Programação Web Front-End
Aula 2 - JavaScript

Profa. Rosangela de Fátima Pereira Marquesone
romarquesone@utfpr.edu.br

Proposta: apresentar as funções para manipulação da API DOM via JavaScript.

Objetivos: espera-se que após essa aula, você tenha habilidade para compreender os seguintes tópicos:

1. Aprender a utilizar JavaScript com DOM HTML
2. Atividade prática com JavaScript

Dicas de aprendizado:

- Execute todos os passos com atenção, compreendendo o que está sendo realizado;
- Procure não copiar código, para ter a prática de digitar o código desenvolvido;
- Pergunte quando tiver alguma dúvida;
- Mantenha um histórico dos códigos desenvolvidos, seja no github ou em algum outro meio de armazenamento (e-mails, google drive, etc.);
- Tenha curiosidade e explore os recursos apresentados.

Tópicos anteriores:

- Compreender o que é HTML
- Compreender o que são tags HTML básicas
- Criar um arquivo .html no Visual Studio (VS) Code
- Abrir o arquivo .html em um navegador
- Visualizar o código-fonte de uma página em um navegador
- Inspecionar a página em um navegador
- Utilizar o Live Server no VS Code
- Aprender a utilizar tags semânticas
- Aprender a inserir links
- Aprender a inserir listas
- Aprender a criar uma página com seu Curriculum Vitae (CV) (atividade prática)
- Aprender a inserir figuras
- Aprender a utilizar a tag semântica <figure>
- Inserir figuras em seu Curriculum Vitae (CV) (atividade prática)
- Aprender a criar formulários
- Criar um formulário (atividade prática)
- Descobrir o que é CSS
- Aprender a sintaxe do CSS
- Aprender os tipos de seletores CSS
- Aprender as formas de inclusão de CSS

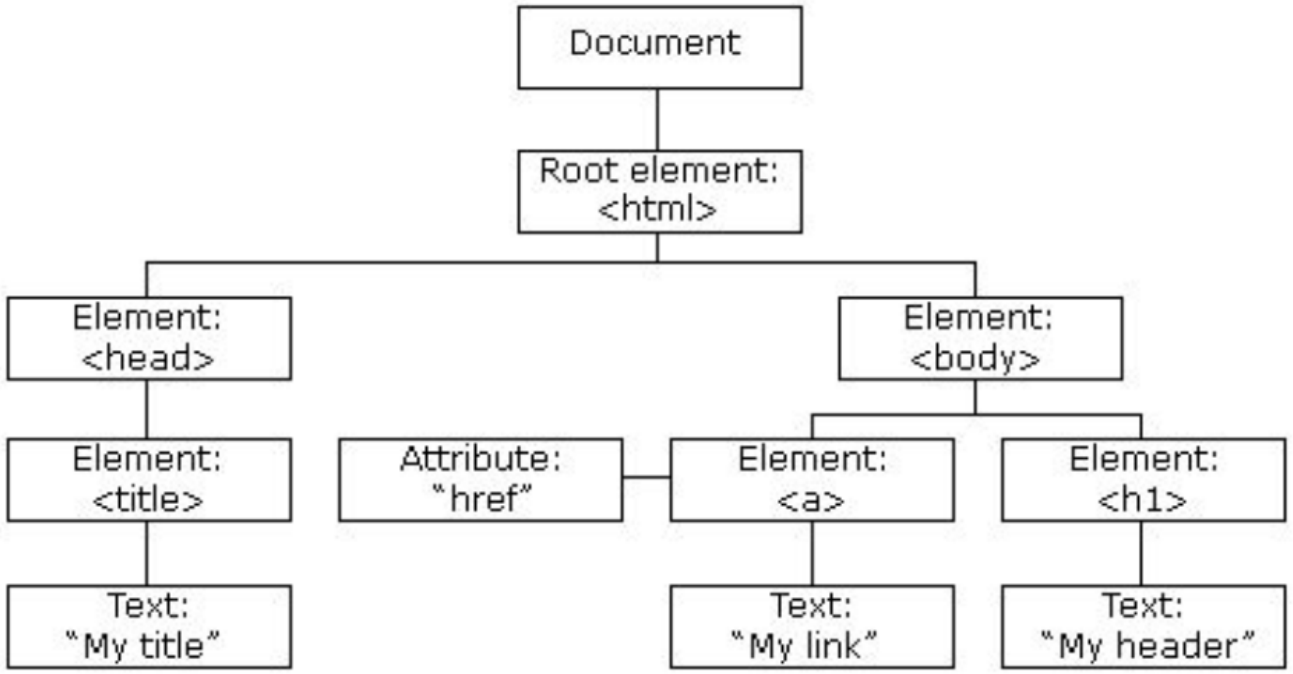
- Aprender a definir cores
- Aprender a alterar as propriedades de texto
- Aprender o conceito de modelo de caixa do CSS
- Aprender a trabalhar com a margem
- Aprender a trabalhar com a borda
- Aprender a trabalhar com o preenchimento (padding)
- Aprender a usar a propriedade display
- Aprender a utilizar a propriedade float
- Aprender a utilizar a propriedade overflow
- Estruturar páginas por meio do modelo de caixa (atividade prática)
- Aprender o conceito de flex-box
- Aprender as propriedades do elemento pai (flex container)
- Aprender as propriedades dos elementos filhos (flex items)
- Descobrir a história da linguagem JavaScript
- Compreender as formas de uso da linguagem JavaScript
- Conhecer as características da linguagem JavaScript
- Atividade prática com JavaScript

