08

Boas Práticas

Transcrição

Quando programamos com o JavaScript, é provável que estejamos trabalhando com outras pessoas, então, o que acontece se outro desenvolvedor decidir deixar de utilizar a tag `<h1>`, substituindo-a por `<h2>`?

```
<header>
    <div class="container">
        <h2>Aparecida Nutrição</h2>
    </div>
</header>
```

Ao abrirmos a página no navegador, será exibida a seguinte mensagem no console:

```
Uncaught TypeError: Cannot set property 'textContent' of null
```

A mensagem indica que a propriedade `textContent` é nula e apontará a linha na qual ocorreu o erro, no caso, na tag `<script>`.

```
<!-- ... -->
        </section>
    </main>
<script>
    var titulo = document.querySelector("h1");

    titulo.textContent = "Aparecida Nutricionista";
</script>
```

O `h1` não foi encontrado porque ele deixou de existir na nossa página e, sendo assim, impossibilitado de ser selecionado. O problema de se buscar uma tag no DOM é que o código JavaScript fica muito atrelado ao HTML. Caso o desenvolvedor decida fazer alterações no HTML, o código deixará de funcionar. Por isso, é uma boa prática não buscarmos por uma tag HTML específica, e usar outras opções que a função `querySelector()` nos disponibiliza.

Além das tags HTML, o `querySelector()` nos permite buscar por um elemento por meio da **classe**, do **id**, ou seja, dos **seletores CSS**. Vamos testar esse recurso adicionando uma classe dentro do `<h1>`:

```
<header>
    <div class="container">
        <h2 class="titulo">Aparecida Nutrição</h2>
    </div>
</header>
```

No código JS inserido na parte de baixo do arquivo, em vez de realizarmos a busca no `document.querySelector()` pela tag `<h1>`, usaremos o seletor CSS que retornaria o conteúdo na mesma tag. Iremos utilizar o seletor CSS para classe: `.` (ponto).

```
<script>
    var titulo = document.querySelector(".titulo");
    titulo.textContent = "Aparecida Nutricionista";
</script>
```

Agora, serão retornados os elementos da classe `.titulo`, e **Aparecida Nutricionista** reaparece na página. Se outro desenvolvedor modificar novamente a tag `h1` no código HTML, nosso código não será prejudicado e o JavaScript continuará sendo executado, pois deixamos as responsabilidades desacopladas.

Trata-se de uma boa prática separarmos o código JavaScript do HTML, no entanto, o código ainda está completo no `index.html`, atuando independentemente. O trecho JS está salvo na tag `<script>`. É interessante desacoplarmos os códigos em arquivos diferentes, com diferentes extensões, `.js`, `.html` - já temos o código CSS separado em um arquivo `.css`.

A seguir, criaremos o arquivo `principal.js` na pasta **/js**, com todo o código JavaScript. O próximo passo será mover o código da tag `<script>` para dentro desse arquivo:

```
var titulo = document.querySelector(".titulo");
titulo.textContent = "Aparecida Nutricionista";
```

A tag `<script>` continuará na página `index.html`, porém, ela não ficará vazia, e apontará para o arquivo JavaScript externo que criamos, por meio do atributo `src` (referente ao termo *source*). Como o arquivo `principal.js` está na pasta `js`, especificaremos o caminho completo no atributo:

```
<!-- ... -->
        </section>
    </main>
<script src="js/principal.js"></script>

</body>
```

Quando recarregarmos a página, o arquivo continuará funcionando corretamente. Vale reforçar que é recomendado deixarmos o código HTML separado do JavaScript.

Nesta aula, nós abordamos vários temas sobre a linguagem JavaScript, vimos que ela ganhou relevância por estar sendo usada de diferentes formas, no navegador, banco de dados, placas Arduíno. É importante que desenvolvedores tenham, pelo menos, um conhecimento básico de como funciona a linguagem.

Vimos algumas funções como `console()` e `alert()`, que nos permitem exibir mensagens ao usuário, e conhecemos a utilidade do console de desenvolvedor, presente nos navegadores. Podemos executar códigos JavaScript no console, com isso, conseguimos testar algumas coisas diretamente no navegador.

Vimos também como realizar a busca por algum elemento do HTML, no código JavaScript. Para isto, utilizamos a variável `document`, que contém todo o conteúdo HTML da página. Quando manipulamos o `document`, conseguimos manipular o que será exibido ao usuário. Essa manipulação foi realizada por meio da seleção de trechos da tela, feita com o `querySelector()`. Vimos que este método busca por nome de tags - o que pode trazer problemas -, assim como seletores de CSS.

Na aula, mostramos como criar uma variável no JavaScript e alterar um conteúdo de texto utilizando a propriedade `textContent` .

Aguardo você nas aulas seguintes, espero que tenha gostado da introdução do curso.