Expressões regulares em JavaScript

A localização de dados e padrões são importantes nas expressões regulares, permitindo verificar se os dados inseridos observam ou respeitam a regra de negócio, conforme pormenoriza-se:

O padrão RegEx determina que os colchetes sejam elementos de agrupamento do caractere ou caracteres que serão procurados, logo [0] significa que deseja-se buscar ou localizar todos os zeros num conjunto de dados, já [04] significa que deseja-se pesquisar ou localizar todos os zeros ou quatros num conjunto de dados.

Observando a aludida premissa infere-se que [0-9] pesquisará ou localizará todos os números dentro de um conjunto de dados.

Vale ressaltar que as possibilidades mencionadas acima buscam os caracteres sem uma ordem definida.

Na hipótese de ser importante a busca com uma ordem definida o padrão [1][1] poderá ser utilizado, ou seja, colchetes com o caractere desejado, seguido de colchetes com o caractere desejado, resultando na busca pela expressão 11.

Há também como utilizar os quantificadores, i.e., [1]{2} o número dentro das chaves determina a sequência de caracteres 1 que será buscada.

Os quantificadores são importantes para caso em que há necessidade de pesquisar uma sequência numérica, ou seja, [0-9]{9} significa que serão buscadas ou procuradas expressões com nove dígitos numéricos.

Ainda vale destacar que há como buscar os elementos no final de uma expressão ou linha de dados, utilizando o símbolo $, algo como [0-9]{9}$ determinará a busca ou localização de nove dígitos ou caracteres numéricos no final de uma linha de dados ou expressão. No mesmo tema o símbolo ^ significa a busca no início da linha de dados ou expressão, como ^[0-9]{9}.

Para buscar letras basta colocar a letra desejada dentro dos colchetes, como [b] e no caso de desejar buscar qualquer letra utilizar [a-z].

Para ampliar o entendimento apresenta-se a expressão de validação do padrão de dígitos do CPF, observando que outros recursos deverão ser utilizados para validação mais ampla das regras inerentes ao CPF:

```
[0-9]{3}[.][0-9]{3}[.][0-9]{3}[-][0-9]{2}
```

