O que são:

Padrões utilizados para encontrar, modificar ou validar determinados padrões em strings.

Ponto (.) demonstra a busca por qualquer caractere, ele é um coringa, exceto em quebra de linha.

Ex -> / a.c / (significa caractere ‘a’ seguido de qualquer caractere + caractere ‘c’)

= a1cdfc -> true

Caso você queira especificamente o caractere ponto (.), ou seja, o seu valor literal e não qualquer caractere utilize uma barra invertida ou colchetes, dentro de colchetes o caractere ponto(.) também tem seu valor literal -> [.] é igual a \.

Ex -> / \.ato / (significa caractere ‘.’ Seguido de ‘ato’)

Ex -> / .ato / (significa qualquer caractere seguido de ‘ato’)

Barra vertical (|) demosntra ‘ou’

Ex -> / abc | def / (significa que se um desses dois padrões for encontrado o retorno será true)

colchetes [ ] ->  servem para corresponder apenas um dos caracteres listados. Permite selecionar um dos caracteres dentro dos colchetes, não importa a ordem.

Ex -> / [g]ato / (significa ‘g’ seguido de ‘ato’)

Ex -> / [gp]ato / (significa ‘g’ ou ‘p’ seguido de ‘ato’)

/ [a-z] / -> representa qualquer caractere do alfabeto (somente minúsculo)

/ [A-Z] / -> representa qualquer caractere maiusculo do alfabeto

/ [1-9] / -> representa qualquer numero

As chaves “{ }” são utilizadas para repetição quantificada. Um número dentro de chaves “{2}” quer dizer que o item anterior corresponde exatamente 2 vezes.

Ex -> / [A-Z]{4} / (significa 4 caracteres maiúsculos do alfabeto em sequência )

Ex -> / [A-Z]{4,6} / (significa um intervalo de 4 a 6 caractere maiúsculos do alfabeto em sequência

/ -[\d]+ / -> (significa uma sequencia do caractere ‘-‘ seguido de um numero zero ou mais vezes até o infinito).O asterisco (\*) tem a mesma função do (+) . / -[\d]+ / = / -[\d]\* /

O sinal de interrogação (?) é parecido , mas em vez de ser zero ou mais vezes é zero ou uma vez

Ex -> -1 = true , -122453453 = true , -43243 = true

\d é um metacaractere, um curinga que casa com um dígito (0 a 9), ou seja [1-9] é igual a [\d]

\w – Representa um caractere alfanumérico, incluindo ocorrências maiúsculas e minúsculas das letras e o caractere “\_” (Alfanumérico é um conjunto de caracteres alfabéticos e numéricos)

Ex -> / [ \w ]{3} / , 123 = true, aBc = true, 13B = true ,

Exercicio: Validacão de uma placa de carro, o padrão de uma placa de carro é quatro letras maiúsculas seguido de um traço ‘-‘ seguido de três números.

Resposta = / [A-Z]{4}-\d{4} /

Acento circunflexo (^) -> determina que uma expressão tem que começar com determinada sequência obrigatoriamente.

Ex -> / [A-Z]{4}-\d{4} / , 23YYYY-12343 = true

(a expressão acima não tem que começar obrigatoriamente com a sequencia da placa, o código regex nos diz que a sequência deve ser encontrada na string, seja no começo no meio ou no final)

Ex2 -> / ^[A-Z]{4}-\d{4} / 23YYYY-12343 = false

YYYY-1234 = true

Percebeu a diferença.

Mas nós ainda temos um problema, pois obrigatoriamente a string deve começar com o padrão da placa, mas após o padrão ela pode receber qualquer caractere que vai ser considerada valida, ex: YYYY-12343 = true, para resolver esse problema usamos o cifrão ($)

Cifrão ($) -> determina que uma expressão tem que terminar com determinada sequência obrigatoriamente.

Com isso para resolvermos o problema da placa usaremos o código / ^[A-Z]{4}-\d{4}$ / , 23YYYY-12343 = false,

YYYY-12343 = false, YYYY-1234 = true

O \b (beta) é uma âncora assim como seus primos ^ (alfa) e $ (omega). Uma vez adicionado em um dos lados da regex vai capturar padrão especificado no inicio, final ou palavra exata.

A diferença é que o (^) e o ($) são usados para strings, e o (\b) é usado para palavras, ex: gabriel Camargo bunhak é apenas uma string pois considera os espaços, mas sabemos que nessa string contém 3 palavras.

Ex -> / \w+ato / , O pato tem sapato de gato xxatoyy ato

/ \w+ato\b /, O pato tem sapato de gato xxatoyy ato

Expressão regular que identifica tags html: