

6. Array no JavaScript

Array

- Variável estruturada que mantém elementos que possuem características em comum, de forma ordenada:

</>

```
let frutas = ['Abacate', 'Uva', 'Limão'];
```

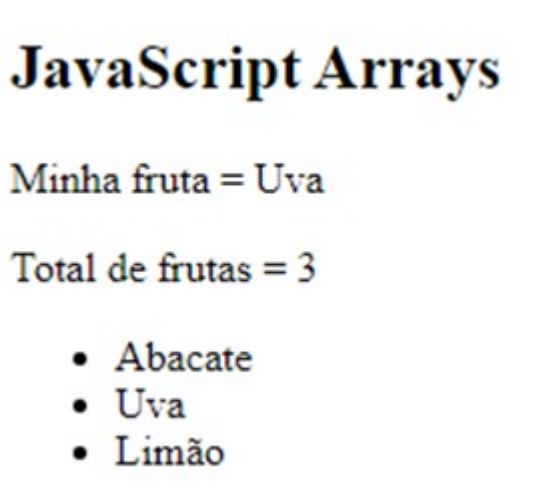
- Uma das maneiras mais comuns para percorrer e acessar cada um dos elementos de um *array* é com um laço de repetição do tipo `for`:

</>

```
```

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Arrays</h2>
<p id="texto1"></p>
<p id="texto2"></p>
<p id="texto3"></p>
<script>
var frutas = ['Abacate', 'Uva', 'Limão'];
var minhaFruta = frutas[1];   // Uva
var fTam = frutas.length;     // Informa o total de elementos no array
document.getElementById("texto1").innerHTML = 'Minha fruta = ' + minhaFr
document.getElementById("texto2").innerHTML = 'Total de frutas = ' + fTa

var text = "<ul>";
for (let i = 0; i < fTam; i++) {          // For que percorre o array e
  text += "<li>" + frutas[i] + "</li>";   // monta o HTML de lista
}
text += "</ul>";                                          // finaliza o HTML d
```



SAIBA MAIS

- https://www.w3schools.com/js/js_arrays.asp
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array

DOM – Visão geral

O Document Object Model (DOM) é a representação dos objetos que compõem a estrutura e o conteúdo de um documento na Web. Com base nessa estrutura de árvore de objetos, o JavaScript consegue acessar e alterar todos os elementos em um documento HTML.

#ParaTodosVerem

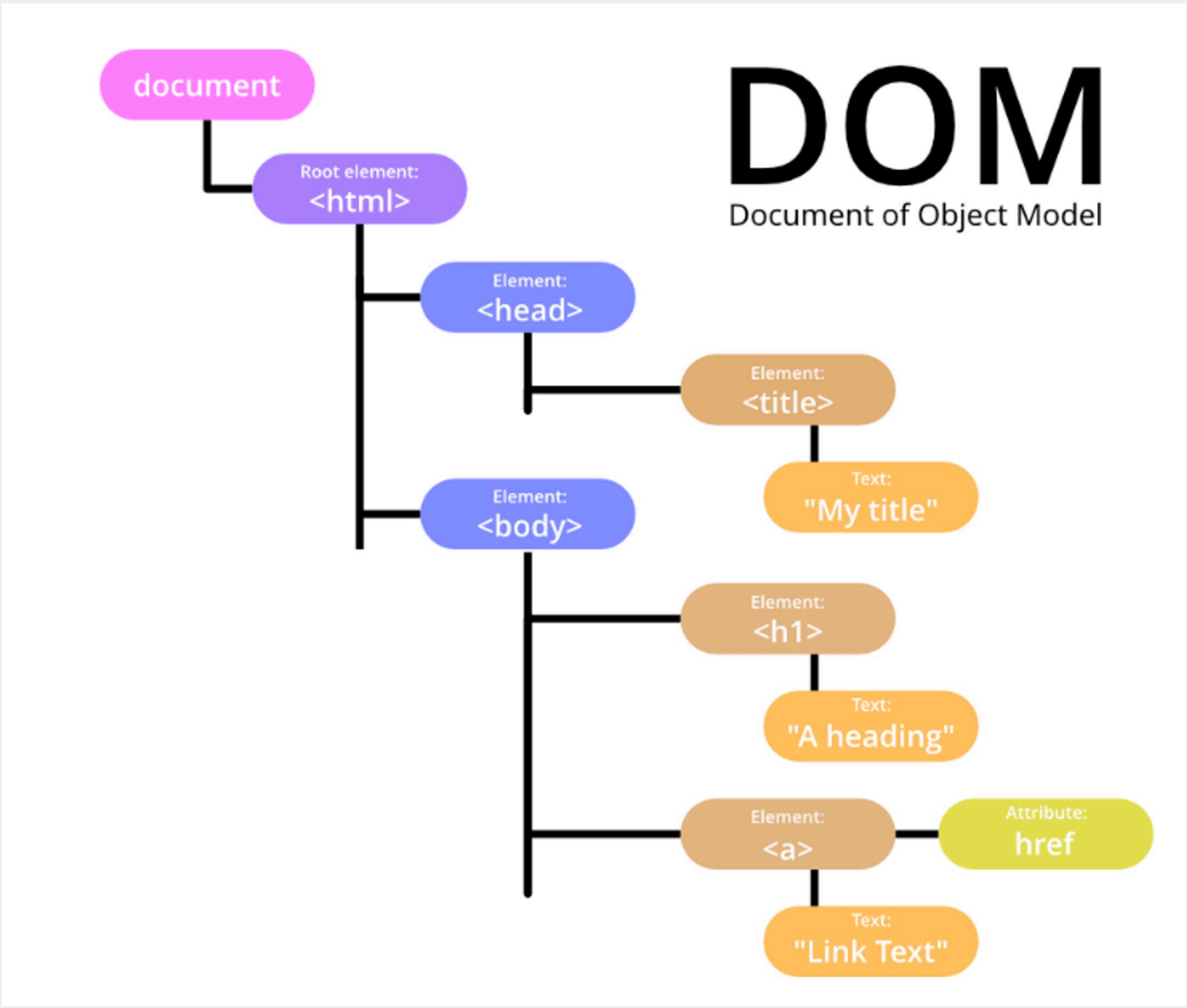


Figura 3: Document Object Model (DOM). Fonte: Geeks for Geeks (2022).

Estrutura:

Node (ou Nó): na árvore de objetos, representa a classe base do DOM, na qual outros objetos da API DOM são baseados, permitindo assim que diferentes tipos de objetos sejam usados de forma semelhante.

Elemento: representa um elemento de programa como um pacote, classe ou método.

Attr: é usado para representar um atributo de um elemento.

Texto: é o conteúdo interno de um **elemento** ou **Attr**.

Document: representa o documento como um todo.

SAIBA MAIS

- https://developer.mozilla.org/pt-BR/docs/Web/API/Document_Object_Model/Introduction
- https://www.w3schools.com/js/js_htmldom.asp

Métodos mais comuns do DOM para selecionar elementos HTML

Métodos do DOM utilizados pelo JavaScript para selecionar elementos HTML

Para realizar alguma operação em um elemento HTML, precisamos encontrar esse elemento no JavaScript, para então alterar seu valor interno, seu estilo, seu foco, e assim por diante. Para isso, temos os métodos descritos a seguir.

```
document.getElementById()
```

Seleciona o elemento com base no seu id. Em geral, o id é único, logo o método retorna o elemento exclusivamente. Se houver mais elementos com o mesmo id, ele retornará o primeiro que encontrar.

```
document.getElementsByClassName()
```

Seleciona o elemento com base no seu nome de classe. Pode retornar um array de elementos que tenham a mesma classe.

```
document.querySelector('nome')
```

Seleciona um único elemento, o primeiro que satisfaz seu parâmetro, que pode ser um id, um nome de classe ou mesmo uma tag HTML.

```
document.querySelectorAll('.class1 p')
```

Seleciona todos os elementos que satisfazem o seu parâmetro, que pode ser um id, um nome de classe ou mesmo uma tag HTML. No exemplo, são retornados os elementos `<p>` que estão subordinados (agrupados) em uma classe `.class1`.

Atribuindo eventos JavaScript aos elementos do DOM corretamente

Para manipular eventos de forma moderna e seguindo as melhores práticas, utilizamos o método ***addEventListener*** para associar eventos aos elementos do DOM (ao invés de utilizar atributos HTML diretamente nos elementos). Além disso, garantimos que nossa interação com os elementos da página aconteça somente após o carregamento completo do DOM utilizando o evento ***DOMContentLoaded***.

Principais eventos JavaScript

Evento	Descrição
click	Disparado quando um elemento é clicado.
mouseover	Disparado quando o ponteiro do mouse passa sobre um elemento.
mouseout	Ocorre quando o ponteiro do mouse sai de um elemento.
keydown	Disparado quando uma tecla é pressionada.
keyup	Disparado quando uma tecla é liberada.
change	Disparado quando o valor de um elemento de entrada muda.
submit	Disparado quando um formulário é enviado.
focus	Ocorre quando um elemento recebe foco.
blur	Disparado quando um elemento perde o foco.

addEventListener

O método ***addEventListener*** permite adicionar múltiplos manipuladores de eventos a um elemento sem sobrescrever manipuladores de eventos existentes.

Exemplo de uso:

```
</>
```

```
//Adiciona evento de click a um botão com id=meuBotao
document.getElementById('meuBotao').addEventListener('click', function() {
    alert('Botão clicado!');
});

//Adiciona evento de change a um input com id=cpf
document.getElementById('cpf').addEventListener('change', function() {
    alert('Realizando validação do CPF!');
});
```

```

```

DOMContentLoaded

Além dos eventos padrão, como o click, change, focus, etc., é importante ficarmos atento ao ***DOMContentLoaded***, que é disparado quando o HTML inicial foi completamente carregado e analisado pelo navegador, ou seja, no momento de seu disparo todos os elementos do DOM já estão disponíveis para serem acessados pelo JavaScript, e assim garantimos que a manipulação da página irá ocorrer sem erros inesperados.

Exemplo de uso:

```
</>
```

```
document.addEventListener('DOMContentLoaded', function() {
    //Imprime no console um aviso quando a página foi carregada
    console.log('O DOM foi completamente carregado e analisado.');
    // Inserimos aqui nosso código JavaScript que manipula elementos do DO
});
```

Agora vamos ver um exemplo completo:

```
</>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de Eventos com JavaScript</title>
</head>
<body>

    <h2>Exemplo de Múltiplos Eventos com JavaScript</h2>

    <button id="botaoClique">Clique em mim</button>
    <button id="botaoHover">Passe o mouse sobre mim</button>
    <p id="mensagemClique"></p>
    <p id="mensagemHover"></p>

