JavaScript: Manipulação do DOM, Seletores e Eventos

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- Utilização de Seletores
- Manipulação do DOM
- Eventos

Seletores

Relembrando CSS

Ao estudar CSS aprendemos
que seletores são usados para
identificar elementos no HTML.
Existe uma estrutura de estilo que envolve:

seletor, declaração, propriedade, valor

Exemplo

```
h1 {
  font-family: Arial;
  font-size: 20pt;
  color: blue;
}
```

Tipos de seletores

- Por **TagName**
  - texto com **nome** do elemento

```
h1 {
  color: blue
}
```

- Por **Id**
  - texto com prefixo **#** *(hash)*

```
#btn-1 {
  background-color: red
}
```

- Por **ClassName**
  - texto com prefixo **.**

```
.subtitulo {
  font-size: 10px;
}
```

- Seletor **universal**
  - ***** seleciona todos os elementos

```
* {
  font-family: Arial, Helvetica, sans-serif
}
```

Seletores Combinados

```html
<div>
  <p>
    Parágrafo Filho 1
  </p>
  <a>
    <p>Parágrafo Neto 1</p>
  </a>
  <p>
    Parágrafo Filho 2
  </p>
  <a>
    <p>Parágrafo Neto 2</p>
  </a>
</div>
```

Antes

Parágrafo Filho 1

Parágrafo Neto 1

Parágrafo Filho 2

Parágrafo Neto 1

Filhos

Sintaxe: **a > b**

Seleciona todos os elementos “b” que são filhos diretos do elemento “a”.

Exemplo: para selecionar todo os itens de uma lista

```css
div > p {
  color: blue
}
```

Depois

Parágrafo Filho 1

Parágrafo Neto 1

Parágrafo Filho 2

Parágrafo Neto 2

Descendentes

Sintaxe: **a b**

Seleciona todos os elementos “b” que são descendentes do elemento “a”, não necessariamente filhos.

Exemplo: para selecionar todo os itens de uma lista

```css
div p {
  color: blue
}
```

Depois

Parágrafo Filho 1

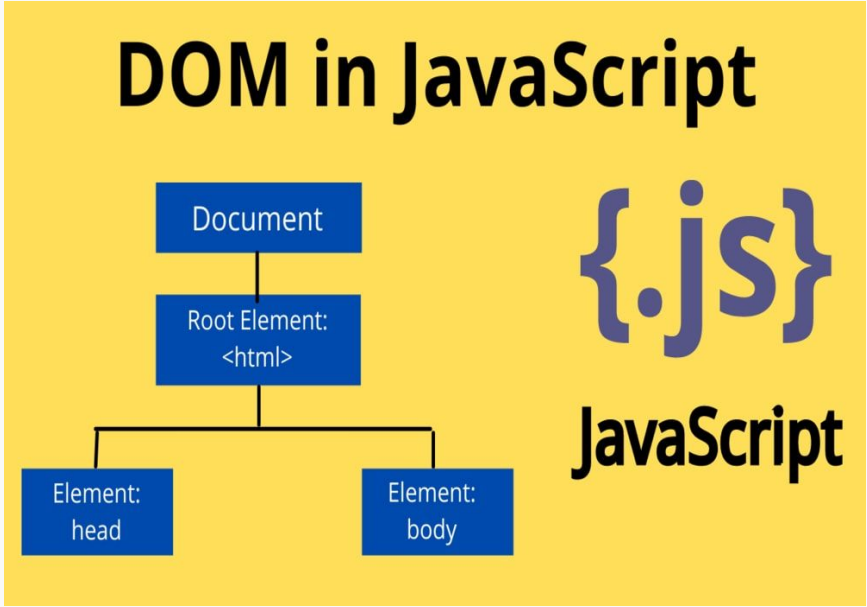
Parágrafo Neto 1

Parágrafo Filho 2

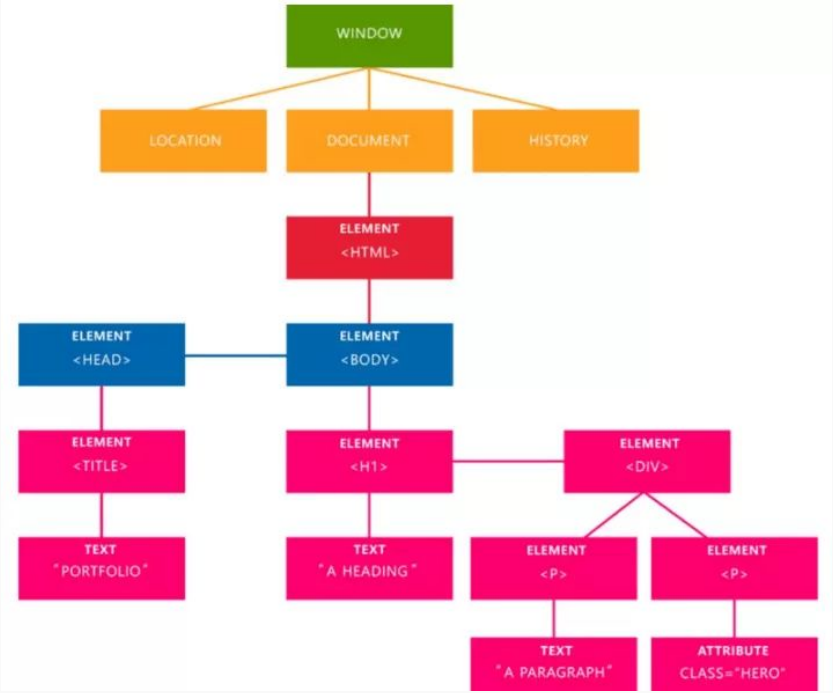
Parágrafo Neto 1

Manipulação do DOM: Porque?

