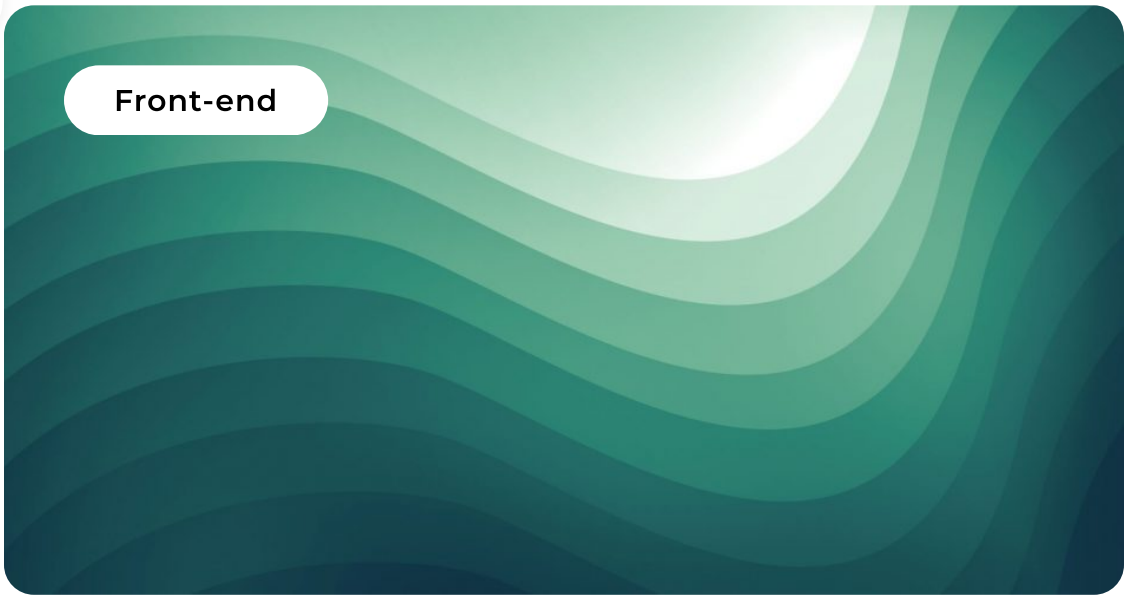

27/07/2021 14875 0

Selecionando elementos HTML no Javascript

Seletores são métodos que selecionam tags HTML (e seu conteúdo) em qualquer código HTML, seja um código fonte de uma página web ou parte dela. Isso é muito útil quando queremos alterar o código HTML, seja adicionando novos elementos a uma determinada tag ou alterando o que já existe

nela.

Acho que a melhor maneira de entender como isso funciona, é criando um código HTML com algumas tags diferentes para usarmos como exemplo.

```
<div id = "colegas-de-classe" class = "conteudo">
  <ul class = "terceiro-ano">
   <li class = "amigo">João</li>
   <li class = "inimigo">Fernanda</li>
   <li class = "amigo">Amanda</li>
  </ul>
</div>

<div id = "frase-motivacional" class = "conteudo">
  <p id = "frase-do-dia">Comece acreditando que é
possível</p>
</div>
```

Observe como as tags estão estruturadas nesse exemplo (suas classes, seus ids, quais tags estão dentro de quais tags, etc.). Isso é muito importante para entendermos esse conceito, inclusive, recomendo que você volte neste código de exemplo durante a leitura.

Vale lembrar que para utilizar os métodos seletores, precisamos especificar o objeto onde estes elementos vão ser utilizados e, em uma página web, por exemplo, esse objeto é o

`document`. Assim, os seletores sempre vão se apresentar na forma `document.metodoSeletor()`.

getElementById()

Este método é o mais simples de todos, e retorna um elemento correspondente ao id passado como parâmetro (em formato string). No entanto, caso mais de um elemento possua o id passado como parâmetro, a função retorna o primeiro elemento encontrado. Veja o código abaixo.

```
let frase = document.getElementById("frase-do-dia");
```

Esse comando cria uma variável chamada `frase` e armazena nessa variável o elemento cujo id é igual ao id passado como parâmetro. Caso este id não exista dentro do código HTML, o retorno da função é `null`.

getElementsByClassName()

Como o próprio nome já diz, essa função vai retornar os elementos que possuem uma mesma classe passada como parâmetro em formato string.

Enquanto a função anterior retorna um único elemento, esta função retorna uma HTMLCollection (uma coleção) de

elementos. No entanto, caso o seletor não exista na página HTML, o retorno é uma HTML Collection vazia. No nosso código, temos alguns itens nas classes “amigo” e “inimigo”, que podemos ser selecionar da seguinte maneira:

```
const amigos =
document.getElementsByClassName("amigo");
const inimigos =
document.getElementsByClassName("inimigo");
```

Esse seletor sempre vai retornar uma coleção com todos os elementos da mesma classe. Se eu tiver apenas um elemento com esta classe, o retorno ainda será uma coleção, mas com um único elemento. Assim, se quiséssemos imprimir as variáveis `amigos` e `inimigos` definidas acima com o comando `console.log()`, teríamos o seguinte resultado:

```
HTMLCollection { 0: li.amigo, 1:li.amigo,
length:2}//constante amigos
HTMLCollection { 0: li.inimigo, length:1} //constante
inimigos
```

Podemos ver na impressão da constante amigos, que temos na posição de índice 0 desta coleção um elemento `li.amigo`, e na posição de índice 1 outro elemento `li.amigo`. Assim, essa coleção possui length (comprimento) de 2. Além disso, na

impressão da constante inimigos, temos um elemento `li.inimigo` na posição de índice 0 da coleção, que possui length de 1.

Para acessarmos os elementos das coleções, podemos utilizar a notação de colchetes utilizada em vetores, ou seja, `inimigos[0]` ou `amigos[1]`.

getElementsByTagName()

Esta função é semelhante à anterior, porém cria uma coleção com todas as tags com o mesmo nome. Assim, caso não exista nenhuma tag com o nome especificado, o retorno é uma HTML Collection vazia.

```
let nomeUsuario =
document.getElementsByTagName("nome");
```

O retorno do método `getElementsByTagName()` é uma HTML Collection, assim como o retorno de `getElementsByClassname()`. Veja o resultado de `console.log(conteudos)`:

```
HTMLCollection { 0: div#colegas-de-classe.conteudo,
1: div#frase-motivacional.conteudo, 2: div#nome-
formulario.conteudo, length: 3, … }
```

Todos os objetos com a classe `div` foram selecionados, tendo seus ids e suas classes especificadas, mesmo que estas tenham sido irrelevantes na seleção.

Assim como no método `getElementsByClassName()`, os objetos dessa coleção devem ser acessados com a notação de colchetes.

getElementsByName()

Esse método retorna uma NodeList contendo os elementos de código selecionados pelo atributo `name` das tags HTML.

```
let opiniaoUsuario =
document.getElementsByName("opiniao");
```

Como temos apenas um elemento com o atributo `name` selecionado, quando tentarmos imprimir a variável `opiniaoUsuario`, teremos uma coleção com um único objeto. O acesso a este objeto, assim como nos exemplos anteriores, deve ser com a notação de colchetes. Assim, caso não seja possível encontrar nenhum elemento com aquele atributo name, o retorno é uma NodeList vazia.

querySelector()

Diferente dos outros métodos, o `querySelector()` não recebe um atributo de uma tag HTML ou uma tag como parâmetro, mas recebe o mesmo seletor que seria utilizado em CSS, no formato string. É importante lembrar que assim como no CSS, a utilização de "." para indicar classes bem como a utilização de "#" para indicar ids é obrigatória.

```
let terceiroAno = document.querySelector(".terceiro-ano");
let paragrafo = document.querySelector("p");
let fraseDia = document.querySelector("div#frase-motivacional p#frase-do-dia");
let conteudos = document.querySelector(".conteudo");
```

Podemos utilizar o `querySelector()` para selecionar tanto por id quanto por classe, assim como uma combinação destes elementos.

Um ponto interessante do `querySelector()` é que essa função só retorna o primeiro elemento encontrado que tem a mesma configuração do seletor passado como parâmetro. Assim, a variável `conteudos` só vai receber o primeiro elemento com a classe `conteudo`, no formato de um único elemento, não de uma coleção. Se quisermos obter todos os elementos que combinam com um mesmo seletor, devemos utilizar o próximo método da lista. Este método retorna `null` caso nenhum

elemento seja encontrado.

querySelectorAll()

O `querySelectorAll()` é similar ao método `querySelector()` e também utiliza os seletores do CSS, porém retorna uma NodeList com todos os elementos que correspondem ao seletor criado. Assim como nas coleções, podemos acessar os elementos através da notação de colchetes.

```
let paragrafo = document.querySelectorAll("p");
let conteudos =
document.querySelectorAll(".conteudo");
```

Caso um exista um único elemento com o seletor passado como parâmetro, o resultado será uma NodeList com apenas um elemento, e se não existir nenhum elemento com a especificação dada, o retorno é uma NodeList vazia.

Compartilhar artigo:

https://blog.cod3r.com.br/selecionando-elem