Passo 1 - Aprender a utilizar JavaScript com DOM HTML

Como vimos na aula anterior, uma das funcionalidades providas pelo JavaScript é a manipulação de Document Object Model (DOM), ou Modelo de Documento por Objetos.

O DOM é uma interface de programação para os documentos HTML. Ele representa a página de forma que os programas possam alterar a estrutura do documento, modificando seu estilo e conteúdo.

Cada página carregada em um navegador possui um objeto chamado Document, conforme exemplificado na figura a seguir.



Dessa forma, a interface Document serve como um ponto de entrada para o conteúdo da página (Ex.: <body> e <p>).

Veja a seguir um resumo dos principais conceitos relacionados ao DOM HTML:

- **Árvore de Elementos**: O DOM HTML organiza a estrutura de uma página web como uma árvore hierárquica, na qual o elemento <html> é a raiz e os elementos <head> e <body> são seus filhos. Outros elementos HTML, como , <p>, <a>, etc., são aninhados dentro desses elementos pai, formando uma estrutura em árvore que representa a página.

- **Elementos HTML como Objetos**: No DOM HTML, cada elemento HTML é representado como um objeto em JavaScript. Esses objetos têm propriedades e métodos que podem ser usados para acessar e manipular o conteúdo e os atributos dos elementos.

- **Acesso aos Elementos**: possibilita acessar elementos HTML no DOM HTML usando seletores, como getElementById, getElementsByClassName, getElementsByTagName, etc. Esses métodos permitem que você selecione elementos específicos na estrutura DOM.

- **Manipulação do Conteúdo**: possibilita alterar o conteúdo de elementos HTML, como texto e atributos, usando propriedades como innerHTML, value, setAttribute, getAttribute, entre outras.

- **Eventos**: O DOM HTML permite a vinculação de eventos a elementos HTML. Você pode usar o método addEventListener para ouvir eventos, como cliques de mouse, envios de formulários, digitação e muito mais, e responder a esses eventos com funções de retorno.

- **Navegação na Árvore**: Você pode percorrer a árvore DOM para acessar elementos pai, filhos e irmãos. Isso é útil para navegar pela estrutura da página e acessar elementos relacionados.

A seguir veremos os seguintes exemplos de APIs em scripting de páginas web e XML usando o DOM.

- document.getElementsByTagName(name)
- document.getElementById(id)
- element.getAttribute()
- document.createElement(name)
- parentNode.appendChild(node)
- document.createTextNode(data)
- element.innerHTML
- element.addEventListener()

document.getElementsByTagName(name)

Parte da API DOM em JavaScript que permite selecionar elementos HTML em uma página da web com base em seus nomes de tag. Essa função retorna uma coleção (não uma matriz) de todos os elementos com a tag especificada.