    <script>
    //Adiciona evento de carregamento de página ao documento
    document.addEventListener('DOMContentLoaded', function() {

        // Atribuindo evento de clique ao primeiro botão
```

Explicação do Exemplo:

1. **DOMContentLoaded**: Garante que o código dentro da função só seja executado após o DOM ter sido completamente carregado e analisado.
2. **addEventListener para Clique**: Adiciona um manipulador de evento click ao primeiro botão, que atualiza o conteúdo do parágrafo mensagemClique quando o botão é clicado.
3. **addEventListener para Mouseover**: Adiciona um manipulador de evento mouseover ao segundo botão, que atualiza o conteúdo do parágrafo mensagemHover quando o mouse passa sobre o botão.
4. **addEventListener para Mouseout**: Adiciona um manipulador de evento mouseout ao segundo botão, que atualiza o conteúdo do parágrafo mensagemHover quando o mouse sai do botão.

Resultado Esperado

- Ao clicar no botão "Clique em mim", o parágrafo mensagemClique exibirá "O botão de clique foi clicado!".
- Ao passar o mouse sobre o botão "Passe o mouse sobre mim", o parágrafo mensagemHover exibirá "Você passou o mouse sobre o botão!".
- Ao tirar o mouse do botão "Passe o mouse sobre mim", o parágrafo mensagemHover exibirá "Você tirou o mouse do botão!".

SAIBA MAIS

- https://www.w3schools.com/js/js_htmldom_elements.asp
- https://developer.mozilla.org/pt-BR/docs/Web/API/Document/getElementById
- https://developer.mozilla.org/pt-BR/docs/Web/API/Document/getElementsByClassName
- https://developer.mozilla.org/pt-BR/docs/Web/API/Document/getElementsByName
- https://developer.mozilla.org/pt-BR/docs/Web/API/Document/getElementsByTagName
- https://developer.mozilla.org/pt-BR/docs/Web/API/Document/querySelector
- https://developer.mozilla.org/pt-BR/docs/Web/API/Document/querySelectorAll

Praticando comandos no JavaScript

Agora que vimos algumas das principais estruturas do JavaScript, vamos ver essas operações em funcionamento, para verificar como elas podem auxiliar a incrementar a interatividade do documento web, adicionando dinamicidade aos elementos da página HTML.

Exemplo 1: Alterando um conjunto de elementos HTML

EXPERIMENTE

PRÁTICA: Selecionando elemento HTML com JavaScript

Um grupo de elementos `<p>` HTML, subordinados a uma `<div class="textos">`, é alterado conjuntamente via laço de repetição. Verifique os comentários ao lado do código, para acompanhar a lógica do programa.

Código HTML com JavaScript

```html
<!DOCTYPE html>
<head>
    <meta charset="utf-8">
    <title>Teste de Comandos JS</title>
    <style>
        html {
            font-family: 'Lucida Sans';
        }

        body {
            width: 70%;
            max-width: 900px;
            min-width: 500px;
            margin: 0 40px;
        }
    </style>
</head>
<body>
    <h2>Comandos JS</h2>
    <h4>Laço de repetição</h4>
    <div class="textos">
        <p class="texto1"></p>
        <p class="texto2"></p>
        <p class="texto3"></p>
    </div>
    <script>
        var textos = document.querySelectorAll('.textos p');
        for (let i = 0; i < textos.length; i++) {
            textos[i].textContent = i + 1;
        }
    </script>
</body>
</html>
```

Observe no código HTML:

O elemento HTML

`<div class="textos">` agrupa outros 3 elementos `<p>`.

um dos elementos de **class="textos"** é acessado pelo laço de repetição

`for (... ; ...; ...)` .

O método **`querySelectorAll('.textos p')`**

retorna em um array

var textos, os elementos **<p>** agrupados em uma **class="textos"**, que são:

