SIMULADOS & TUTORIAIS

TUTORIAIS JAVASCRIPT

O Objeto RegExp

TUTORIAIS JAVASCRIPT

O Objeto RegExp

Introdução

O que é RegExp?

RegExp é uma forma mais curta para **Expressões Regulares**. Uma expressão regular é um objeto que descreve um padrão de caracteres. Quando você procura uma palavra em um texto, você poderá usar um padrão para descrever o que você está procurando. Um padrão simples pode ser um simples caractere. Padrões mais complicados pode consistir de mais caracteres, e pode ser usado para verificar a formatação, substituição de textos, entre outros.

Sua sintaxe é a seguinte:

```
var padrao = new RegExp(pattern,modifiers);

ou mais simplesmente:

var padrao = /pattern/modifiers;
```

Modificadores das Expressões Regulares

Modificadores são usados para executar procuras globais e casos sensitivos. Existem dois tipos de modificadores:

- **Modificador i** – é usado para executar procuras com casos sensitivos.
- **Modificador g** – é usado para executar uma procura global, ou seja, encontrar todas as ocorrências que forem encontradas e não parar na primeira ocorrência achada.

Vejamos um exemplo prático:

Exemplo 1

Nesse exemplo vamos utilizar os modificadores **i** e **g** para mostrar algumas situações de procura em um texto.

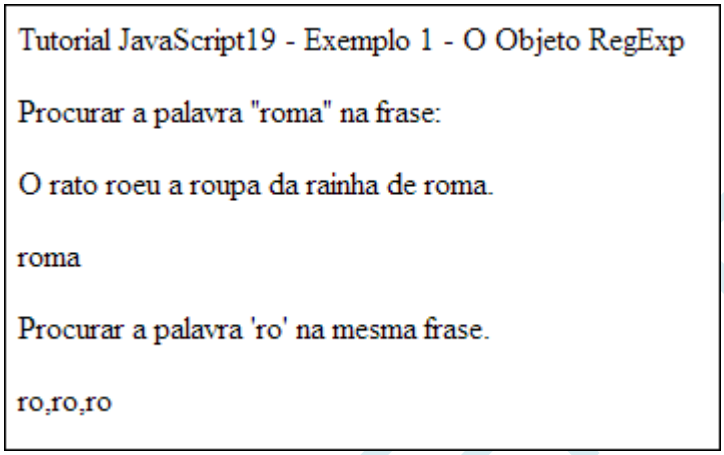
Para isso:

1. Digite o código abaixo no seu editor de texto e salve-o como: **js19-exemplo1.html**.

```
<!DOCTYPE html>
<html>
Tutorial JavaScript19 - Exemplo 1 - O Objeto RegExp
<head>
   <title>Tutorial JavaScript19 - Exemplo 1 - O Objeto RegExp</title><p>
        Procurar a palavra "roma" na frase:<p>
        O rato roeu a roupa da rainha de roma.<p>
<script type="text/javascript">
        // Cria uma variável e atribui uma string a ela.
        var str = "O rato roeu a roupa da rainha de roma";
        var padrao = /roma/i;
```

```
        document.write(str.match(padrao) + "<p>");
        document.write("Procurar a palavra 'ro' na mesma frase.<p>");
        var padrao = /ro/g;
        document.write(str.match(padrao));
</script>
</head>
<body>
</body>
</html>
```

2. O resultado desse código após executado no browser será o seguinte:



Brackets (ou colchetes)

Os colchetes são usados para encontrar uma faixa de caracteres definidos dentro deles. Os utilizados em expressões regulares são os seguintes:

Expressão	Descrição
[abc]	Encontra qualquer caractere que se encontram entre os colchetes.
[^abc]	Encontra qualquer caractere que não seja nenhum que estão dentro dos colchetes.
[0-9]	Encontra qualquer dígito de 0 a 9.
[A-Z]	Encontra qualquer caractere maiúsculo de **A** a **Z**.
[a-z]	Encontra qualquer caractere minúsculo de **a** a **z**.
[A-z]	Encontra qualquer caractere maiúsculo de **A** até o minúsculo de **z**.
[adgk]	Encontra qualquer caractere no conjunto dado.
[^adgk]	Encontra qualquer caractere fora do conjunto dado.
(red|blue|green)	Encontra qualquer das alternativas especificadas.

Vejamos algumas expressões:

A expressão [abc]

Essa expressão é usada para encontrar qualquer caractere entre os colchetes e retorna os caracteres encontrados. Os caracteres dentro dos colchetes podem ser quaisquer caracteres ou faixa de caracteres, por exemplo **[a-d]**.

