

O que é uma Expressão Regular?

Uma expressão regular, ou **Regex**, são padrões utilizados para identificar determinadas combinações ou cadeias de caracteres em uma string. Ela faz parte do dia a dia de todos os programadores e administradores de infra.

Por meio dela, podemos validar a entrada de usuários ou encontrar alguma informação em logs, documentação ou saída de comando. O mais legal é que as Regex são escritas independentes de uma linguagem, como JavaScript ou C#.

As expressões são definidas em sua própria linguagem formal e uma vez aprendida, podemos aplicar o conhecimento dentro da linguagem de nossa preferência. Em outras palavras, linguagens como Java, JavaScript, C# e várias outras possuem uma implementação das expressões regulares e sabem interpretá-la.

Onde queremos chegar? Entender, por exemplo, uma Regex assim:

```
<(img)\s+src="(.)" (?:>(?:.*<\/\1>|\s+\/>
```

Veja por exemplo, o exemplo abaixo...

Expressões regulares

Target string (alvo)

imagem.png

Pattern (expressão regular)

.*png

Executar Regex

☐ Mostra índice ☐ Mostra grupos

1 Matches (resultados)

imagem.png

Highlight

imagem.png

Repare que já estamos usando um vocabulário focado na Regex. A string que queremos usar na busca, no exemplo da imagem, a string `imagem.png` é chamada de *alvo*, ou *target*. A expressão regular, na imagem `.*png`, estamos chamando de *pattern*. Os resultados são os `*matches`. *

Já estamos aplicando a primeira Regex, estamos procurando algum texto que termine com `png`.

O ponto (`.`) é um caractere especial e significa **qualquer caractere**. Ou seja, a Regex não interpreta o ponto literalmente e sim faz um *match* para qualquer `*char`. *

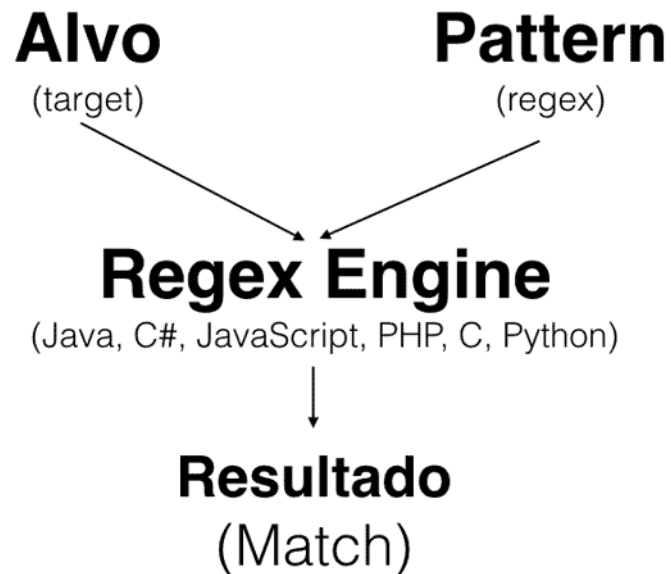
O asterisco (`*`) é outro **meta-char** com o significado "zero, um ou mais vezes". Ao adicionar o asterisco (`*`), conseguimos definir a quantidade, por isso ele também é chamado de **quantifier**.

O ponto (`.`) e asterisco (`*`) fazem parte dos **metacaracteres** que veremos durante o curso.

Apresentando o código

Qualquer Regex precisa ser interpretada por meio de uma *Regex engine*. Esse motor é uma parte de software que processa a expressão, tentando achar o padrão dentro da string dada, devolvendo os resultados. Normalmente a *Regex engine* faz parte da aplicação maior, para executar uma validação ou busca de uma string.

No nosso caso, usamos uma *Regex engine* baseada na linguagem JavaScript. No treinamento, veremos ainda como usar expressões regulares em outras linguagens, como Java, C# ou Ruby.



Vamos dar uma olhada na função `executaRegex`, dentro do arquivo `regex.js`. Nela criamos o objeto que sabe interpretar a expressão regular. O JavaScript chama esse objeto de `RegExp`:

```
var objetoRegex = new RegExp(textoPattern, 'g');
```

No construtor passamos o *pattern*, aquilo que colocamos no `input` *Pattern* da página `index.html`. A letra `g` é uma flag do mundo JavaScript e significa `global`, para aplicar a regex na string inteira (e não parar no primeiro *match*) .

Após a inicialização do objeto regex, podemos executá-lo usando o método `exec`, que recebe como parâmetro o alvo:

```
while (resultado = objetoRegex.exec(textoTarget)) {  
  
    //codigo omitido  
}
```

O método `exec` devolve um resultado que possui a string *match* *e outras informações, como a posição/index. *Repare que estamos fazendo um laço sempre pedindo o próximo resultado.

O resto do código do arquivo `regex.js` se preocupa com a leitura, validação e apresentação dos dados.