```
<p class="texto1"></p>
```

```
<p class="texto2"></p>
```

```
<p class="texto3"></p>
```

O valor **textos.length** informa o total de elementos do *array*.

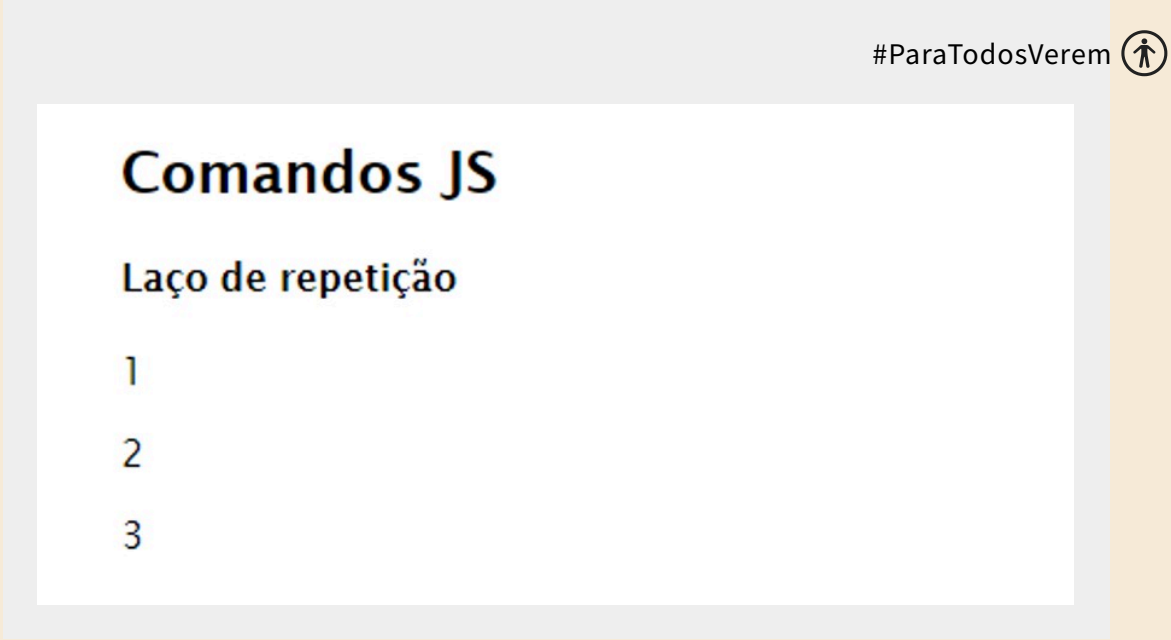


Figura 4: Alterando um grupo de elementos HTML com JavaScript. Fonte: A autora (2023).

Exemplo 2: Alterando os atributos de elementos HTML

EXPERIMENTE

PRÁTICA: Selecionando elemento HTML com JavaScript

O uso de um botão HTML (`<button>`) permite chamar um trecho em JavaScript que acessa o elemento HTML `<img>` e alterar o valor do seu atributo `src` .

Código HTML com JavaScript

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
    <meta charset="utf-8">
    <title>Teste de Comandos JS</title>
    <style>
    html {
        font-family: 'Lucida Sans';
    }

    body {
        width: 70%;
        max-width: 900px;
        min-width: 500px;
        margin: 0 40px;
    }

    button {
```

Para que o exemplo funcione, você deve ter no mesmo diretório de seu HTML uma pasta contendo uma imagem de uma lâmpada acesa e outra apagada. Se quiser pode efetuar o download de imagens **aqui**.

#ParaTodosVerem

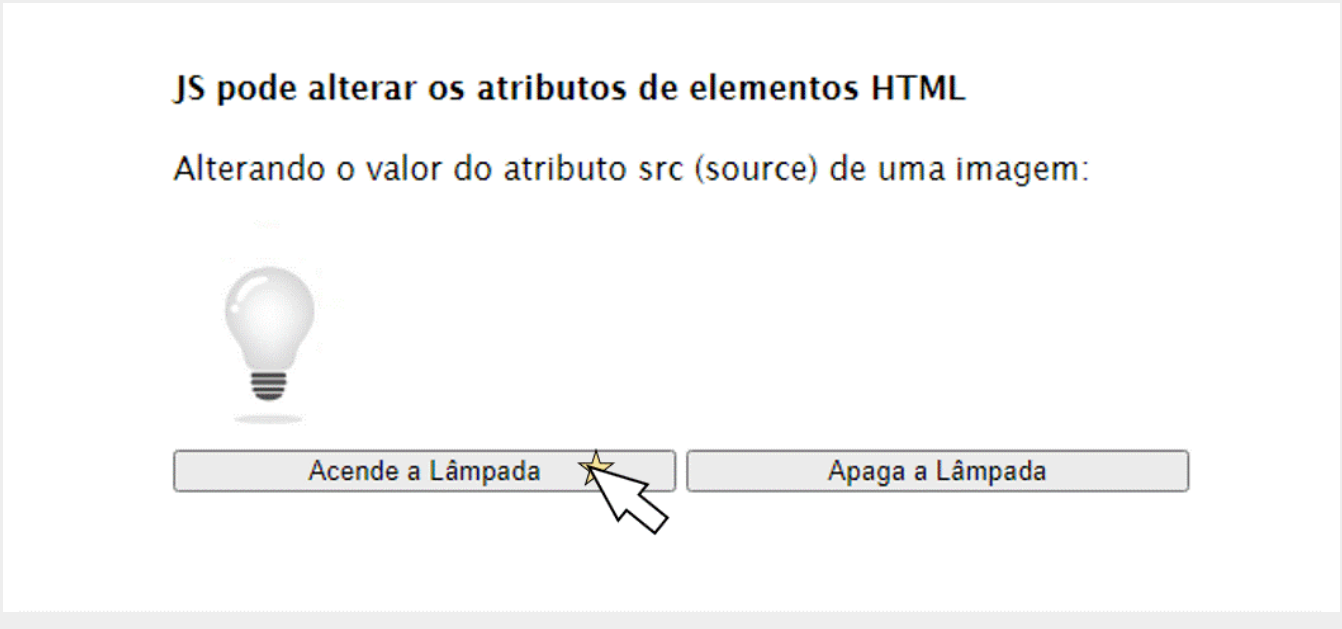


Figura 5: Alterando atributos HTML com JavaScript. Fonte: A autora (2023).

Exemplo 3: Alterando o conteúdo interno de elementos HTML

EXPERIMENTE

PRÁTICA: Troca de conteúdo de elemento HTML com JavaScript

O uso de um botão HTML (`<button>`) permite chamar um trecho em JavaScript que acessa o elemento HTML `<p>` e alterar o valor do seu conteúdo interno.

