
[abc]
E-BOOK
GUIA
RÁPIDO DE
REGEX
{n}
WWW.TREINAWEB.COM.BR
TREINAWEB

SUMÁRIO

INTRODUÇÃO

Seja bem-vindo! Este é um guia rápido de RegEx do TreinaWeb.

RegEx é uma sigla em inglês que vem de “Regular Expression”, ou, em português, “Expressão Regular”, uma das ferramentas mais úteis para desenvolvedores em geral.

Expressões regulares são expressões que determinam formatos ou padrões de texto. As expressões regulares são utilizadas principalmente para extrair e validar informações. Sendo assim, você pode, por exemplo, determinar um formato de texto para extrair informações de uma página ou montar uma expressão para validar a entrada de um usuário... As possibilidades com expressões regulares são muitas!

Boa leitura e bons estudos!

RegEx

1. Componentes das expressões regulares

Uma expressão regular tem três componentes basicamente:

- A expressão regular em si, responsável por definir o padrão a ser aplicado na entrada;

- A entrada de dados, que constitui os dados que irão sofrer a ação da expressão regular;

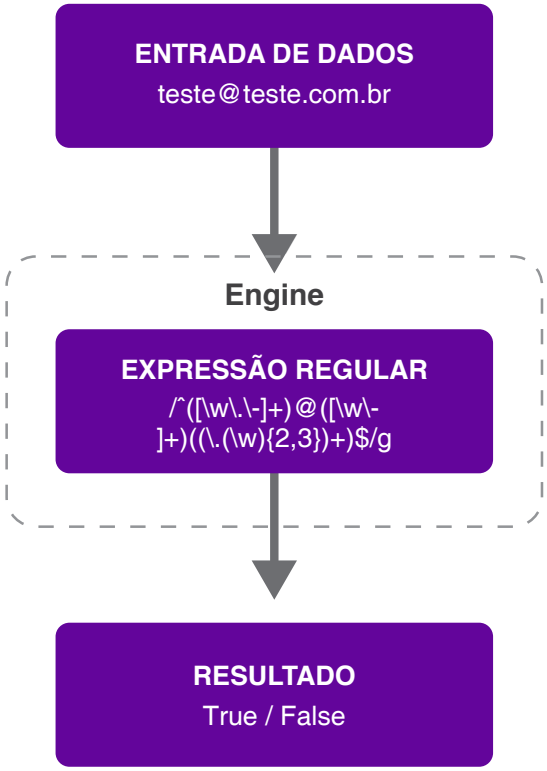
- A engine, que é o componente responsável por aplicar a expressão.

Por exemplo: vamos imaginar que precisamos de uma expressão regular para validar um e-mail. Neste caso, teríamos as seguintes situações:

- A expressão regular deveria descrever o formato válido de um e-mail;

- A entrada de dados seria o e-mail informado por um usuário para ser validado;

- A engine seria a linguagem utilizada para a aplicação da expressão regular.



2. Formato das expressões regulares

Uma expressão regular precisa obrigatoriamente estar contida entre barras (/). Após a expressão, podem existir algumas indicações de execução que são chamadas de flags.

```
/<meta-caracteres>/[flags]
```

As flags indicam de maneira geral parâmetros especiais para a execução da expressão regular pela engine. As flags são opcionais, além de que sua aceitação irá depender da engine que irá executar a expressão regular.

As flags mais comuns são:

i (Ignore case) - Diferenças entre caracteres em caixa alta ou caixa baixa não devem ser levadas em consideração.

