EXPRESSÕES REGULARES

(AULA 17)

CURSO BÁSICO DE PROGRAMAÇÃO COM JAVASCRIPT

MAYARA MARQUES

mmrosatab@gmail.com

JS

SUMÁRIO

- O que são expressões regulares?
- Exemplos de expressões regulares
- Sintaxe das expressões regulares
- Expressões regulares no JavaScript
- Exemplos
- Mão na massa

O QUE SÃO EXPRESSÕES REGULARES?

As expressões regulares (ou regex) são **padrões de texto usados para encontrar ou manipular cadeias de caracteres (strings)** de acordo com regras específicas.

O QUE SÃO EXPRESSÕES REGULARES?

Expressões regulares podem ser vistas como uma forma de **buscar padrões em texto** e até mesmo **validar se uma string segue um formato específico**.

EXEMPLOS DE EXPRESSÕES REGULARES

Encontrar um número de telefone celular:

- \d{2}-\d{5}-\d{4}

Essa expressão busca por um número de telefone no formato 21-12345-6789. Ela está buscando 2 dígitos seguidos de um hífen, mais 5 dígitos, outro hífen e mais 4 dígitos.

EXEMPLOS DE EXPRESSÕES REGULARES

Encontrar uma data no formato DD/MM/AAAA:

- \d{2}/\d{2}/\d{4}

A expressão busca uma data com 2 dígitos para o dia, seguidos de uma barra, 2 dígitos para o mês, outra barra, e 4 dígitos para o ano.

SINTAXE DAS EXPRESSÕES REGULARES

- ****:Utilizado para escapar caracteres especiais.
- **\d**: Corresponde a qualquer dígito numérico (0-9).
- **\D**: Corresponde a qualquer caractere que não seja um dígito.
- **\w**: Corresponde a qualquer caractere alfanumérico (a-z, A-Z, 0-9) e o caractere _.
- **\W**: Corresponde a qualquer caractere que não seja alfanumérico.
- **\s**: Corresponde a qualquer caractere de espaço em branco (espaço, tabulação, nova linha, etc.).
- **\S**: Corresponde a qualquer caractere que não seja espaço em branco.
- **\b**: Marca o limite de uma palavra (início ou fim).
- **\t**: Corresponde a uma tabulação (tab).
- **\n**: Corresponde a uma nova linha.
- **\1**: Subcorrespondência de um grupo de captura anterior que corresponde ao mesmo texto que esse grupo.