Por exemplo, se você deseja selecionar todos os elementos <p> (parágrafos) em uma página da web, você pode fazer o seguinte:

```
var textos = document.getElementsByTagName("p");
```

Esse comando retorna uma coleção de todos os elementos <p> na página.

Você pode acessar e manipular esses elementos por meio da coleção usando índices, assim como faria com uma matriz. Por exemplo, para acessar o primeiro parágrafo:

```
var primeiroParagrafo = textos[0];
```

Confira outro exemplo:

```
<html>
<head>
</head>
<body>
<h2>Aprendendo JavaScript</h2>
<script type="text/javascript">
alert(document.getElementsByTagName("h2")[0].innerHTML);
```

```
</script>
</body>
</html>
```

document.getElementById(id) e element.getAttribute()

document.getElementById(id) é uma função para selecionar um elemento HTML com base no valor de seu atributo id. Cada elemento HTML pode ter um atributo id exclusivo que o identifica de maneira única na página.

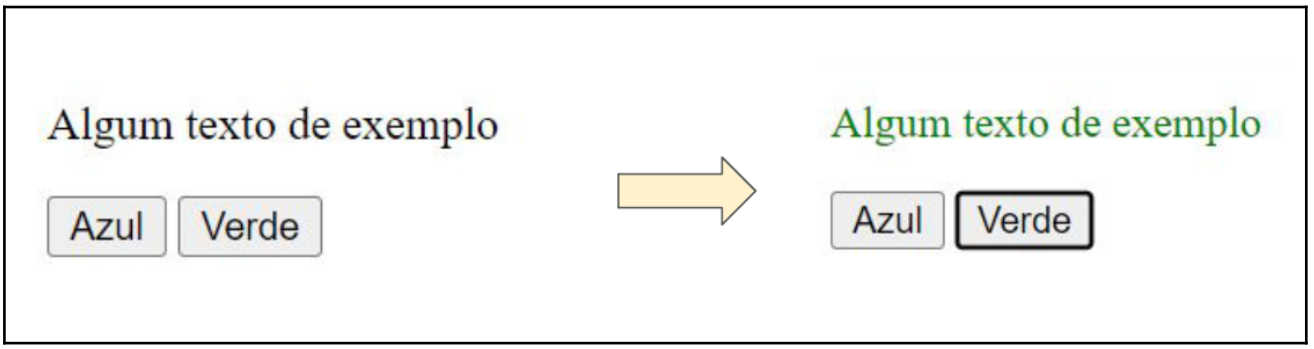
Quando você chama document.getElementById(id), ele retorna o elemento que possui o atributo id correspondente ao valor id passado como argumento.

element.getAttribute(attribute) é uma função que permite acessar o valor de qualquer atributo específico de um elemento HTML. Você passa o nome do atributo que deseja acessar como argumento, e a função retorna o valor desse atributo. Veja um exemplo a seguir:

```
var imagem = document.getElementById("minhaImagem");
var src = imagem.getAttribute("src");
```

Você também pode acessar o atributo diretamente pelo elemento obtido, conforme exemplo a seguir.

```
<html>
<head>
<script>
function mudarCor(novaCor) {
var elemento = document.getElementById("texto");
elemento.style.color = novaCor;
}
</script>
</head>
<body>
<p id="texto">Algum texto de exemplo</p>
<button onclick="mudarCor('blue');">Azul</button>
<button onclick="mudarCor('green');">Verde</button>
</body>
</html>
```



document.createElement(name), parentNode.appendChild(node) e document.createTextNode(data)

document.createElement(name) é uma função para criar um novo elemento HTML com base no nome da tag especificado como argumento. Ela retorna um novo elemento vazio com a tag especificada, que pode ser posteriormente personalizado e manipulado antes de ser adicionado à página.

Exemplo:

```
var novoDiv = document.createElement("div");
```

parentNode.appendChild(node) é uma função que permite adicionar um nó (elemento, texto, etc.) como um filho de outro nó. O nó de destino é referenciado como parentNode, e o nó que você deseja adicionar é referenciado como node. Isso é normalmente usado para adicionar o elemento criado com createElement (ou qualquer outro nó) a um elemento existente na página.