Na última aula estudamos funções no JS. Agora iremos aprender como **utilizar o JavaScript** (e suas funções) para **manipular o HTML**: DOM (Document Element Model - modelo do objeto do documento).



Manipulação do DOM: O que é?



Ele fornece uma **representação estruturada** do documento como se fosse uma espécie de árvore com galhos, definindo métodos que permitam alterar a estrutura, estilo e conteúdo do documento.

Manipulação do DOM: Funcionamento

O navegador lê e renderiza o HTML e, em seguida, cria um **conjunto de comandos** que podem acessar, alterar e criar elementos no “site”.

Todos os comandos devem ser acessados por variáveis de escopo global:
window*, *document*, *location*, *history

Garantindo a renderização

Precisamos garantir que o **script** só seja **executado depois** que a página/**HTML seja renderizado**: só depois disso podemos acessar os elementos do layout.

```
<body>
    <!-- seu site -->
</body>
  <script src="index.js"></script>
</html>

<!-- ou -->

<head>
  <script src="index.js" defer></script>
</head>
```

INTERVALO DE AULA

DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20
Retorno: 20:40

Manipulação do DOM

Acessar: getElement

- document.**getElementById**(“id”)
  - retorna **um** elemento
- document.**getElementsByClassName**(“class”)
  - retorna um array com **todos** os elementos com a classe
- document.**getElementsByTagName**(“p”)
  - retorna um array com **todos** os elementos com a da tag
- document.**querySelector**(“.class”)
  - retorna o **primeiro elemento** pelo seletor

Alterar

- *elementoAcessado*.**innerHTML =**
  - altera o texto
- *elementoAcessado*.**style.***alteração*
  - altera o estilo do elemento

```html
<!-- HTML -->
<h1>Aula 02 - DOM</h1>
<div id="conteudo-principal">
  <h2 class="subtitulo">Porque aprender?</h2>
  <h2 class="subtitulo">Funcionamento</h2>
</div>
```

```javascript
// JavaScript
const subtitulo = document.getElementsByClassName('subtitulo')[0]
subtitulo.innerHTML = 'Porque aprender a utilizar o DOM?!'
```

Manipulação do DOM

Adicionar

- *elementoAcessado*.**innerHTML +=**
- *elementoPai*.**appendChild**(*elementoFilho*)

Criar

const ***nomeDoElementoASerCriado*** = document.**createElement**(*'elemento'*)

Remover

const ***nomeDoElementoASerRemovido*** = document.**removeChild**(*'elemento'*)

```
// ==== Criar elemento ====
const divQueCriamos = document.createElement('div')
divQueCriamos.innerHTML = '<p>Este é um parágrafo</p>'

// ==== Adicionar elemento ====
divPrincipal.appendChild(divQueCriamos)

// ==== Remover elemento ====
divPrincipal.removeChild(divQueCriamos);
```

Eventos

O que é?

Evento é um processo que possibilita **alterar** o valor ou o status de um **elemento HTML**

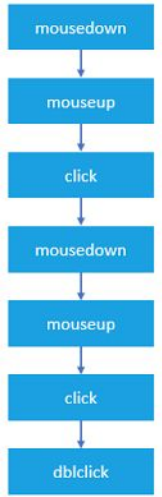
Sintaxe

Para capturar o evento de um elemento HTML, usamos:

```
<elemento evento="nomeDaFuncao()" >
```

Exemplos + comuns

- *onchange*
  - um elemento ou seu valor foram alterados
- *onclick*
  - usuário clicou no elemento
- *onmouseover*
  - usuário posicionou mouse em cima do elemento
- *onmouseout*
  - usuário tirou o mouse de cima do elemento



Manipulação do DOM: addEventListener()

O método addEventListener() permite configurar **funções a serem chamadas** quando um evento especificado acontece.
Por exemplo: quando um usuário *clica em um botão*

```html
<!-- HTML -->
<p>Clique no botão</p>
<button id="meu-botao">Clique Aqui</button>
<div id="conteudo"></div>
```

```javascript
// JavaScript
const botao = document.getElementById("meu-botao");

botao.addEventListener("click", function() {
  document.getElementById("conteudo").innerHTML = "Hello World";
});
```

Desafio 1

Crie um site (*arquivo HTML*) que,
ao ser renderizado, adicione (*arquivo JS*)
5 textos quaisquer, um ao lado outro,
com a cor azul (*arquivo css*).

Desafio 2

Agora substitua as repetições
por uma **função**.

Desafio 3

Crie um site em que toda vez que
o usuário clicar em um botão,
mostre quantos cliques
ele deu no botão.

Desafio 4

Faça um site com uma **lista de itens** (*por exemplo, de brinquedos*), que tenha:

- um campo de input do item
- um botão para criar um item na lista

Material Complementar

- Entendendo o DOM (Document Object Model) - Tableless
- Examples of web and XML development using the DOM - APIs da Web | MDN
- Elementos HTML - HTML: Linguagem de Marcação de Hipertexto | MDN
- Event reference | MDN
- Diferenças entre async e defer - by Eduardo Rabelo

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>