Código HTML com JavaScript

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
    <meta charset="utf-8">
    <title>Teste de Comandos JS</title>
    <style>
    html {
        font-family: 'Lucida Sans';
    }

    body {
        width: 70%;
        max-width: 900px;
        min-width: 500px;
        margin: 0 40px;
    }

    button {
```

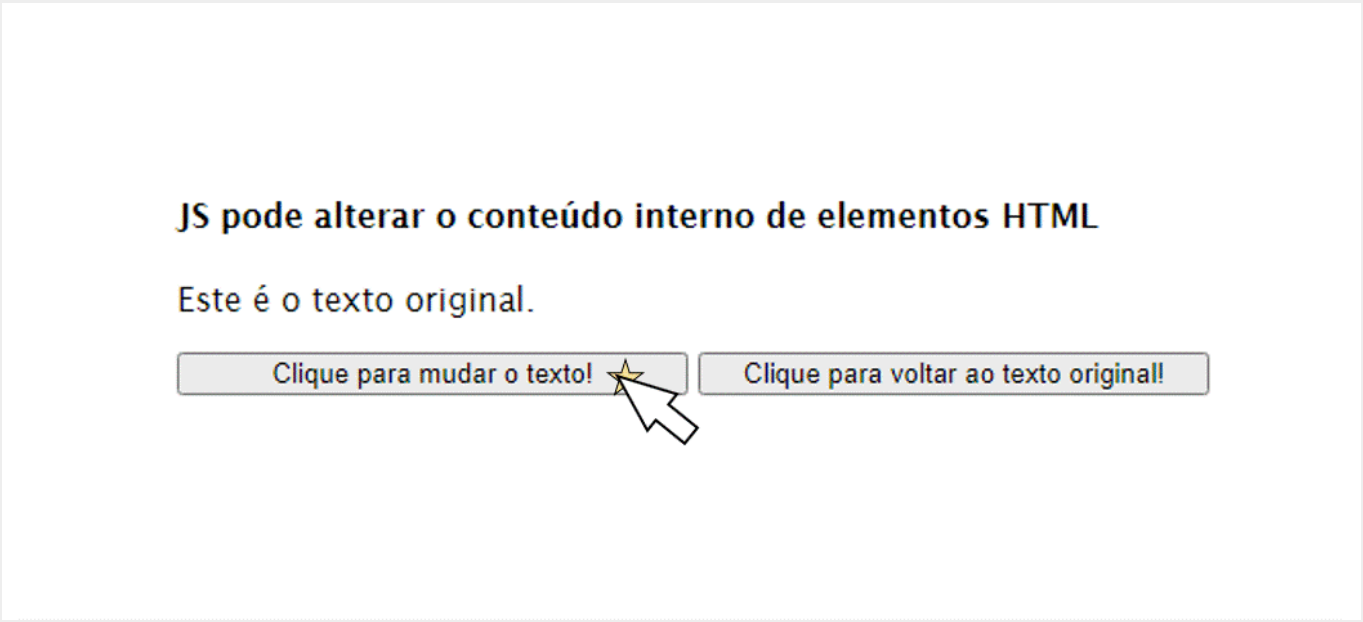


Figura 6: Alterando conteúdo HTML com JavaScript. Fonte: A autora (2023).

Exemplo 4: Alterando o estilo de elementos HTML

EXPERIMENTE

PRÁTICA: Troca de estilo de elemento HTML com JavaScript

O uso de um botão HTML (`<button>`) permite chamar um trecho em JavaScript que acessa o elemento HTML `<p>` e alterar o valor do seu CSS `style.fontSize` .

Código HTML com JavaScript

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
    <meta charset="utf-8">
    <title>Teste de Comandos JS</title>
    <style>
    html {
        font-family: 'Lucida Sans';
    }

    body {
        width: 70%;
        max-width: 900px;
        min-width: 500px;
        margin: 0 40px;
    }

    button {
```