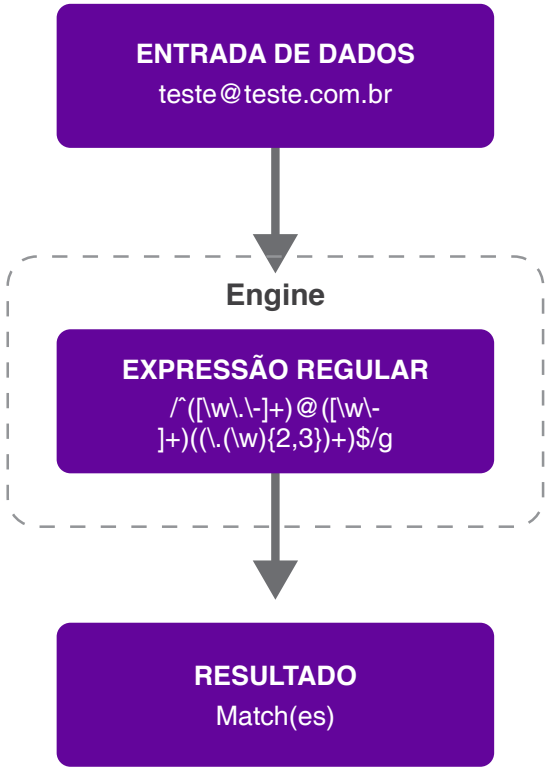
g (Global source) - Permite captura de dados de maneira global na entrada de dados, além de guardar os índices de ocorrência e permitir a navegação entre elas

m (Multiline) - Flag para informar que a entrada é constituída por múltiplas linhas, fazendo com que a expressão regular seja aplicada em cada uma das linhas.

3. Matches

Um match é uma parte do texto que corresponde ao formato especificado pela expressão regular. Se pensarmos em uma expressão regular para validação do formato de um dado de entrada, poderíamos considerar pelo menos um match da expressão regular sob a entrada.

Poderíamos representar os matches com o diagrama apresentado anteriormente:



4. Meta-caracteres

Os meta-caracteres são caracteres especiais que possuem um significado nas expressões regulares.

Eles servem para indicar para a engine como ela deve fazer a análise da entrada de dados. Um conjunto destes meta-caracteres formam uma expressão regular, ou RegEx.

Cada meta-caractere tem uma função específica e que pode mudar dependendo do contexto inserido. Você pode fazer diversas combinações e realizar expressões mais complexas com estas combinações.

Os principais meta-caracteres serão descritos a seguir.

4.1. Início (^) e fim ($)

ˆ (função "Início") - Indica "começo de linha" em uma expressão regular. Sintaxe: **/^<...>/**

$ (função "Fim") - Indica "final de linha" em uma expressão regular. Sintaxe: **/<...>$/**

Veja a expressão regular abaixo:

Expressão:	/^[1,9]/g
Entrada:	0,1,2,3,4
Saída:	0,1,2,3,4

Veja que não houveram matches dentro da entrada de dados, pois a expressão regular inserida diz que procuramos uma linha que **comece** com um caractere de 1 a 9.

Se mudarmos a entrada de dados...

Expressão:	/^[1,9]/g
Entrada:	1,2,3,4,5
Saída:	1,2,3,4,5

Agora o match passa a acontecer, já que a entrada de dados começa com o número 1.

O funcionamento é muito similar para o meta-caractere de fim de linha, que, como o próprio nome sugere, faz a busca no final da entrada de dados:

Expressão:	/^[1,9]/g
Entrada:	1,2,3,4,5
Saída:	1,2,3,4,5

4.2. Grupo de caracteres ([e])

[(função "Início de grupo") - Indica o começo de um grupo de caracteres. Sintaxe: **/[<...>]/**

] (função "Fim de grupo") - Indica o fim de um grupo de caracteres. Sintaxe: **/[<...>]/**

Por exemplo: vamos imaginar que desejamos obter todos os matches do nome "João", independente de estarem escritos com letra maiúscula ou minúscula e com til ou sem til. Precisamos então informar a engine que o primeiro caractere pode ser "j" ou "J". Também precisamos dizer à engine que o terceiro caractere pode ser "a" ou "ã".