Sua sintaxe é a seguinte:

```
new RegExp("[abc]")

ou simplesmente:

/[abc]/
```

Vejamos um exemplo prático:

Exemplo 2

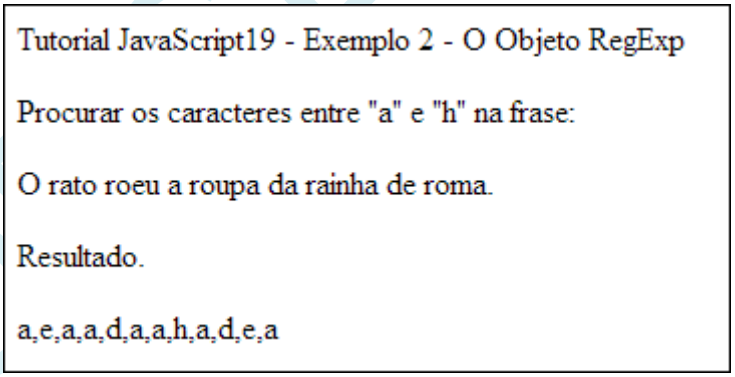
Nesse exemplo vamos utilizar a expressão **[a-h]** para mostrar os caracteres que estão entre os colchetes em uma string.

Para isso:

1. Digite o código abaixo no seu editor de texto e salve-o como: **js19-exemplo2.html**.

```
<!DOCTYPE html>
<html>
Tutorial JavaScript19 - Exemplo 2 - O Objeto RegExp
<head>
        <title>Tutorial JavaScript19 - Exemplo 2 - O Objeto RegExp</title><p>
        Procurar os caracteres entre "a" e "h" na frase:<p>
        O rato roeu a roupa da rainha de roma.<p>
        <script type="text/javascript">
        // Cria uma variável e atribui uma string a ela.
        var str = "O rato roeu a roupa da rainha de roma";
        var padrao = /[a-h]/g;
        document.write("Resultado.<p>");
        document.write(str.match(padrao) + "<p>");
</script>
</head>
<body>
</body>
</html>
```

2. O resultado desse código após executado no browser será o seguinte:



A expressão [^abc]

Essa expressão é usada para encontrar qualquer caractere que não esteja entre os colchetes. Os caracteres dentro dos colchetes podem ser quaisquer caracteres ou extensão de caracteres.

Sua sintaxe é a seguinte:

```
new RegExp("[^abc]")

ou simplesmente:

/[^abc]/
```

Vejamos um exemplo prático:

Exemplo 3

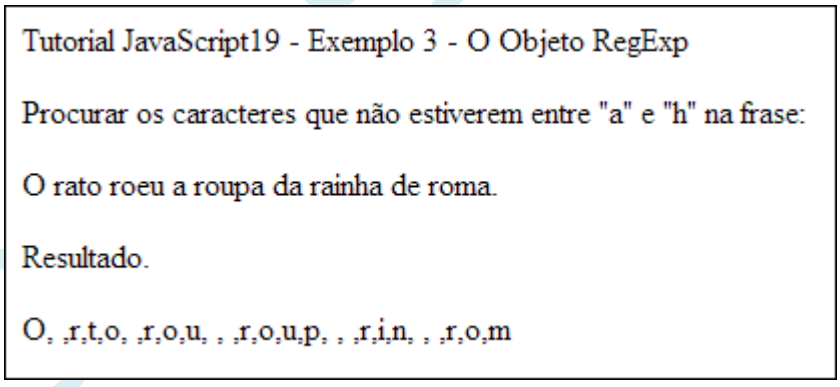
Nesse exemplo vamos utilizar a expressão **[^a-h]** para mostrar os caracteres que não estão entre os colchetes em uma string.

Para isso:

1. Digite o código abaixo no seu editor de texto e salve-o como: **js19-exemplo3.html**.

```
<!DOCTYPE html>
<html>
Tutorial JavaScript19 - Exemplo 3 - O Objeto RegExp
<head>
        <title>Tutorial JavaScript19 - Exemplo 3 - O Objeto RegExp</title><p>
        Procurar os caracteres que não estiverem entre "a" e "h" na frase:<p>
        O rato roeu a roupa da rainha de roma.<p>
        <script type="text/javascript">
        // Cria uma variável e atribui uma string a ela.
        var str = "O rato roeu a roupa da rainha de roma";
        var padrao = /[^a-h]/g;
        document.write("Resultado.<p>");
        document.write(str.match(padrao) + "<p>");
</script>
</head>
<body>
</body>
</html>
```

2. O resultado desse código após executado no browser será o seguinte:



A expressão [0-9]

Essa expressão é usada para encontrar qualquer dígito entre 0 e 9 em uma string dada. Você pode determinar qualquer faixa de números entre esses caracteres (por exemplo: [0-5],[3-8], e assim por diante). O retorno será os caracteres encontrados.