#ParaTodosVerem

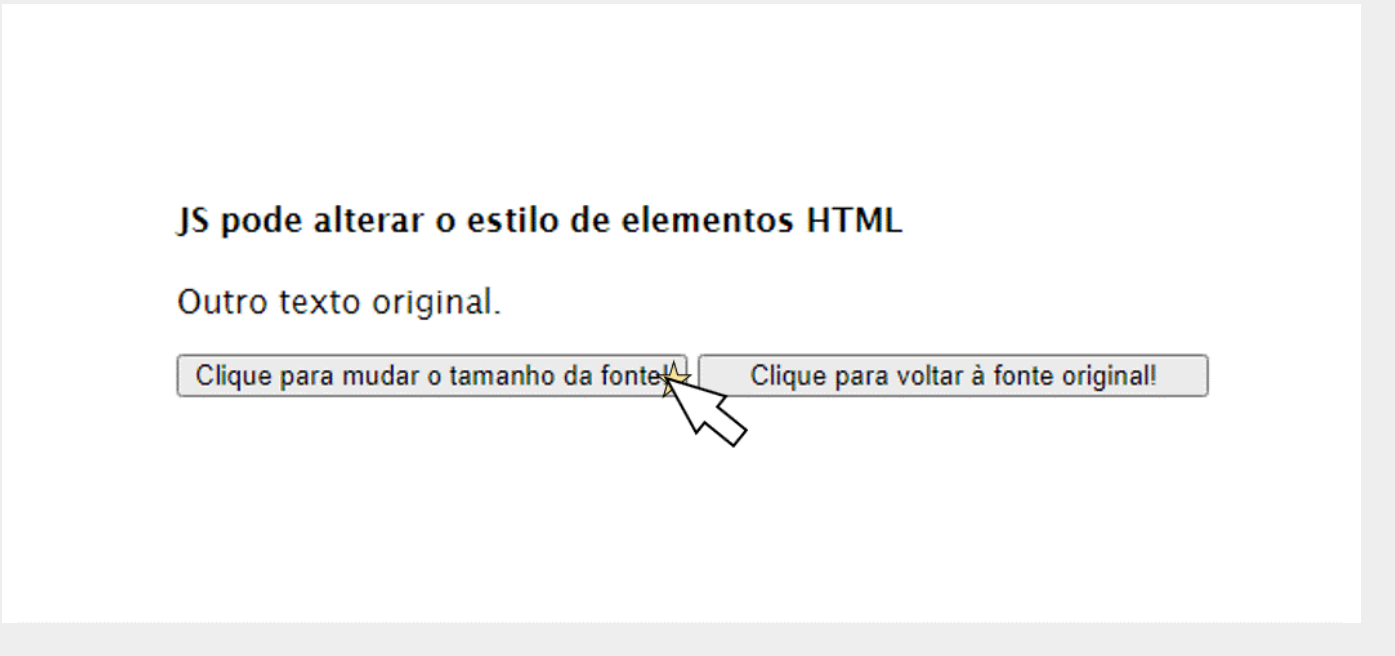


Figura 7: Alterando estilo HTML com JavaScript. Fonte: A autora (2023).

Exemplo 5: Ocultando ou exibindo um elemento HTML

EXPERIMENTE

PRÁTICA: Troca de estilo para ocultar ou exibir elemento HTML com JavaScript

O uso de um botão HTML (`<button>`) permite chamar um trecho em JavaScript que acessa o elemento HTML `<p>` e alterar o valor do seu CSS `style.display.`

Código HTML com JavaScript

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
    <meta charset="utf-8">
    <title>Teste de Comandos JS</title>
    <style>
    html {
        font-family: 'Lucida Sans';
    }

    body {
        width: 70%;
        max-width: 900px;
        min-width: 500px;
        margin: 0 40px;
    }

