E-BOOK
DO ESTUDANTE

Manipulação de DOM

Manipulando a natureza!

RESILIA

SUMÁRIO

Manipulação de DOM

Manipulando a natureza!

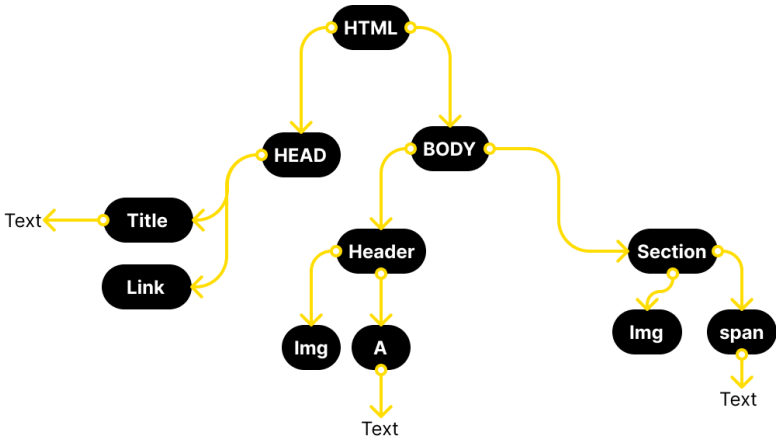
Neste ebook iremos falar sobre como **manipular os elementos existentes em um arquivo HTML usando o Javascript**. Esta manipulação nos permite deixar os elementos aptos para a interação e mudá-los. Sempre quando falamos em manipulação estamos falando de selecionar um elemento do HTML e acessar suas propriedades, seus valores e poder alterá-los. Para manipularmos os elementos iremos usar o DOM (Document Object Model), ele representa todos os elementos existentes em uma página, através dele é possível acessar propriedades e valores.

Para acessarmos o DOM precisaremos usar a propriedade document que sempre irá representar o HTML que está carregando o arquivo Javascript. Dentro da propriedade document, existem alguns métodos que nos permite, selecionar, adicionar, criar elementos HTML, através dele conseguimos adicionar atributos, remover ou alterar valores de um elemento. Pense na propriedade document como um método que nos dá acesso aos elementos existentes no HTML para podermos manipulá-los dentro do Javascript.

Está pronto para se aprofundar neste tópico e começar a deixar suas páginas interativas e funcionais permitindo que estejam mais próximas daquelas que usamos em nosso dia-a-dia?

CONTEXTUALIZANDO

Pense no DOM como uma árvore, logo na raiz temos o HTML, em seguida possuímos duas ramificações: header e o body. Dentro de cada um temos mais ramificações ao longo que nós vamos adicionando elementos dentro do body ou header assim como podemos ver um exemplo na imagem a seguir:



Em nosso dia-a-dia estamos constantemente interagindo com botões, campos de textos e diversos outros elementos em sites na web ou em aplicativos mobile, pense que para cada interação existir é necessário que haja uma árvore que forneça o acesso deste elemento ao Javascript.

Por exemplo: quando clicamos no botão de enviar uma mensagem, o site tem que acessar o valor dentro do campo de mensagem (normalmente se utiliza a tag textarea) e os enviar para o remetente. Quando chega ao remetente, ele recebe o dado e cria uma nova caixa de mensagem com o valor enviado por nós. Todo esse percurso de acessar o campo de texto, logo após o valor e criar uma nova caixa de mensagem somente é possível graças ao DOM. O DOM é usado diariamente por diversas empresas para permitir o acesso a elementos do HTML ao Javascript, e consecutivamente permitir que o site seja funcional. Vemos isso em todos os sites na web, vamos compreender como usá-lo e os seus métodos?

MÉTODOS ÚTEIS

Dentro do DOM temos os tipos de atributos que nos ajudam a selecionar elementos. Iremos iniciar vendo os métodos de **getElement,** ao total temos quatro variantes mais utilizadas:

- **getElementById** (É usado para selecionar um elemento através do atributo id);
- **getElementsByClassName** (É usado para selecionar todos os elementos que contém o mesmo atributo class);
- **getElementsByName** (É usado para selecionar todos os elementos que contém o mesmo atributo name);
- **getElementsByTagName** (É usado para selecionar todos os elementos que sejam da mesma tag).

Esses métodos podem ser acessados pelo método document. Exemplo:

```
document.getElementById("nome")
document.getElementsByClassName("link")
document.getElementsByName("estado")
document.getElementsByTagName("h1")
```

Obs: Lembrando que o método document sempre será o arquivo HTML.

Ainda na seção de selecionar elementos temos um método que é bastante utilizado pela sua flexibilidade. Como vimos no exemplo acima, para cada seletor existe um método getElement, porém também temos o método querySelector que nos permite selecionar os elementos através de todos os seletores, como se fosse uma junção dos métodos do getElement. Sua sintaxe é:

- `querySelector("#nome")` - É usado o símbolo # (hashtag) juntamente com um valor para selecionar um elemento através do atributo id;
- `querySelector(".link")` - É usado o símbolo . (ponto) juntamente com um valor para selecionar o último elemento encontrado que contenha o mesmo valor do atributo class;
- `querySelector("h1")` - É usado para selecionar o último elemento encontrado que seja da mesma tag (neste caso a tag h1).

Note que como o método querySelector nos dá uma maior flexibilidade na hora de escolher o meio de seleção dos elementos. Se quisermos selecionar todos os elementos que contém a mesma tag ou a mesma class utilizamos o método querySelectorAll seguindo a mesma sintaxe do querySelector, só que, retornando um conjunto de elementos.

Caso precise selecionar os elementos head e body, conseguimos através de seus próprios métodos sendo eles:

```
document.body
document.head
```

Isso nos permite selecionar rapidamente caso precisemos deles.

Entrando agora em métodos de manipulação, eles nos permitem manipular, modificar atributos, valores e até mesmo o estilo (CSS) dos nossos elementos. O primeiro grupo de elementos que iremos ver será o grupo de manipulação de texto, por exemplo: quando temos um elemento de tag h1 com um valor “Olá mundo” e que quero o selecionar o seu conteúdo textual podemos utilizar os seguintes métodos:

`textContent`
`innerText`

Estes dois métodos nos permitem selecionar e manipular o conteúdo de texto de um elemento previamente selecionado. Veja o exemplo abaixo:

`*html*`

```
<!-- index.html -->
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <a href="/" class="link">Inicio</a>

    <script src="./main.js"></script>
</body>
</html>
```

`*Javascript*`

```
// main.js
let valorLink = document.querySelector(".link").innerText;
console.log(valorLink);
```

Note que em nosso código HTML, nós temos dentro do body uma tag a com uma class de valor “link” e um valor textual de “Inicio”. Dentro do Javascript, a primeira coisa que fazemos é criar uma variável de nome “valorLink”, logo após selecionamos um elemento através do método querySelector e além de um elemento que contenha a class “link” após selecionarmos o elemento que contém a class “link”. Nós utilizamos o método innerText para pegar o valor textual do elemento selecionado. No caso o valor textual é “Inicio”, neste exemplo nós usamos o método innerText que poderia ser facilmente trocado pelo método “textContent”, pois teríamos o mesmo resultado.

Algo muito útil que o DOM disponibiliza é o método style, através dele conseguimos selecionar e manipular o valor de um atributo de estilo (CSS), ou seja, além de selecionar e manipular elementos, nós conseguimos também manipular os estilos de tais elementos. Isso nos permite criar interações de cores e outras infinitas possibilidades, para utilizarmos o método style, precisamos, anteriormente, selecionar o elemento que queremos aplicar ou mudar um estilo CSS. Logo após declarar qual a propriedade CSS na qual iremos manipular, lembre-se sempre de utilizar a abordagem camelCase.

Por exemplo: a propriedade “background-color” irá ficar backgroundColor, e por fim, após declarar o atributo CSS, você deve atribuir um valor caso for alterar ou aplicar um novo valor, como podemos ver no exemplo abaixo:

```
document.querySelector(".link").style.backgroundColor = "red"
```

Isso nos ajuda a manipular estilos CSS dentro do Javascript e abre uma grande porta para infinidade de funcionalidades que podemos programar. O conteúdo abordado até o momento possui uma complexidade mediana, dessa forma recomendamos que você pratique o que vimos até o momento com as atividades abaixo, lembrando que não é obrigatório, porém é algo que irá te ajudar a reter e entender melhor o funcionamento do DOM no Javascript.

Atividade: Ajudando uma loja

Uma loja precisa de ajuda para notificar seus clientes quando estão abertas ou quando estão fechadas, para isso é preciso que criemos uma página que contenha um elemento de título com o valor “Aberto” um elemento de botão com o valor “Fechar” que ao ser clicado (para isso utilize o atributo onclick) deve mudar valor textual do elemento de título para “Fechado” e um botão de valor “Abrir” que ao ser clicado deve mudar valor textual do elemento de título para “Aberto”.

Atividade: Ajudando uma loja 2.0

A loja agora precisa de ajuda para melhorar o sistema construído anteriormente:
Quando o valor textual do elemento de título estiver “Aberto” o botão de valor textual “Abrir” deve desaparecer da tela, ficando somente o botão de valor textual “Fechar”;
Quando o valor textual do elemento de título estiver “Fechado” o botão de valor textual “Fechar” deve desaparecer da tela ficando somente o botão de valor textual “Abrir”.

.

Recomendamos que treine bastante o conteúdo que vimos até o momento, pois são tópicos que exigem muita repetição e treino. O Javascript é uma linguagem na qual aprendemos mais na prática do que lendo sobre os conceitos. Por isso, não deixe que o medo e a insegurança impeçam a realização desses exercícios para que assim você domine essa linguagem.

Para refletir

- Como selecionamos um elemento através do ID pelo querySelector?
- Qual a diferença entre os métodos getElement e o método querySelector?
- Por que devemos sempre colocar em prática algum conteúdo novo que estudamos em Javascript?
- Você tem medo ou se sente inseguro ao tentar desenvolver em Javascript?

CRIANDO ELEMENTOS

Agora que conhecemos alguns métodos úteis dentro do DOM, precisamos também nos aprofundar e entender como criamos elementos HTML dentro do Javascript e os inserimos na tela. Para realizar essa ação iremos utilizar o método createElement, este método nos permite criar um elemento HTML para podermos manipular e posteriormente inseri-lo na tela. O método pede como argumento a tag do elemento que ele irá criar por exemplo:

```
let titulo = document.createElement("h1")
```

Neste caso estamos criando um elemento <h1></h1> e o armazenando na variável de nome “titulo”. Após criarmos o elemento, ele se torna apto a manipulação, se precisarmos aplicar um valor textual a ele poderíamos usar o método textContent e atribuir o valor textual desejado, por exemplo:

```
let titulo = document.createElement("h1")

titulo.textContent = "Olá mundo"
```

Neste caso estamos pegando a variável título e inserindo um novo valor textual para ela (“Olá mundo”).

Para atribuirmos algum tributo ao elemento criado, utilizamos o método addAtribute que nos permite adicionar um atributo com um valor ao elemento. O método pede como argumento que passemos o nome do atributo e o valor do atributo.

Exemplo:

```
let titulo = document.createElement("h1")

titulo.textContent = "Olá mundo"

titulo.setAttribute("class","tituloPrincipal")
```

Neste caso estamos pegando nossa variável título e adicionando o atributo class com o valor de tituloPrincipal. Agora nosso elemento já está pronto para ser inserido na tela, para isso iremos utilizar a propriedade append, esta propriedade nos permite inserir um elemento como filho de outro elemento.

Exemplo:

```
let titulo = document.createElement("h1")

titulo.textContent = "Olá mundo"

titulo.setAttribute("class","tituloPrincipal")

document.body.append(titulo)
```

Note que estamos selecionando o “body”, ou seja, precisamos antes selecionar o elemento onde iremos inserir nosso título que criamos, após selecionado, utilizamos o método append que nos permite inserir o elemento na página como filho do body, fazendo assim nosso título aparecer na página. Este é um jeito de criar elementos através de código Javascript, existe um jeito mais simples de criar um elemento dentro do Javascript e ao mesmo tempo inseri-lo na tela, este método é realizado usando o innerHTML que nos permite inserir um valor de texto que será interpretado como um código HTML, possibilitando, de forma rápida, a inserção de elementos através do Javascript.

Exemplo:

```
document.body.innerHTML = "<h1 class='tituloPrincipal'>Olá mundo</h1>"
```

Note que o código ficou muito mais enxuto e limpo, porém ele perdeu toda a interatividade que tínhamos antes com os métodos. Quando for escolher um dos dois, elenque sempre os pontos positivos e negativos de cada um e levante também qual a necessidade do projeto.

Agora que vimos as duas possibilidades da criação de um elemento dentro do Javascript, novos horizontes se abrem. O que, talvez, anteriormente era uma voz bem baixa, agora começa a ganhar tom, “Será que eu consigo fazer o Javascript interagir com a minha página? Será que os sites fazem isso?” A resposta é sim! O Javascript foi criado para que possa existir esta interação, e como vimos, nós podemos criar elementos, selecionar alguns já existentes, ou modificá-los/programá-los para em algum momento ele ser alterado. Por exemplo: se precisarmos criar uma interação para inserir uma lista de frutas na página ficaria:

```
let frutas = ["abacaxi", "maçã", "pera", "uva"]
for(let i = 0; i< frutas.length; i++){
    document.body.innerHTML = "<p>" + frutas[i] + "</p>"
}
```

Perceba que temos uma variável de nome “frutas” que esta contendo um array com um total de quatro frutas, logo após temos um laço de repetição do tipo for que está criando uma variável de nome “i”, atribuindo o valor zero a esta variável, a condição do laço de repetição é enquanto o valor de “i” que iniciou em zero for menor que a quantidade total de itens dentro do array frutas, que neste caso é quatro, ou seja, ele dará quatro voltas, sendo elas a volta zero, um, dois e três.

Para cada volta o laço de repetição irá incrementar de um em um o valor de “i”. Dentro do for, nós temos nosso código que será executado e repetido, estaremos selecionando o “body” e inserindo um texto que seria interpretado como código HTML, em nosso caso estamos inserindo a soma dos textos `"<p>" + frutas[i] + "</p>"`.Se repararmos no meio, entre os dois textos, temos nossa variável e estamos selecionando um valor dentro do array através do seletor de index os [] (colchetes), essa parte do código está nos permitindo pegar em cada volta um valor diferente dentro do array frutas.

Exemplo:

volta 0: i = 0 ; " " + frutas[0] + " " = " abacaxi "; i + 1 = 1;

volta 1: i = 1 ; " " + frutas[1] + " " = " maçã "; i + 1 = 2;

volta 2: i = 2 ; " " + frutas[2] + " " = " pera " i + 1 = 3;

volta 3: i = 3 ; " " + frutas[3] + " " = " uva " i + 1 = 4;

volta 3: i = 4; STOP!

Lembre-se sempre que os índices sempre começam em 0!

Note que fizemos um código de quatro linhas que consegue inserir uma lista de frutas que pode sempre aumentar, seja a quantidade de frutas que existir todas elas sempre irão aparecer na tela, isso permite para nós não precisar ficar escrevendo no HTML linhas repetidas com somente o valor mudando e tornou nosso código muito mais reutilizável, já que se precisarmos mudar de tag ou a quantidade de frutas, seriam alterações pequenas que afetariam o todo, sei também que a complexidade deste código aumentou, por isso eu proponho uma atividade para treinar tudo aquilo que vimos até o momento. O que acha? Vamos fixar o conteúdo que vimos com uma atividade que nos ajudará a compreender melhor esta interação e nos fará exercitar um pouquinho de Javascript?

Atividade: Obtendo tronos

Você deve criar um array de top dez coisas que você mais gosta, seja filmes, séries, desenhos, livros, conteúdos, linguagens, qualquer coisa que você goste e que considere que está em seu top dez.

Após a criação do array, deve inserir cada um dos dez elementos na tela, lembre-se de utilizar aquilo que nós aprendemos e utilizamos nos exemplos.

O foco desta atividade é praticar a interação entre o Javascript e elementos HTML, se sentir confortável, fique à vontade para estilizar o projeto e publicar em seu GitHub, pois isso daria um belo de projeto para seu portfólio!

Este e-book teve o objetivo de apresentar o conceito de DOM e aprofundar mais ainda o seu conhecimento dentro da linguagem Javascript. Cada tópico presente foi amplamente pesquisado e desenvolvido para que cada um deles estivesse mais ligado ao que um programador no mercado de trabalho utiliza diariamente. Cada tópico, seção e método aqui descritos são super importantes para o seu desenvolvimento e o da sua carreira, o Javascript é difícil e cansativo, mas respire fundo, se preciso medite e tente outra vez, pois nós aprendemos mais nos erros do que nos acertos, então erre bastante para ter um acerto gigantesco futuramente!

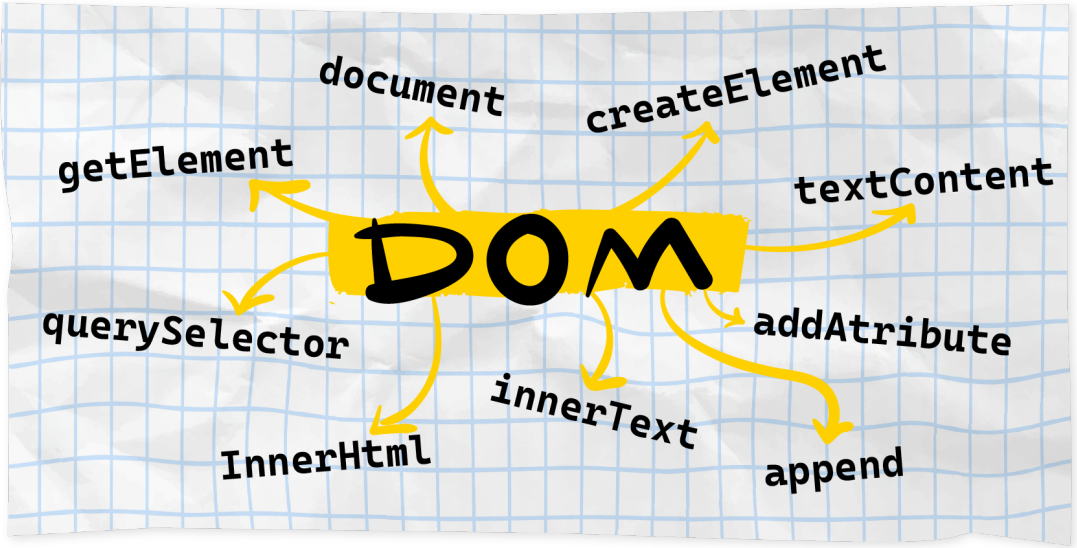
Para refletir

- O DOM nos permite interagir com elementos ou manipular os elementos do HTML?
- Como fazemos para atribuir um atributo a um elemento no Javascript? E como fazemos para remover?
- O que você absorveu neste e-book e levará para sua carreira e vida pessoal?

Para ir além

- Aprofundamento teórico e prático ao DOM:
  >Introdução ao DOM - APIs da Web | MDN (mozilla.org)<
- Post sobre o Dom e seus métodos de forma simples:
  >O que é DOM do JavaScript? (Document Object Model)<

NUVEM DE PALAVRAS



Até a próxima e
#confianoprocesso

RESILIA