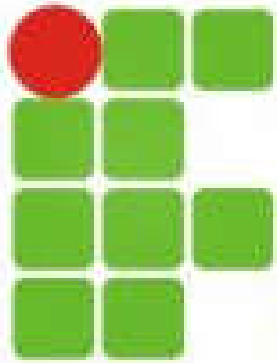
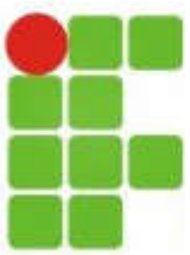
Fundamentos de Web Design 2

Professor Eng. Dr. Will Roger Pereira

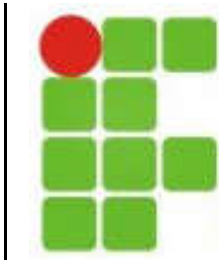


Conteúdo



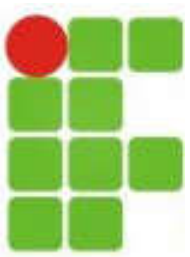
- Expressões regulares.

Introdução



- Expressão regular é, na teoria de linguagens formais, uma sequência de caracteres que define um **padrão de busca**;
- Normalmente utilizado em operações de busca em strings, para verificação de presença ou substituição;
- Ela institui vários conceitos, como metacaracteres, agrupamento, OU booleano e quantificação;
- Utilize com muito cuidado e preste atenção na sua expressão regular, pois um símbolo, e.g. ^, pode ter significados diferentes dependendo do contexto.

Sintaxe em JS



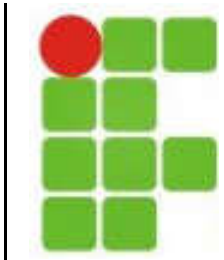
- Em Javascript, uma expressão regular está contida entre barras (/):

/expressão regular/

var exp = **/jovem/**

- A expressão regular exp procura pela string “jovem”.

Operações com Expressões Regulares

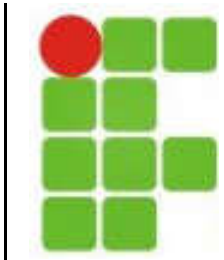


- Para exemplificar, uma expressão regular será chamada de **regexp**, e uma string de **string**;

- Operações:
  - Testar presença;
  - Retornar primeira combinação;
  - Substituir as combinações encontradas em uma string;
  - Dividir uma string em array baseado na regexp;

- **Seja curioso e procure, há várias funcionalidades!!!**

Teste de Presença

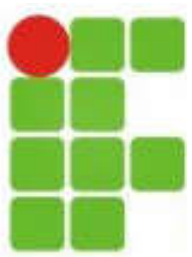


- Testa a expressão regular na string;
- Sintaxe:

```
regexp.test(string)
```

- Retorna **true** caso a expressão regular encontre correspondência na string, e **false** caso contrário;
- Ótimo para verificações em formulários.

Retorno da Primeira Combinação



- Testa a expressão regular na string e retorna a primeira combinação;

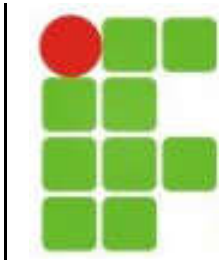
- Sintaxe:

```
regexp.exec(string)
```

- Retorna **uma string,** contendo a primeira combinação, caso se aplique, e **null** caso contrário;

- Ótimo para verificar o que foi combinado através de uma expressão regular mais complexa.

Substituição em uma String



- Testa a expressão regular na string, realiza a substituição da(s) combinação(ões), e retorna a string modificada;

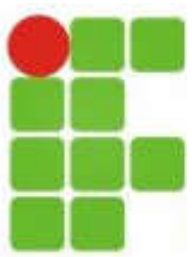
- Sintaxe:

```
string.replace(regexp, novastring)
```

- Nas combinações, novastring irá substituir a string encontrada;

- Caso nenhuma combinação for realizada, a string não será modificada.

Regra Básica



- Foco nas exceções;

- Os metacaracteres podem aparentar ser uma coisa mas podem representar outra completamente diferente;

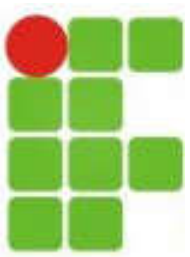
/abc/ → "abc"

/a.c/ ≠ somente "a.c"

/a.c/ → "abc", "a1c", "a=c", "a.c" ... exceto "a\nc"

- E.g., o metacaractere **.** representa qualquer caractere, exceto **\n**.