Expressão:	/[Jj]o[aã]o/gm
Entrada:	joao joão João Joao Zoao
Saída:	joao joão João Joao Zoao

4.3. Ponto (.)

. (função "Caractere "coringa") - Representa qualquer outro caractere. Sintaxe: **/./**

Por exemplo: vamos imaginar que desejamos recuperar todas as palavras que possuam quatro letras e que terminem com o sufixo "ato".

Expressão:	/.ato/gm
Entrada:	gato mato rato martelato
Saída:	gato mato rato martelato

4.3.1. Operações lógicas

Também temos meta-caracteres para representar as principais operações lógicas: AND, OR e NOT.

4.3.2. AND (.*)

.* (função "Operação lógica AND") - Realiza a operação lógica AND entre duas expressões. Sintaxe: **/<expressão 1>.*<expressão 2>/**

Vamos imaginar que desejamos recuperar todos os nomes em uma lista que comecem e terminem com vogais.

Expressão:	/^[AEIOU].*[aeiou]$/gm
Entrada:	Amanda Eduardo Alana Emanuel
Saída:	Amanda Eduardo Alana Emanuel

4.3.3. OR (|)

| (função "Operação lógica OR") - Realiza a operação lógica OR entre duas expressões. Sintaxe: **/<expressão 1>|<expressão 2>/**

Vamos imaginar que queiramos capturar todos os mercados de médio e grande porte, sendo eles supermercados ou hipermercados, de uma entrada de dados.

Expressão:	/^(super|hiper)mercado$/gm
Entrada:	Supermercado Hipermercado Minimercado
Saída:	Supermercado Hipermercado Minimercado

4.3.4. NOT (^)

^ (função "Operação lógica NOT") - Realiza a operação lógica NOT em uma expressão. Sintaxe: **/[^<expressão>]/**

Vamos imaginar que queiramos capturar, a partir de uma lista de nomes, todos os que não sejam iniciados por vogais.

Expressão:	/^[^AEIOU].*/gm
Entrada:	Amanda Eduardo Alana Marcelo
Saída:	Amanda Eduardo Alana Marcelo

4.4. Meta-caracteres de quantificação

Expressões regulares também permitem a quantificação de ocorrências de caracteres através de meta-caracteres específicos.

4.4.1.Chaves ({ e })

ˆ (função "Repetição de expressão") - Permite detectar/determinar a repetição de uma expressão por n vezes ou dentro de um intervalo. Sintaxe: **/<expressão>{nmin, nmax}/**

Por exemplo: vamos escrever uma expressão regular que rastreie todas as palavras que contenham 9 letras, sendo que as duas últimas sejam o sufixo "ão".

Expressão:	/^.{7}ão$/gm
Entrada:	Graduação Exposição Impressão Locução
Saída:	Graduação Exposição Impressão Locução

4.4.2. Plus (+)

+ (função "1 ocorrência ou mais") - Verifica se o caractere que estiver à esquerda ocorre pelo menos uma vez na entrada de dados. Sintaxe: **/<expressão>+/**

Por exemplo: vamos recuperar todas as ocorrências do caractere "a" dentro de uma entrada.

Expressão:	/a+/gm
Entrada:	Música Livro Tecnologia
Saída:	Música Livro Tecnologia Alambrado

4.4.3. Interrogação (?)

? (função "0 ou 1 ocorrência") - Verifica se o caractere que estiver à esquerda ocorre nenhuma vez ou pelo menos uma vez. Faz o papel de "opcional". Sintaxe: **/<expressão>?/**

Por exemplo: vamos recuperar todas as palavras que possuam o radical “cair”. Sendo assim, devemos capturar palavras como “cai”, “cair”, “caio” e “caiu”. Veja que temos opcionalmente o radical “cai” com as terminações “r”, “o” e “u”

Expressão:	/ ^cai[r,u,o]?$/gm
Entrada:	cai caiu caio cair caída
Saída:	cai caiu caio cair caída

4.4.4. Asterisco (*)

*** (função “0 ou mais ocorrências”)** - Verifica se o caractere que estiver à esquerda ocorre nenhuma vez ou uma ou várias vezes. Sintaxe:
/<expressão>*/

Por exemplo: vamos recuperar todas as vezes que o usuário digitou “oi”, tendo ele digitado “oi”, “oii” ou “oiiii”.