SINTAXE DAS EXPRESSÕES REGULARES

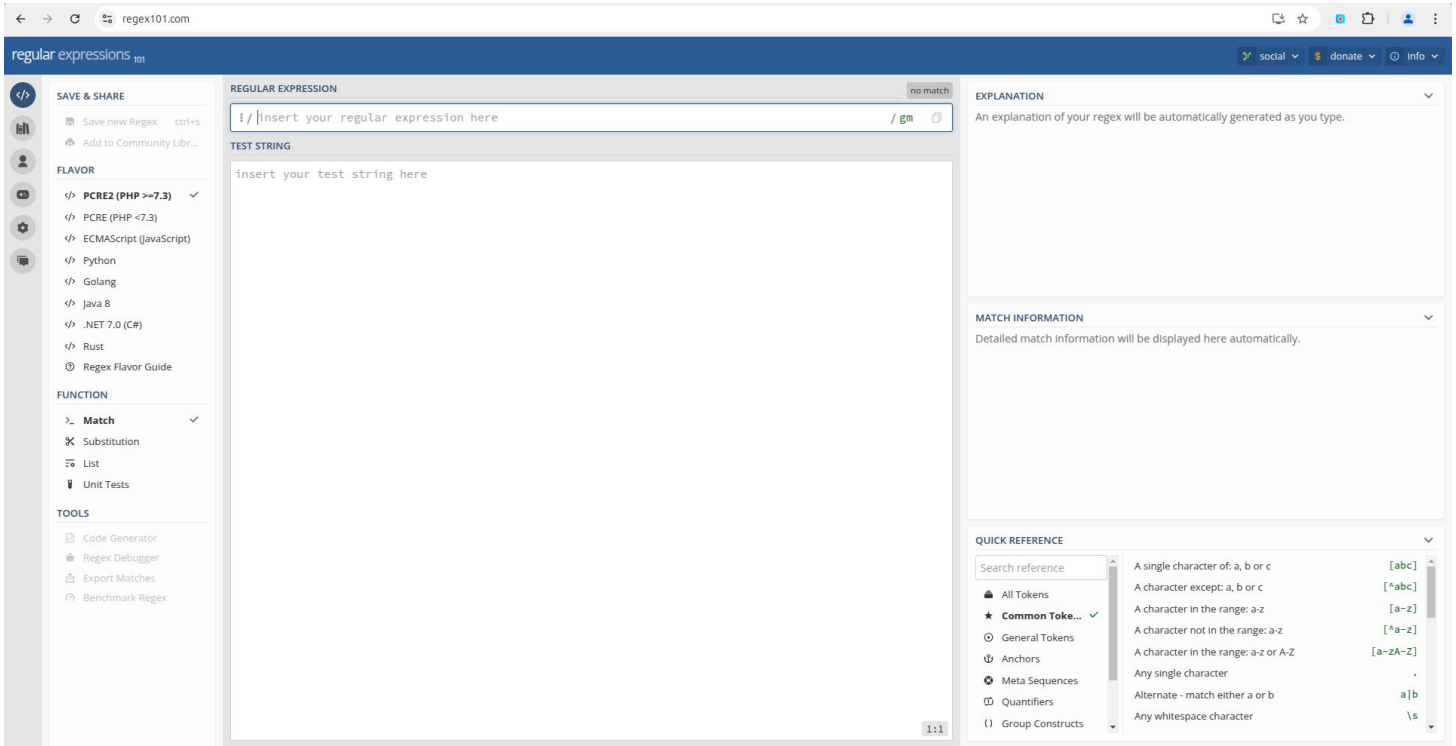
- **^**: Indica o início de uma string.
- **$**: Indica o final de uma string.
- **.**: Corresponde a qualquer caractere, exceto uma nova linha.
- **[]**: Conjunto de caracteres.
- **|**: Alternância.
- *: Corresponde a zero ou mais repetições do elemento anterior.
- **+**: Corresponde a uma ou mais repetições do elemento anterior.
- **?**: Torna o elemento anterior opcional (zero ou uma vez).
- **{n,m}**: Corresponde a entre n e m repetições do elemento anterior.
- **()**: Grupo de captura, usado para agrupar expressões.

REGEX101

O site **regex101** permite criar expressões regulares e testá-las.

link: https://regex101.com/

REGEX101


regex101.com
regular expressions 101
social
donate
info
SAVE & SHARE
Save new Regex ctrl+s
Add to Community Libr...
FLAVOR
PCRE2 (PHP >=7.3)
PCRE (PHP <7.3)
ECMAScript (JavaScript)
Python
Golang
Java 8
.NET 7.0 (C#)
Rust
Regex Flavor Guide
FUNCTION
Match
Substitution
List
Unit Tests
TOOLS
Code Generator
Regex Debugger
Export Matches
Benchmark Regex
REGULAR EXPRESSION
no match
insert your regular expression here
/ gm
TEST STRING
insert your test string here
1:1
EXPLANATION
An explanation of your regex will be automatically generated as you type.
MATCH INFORMATION
Detailed match information will be displayed here automatically.
QUICK REFERENCE
Search reference
All Tokens
Common Toke...
General Tokens
Anchors
Meta Sequences
Quantifiers
Group Constructs
A single character of: a, b or c [abc]
A character except: a, b or c [^abc]
A character in the range: a-z [a-z]
A character not in the range: a-z [^a-z]
A character in the range: a-z or A-Z [a-zA-Z]
Any single character .
Alternate - match either a or b a|b
Any whitespace character \s

EXPRESSÕES REGULARES NO JAVASCRIPT

“Expressões regulares são padrões utilizados para selecionar combinações de caracteres em uma string. Em JavaScript, expressões regulares também são objetos. Elas podem ser utilizadas com os métodos exec e test do objeto RegExp, e com os métodos match, replace,search, e split do objeto String. “

Texto extraído de https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Regular_expressions

EXPRESSÕES REGULARES NO JAVASCRIPT

- Maneiras de construir uma expressão regular:

```
const re = /ab+c/
```

Usando uma expressão literal, que consiste em um padrão fechado entre barras.

EXPRESSÕES REGULARES NO JAVASCRIPT

- Maneiras de construir uma expressão regular:

```
let re = new RegExp("ab+c")
```

Chamando o construtor do objeto RegExp.

EXPRESSÕES REGULARES NO JAVASCRIPT

Usando os métodos **match**, **replace**,**search**, e **split** do objeto String

O método **match**() retorna **apenas a primeira correspondência** de um padrão de pesquisa como um **array com detalhes da captura**.

```
const phrase = "abacate abacaxi abajur abalou"
const regex = /aba/
phrase.match(regex) // ['aba', index: 0, input: 'abacate abacaxi abajur abalou', groups:
undefined]
```

EXPRESSÕES REGULARES NO JAVASCRIPT

Usando **match**() com o operador global **g** o retorno será todas as correspondências como um array de strings.

```
const phrase = "abacate abacaxi abajur abalou"
const regex = /aba/g

phrase.match(regex) //(4) ['aba', 'aba', 'aba', 'aba']
```

EXPRESSÕES REGULARES NO JAVASCRIPT

Entrando ocorrências de um padrão para números de celulares.

```
const phrase = "my cellphone is 2191234-5678 and her cellphone is 2199876-5432"
phrase.match(/\d+\-\d+/g) // (2) ['2191234-5678', '2199876-5432']
```

EXPRESSÕES REGULARES NO JAVASCRIPT

O método **replace**() retorna uma nova string com algumas ou todas as correspondências de um padrão substituídas por um determinado caractere (ou caracteres).

```
const phrase = "abacate abacaxi abajur abalou"
const regex = /aba/
phrase.replace(regex, 'epa') // 'epacate abacaxi abajur abalou'
```

EXPRESSÕES REGULARES NO JAVASCRIPT

Método **replace** usando operador global **g**

```
const phrase = "abacate abacaxi abajur abalou"
const regex = /aba/g
phrase.replace(regex, 'epa') // 'epacate epacaxi epajur epalou'
```

EXPRESSÕES REGULARES NO JAVASCRIPT

Substituindo palavras iniciadas por ‘aba’ seguidas ou não de caracteres pela palavra ‘pia’.

```
const phrase = "abacate abacaxi abajur abalou"
const regex = /aba\w*/g
phrase.replace(regex, 'pia') // 'pia pia pia pia'
```

EXPRESSÕES REGULARES NO JAVASCRIPT

O método **search**() realiza uma busca por uma ocorrência entre uma expressão regular e uma String. O retorno é o índice na string do primeiro trecho que satisfaz a expressão regular ou valor -1 caso nenhuma ocorrência seja encontrada.

```
const phrase = "abacate abacaxi abajur abalou"
phrase.search(/ba/) // 1
```

EXPRESSÕES REGULARES NO JAVASCRIPT

O método **search**() realiza uma busca por uma ocorrência entre uma expressão regular e uma String. O retorno é o índice na string do primeiro trecho que satisfaz a expressão regular ou valor -1 caso nenhuma ocorrência seja encontrada.

```
const phrase = "abacate abacaxi abajur abalou"
phrase.search(/70/) // -1
```

EXPRESSÕES REGULARES NO JAVASCRIPT

O método **split**() divide uma string em várias substrings, organizando-as em um array e retornando esse array. A divisão ocorre com base em um padrão, que é especificado como o primeiro argumento na chamada do método.

```
const phrase = "abacate abacaxi abajur abalou"
const regex = /ba/g
phrase.split(regex) // (5) ['a', 'cate a', 'caxi a', 'jur a', 'lou']
```

EXPRESSÕES REGULARES NO JAVASCRIPT

Caso a ocorrência não seja encontrada, um array com a cópia da string é retornado.

```
const phrase = "abacate abacaxi abajur abalou"
const regex = /bb/g
phrase.split(regex) // ['abacate abacaxi abajur abalou']
```

EXPRESSÕES REGULARES NO JAVASCRIPT

Usando métodos **exec** e **test** do objeto RegExp.

O método **exec**() executa a busca por um padrão em uma determinada string. Retorna um array, ou null.

```
// Criando o objeto RegExp
const regex = new RegExp('(\\d+)')  // Expressão regular para capturar números
const texto = 'Eu tenho 123 maçãs e 456 bananas'

const resultado = regex.exec(texto)
console.log(resultado) // ["123", "123"]
```

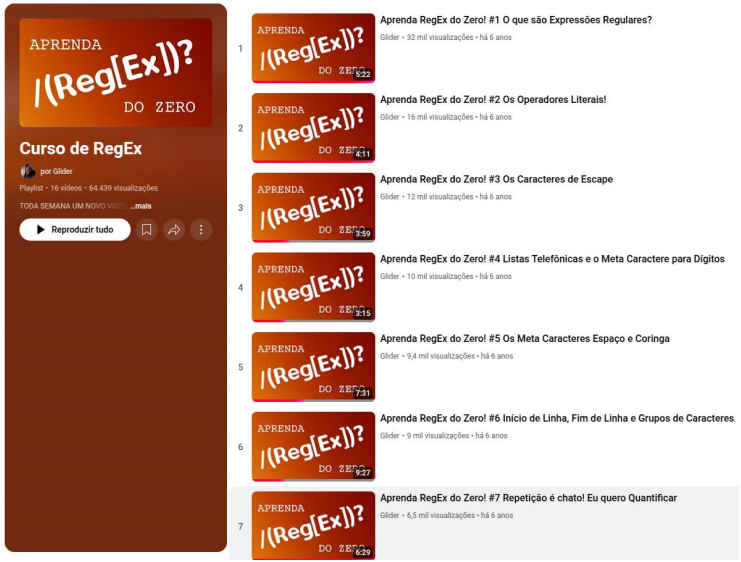
EXPRESSÕES REGULARES NO JAVASCRIPT

Usando métodos **exec** e **test** do objeto RegExp.

O método **test**() é um dos mais comuns quando você está lidando com expressões regulares em JavaScript. Ele verifica se uma expressão regular corresponde a alguma parte de uma string e retorna true ou false.

```
// Criando o objeto RegExp
const regex = new RegExp('abc')  // Expressão regular para encontrar "abc"
const texto = 'Eu tenho de abc e ABC'

const resultado = regex.test(texto)
console.log(resultado) // true (porque "abc" é encontrado no texto)
```



[Aprenda RegEx do Zero!]

Playlist curso gratuito de expressões regulares

Bónus

MÃO NA MASSA

MÃO NA MASSA

1. Escreva uma expressão regular que verifique se uma string começa com um número. Se sim, exiba o número.

2. Dado o texto "O vigia estava ansioso", extraia todas as palavras que têm exatamente 5 letras.

3. Crie uma expressão regular que valide e-mails no formato usuario@dominio.com.

4. Dado o texto "A conta total foi de 123, mas recebi um desconto de 23", extraia todos os números.

5. Substitua todas as datas no formato dd/mm/yyyy pelo texto "DATA" em:

"As datas importantes são 25/12/2023 e 01/01/2024."

MÃO NA MASSA

6. Escreva uma expressão regular para verificar se uma string contém um CEP no formato brasileiro XXXXX-XXX

7. Dado o texto "Pedro e Paula foram para a praia", encontre todas as palavras que começam com "p" ou "P".

8. Dado o texto com URLs: "Acesse https://google.com ou http://example.org para mais informações." Extraia apenas os domínios (google.com, example.org).

9. Crie uma expressão regular para validar números no formato: (XX) XXXXX-XXXX.

10. Dado o texto "Use #JavaScript e #regex para resolver problemas.", extraia todas as hashtags.

REFERÊNCIAS

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions
- https://www.w3schools.com/python/python_regex.asp
- https://regex101.com/