Letras Maiúsculas e Minúsculas



- As expressões regulares são case-sensitive, i.e., diferenciam letras maiúsculas e minúsculas;

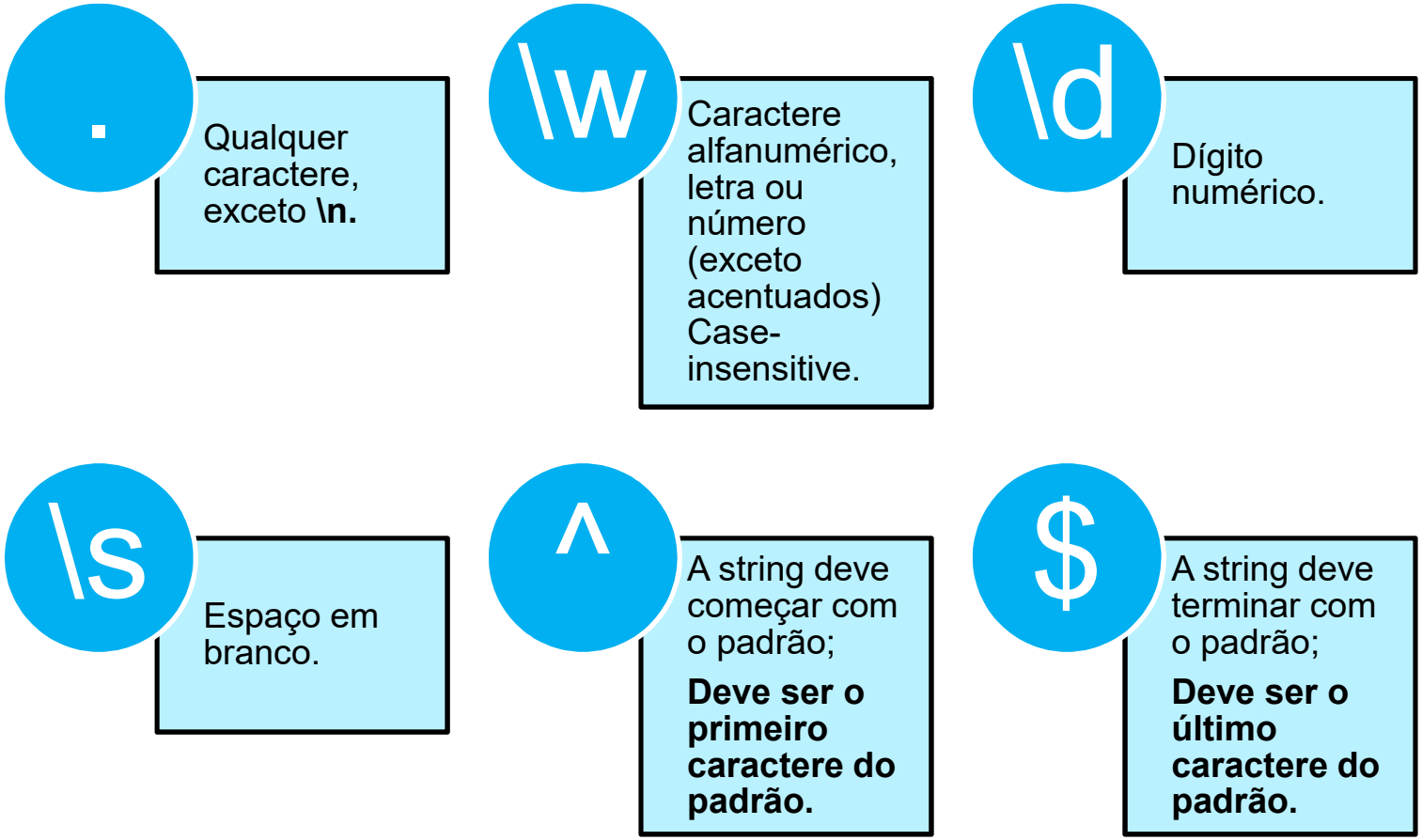
- Em Javascript, caso deseje-se suprimir esta diferenciação, acrescente um **i** após a expressão regular:

/jovem/ → "jovem"

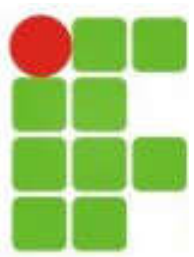
/jovem/i → "jovem", "JOVEM", "JoVeM"

