1. O que é Regex?

Regex, abreviação de "Regular Expression" (Expressão Regular), é uma sequência de caracteres que define um padrão de busca. É utilizada para encontrar, validar ou substituir texto em strings.

1. Caracteres básicos:

* .: Corresponde a qualquer caractere, exceto nova linha.
* ^: Início da linha.
* $: Fim da linha.
* \: Escape de caracteres especiais.

1. Quantificadores:

* \*: Zero ou mais ocorrências.
* +: Uma ou mais ocorrências.
* ?: Zero ou uma ocorrência.
* {n}: Exatamente n ocorrências.
* {n,}: No mínimo n ocorrências.
* {n,m}: Entre n e m ocorrências.

1. Classes de caracteres:

* [abc]: Corresponde a qualquer caractere dentro dos colchetes.
* [^abc]: Corresponde a qualquer caractere que não esteja dentro dos colchetes.
* [a-z]: Corresponde a qualquer caractere minúsculo de a a z.
* [A-Z]: Corresponde a qualquer caractere maiúsculo de A a Z.
* [0-9]: Corresponde a qualquer dígito de 0 a 9.

1. Metacaracteres:

* \d: Dígito (equivalente a [0-9]).
* \D: Não dígito.
* \w: Caractere de palavra (letras, dígitos e underscore).
* \W: Não caractere de palavra.
* \s: Espaço em branco.
* \S: Não espaço em branco.

1. Grupos e captura:

* (...): Cria um grupo de captura.
* (?:...): Cria um grupo sem captura.
* \1, \2, ...: Referência a grupos capturados.

1. Alternância:

* |: Operador OR (ou).

1. Âncoras:

* \b: Limite de palavra.
* \B: Não limite de palavra.

1. Lookahead e Lookbehind:

* (?=...): Lookahead positivo.
* (?!...): Lookahead negativo.
* (?<=...): Lookbehind positivo.
* (?<!...): Lookbehind negativo.

1. Flags:

* i: Ignora maiúsculas/minúsculas.
* g: Busca global (todas as ocorrências).
* m: Multiline (^ e $ correspondem ao início e fim de cada linha).

1. Exemplos práticos:

a. Validar um e-mail:

regexCopiar

^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$

b. Validar um número de telefone brasileiro:

regexCopiar

^\(?[1-9]{2}\)? ?(?:[2-8]|9[1-9])[0-9]{3}\-?[0-9]{4}$

c. Encontrar tags HTML:

regexCopiar

<([a-z]+)([^<]+)\*(?:>(.\*?)<\/\1>|\s+\/>)

1. Dicas de uso:

* Sempre teste suas expressões regulares com diversos casos.
* Use ferramentas online como regex101.com para testar e depurar suas expressões.
* Comece com padrões simples e vá aumentando a complexidade conforme necessário.
* Documente suas expressões regulares complexas para facilitar a manutenção.

1. Limitações:

* Expressões regulares não são adequadas para analisar HTML ou XML complexos.
* Expressões muito complexas podem ser difíceis de manter e entender.
* Algumas implementações de regex podem ter diferenças sutis entre linguagens.

1. Ferramentas e bibliotecas:

A maioria das linguagens de programação modernas suporta regex nativamente ou através de bibliotecas. Alguns exemplos:

* Python: módulo re
* JavaScript: objeto RegExp
* Java: classe java.util.regex.Pattern
* C#: classe System.Text.RegularExpressions.Regex
* PHP: funções preg\_\*

Lembre-se de que a prática é fundamental para dominar as expressões regulares. Comece com padrões simples e vá aumentando a complexidade à medida que ganha confiança. Regex é uma habilidade valiosa em programação e processamento de texto, mas requer tempo e experiência para ser dominada completamente.