Veja um exemplo de como adicionar o novo criado anteriormente a um elemento pai (por exemplo, o <body>):

```
var body = document.body;
body.appendChild(novoDiv);
```

Assim, ao combinar essas duas funções, você pode criar um novo elemento com createElement e, em seguida, adicioná-lo como um filho a um elemento existente com appendChild.

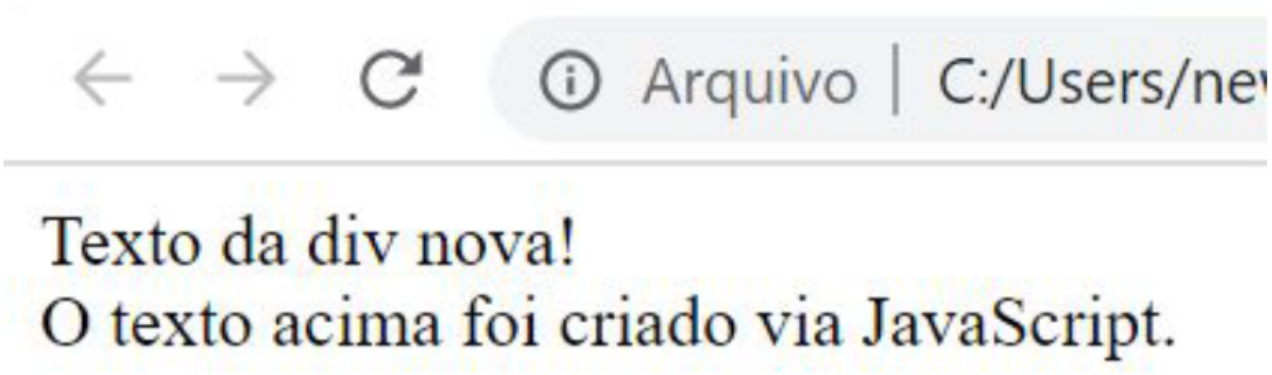
document.createTextNode(data) é uma função para criar e inserir nós de texto em elementos HTML. Um nó de texto é uma sequência de texto que pode ser inserida em um elemento HTML para exibir texto na página.

```
var texto = "Aprendendo JavaScript.";
var node = document.createTextNode(texto);
```

Veja um exemplo utilizando as 3 funções.

```
<html>
<head>
<title>Exemplo de JavaScript</title>
</head>
<body>
```

```
<div id="div1">O texto acima foi criado via JavaScript.</div>
<script>
document.body.onload = addElemento;
function addElemento () {
var divNova = document.createElement("div");
var conteudoNovo = document.createTextNode("Texto da div nova!");
divNova.appendChild(conteudoNovo);
var divAtual = document.getElementById("div1");
document.body.insertBefore(divNova, divAtual);
}
</script>
</body>
</html>
```



element.innerHTML

element.innerHTML é uma propriedade da API DOM para acessar ou modificar o conteúdo HTML interno de um elemento HTML específico. Essa propriedade é usada para ler ou definir o conteúdo HTML de um elemento, permitindo a manipulação do conteúdo de forma dinâmica.

Para acessar o conteúdo HTML de um elemento usando element.innerHTML, você pode simplesmente acessar a propriedade, conforme exemplo a seguir:

```
var meuElemento = document.getElementById("meuElemento");
var conteudoHTML = meuElemento.innerHTML;
console.log(conteudoHTML);
```

No exemplo acima, conteudoHTML conterá o conteúdo HTML interno do elemento com o id "meuElemento".

Você também pode usar element.innerHTML para definir o conteúdo HTML de um elemento. Isso permite que você atualize dinamicamente o conteúdo do elemento. Veja um exemplo:

```
var meuElemento = document.getElementById("meuElemento");
meuElemento.innerHTML = "<p>Novo conteúdo HTML</p>";
```

Perceba que ao usar element.innerHTML para definir o conteúdo, pois ele substituirá completamente o conteúdo existente do elemento.

element.addEventListener()

element.addEventListener() é um método da API DOM para vincular funções de retorno de chamada (*event handlers*) a elementos HTML para lidar com eventos específicos. Eventos podem ser compreendidos como ações que ocorrem no navegador, como cliques de mouse, pressionamentos de tecla, carregamento de páginas, etc.

O método addEventListener() é amplamente usado para criar interatividade em páginas da web, pois permite responder a ações do usuário e realizar ações com base nesses eventos.

