04

Filtrando com expressão regular

Transcrição

Estamos com uma busca funcional, conseguimos filtrar a lista pelo nome do paciente, e quando apagamos o termo pesquisado, a lista da tabela é exibida completamente. Para tornar a nossa filtragem melhor, seria interessante que, ao digitarmos uma letra no campo de busca, todos os nomes com essa letra fossem listados. Por exemplo, ao digitarmos a letra "P", todos os pacientes que começassem com essa letra seriam exibidos. Como "Pedro", "Paulo" e "Pablo".

Conforme formos digitando as letras, a filtragem seria atualizada e a busca ficaria mais interativa.

Para que isso aconteça, é preciso compararmos letra por letra pelos termos buscados com os nomes cadastrados na lista. Seria trabalhoso fazer isso manualmente, mas o JavaScript, além de outras linguagens de programação, já possuem uma solução para a busca de texto: **expressões regulares**.

As **expressões regulares** são um tipo especial de texto, que nos auxilia a buscarmos por strings, facilitando quando o termo for maior. Pode ser uma busca simples, como no nosso caso, em que queremos identificar quais nomes contêm determinadas letras; ou casos complexos, se queremos pesquisar se o parágrafo contém a palavra "nome", por exemplo, é como quando os editores de texto buscam por uma palavra usando o comando "CTRL + F".

Criando expressões regulares no JavaScript

É bastante simples criar expressões regulares. Criaremos uma variável, que no caso chamaremos `expressao`, e em seguida colocaremos uma expressão regular dentro dela. Vamos gerar um objeto especial do JS, adicionando `new` e o nome `RegExp()`:

```
var expressao = new RegExp();
```

Com a nova linha o trecho ficará da seguinte maneira:

```
var campoFiltro = document.querySelector("#filtrar-tabela");

campoFiltro.addEventListener("input", function(){
    console.log(this.value);
    var pacientes = document.querySelectorAll(".paciente");
```

```
    if (this.value.length > 0){
        for (var i = 0; i < pacientes.length; i++){
            var paciente = pacientes[i];
            var tdNome = paciente.querySelector(".info-nome");
            var nome = tdNome.textContent;
            var expressao = new RegExp();
            if (nome != this.value){
                paciente.classList.add("invisivel");
            } else {
                paciente.classList.remove("invisivel");
            }
        }
```

Nós poderemos passar dois parâmetros para o objeto, sendo o primeiro o texto que queremos buscar, no caso, o termo digitado no campo de busca (`this.value`), e o segundo parâmetro será referente às características dos termos que devem ser buscados. É importante que a busca não seja *case sensitive*, que é a diferenciação entre letras maiúsculas e minúsculas. Devem ser buscadas tanto letras maiúsculas como minúsculas, e passaremos a letra "i" como segundo parâmetro, para indicarmos a opção **case insensitive**:

```
var campoFiltro = document.querySelector("#filtrar-tabela");

campoFiltro.addEventListener("input", function(){
    console.log(this.value);
    var pacientes = document.querySelectorAll(".paciente");

    if (this.value.length > 0){
        for (var i = 0; i < pacientes.length; i++){
            var paciente = pacientes[i];
            var tdNome = paciente.querySelector(".info-nome");
            var nome = tdNome.textContent;
            var expressao = new RegExp(this.value, "i");
            if (nome != this.value){
                paciente.classList.add("invisivel");
            } else {
                paciente.classList.remove("invisivel");
            }
        }
```

Porém, como utilizamos a expressão regular para buscar um texto específico na tabela? Em vez de compararmos com o nome inteiro do paciente (como estávamos fazendo), vamos pedir para a expressão regular verificar se um pedaço do nome do paciente possui as letras digitadas no campo de busca. Para isso, a expressão regular tem o método `test()` , com a qual passaremos o que queremos testar:

```
var campoFiltro = document.querySelector("#filtrar-tabela");

campoFiltro.addEventListener("input", function(){
    console.log(this.value);
    var pacientes = document.querySelectorAll(".paciente");

    if (this.value.length > 0) {
        for (var i = 0; i < pacientes.length; i++){
            var paciente = pacientes[i];
            var tdNome = paciente.querySelector(".info-nome");
```

```
            var nome = tdNome.textContent;
            var expressao = new RegExp(this.value, "i");
            if (expressao.test(nome)) {
                paciente.classList.add("invisivel");
            } else {
                paciente.classList.remove("invisivel");
            }
        }
    } else {
        for (var i = 0; i < pacientes.length; i++) {
            var paciente = pacientes[i];
            paciente.classList.remove("invisivel");
        }
    }
});
```