    button {
```

#ParaTodosVerem

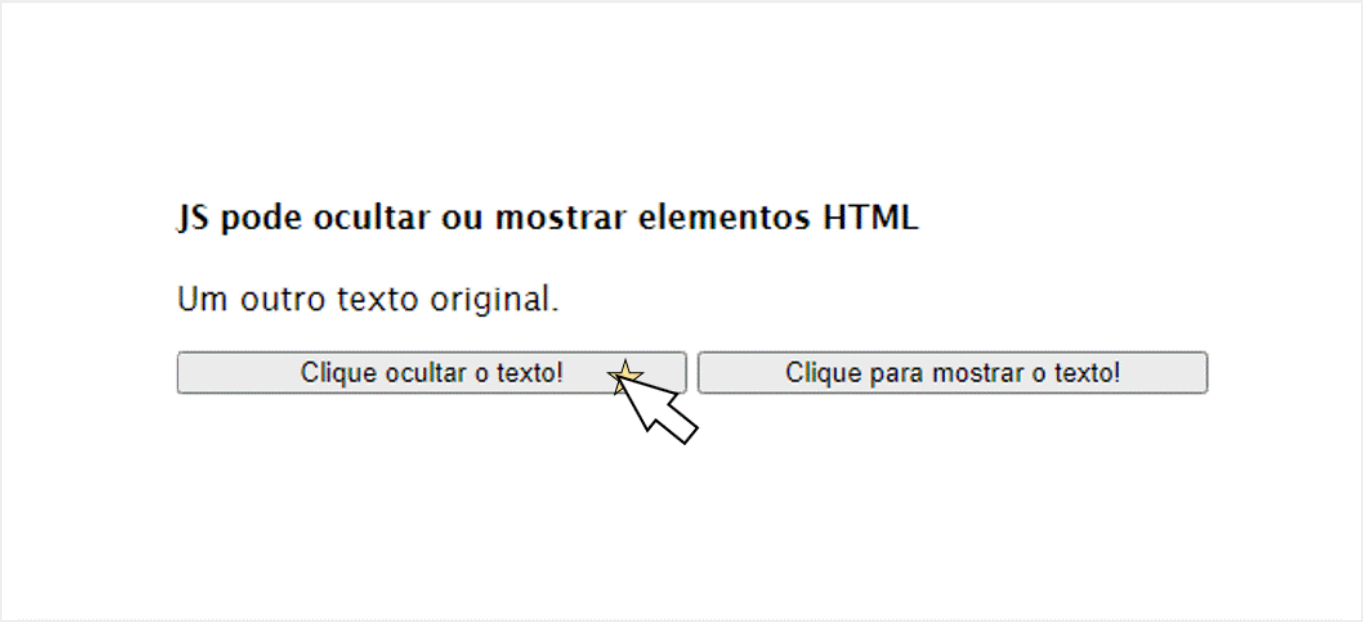


Figura 8: Alterando estilo HTML com JavaScript. Fonte: A autora (2023).

Atividade Formativa 1

Execute o exercício proposto na **videoaula-tutorial** a seguir, para criar um **jogo para adivinhar o número**, que usa HTML, CSS e JavaScript. Nesta prática, além de criar vários elementos HTML, você poderá verificar como podemos adicionar lógica de programação e dinamicidade para incrementar a interação entre usuário e a página *web*. Indicamos sempre o uso de um ambiente de desenvolvimento adequado, como o VS Code (https://code.visualstudio.com/download), que sempre auxilia na escrita dos códigos.

Vamos criar um jogo que pede ao usuário um palpite sobre um número secreto, selecionado aleatoriamente. Ele tem 10 chances de acertar, e o jogo ainda dá dicas se o usuário está perto ou não de adivinhar o número.

Para isso, o jogo usa elementos HTML, estilo CSS e programação com JavaScript para: criar e manipular variáveis, acessar e alterar os elementos HTML, além de controlar o fluxo do jogo para permitir até 10 palpites e verificar se algum palpite adivinha o número secreto.

Você pode utilizar os códigos de referência, apresentados a seguir, para incluir o código JavaScript.

ATENÇÃO: O código apresentado apresenta duas características específicas:

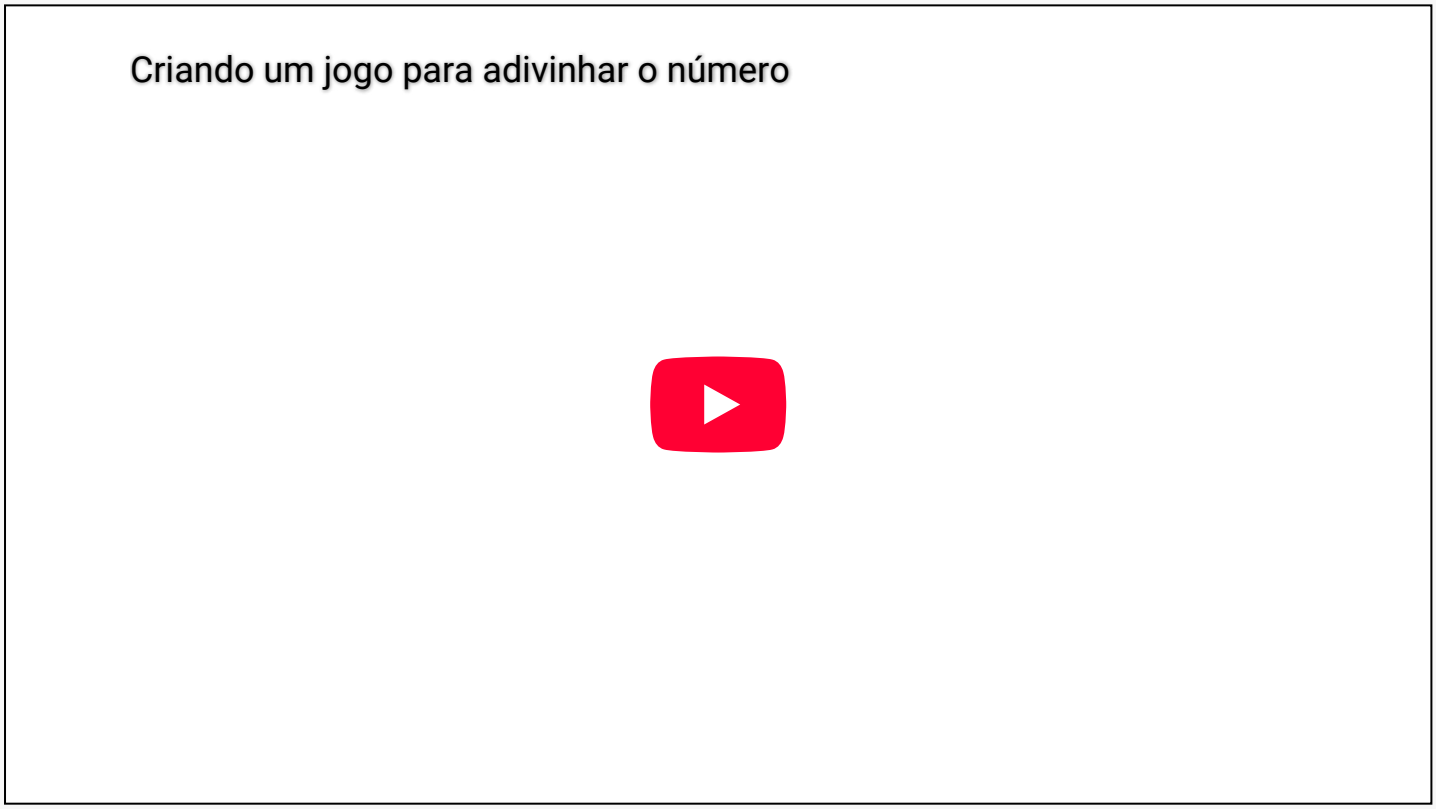
- Utilização da palavra reservada “var” para a declaração de variáveis JavaScript
- Atribuição de eventos JavaScript aos elementos utilizando um atributo diretamente no HTML (por exemplo: onclick="conferirPalpite();")

Ao realizar a atividade você deve melhorar o código para ele ficar mais aderente às boas práticas que foram mostradas nos exemplos desta Unidade, ou seja:

- Substitua a utilização de “var” por “let”
- Realize a atribuição de eventos utilizando a função addEventListener
- Utilize o evento DOMContentLoaded para garantir que a página esteja carregada antes de atribuir eventos

Mãos à obra!

Criando um jogo para adivinhar o número



Este tutorial é baseado no exemplo **Um primeiro mergulho no JavaScript**, disponível em https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/A_first_splash.

DICA

Códigos de referência

Estrutura HTML

