

Aula 06 - Expressões regulares

Aula: Validações de Dados com Expressões Regulares

1 O que são Expressões Regulares?

Expressões regulares (**regex**) são padrões usados para encontrar, validar e manipular textos de forma eficiente. Elas são amplamente usadas para validar formulários, filtrar informações e até mesmo substituir trechos de texto.

◆ Exemplos de uso de regex:

- ✓ Verificar se um e-mail está no formato correto.
- ✓ Validar um CPF ou CNPJ.
- ✓ Garantir que uma senha tenha letras, números e caracteres especiais.
- ✓ Extrair números de um texto.

2 Sintaxe das Expressões Regulares

1. Delimitadores

Em JavaScript, expressões regulares são escritas entre barras `/.../` ou usando o construtor `new RegExp()`.

```
let regex1 = /teste/; // Forma literal
let regex2 = new RegExp("teste"); // Usando o construtor
```

2. Caracteres Especiais

Os caracteres especiais modificam o comportamento das expressões regulares.

Símbolo	Significado	Exemplo
---------	-------------	---------

<code>^</code>	Início da string	<code>/^a/</code> → Testa se começa com "a"
<code>\$</code>	Fim da string	<code>/a\$/</code> → Testa se termina com "a"
<code>.</code>	Qualquer caractere	<code>/a.b/</code> → "a+b", "acb", "a#b"
<code>\d</code>	Qualquer dígito (0-9)	<code>/\d+/</code> → "123"
<code>\w</code>	Qualquer letra, número ou –	<code>/\w+/</code> → "abc_123"
<code>\s</code>	Qualquer espaço em branco	<code>/\s+/</code> → " "
<code>\b</code>	Delimita palavras	<code>/\bcat\b/</code> só encontra "cat", mas não "catalog"
<code>+</code>	1 ou mais ocorrências	<code>/\d+/</code> → "123"
<code>*</code>	0 ou mais ocorrências	<code>/\d*/</code> → "123" ou ""
<code>?</code>	0 ou 1 ocorrência	<code>/a?b/</code> → "b" ou "ab"
<code>{n}</code>	Exatamente n ocorrências	<code>/\d{3}/</code> → "123"
<code>{n,}</code>	Pelo menos n ocorrências	<code>/\d{3,}/</code> → "1234"
<code>{n,m}</code>	Entre n e m ocorrências	<code>/\d{2,4}/</code> → "12", "1234"

3 Aplicação: Expressões Regulares em JavaScript

Em JavaScript, usamos o método `.test()` para verificar se uma string corresponde a uma regex.

Exemplo 1: Validar um e-mail

```
let regexEmail = /^[a-z0-9._-]+@[a-z]+\.[a-z]{2,4}$/i;

console.log(regexEmail.test("teste@email.com")); // true
console.log(regexEmail.test("emailerrado@com")); // false
```

Exemplo 2: Validar um número de telefone

```
let regexTelefone = /^(\d{2}) \d{4,5}-\d{4}$/;

console.log(regexTelefone.test("(11) 98765-4321")); // true
console.log(regexTelefone.test("11987654321")); // false
```

Exemplo 3: Validar um CPF

```
let regexCPF = /^\d{3}\.\d{3}\.\d{3}-\d{2}$/;

console.log(regexCPF.test("123.456.789-09")); // true
console.log(regexCPF.test("12345678909")); // false
```

4 Criando Validações em Formulários

Vamos aplicar essas expressões regulares para validar um formulário em tempo real.

```
<form id="formulario">
  <label for="email">E-mail:</label>
  <input type="text" id="email">
  <span id="erroEmail" style="color: red;"></span>

  <label for="cpf">CPF:</label>
  <input type="text" id="cpf">
  <span id="erroCPF" style="color: red;"></span>

  <button type="submit">Enviar</button>
</form>

<script>
  document.getElementById("formulario").addEventListener("submit", function(event) {
    event.preventDefault(); // Evita o envio do formulário
```

```

let email = document.getElementById("email").value;
let cpf = document.getElementById("cpf").value;

let regexEmail = /^[a-z0-9._-]+@[a-z]+\.[a-z]{2,4}$/i;
let regexCPF = /^\d{3}\.\d{3}\.\d{3}-\d{2}$/;

let erroEmail = document.getElementById("erroEmail");
let erroCPF = document.getElementById("erroCPF");

if (!regexEmail.test(email)) {
    erroEmail.innerText = "E-mail inválido!";
} else {
    erroEmail.innerText = "";
}

if (!regexCPF.test(cpf)) {
    erroCPF.innerText = "CPF inválido!";
} else {
    erroCPF.innerText = "";
}
});
</script>

```


i Esse formulário valida o e-mail e CPF antes do envio. Se houver erro, a mensagem será exibida abaixo do campo correspondente.

5 Exercícios para os Alunos

1 Crie uma expressão regular para validar uma senha com:

- Pelo menos 8 caracteres
- Pelo menos uma letra maiúscula
- Pelo menos um número


- Pelo menos um caractere especial

 **Dica:** Use `/^(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/` .

2 Crie um formulário com um campo de telefone e valide se ele está no formato `(XX) XXXXX-XXXX` .

3 Crie uma regex para validar URLs, aceitando formatos como:

- `https://www.site.com`
- `http://site.com.br`
- `www.site.com`

 **Dica:** Use `/^(https?:\/\/)?(www\.)?[a-z0-9-]+\.[a-z]{2,}+$/` .