Expressão:	/ ^oi*$/gm
Entrada:	oi oii oiii oioi
Saída:	oi oii oiii oioi

4.5. Outros meta-caracteres

4.5.1. Hífen (-)

- (função “Intervalo”) - Verifica matches que correspondam a um intervalo de caracteres. Sintaxe:
/[<início>-<fim>]/

Por exemplo: vamos buscar os nomes de pessoas que comecem com as letras de A até M.

Expressão:	/^[A-M].*$/gm
Entrada:	Gabriela Catarina Marina Natália

Saída:	Gabriela Catarina Marina Natália

4.5.2. Dígito e não-dígito (\d e \D)

\d (função "Dígito") - Captura os caracteres que sejam correspondentes a números. Sintaxe: **/\d/**

\D (função "Não-dígito") - Captura os caracteres que sejam correspondentes a letras. Sintaxe: **/\D/**

Por exemplo: vamos buscar todos os números em uma entrada de dados.

Expressão:	/\d/gm
Entrada:	123 Teste Teste_123
Saída:	123 Teste Teste_123

Poderíamos também capturar exclusivamente as letras dessa mesma entrada.

Expressão:	/\D/gm
Entrada:	123 Teste Teste_123
Saída:	123 Teste Teste_123

4.5.3. Alfanumérico e não-alfanumérico (\w e \W)

\w (função "Alfa-numérico") - Captura os caracteres que sejam alfanuméricos, mais o underline. Sintaxe: **/\w/**

\w (função "Não-alfa-numérico") - Captura os caracteres que não sejam alfanuméricos (como caracteres especiais. Sintaxe: **/\W/**

Por exemplo: podemos querer capturar todos os caracteres que sejam considerados alfa-numéricos.

Expressão:	/\w/gm
Entrada:	Teste 1234 Teste_123 Teste$%Teste $%&*&

Saída:	Teste 1234 Teste_123 Teste$%Teste $%&*&

Também podemos recuperar matches com caracteres especiais.

Expressão:	/\w/gm
Entrada:	Teste 1234 Teste_123 Teste$%Teste $%&*&
Saída:	Teste 1234 Teste_123 Teste$%Teste $%&*&

4.5.4. Espaço e não espaço (\s e \S)

\s (função "Espaço") - Captura os caracteres correspondentes a espaços (espaços propriamente ditos, tabulações, quebras de linha e retornos).
Sintaxe: **/\s/**

\S (função "Não-espaço") - Captura os caracteres que não sejam correspondentes a espaços.
Sintaxe: **/\S/**

Por exemplo: se quisermos capturar todos os espaços em uma entrada.

Expressão:	/\s/gm
Entrada:	Isso é um espaço Isso é um espaço com tabulação 123 45
Saída:	Isso é um espaço Isso é um espaço com tabulação 123 45

Também poderíamos recuperar todos os caracteres que não correspondessem a espaço.

Expressão:	/\s/gm
Entrada:	Isso é um espaço Isso é um espaço com tabulação 123 45
Saída:	Isso é um espaço Isso é um espaço com tabulação 123 45

4.5.5. Bordas: word boundary (\b e \B)

\b (função "Borda inicial") - Captura as bordas iniciais das sequências de caracteres (primeiro caractere). Sintaxe: **/\b<caractere>/**

\B (função "Borda final") - Captura caracteres que não estejam na borda de uma sequência de caracteres. Sintaxe: **/\B<caractere>/**

Por exemplo: podemos capturar todas as iniciais com a letra "e" ou "E".