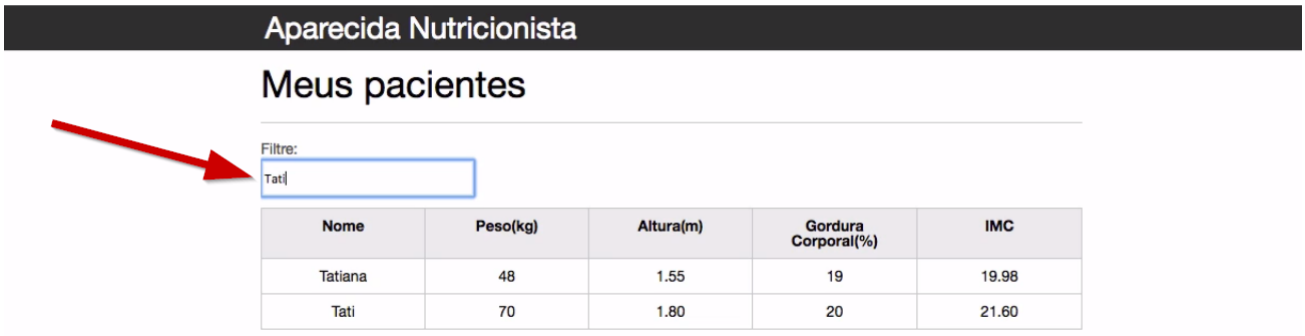
Esse teste irá retornar **verdadeiro** caso o nome contenha a expressão, ou **falso** caso não contenha. Como estamos testando se o nome não contém a expressão (por isso adicionaremos a classe `invisivel`), utilizaremos novamente o **operador de negação** (`!`). Logo, se o teste falhar, adicionaremos a classe; se não, ela será removida:

```
var campoFiltro = document.querySelector("#filtrar-tabela");

campoFiltro.addEventListener("input", function(){
    console.log(this.value);
    var pacientes = document.querySelectorAll(".paciente");

    if (this.value.length > 0) {
        for (var i = 0; i < pacientes.length; i++) {
            var paciente = pacientes[i];
            var tdNome = paciente.querySelector(".info-nome");
            var nome = tdNome.textContent;
            var expressao = new RegExp(this.value, "i");

            if (!expressao.test(nome)) {
                paciente.classList.add("invisivel");
            } else {
                paciente.classList.remove("invisivel");
            }
        }
    } else {
        for (var i = 0; i < pacientes.length; i++) {
            var paciente = pacientes[i];
            paciente.classList.remove("invisivel");
        }
    }
});
```

Agora a filtragem será feita a cada letra, deixando a busca mais dinâmica.

Vamos testar cadastrando mais um paciente com nomes que comecem com a mesma letra. Após adicionarmos os dados do "Pedro" e da "Tati", testaremos digitar a letra "P".

Em seguida, se apagarmos os termos do campo de busca, a lista de pacientes será exibida completamente na tabela. Nossa busca também será bem sucedida se buscarmos pelo termo "Tati".



Vimos usando um pouco da lógica da expressão regular e melhoramos nossa filtragem, que ficou mais elaborada. Se você se interessou pelas expressões regulares, temos um curso **Expressões regulares: Capturando textos de forma mágica** (https://cursos.alura.com.br/course/expressoes-regulares) na Alura, que se aprofunda no assunto.

Nesta aula, implementamos a busca! No código do `filtra.js`, nós utilizamos o evento de `input`, e com o conteúdo abordado no curso, conseguimos uma lógica interessante. Mesmo sem a expressão regular, já havíamos implementado uma busca bastante eficiente. Também vimos o uso do `this`, do `value`, e que o `length` nos garante o poder de realizar uma busca na tabela, percorrendo-a.