Sua sintaxe é a seguinte:

```
new RegExp("[0-9]")

ou simplesmente:

/[0-9]/
```

Vejamos um exemplo prático:

Exemplo 4

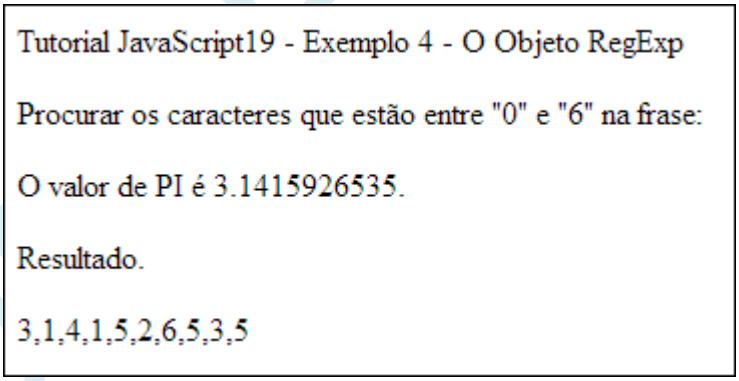
Nesse exemplo vamos utilizar a expressão **[0-6]** para procurar e mostrar os caracteres que estão entre os colchetes em uma string dada.

Para isso:

1. Digite o código abaixo no seu editor de texto e salve-o como: **js19-exemplo4.html**.

```
<!DOCTYPE html>
<html>
Tutorial JavaScript19 - Exemplo 4 - O Objeto RegExp
<head>
        <title>Tutorial JavaScript19 - Exemplo 4 - O Objeto RegExp</title><p>
        Procurar os caracteres que estão entre "0" e "6" na frase:<p>
        O valor de PI é 3.1415926535.<p>
        <script type="text/javascript">
        // Cria uma variável e atribui uma string a ela.
        var str = "O valor de PI é 3.1415926535";
        var padrao = /[0-6]/g;
        document.write("Resultado.<p>");
        document.write(str.match(padrao) + "<p>");
</script>
</head>
<body>
</body>
</html>
```

2. O resultado desse código após executado no browser será o seguinte:



Metacaracteres

Metacaracteres são caracteres com um significado especial. São os seguintes os metacaracteres:

Metacaractere	Descrição
.	Encontra um simples caractere, exceto new line (nova linha) ou final de linha.
\w	Encontra um caractere palavra.
\W	Encontra um caractere não-palavra.
\d	Encontra um dígito.
\D	Encontra um caractere não dígito.
\s	Encontra um caractere espaço em branco.
\S	Encontra um caractere não espaço em branco.
\b	Encontra uma correspondência no início e final de uma palavra.
\B	Encontra uma não correspondência no início e final de uma palavra.
\0	Encontra um caractere nulo.

\n	Encontra um caractere new line (nova linha).
\f	Encontra um caractere form feed.
\r	Encontra um caractere retorno de carro (carriage return).
\t	Encontra um caractere de tabulação (tab).
\v	Encontra um caractere tab vertical.
\xxx	Encontra o caractere especificado por um número octal xxx.
\xdd	Encontra o caractere especificado por um número hexadecimal dd.
\uxxxx	Encontra o caractere Unicode especificado por um número hexadecimal xxxx.

Quantificadores

São os seguintes os quantificadores:

Quantificador	Descrição
n+	Corresponde a qualquer seqüência que contém pelo menos um n.
n*	Corresponde a qualquer seqüência que contém zero ou mais ocorrências de n.
n?	Corresponde a qualquer seqüência que contém zero ou uma ocorrência de n.
n{x}	Corresponde a qualquer string que contém uma seqüência de X de n.
n{x,y}	Corresponde a qualquer string que contém uma seqüência de X e Y de n.
n{x,}	Corresponde a qualquer string que contém uma seqüência de pelo menos X de n.
n$	Corresponde a qualquer string com n no final dela.
^n	Corresponde a qualquer string com n no início dela.
?=n	Corresponde a qualquer string que é seguida por uma string especifica n.
?!n	Corresponde a qualquer string que não é seguida por uma string especifica n.

Propriedades e Métodos

O objeto **RegExp** também dispõe de algumas propriedades e métodos. Vejamos quais são:

Propriedades:

Propriedade	Descrição
global	Especifica se o modificador "g" está setado.
ignoreCase	Especifica se o modificador "i" está setado.
lastIndex	O índice pelo qual deve iniciar a próxima procura.
multiline	Especifica se o modificador "m" está setado.
source	O texto do próximo padrão RegExp.

Métodos:

Método	Descrição
compile()	Compila uma expressão regular.
exec()	Testa se determinada expressão é encontrada em uma string dada. Retorna a primeira ocorrência.
test()	Testa se uma determinada expressão é encontrada em uma string dada. Retorna true ou false.