Para aplicar a aludida expressão em JavaScript apresenta-se a seguinte marcação 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             <hr>
12             <label for="email"><b>CPF</b></label>
13             <input type="text" placeholder="Insira o CPF"
onblur="verificaCpf()" name="cpf" id="cpf" required>
14             <hr>
15             <button type="submit" id="btn" class="btn">Confirmar
cadastro</button>
16           </div>
17         </form>
18         <script src="aularegexp.js" type="text/javascript"></script>
19  </body>
20 </html>
```

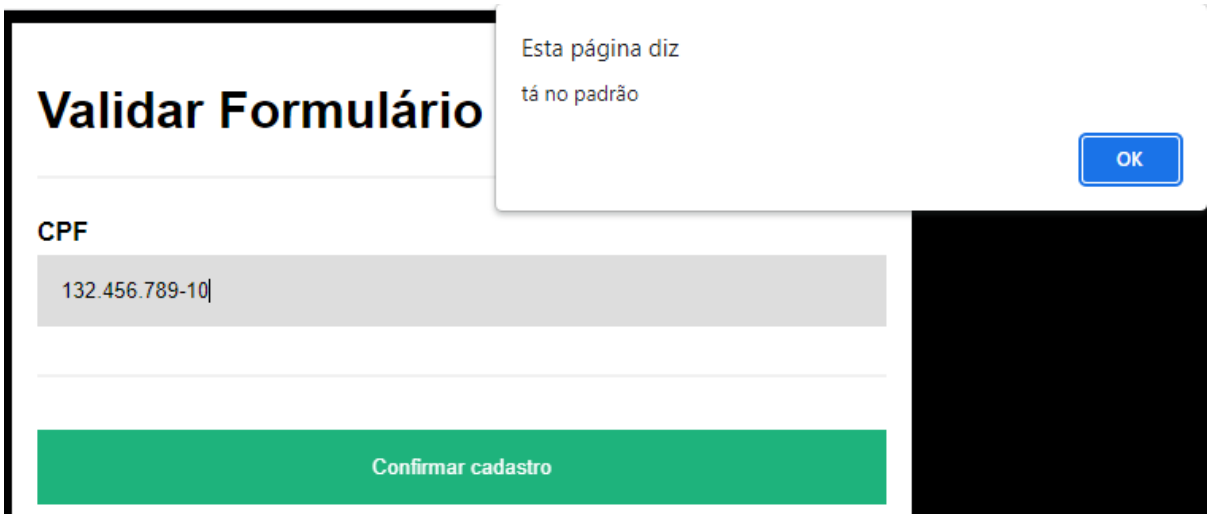
Observa que a linha 18 do código acima vincula o arquivo aularegexp.js, permitindo o acionamento da função verificaCpf() no evento onblur da linha 13. Importante apresentar o código do arquivo aularegexp.js:

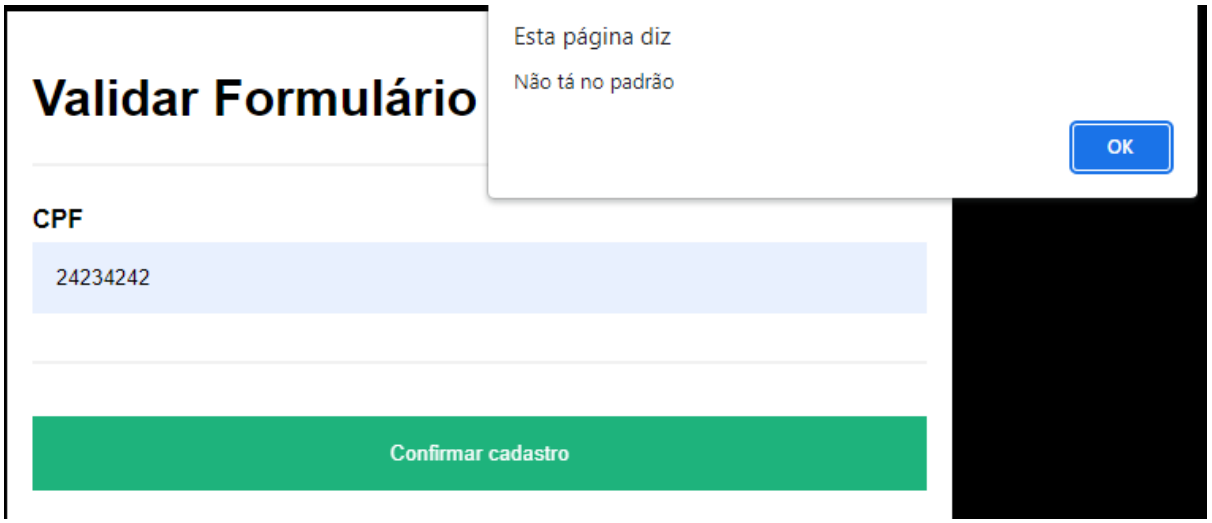
```
1  function verificaCpf() {
2    let expressao = /[0-9]{3}[.][0-9]{3}[.][0-9]{3}[-][0-9]{2}/;
3    let cpf = document.querySelector('input[name="cpf"]').value;
4    if(expressao.test(cpf)){
5      alert('tá no padrão');
6    }else{
7      alert('Não tá no padrão');
8    }
9  }
```

Observa-se que na linha 2 do código acima a variável expressao recebe a expressão regular contendo o padrão de dígitos do CPF e na linha 3 a variável cpf recebe o conteúdo do input com name igual a cpf.

Na linha 4 é iniciada uma estrutura condicional para verificar se os valores ou caracteres inseridos no input de name cpf estão de acordo com o padrão de digitos do CPF ou

expressão regular definida na linha 2. Salienta-se a o teste realizado no if/se da linha 4 retorna true, se os dados ou caracteres digitados estarem de acordo com o padrão definido na expressão regular definida na linha 2, ou false senão(else) estiver de acordo, conforme verifica-se nas imagens:





Ainda foi utilizado para a construção deste exemplo o código css acionado na linha 5 do código de marcação/HTML, conforme exibe-se:

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

  * {
    box-sizing: border-box;
  }
```

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

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;
}
```

Por ora vale indicar o site https://regex101.com/ que permite o aperfeiçoamento do entendimento sobre as expressões regulares.