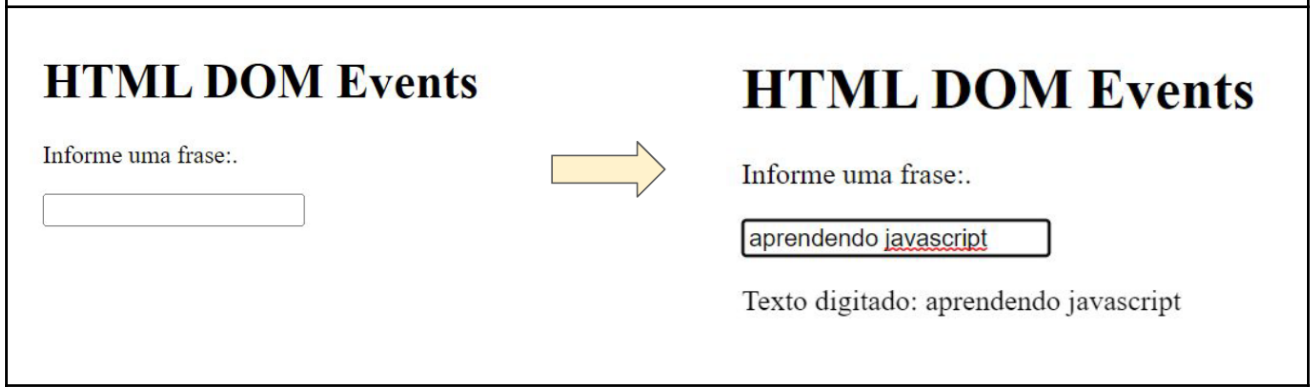
Veja um exemplo:

```
var meuBotao = document.getElementById("meuBotao");

meuBotao.addEventListener("click", function() {
  alert("Botão clicado!");
});
```

Nesse outro exemplo, é possível utilizar o evento oninput, para mostrar o texto digitado.

```
<!DOCTYPE html>
<html>
<body>
<h1>HTML DOM Events</h1>
<p>Informe uma frase:.</p>
<input type="text" id="myInput" oninput="mostraTexto()">
<p id="demo"></p>
<script>
function mostraTexto() {
  let texto = document.getElementById("myInput").value;
  document.getElementById("demo").innerHTML = "Texto digitado: " + texto;
}
</script>
</body>
</html>
```



Passo 2 - Atividade prática com JavaScript

Para validar o conhecimento visto nos tópicos anteriores, realize as atividades a seguir, enviando-as no moodle da disciplina.

1 - Crie um parágrafo:
- Crie um novo elemento <p> usando document.createElement().
- Crie um nó de texto com algum conteúdo usando document.createTextNode().
- Adicione o nó de texto ao parágrafo.
- Adicione o parágrafo ao <body> da página.

2. Lista de Compras:
- Crie uma lista não ordenada <ul> usando document.createElement().
- Crie vários elementos de lista <li> usando document.createElement().
- Crie nós de texto para cada item da lista usando document.createTextNode().
- Adicione os nós de texto aos elementos <li>.
- Adicione os elementos <li> à lista <ul>.
- Adicione a lista <ul> ao <body>.

3. Troca de Texto:
- Selecione um elemento com um id usando document.getElementById().
- Altere o texto do elemento selecionado usando a propriedade textContent ou innerHTML

4. Contador de Cliques:
- Crie um botão e um parágrafo.
- Quando o botão for clicado, incremente um contador e atualize o texto do parágrafo com o número de cliques.

5. Contador de Caracteres:
- Crie um campo de texto (<input type="text">) e um parágrafo.
- Use o evento oninput para contar e exibir o número de caracteres inseridos no campo de texto em tempo real.

Entregar um arquivo .zip com os códigos realizados em cada atividade.

Considerações finais

Caso tenha chegado até aqui, você conseguiu completar o conteúdo do segundo tutorial sobre JavaScript. A partir desses recursos, você passa a compreender a base para o desenvolvimento de funcionalidades via JavaScript. Nas aulas seguintes veremos ainda mais funcionalidades para tornar as páginas ainda mais dinâmicas.

Bom estudo!