```
<!DOCTYPE html>
<head>

    <meta charset="utf-8">
    <title>Jogo Adivinhe um número</title>
    <style>
        html {
            font-family: Verdana, Geneva, Tahoma, sans-s
        }

        body {
            width: 70%;
            max-width: 900px;
            min-width: 500px;
            margin: 0 40px;
        }
    </style>
</head>

<body>

    <h1>Jogo Adivinhe um número</h1>
    <p>Selecionamos um número aleatório entre <b>1</b> e
    <p>Veja se consegue adivinhar em <b>10</b> chances o
    <p>Nós lhe diremos se seu palpite está com valor <b>

    <div class="form">
        <label for="campoPalpite">Digite seu palpite: </
        <input type="text" id="campoPalpite" class="camp
        <input type="submit" value="Enviar palpite" clas
    </div>

    <div class="cjtoResultados">
        <p class="palpites"></p>
        <p class="ultimoResultado"></p>
        <p class="baixoOuAlto"></p>
    </div>

    <script>
        // O Código Javascript será inserido aqui.
    </script>

</body>
</html>
```

JavaScript para tratar variáveis

```
// Cria um número aleatório entre 1 e 100
  var numeroAleatorio = Math.floor(Math.random() * 100)

// Cria variáveis ligadas a elementos com as classes ind
// para ATUALIZAR esses elementos de acordo com as jogad
  var palpites = document.querySelector('.palpites');
  var ultimoResultado = document.querySelector('.ultimoR
  var baixoOuAlto = document.querySelector('.baixoOuAlto

// Cria variáveis para elementos INPUT com as classes in
// para OBTER DADOS desses elementos de acordo com as jo
  var envioPalpite = document.querySelector('.envioPalpi
  var campoPalpite = document.querySelector('.campoPalpi
  var contagemPalpites = 1;  // Inicia a contagem dos pa
  var botaoReinicio;
```

JavaScript para tratar fim de jogo

```
function configFimDeJogo() {
      campoPalpite.disabled = true;  // desabilita campo
      envioPalpite.disabled = true;  // desabilita campo
      botaoReinicio = document.createElement('button');
      botaoReinicio.textContent = 'Iniciar novo jogo';
      document.body.appendChild(botaoReinicio);
      botaoReinicio.addEventListener('click', reiniciarJ
}
```

JavaScript para tratar reinício de jogo

```
function reiniciarJogo() {
      contagemPalpites = 1;

      var reiniciarParas = document.querySelectorAll('.c
      for (var i = 0; i < reiniciarParas.length; i++) {
          reiniciarParas[i].textContent = '';
      }
      botaoReinicio.parentNode.removeChild(botaoReinicio
      campoPalpite.disabled = false;
      envioPalpite.disabled = false;
      campoPalpite.value = '';
      campoPalpite.focus();
      ultimoResultado.style.backgroundColor = 'white';
      numeroAleatorio = Math.floor(Math.random() * 100)
}
```

JavaScript para conferir palpite

```
function conferirPalpite() {
 var palpiteUsuario = Number(campoPalpite.value);
 if (contagemPalpites === 1) {
     palpites.textContent = 'Palpites anteriores: ';
 }
 palpites.textContent += palpiteUsuario + ' '; // Inform

 // IMPORTANTE: este código está incompleto nesta parte.
 // Assista a VIDEOAULA TUTORIAL para completar a lógica

 contagemPalpites++;
 campoPalpite.value = '';
 campoPalpite.focus();
}
```

Conclusão

Olá!

Com esta unidade, finalizamos o planejado na disciplina para o desenvolvimento *web* ***front-end***! Vimos como a linguagem de programação **JavaScript** aumenta consideravelmente o poder de interação e dinamicidade da aplicação com o cliente, a partir de um navegador. E tudo isso sem necessariamente precisar recorrer ao lado servidor da aplicação, aproveitando a capacidade de processamento computacional da máquina do usuário. 😉

Na próxima unidade, começaremos a praticar programação ***back-end*** com **PHP**, uma linguagem de programação feita para ser executada em um servidor **HTTP**, que processa páginas **HTML dinâmicas** para envio ao navegador.

Em resumo, em uma aplicação *web*, temos:

- Lado ***front-end***, ou lado **cliente**, que interage com o usuário e garante que os dados que serão enviados ao lado servidor estejam no formato adequado (faz a validação dos dados). Nele, desenvolvemos com as linguagens **HTML**, **CSS** e **JavaScript**.
- Lado ***back-end***, ou lado **servidor**, que processa as solicitações do usuário e devolve páginas **HTML** geradas dinamicamente, de acordo com a requisição recebida do cliente. Nele, utilizaremos as linguagens **PHP** e **SQL**, que serão vistas nas próximas unidades de estudo.

Até lá!

Referências Bibliográficas

ALVES, W. P. **Desenvolvimento e *design* de *sites***. São Paulo: Erica, 2014.

MILETTO, E. M.; BERTAGNOLLI, S. C. **Desenvolvimento de software II**: Introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014.

TERUEL, E. C. **HTML 5**: Guia prático. 2. ed. Porto Alegre: Bookman, 2014.