JavaScript

Expressões Regulares

Regex

Validação de Formulário

```
function isCEP(cep) {
    const re = /^\d{2}\.?\d{3}-?\d{3}$/;
    return re.test(cep);
}
```

Regex

Caractere: \

Uma barra invertida que preceda um caractere não especial significa que o caractere seguinte é especial e não deve ser interpretado de forma literal. Por exemplo, o caractere 'b' quando não precedido de uma barra invertida significará uma ocorrência do próprio caractere 'b' minúsculo, porém se precedido da barra invertida '\b' ele passará a significar a ocorrência do caractere especial [fronteira do caractere](#special-word-boundary).
Quando a barra invertida preceder um caractere especial isso significará que o próximo caractere deve ser interpretado de forma literal. Por exemplo o padrão /a*/, que selecionará a ocorrência de zero ou mais caracteres 'a' quando utilizado sem a \ para escape. Por outro lado no padrão /a*/ o asterisco deixa de ter seu significado especial, pois a '\' de escape fará com que o '*' seja interpretado de forma literal, passando o padrão a selecionar o caractere 'a' seguido do caractere '*'.

Regex

Caractere: ^

Corresponde ao início do texto. Se a flag multilinhas é setada para true, também se aplica imediatamente após um caractere de quebra de linha.
Por exemplo, /^A/ não corresponde ao 'A' em "Um Alvo", mas corresponde ao 'A' em "Alvo Encontrado".

Regex

Caractere: $

Corresponde ao final do texto. Se a flag multilinhas é setada para true, também se aplica imediatamente antes de um caractere de quebra de linha.
Por exemplo, /r$/ não corresponde ao 'r' em "corre", mas acha correspondência em "correr".

Regex

Caractere: *

Corresponde a expressão que o precede repetida 0 ou mais vezes. Equivalente a {0,}
Por exemplo, /bo*/ acha uma correspondência para 'boooo' em "Scoob doo" e 'b' em "A bird warbled", mas nenhuma em "A goat grunted".

Regex

Caractere: +

Corresponde a expressão que o precede repetido 1 ou mais vezes. Equivalente a {1,}. Por exemplo, /a+/ acha correspondência para o 'a' em "candy" e todos os "as" em "caaaaaaandy", mas nâo encontra em "cndy".

Regex

Caractere: ?

Corresponde a expressão que o precede repetido 0 ou 1 vez. Equivalente à {0,1}.
Por exemplo, /e?le?/ encontra o 'el' em "angel" e o 'le' em "angle" e também o 'l' em "oslo".
Se usado imediatamente após qualquer um dos quantificadores *, +, ? ou {}, faz o quantificador não guloso (combinando o número mínimo de vezes), como um oposto para o padrão que é guloso (combinar o número máximo possível). Por exemplo, aplicando /\d+/ em "123abc" encontra "123". Mas aplicando /\d+?/, apenas "1" será encontrado.

Regex

Caractere: .

(O ponto decimal) corresponde com qualquer caracter, exceto o caracter de nova linha.
Por exemplo, /.n/ acha correspondência para o 'an' e 'on' em "nove dias restantes para onze de agosto.", mas não encontra 'no' em 'nove'.

Regex

Caractere: x | y (ou)

Pesquisa correspondência em 'x' ou 'y'.
Por exemplo, /verde|vermelha/ encontra 'verde' em "maçã verde" e 'vermelha' em "maçã vermelha."

Regex

Caractere: { n }

Pesquisa n ocorrências correspondentes ao caracter precedido. Onde, n deve ser um inteiro positivo.
Por exemplo, /a{2}/ não encontra o 'a' em "candy," mas encontra-o se houver a quantidade de a's informarda em "caandy," e os dois primeiros a's em "caaandy."

Regex

Caractere: [xyz]

Um conjunto de caracteres. Pesquisa correspondência para qualquer um dos caracteres entre colchetes. Você pode especificar um intervalo de caracteres usando hífen. Caracteres especiais (como o ponto (.) e o asterisco(*)) não tem significado algum quando está dentro de um conjunto de caracteres. Não necessita utilizar escape neles. Mas, se utilizar escape também irá funcionar. Por exemplo, [abcd] é o mesmo que [a-d]. Com a expressão será encontrado o 'b' em "beijo" e o 'c' em "chop". A expressão /[a-z.]+/ e /[\w.]+/ ambos encontraram as letras que formam "test.i.ng".

Regex

Caractere: [^xyz]

Um conjunto de caracteres negados ou complementados. Isto é, combina com qualquer coisa que não esteja listado entre os colchetes. Você pode especificar um intervalo de caracteres usando hífen. Tudo que funciona no conjunto de caracteres (apresentado acima) também funciona aqui.
Por exemplo, [^abc] é o mesmo que [^a-c]. Com a expressão será encontrado inicialmente 'e' em "beijo" e 'h' em "chop."

Regex

Caractere: [\b]

Pesquisa correspondência com espaço em branco (U+0008). É preciso utilizar os colchetes se você quer encontrar um espaço em branco. (Não confunda-o com \b.)

Regex

Caractere: \b

Pesquisa correspondência em uma fronteira de caractere. Uma fronteira de caractere corresponde a posição onde um caractere/palavra não é seguido ou antecedido por outro caractere/palavra. Isto é, em fronteira de caractere não pode haver nenhum caractere ou espaço, seu tamanho deve ser vazio. (não confunda-o com [\b].)

Exemplos:

/\bmoo/ encontra a substring 'moo' em "moon" ;

/oo\b/ não encontra o 'oo' em "moon", devido o 'oo' ser seguido por 'n' que é um caractere;

/oon\b/ encontra a substring 'oon' em "moon", devido 'oon' ser o fim da string, ou seja, não é seguido por nenhum caractere;

/\w\b\w/ não encontrará nada, pois o caractere nunca será seguido por um não caractere e um caractere.

Regex

Caractere: \B

Pesquisa correspondência que não seja em uma fronteira de caractere. Para a correspondência é associada uma posição onde o caractere anterior e o próximo tem as mesmas características: ambos são caractere/palavra, ou ambos não sejam caractere/palavra. O início e o fim de uma string não considerados como não caractere/palavra.
Por exemplo, /\B../ encontra correspondente 'oo' em "boolean", e /y\B./ encontra correspondente 'ye' em "possibly yesterday."

Regex

Caractere: \d

Encontra correspondência com um número. Equivalente a [0-9].
Por exemplo, /\d/ ou /[0-9]/ encontra correspondente '8' em "Dróide BB8".

Regex

Caractere: \D

Encontra correspondência com um caractere que não seja número. Equivalente a [^0-9].
Por exemplo, /\D/ ou /[^0-9]/ econtra correspondente 'C' em "C3 está ativada."

Regex

Caractere: \w

Encontra correspondência de qualquer caractere alfanumérico incluindo underline. Equivalente a [A-Za-z0-9_].
Por exemplo, /\w/ encontra correspondente 'a' em "apple," '5' em "$5.28," e '3' em "3D."

Regex

Caractere: \W

Encontra correspondência em um não caractere. Equivalente a [^A-Za-z0-9_].
Por exemplo, /\W/ ou /[^A-Za-z0-9_]/ encontra correspondente '%' em "50%."

Regex

Exemplos: CEP

```
function isCEP(cep) {
    const re = /^\d{2}\.?\d{3}-?\d{3}$/;
    return re.test(cep);
}
```

Regex

Exemplos: Email

```
function isEmail(email) {
    // Formato: inicio@meio.fim
    // inicio => mínimo de 1 caractere / menos
    caracteres especiais (w)
    // Deve ter um @
    // Meio => mínimo de 3 caracteres
    // Deve terminar . 2 ou 3 caracteres
    const re = /^\w.+@\w{3}.*\.\w{2,3}$/;
    return re.test(email);
}
```

Regex

Exemplos: Apenas Números

```
function onlyNumbers(text) {
    const re = /^\d+$/;
    return re.test(text);
}
```

Regex

Exemplos: Protocolo (número.ano)

```
function isProtocol(text) { //Exemplo: 123456.2022
    const re = /^\d+\.\d{4}$/;
    return re.test(text);
}
```

Regex
Exercícios

1. CPF com ou sem pontuação
A expressão regular \d{3}\.\d{3}\.\d{3}\/\d{2} casa um CPF como 772.843.809-34. Inclua quantificadores para que a pontuação seja opcional.
A regex resultante deve casar com 77284380934.

2. CPF com pontuação diferente ou espaços
Modifique a solução do exercício 1 para aceitar CPFs escritos com espaços ou pontos entre os grupos de três dígitos, e com hífen - ou barra / antes dos dois dígitos de controle.
A regex resultante deve casar com 772 843 809/34, 772.843.809/34, e continuar casando com as strings aceitas pela solução de 1.1.

3. Âncoras
3.1. Escreva uma regex capaz de encontrar no texto deste parágrafo todas as palavras que terminam com a letra “o”.
3.2. Escreva uma regex capas de encontrar no parágrafo acima todas as palavras que começam e terminam com vogais.

4. Regex para datas
Dia: de 01 a 31, sendo que dias menores que 10 devem ter 0 na frente
Mês: de 01 a 12, sendo que meses menores que 10 devem ter 0 na frente
Ano: 2 ou 4 dígitos

5. Regex para endereço IP
Escreva uma regex que valida um endereço IP. Ex.: 192.168.1.10
Obs.: cada octeto pode receber até 255.