Expressão:	/\b[eE]/gm
Entrada:	Este é o e-book de RegEx do TreinaWeb! Eba!
Saída:	Este é o e-book de RegEx do TreinaWeb! Eba!

Também poderíamos capturar todas as finalizações com os caracteres "e" e "E".

Expressão:	/\B[eE]/gm
Entrada:	Este é o e-book de RegEx do TreinaWeb! Eba!
Saída:	Este é o e-book de RegEx do TreinaWeb! Eba!

4.5.6. Grupos de captura e não-captura ("()" e "(?:)")

() (função "Grupo de captura") - Agrupa uma sequência de caracteres que deve ser capturada. São produzidos sub-matches quando este meta-caractere é utilizado. Sintaxe: **/(<caracteres do grupo>)<meta-caractere a ser aplicado no grupo>/**

(?:) (função "Grupo de não-captura") - Agrupa uma sequência de caracteres que não deve ser capturada. São produzidos sub-matches quando este meta-caractere é utilizado. Sintaxe: **/(?:<caracteres do grupo>)<meta-caractere a ser aplicado no grupo>/**

Por exemplo: podemos capturar a sequência "abc", desde que essa se repita por três vezes.

Expressão:	/(abc){3}/
Entrada:	abcabcabcabdab
Saída:	abcabcabcabdab

Observação:	Neste caso, são produzidos dois matches: - Um correspondente à sequência “abcabcabc”, respeitando a quantificação à frente do grupo de captura; - Outro correspondente ao grupo de captura “abc”.

Podemos também desejar que o match correspondente ao grupo de captura seja desprezado com um grupo de não-captura.

Expressão:	/(?:abc){3}/
Entrada:	abcabcabcabdab
Saída:	abcabcabcabdab
Observação:	Neste caso, um único match é produzido: o correspondente à sequência “abcabcabc”. O match correspondente ao grupo de captura é desprezado.

4.5.7. Sucessão (?=) e não-sucessão (?!)

(?=) (função “Sucessão”) - Agrupa uma sequência de caracteres que deve ser capturada. São produzidos sub-matches quando este meta-caractere é utilizado. Sintaxe: **/(<caracteres do grupo>)<meta-caractere a ser aplicado no grupo>/**

(?!) (função “Não-sucessão”) - Agrupa uma sequência de caracteres que não deve ser capturada. São produzidos sub-matches quando este meta-caractere é utilizado. Sintaxe: **/(?:<caracteres do grupo>)<meta-caractere a ser aplicado no grupo>/**

Por exemplo: podemos desejar capturar a quantidade de litros (que são seguidos por “l”) de uma lista de unidades de medida.

Expressão:	/^\d+(?=\sl)/gm
Entrada:	1 l 30 m 10 l 20 kg
Saída:	1 l 30 m 10 l 20 kg

Também podemos capturar as quantidades de todas as relações que não são expressas em litros:

Expressão: /^\d{2}(?!\sl)/gm

Entrada:

```
1 l
30 m
10 l
20 kg
```

Saída:

```
1 l
30 m
10 l
20 kg
```

5. Quadro-resumo dos principais meta-caracteres RegEx

Meta-caractere	Descrição	Expressão de exemplo	Entrada de exemplo Saída de exemplo
^	Início de linha	/^[1,9]/g	1,2,3,4,5 1,2,3,4,5
$	Fim de linha	/[1,9]$/g	1,2,3,4,5 1,2,3,4,5
[]	Grupo de caracteres	/[Jj]o[aã]o/gm	joão Joao Zoao joão Joao Zoao
.	Caractere "coringa"	/.ato/gm	gato mato martelato gato mato martelato
.*	AND	/^[AEIOU].*[aeiou]$/gm	Amanda Eduardo Alanor Amanda Eduardo Alanor
|	OR	/^(super|hiper)mercado$/gm	Supermercado Hipermercado Minimercado Supermercado Hipermercado Minimercado
^	NOT	/^[^AEIOU].*/gm	Alana Alanor Marcelo Alana Alanor Marcelo
{{}}	Repetição	/^.{7}ão$/gm	Exposição Impressão Locução Exposição Impressão Locução
+	1/N ocorrências	/a+/gm	Música Livro Alambrado Música Livro Alambrado
?	0/1 ocorrência	/ ^cai[r,u,o]?$/gm	caio cair caída caio cair caída
*	0/N ocorrências	/ ^oi*$/gm	oii oiii oioi oii oiii oioi

