

Expressões regulares em JavaScript

As expressões regulares são utilizadas para localizar, validar ou substituir caracteres que poderão compor uma String ou um conjunto de dados. Vale ressaltar que o tema expressões regulares é amplo, portanto estudos adicionais são importantes.

Para o processo de validação de um endereço eletrônico, especificamente e-mail é importante verificar se o caracter especial @ está presente, desta maneira a utilização do recurso mais simples das expressões regulares permite essa checagem, conforme exemplifica-se com parte de processo:

Código do documento de marcação chamado regexp.html:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta name="viewport" content="width=device-width, initial-scale=1">
5     <link rel="stylesheet" type="text/css" href="aula.css">
6   </head>
7   <body>
8     <form action="/action_page.php">
9       <div class="container">
10         <h1>Validar Formulário</h1>
11         <p>Preencha todos os campos para criar sua conta</p>
12         <hr>
13         <label for="email"><b>E-mail</b></label>
14         <input type="text" placeholder="Insira o e-mail"
onblur="verificaMail()" name="email" id="email" required>
15         <hr>
16         <button type="submit" id="btn" class="btn">Confirmar
cadastro</button>
17       </div>
18     </form>
19     <script src="aularegexp.js" type="text/javascript"></script>
20 </body>
21 </html>
```

A vinculação com o código JavaScript é realizada na linha 19 do código acima, permitindo o consumo do arquivo aularegexp.js que detém as seguintes instruções:

```
1 function verificaMail() {
2   let expressao = /@/;
3   let mail = document.querySelector('input[name="email"]').value;
4   if(expressao.test(mail)){
5     alert('Tem @');
6   }else{
```

```
7     alert('Não tem @');
8   }
9 }
```

Na linha 1 do código acima é criada a função `verificaMail`.

Na linha 2 a variável `expressao` é criada e inicializada com a expressão regular `/@/`, valendo observar que entre os símbolos `/ /` deverá estar a palavra ou expressão que deseja-se operar.

Na linha 3 a variável `mail` é criada e inicializada com o conteúdo do input cujo name é igual a e-mail, ou seja o input marcado na linha 14 do documento `regexp.html`.

Na linha 4 inicia-se a estrutura condicional `if` ou `se` para testar ou “procurar” a expressão `@` dentro da variável `mail`, retornando `true` no caso de encontrar ou `false` no caso de não encontrar.

A linha 5 exibe um alerta ou mensagem na tela para os casos que o resultado da linha 4 for `true`.

A linha 6 opera o recurso `else` ou `senão` da estrutura condicional iniciada na linha 5 para exibir a mensagem da linha 7 nos casos em que o resultado da linha 5 for `false`.

A linha 8 detém a chave que encerra a estrutura condicional e a linha 9 detém a chave que encerra a função.

Vale salientar que o input da linha 14 do documento `regexp.html` contém a propriedade `onblur` recebendo a função `verificaMail` `-onblur="verificaMail()"` – o que permite o acionamento da inteligência da função, quando o cursor ou o foco sai do input.

A estilização CSS vinculada pelo código da linha 5 do documento `regexp.html` é

a seguinte:

```
body {
    font-family: Arial, Helvetica, sans-serif;
    background-color: black;
}

* {
    box-sizing: border-box;
}

.container {
    padding: 16px;
    background-color: white;
}

input[type=text], input[type=password] {
    width: 100%;
    padding: 15px;
    margin: 5px 0 22px 0;
    display: inline-block;
    border: none;
    background: #f1f1f1;
}

input[type=text]:focus, input[type=password]:focus {
    background-color: #ddd;
    outline: none;
}

hr {
    border: 1px solid #f1f1f1;
    margin-bottom: 25px;
}

.btn {
    background-color: #04AA6D;
    color: white;
    padding: 16px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 100%;
    opacity: 0.9;
}

.btn:hover {
    opacity: 1;
}
```