O método test()

Esse método testa se uma determinada expressão é encontrada em uma string dada. Se for encontrada retorna **true**, caso contrário retorna **false**.

Sua sintaxe é a seguinte:

```
RegExpObject.test(string)
```

Vejamos um exemplo prático:

Exemplo 5

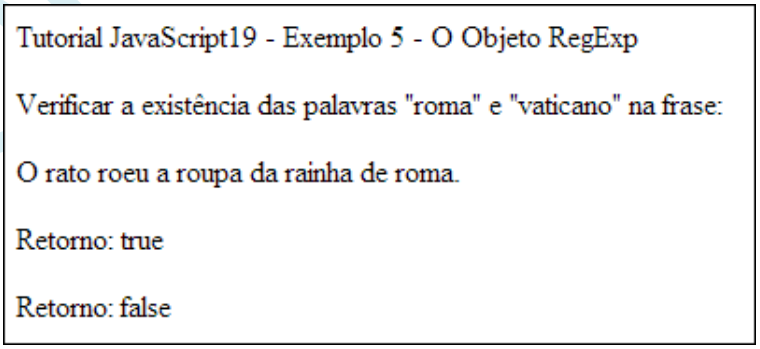
Nesse exemplo vamos utilizar a expressão **[0-6]** para procurar e mostrar os caracteres que estão entre os colchetes em uma string dada.

Para isso:

1. Digite o código abaixo no seu editor de texto e salve-o como: **js19-exemplo5.html**.

```
<!DOCTYPE html>
<html>
Tutorial JavaScript19 - Exemplo 5 - O Objeto RegExp
<head>
        <title>Tutorial JavaScript19 - Exemplo 5 - O Objeto RegExp</title><p>
        Verificar a existência das palavras "roma" e "vaticano" na frase:<p>
        O rato roeu a roupa da rainha de roma.<p>
        <script type="text/javascript">
        var str="O rato roeu a roupa da rainha de roma.";
        // Procura por "roma"
        var padrao = /roma/g;
        var resultado = padrao.test(str);
        document.write("Retorno: " + resultado + "<p>");
        // Procura por "vaticano"
        padrao = /vaticano/g;
        resultado = padrao.test(str);
        document.write("Retorno: " + resultado);
</script>
</head>
<body>
</body>
</html>
```

2. O resultado desse código após executado no browser será o seguinte:



*****

Exercícios de fixação

1) Essa expressão é usada para encontrar qualquer caractere entre os colchetes e retorna os caracteres encontrados.

   a) [abc]
   b) [a...z]
   c) [^abc]
   d) [~abc]

2) O método _____ testa se uma determinada expressão é encontrada em uma string dada.

   a) compile()
   b) exec()
   c) test[]
   d) test()

3) Essa expressão é usada para encontrar qualquer dígito entre 0 e 9 em uma string dada.

   a) [0 to 9]
   b) (0-9)
   c) [0-9]
   d) [0...9]

4) Esse quantificador corresponde a qualquer seqüência que contém zero ou mais ocorrências de n.

   a) n^
   b) n*
   c) ^n
   d) ñ

5) Esse quantificador corresponde a qualquer string com **n** no início dela.

   a) ñ
   b) [^n]
   c) ^n
   d) ~n

6) Encontra qualquer caractere minúsculo de a a z.

   a) (a-z)
   b) [a...z]
   c) (a...z)
   d) [a-z]

7) A expressão _____ encontra qualquer caractere maiúsculo de A até o minúsculo de z.

   a) [A-z]
   b) (A-z)
   c) [A...z]
   d) (A...z)

8) Esse quantificador corresponde a qualquer string com **n** no final dela.

   a) n!
   b) n$

c) $n
d) n*

9) O método _____ testa se determinada expressão é encontrada em uma string dada, e retorna a primeira ocorrência.

a) compile()
b) test()
c) execute()
d) exec()

10) O metacaractere _____ encontra um caractere não dígito.

a) \d
b) \nd
c) \D
d) \ND

*****

Exercícios propostos

1. Crie uma rotina para achar os caracteres minúsculos **b**,**c**,**d**,**e** e **f** da string abaixo e informe quantos caracteres foram encontrados:

```
nome = "Maria das Graças Cabral Florentino da Silva".
```

2. Crie uma rotina para achar os caracteres na string abaixo que não estão no intervalo entre "**m** e **r**":

```
nome = "Alexandre Magno, Rei da Macedônia ".
```

3. Crie uma rotina para encontrar apenas os caracteres maiúsculos na string abaixo:

```
nome = "Alexandre Magno, Rei da Macedônia ".
```

*****