-	Intervalo	/^[A-M].*$/gm	Catarina Marina Natália **Catarina** **Marina** Natália
\d	Dígito	/\d/gm	123 Teste Teste_123 **123** Teste Teste_**123**
\D	Não-dígito	/\D/gm	123 Teste Teste_123 123 **Teste** **Teste**_123
\w	Alfanumérico	/\w/gm	Teste 1234 Teste$%Teste $%&*& **Teste** **1234** **Teste**$%**Teste** $%&*&
\W	Não-alfanumérico	/\W/gm	Teste 1234 Teste$%Teste $%&*& Teste 1234 Teste**$%**Teste **$%&*&**
\s	Espaço	/\s/gm	Isso é um espaço Tem TAB Isso** **é** **um** **espaço ** **Tem TAB
\S	Não espaço	/\S/gm	Isso é um espaço Tem TAB **Isso** **é** **um** **espaço** **Tem** **TAB**
\b	Borda	/\b[eE]/gm	Este e-book **E**ste **e**-book
\B	Não-borda	/\B[eE]/gm	Este e-book Reg Est**e** e-book R**e**g
()	Grupo de captura	/(abc){3}/	abcabcabcabdab **abcabcabc**abdab (2m)
(?:)	Grupo de não-captura	/(?:abc){3}/	abcabcabcabdab **abcabcabc**abdab (1m)
(?=)	Sucessão	/^\d+(?=\sl)/gm	1 l 30 m 20 kg **1** l 30 m 20 kg
(?!)	Não-sucessão	/^\d{2}(?!\sl)/gm	1 l 30 m 20 kg 1 l **30** m **20** kg

6. Expressões RegEx mais utilizadas

Descrição	Expressão
Validação de e-mail	`/^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/`
Validação de CPF ou CNPJ	`/([0-9]{2}[\.]?[0-9]{3}[\.]?[0-9]{3}[\/]?[0-9]{4}[-]?[0-9]{2})|([0-9]{3}[\.]?[0-9]{3}[\.]?[0-9]{3}[-]?[0-9]{2})/`
Validação de CEP	`/^[0-9]{5}-[0-9]{3}$/`
Validação de URL (HTTP, HTTPS, FTP e FILE)	`/^((https?|ftp|file):\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$/`
Validação de nome de usuário (De 3 a 16 caracteres, números, letras e underline)	`/^[a-z0-9_-]{3,16}$/`
Validação de senha (De 6 a 18 caracteres, números, letras, underline e hífen)	`/^[a-z0-9_-]{6,18}$/`
Validação de IP	`/^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/`
Número de telefone (Com nono dígito ou não)	`/^\(?\d{2}\)?[\s-]?[\s9]?\d{4}-?\d{4}$/`
Data (formato dd/mm/yyyy)	`/^(0?[1-9]|[12][0-9]|3[01])([ \/\-])(0?[1-9]|1[012])\2([0-9][0-9][0-9][0-9])(([ -])([0-1]?[0-9]|2[0-3]):[0-5]?[0-9]:[0-5]?[0-9])?$/`
Número positivo	`/^\d+$/`
Número inteiro	`/^-?\d+$/`
Ano (1900 a 2099)	`/^(19|20)\d{2}$/`

Acompanhe-nos nas redes sociais: