



# Manipulando elementos do DOM

## **R**esumo

Nesta aula, você aprenderá maneiras diferentes de “pegar” elementos do seu site para que você possa manipular. Você vai conhecer o `getElementById` e o `getElementsByTagName`. Esses são geralmente os melhores métodos para “pegar” os elementos, mas eles têm limitações. O primeiro, `getElementById`, dá acesso apenas aos elementos que foi atribuído a um id. O segundo, `getElementsByTagName`, é bom para mudanças de vários elementos ao mesmo tempo, mas é um pouco complicado para um bom trabalho mais detalhado.

Felizmente, essas duas opções são apenas dois dos muitos métodos para trabalhar com o Modelo de Objeto de Documento, o DOM. O DOM é um organograma, criado automaticamente pelo navegador quando sua página web é carregada, em um navegador. Todos os elementos do seu site - as tags, os blocos de texto, as imagens, os links, as tabelas, o estilo, atributos e muito mais—estão neste organograma. Isso significa que seu código JavaScript pode pegar qualquer elemento do seu site. Além disso, o JavaScript pode adicionar, mover ou excluir elementos. Se você quisesse, você poderia quase criar um site do zero usando os métodos DOM do JavaScript.

## Conhecendo os elementos do DOM

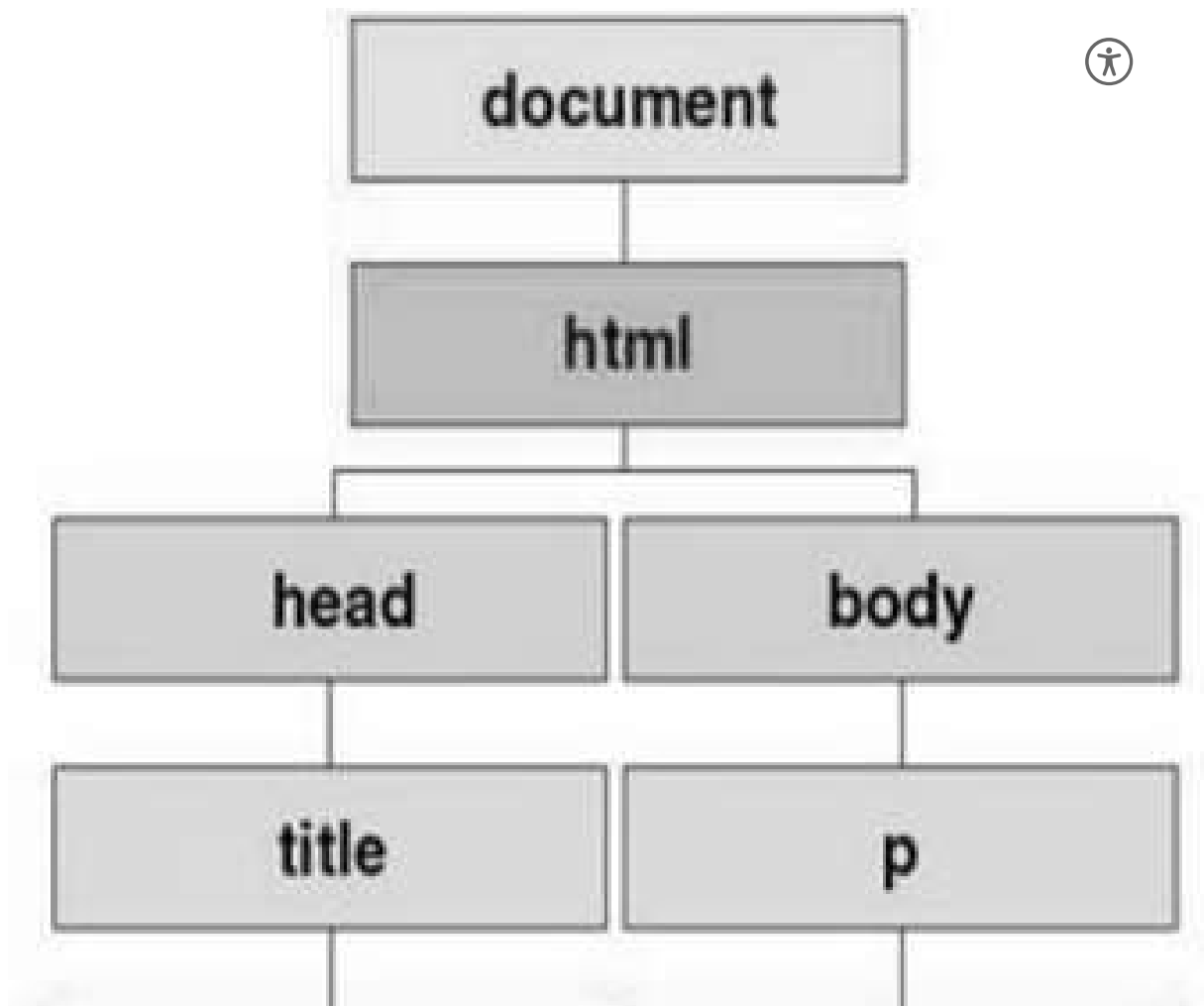


Aqui está um documento html simplificado. Os três níveis superiores do DOM são sempre os mesmos para uma página web padrão. O documento é o primeiro nível. Abaixo do document está o segundo nível, o html. E abaixo do html está o terceiro nível que é composto pela head e body. Abaixo de cada um deles há mais níveis. Veja o código abaixo:

```
<html>
  <head>
    <title>
      Meu título
    </title>
  </head>
  <body>
    <p>
      Texto na página
    </p>
  </body>
</html>
```

Fonte: Autoral

Aqui está a mesma coisa, mostrada como um organograma.



Fonte: Autoral

## Encontrando Elementos

Como você viu, o Document Object Model (DOM) é uma hierarquia de pais e filhos. Tudo no site, excluindo o próprio document, é filho de algo. Então agora vamos falar sobre como seu código JavaScript pode usar essa hierarquia de pais e filhos para ler ou alterar praticamente qualquer coisa em um site. Podemos “pegar” qualquer elemento do nosso site e manipular. Abaixo, um exemplo de como podemos pegar um elemento pelo id.

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



Você também aprendeu como fazer uma coleção de array de todos os elementos de um determinado tipo dentro do documento:

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

Tendo feito um array de parágrafos, você pode separar qualquer parágrafo dentro do array para que você possa, por exemplo, pegar seu conteúdo.

```
var conteudo = elementoP[2].innerHTML;
```

A instrução acima atribui à variável conteúdo o texto do terceiro parágrafo do documento.

Um outro exemplo é se quisermos restringir a pegar o segundo elemento de uma lista; poderíamos fazer assim:

```
var variasLi = document.getElementsByTagName("li")
```


```
var segundoltem = variasLi[0]
```

Dessa forma, conseguimos pegar o segundo elemento da lista para poder manipular.

Depois que temos o elemento, podemos fazer qualquer coisa com ele. Por exemplo, trocar o valor dele ou trocar a cor dele.

## Conteúdo Bônus

Até agora, você aprendeu a usar a hierarquia pai-filho do DOM para direcionar um nó filho de um nó pai, especificando sua ordem em uma coleção semelhante a uma matriz de filhos —

childNodes[0], childNodes[1], childNodes[2] e assim por diante. Mas existem outras maneiras de usar a hierarquia  para segmentação. Para começar, em vez de escrever childNodes[0], você pode escrever firstChild. E em vez de escrever, por exemplo, childNodes[9], para segmentar o 10º filho em uma coleção de 10 filhos, você pode escrever lastChild.

Fazer isso:

```
var targetNode = parentNode.childNodes[0];
```

Fonte: Autoral

É a mesma coisa que fazer assim:

```
var targetNode = parentNode.firstChild;
```

Fonte: Autoral

E se houver, por exemplo, 3 nós filho...

```
var targetNode = parentNode.childNodes[2];
```



É a mesma coisa que fazer isso:

```
var targetNode = parentNode.lastChild;
```

Fonte: Autoral

## Referência Bibliográfica

FLANAGAN, David. **JavaScript: O Guia Definitivo**. 6ª Ed. Porto Alegre: Bookman, 2013.

FREEMAN, Eric. **Use a cabeça!: programação JavaScript**. 1ª Ed. São Paulo: Alta Books, 2016.


## ATIVIDADE PRÁTICA

**Título da Prática:** O que são elementos do DOM?

**Objetivos:** Compreender o uso do Javascript para manipular o DOM

**Materiais, Métodos e Ferramentas:** Para realizar esta prática vamos utilizar o Visual Studio Code

## Prática

Um professor deseja calcular a média das notas de uma turma de estudantes. Escreva um programa em JavaScript que solicite ao professor as  $n$   dos estudantes, uma por uma, e que pare de solicitar notas quando for digitado um valor negativo. Em seguida, o programa deve calcular a média das notas fornecidas e exibir o resultado.

## Resolução

Resposta:

```
let totalNotas = 0;
```

```
let quantidadeEstudantes = 0;
```

```
let nota;
```

```
console.log("Digite as notas dos estudantes (digite um valor negativo para encerrar):");
```

```
while (true) {
```

```
    nota = parseFloat(prompt("Digite a nota do estudante:"));
```

```
    if (nota < 0) {
```

```
        break; // Encerra o loop quando uma nota negativa é digitada
```

```
    }
```

```
    totalNotas += nota;
```

```
quantidadeEstudantes++;
```



```
}
```

```
if (quantidadeEstudantes > 0) {
```

```
    const media = totalNotas / quantidadeEstudantes;
```

```
    console.log("A média das notas é:", media.toFixed(2));
```

```
} else {
```

```
    console.log("Nenhuma nota foi digitada.");
```

```
}
```

**Ir para exercício**