Metacaracteres

- São caracteres que possuem combinação especial:



Exemplos com metacaracteres



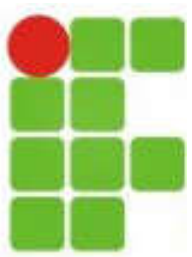
```
/\d\d\d\d\d-\d\d\d/
```

- Indica qualquer informação contendo 5 dígitos, seguido de um hífen (-), e por outros 3 dígitos.

```
/^\d\d\d\d\d-\d\d\d/
```

- Indica qualquer informação **que inicia** por 5 dígitos, seguido de um hífen (-), e por outros 3 dígitos.

Exemplos com metacaracteres



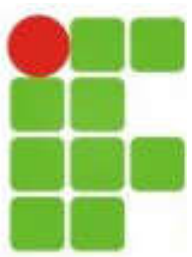
```
/^\d\d\d\d\d-\d\d\d$/
```

- Indica qualquer informação **composta exatamente** por 5 dígitos, seguido de um hífen (-), e por outros 3 dígitos.

```
/^\d\w\w$/
```

- Indica qualquer informação **composta exatamente** por 3 caracteres alfanuméricos, sendo o primeiro um dígito.

Desativando o significado de um metacaractere



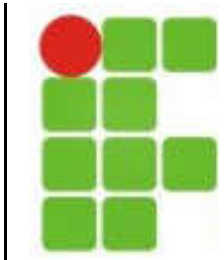
- Há momento que se deseja procurar literalmente um caractere que possui um significado especial, ou seja, é um metacaractere;

- Para suprimir seu significado especial, utiliza-se a barra invertida (\):

/a\.c/ → somente “a.c”

- Isto vale para os vários metacaracteres que vamos aprender.

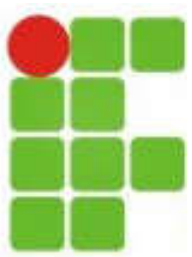
Procurando uma gama de caracteres



- Com expressões regulares, é possível construir um padrão que combine com uma gama de caracteres;
- Na string, quando for encontrado uma ocorrência dentro deste espectro de caracteres, haverá combinação;
- Para isto, utiliza-se os metacaracteres **[** e **]**. Entre os colchetes coloca-se a gama de caracteres desejados no padrão.

/[caracteres]/

Procurando uma gama de caracteres



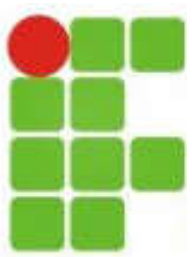
- Busca por uma vogal:

/[aeiou]/ → “a”, “e”, “i”, “o”, “u”

- Suponha que queiramos, entre as letras **X** e **Y**, uma vogal minúscula ou um número, como procederíamos?

/X[aeiou\d]Y/ → “XaY”, “XeY”, “XiY”, “XoY”, “XuY”, “X1Y”, “X2Y”, “X3Y”, “X4Y”, “X5Y”, “X6Y”, “X7Y”, “X8Y”, “X9Y”e “X0Y”.

Buscando uma faixa contínua de caracteres



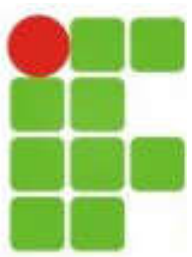
- Referência → Tabela ASCII: 0(48), 9(57).

/[0-9]/ → “0”, “1”, “2”, “3”, ..., “8”, “9”

- Suponha que queiramos, entre as letras **X** e **Y**, um dígito, sem usar metacaractere **\d**, como procederíamos?

/X[0-9]Y/ → “X1Y”, “X2Y”, “X3Y”, “X4Y”, “X5Y”, “X6Y”, “X7Y”, “X8Y”, “X9Y”e “X0Y”.

Buscando uma faixa contínua de caracteres



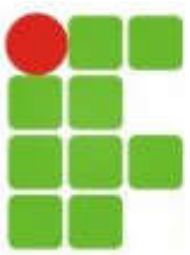
- Referência → Tabela ASCII: A(65), Z(90), a(97), z(122).

/[A-Za-z]/ → Todas as letras não acentuadas

- Suponha que queiramos, entre as letras **X** e **Y**, uma letra minúscula sem acento, sem usar metacaractere **\w**, como procederíamos?

/X[a-z]Y/ → "XaY", "XbY", "XcY", "XdY", "XeY", "XfY",..., "XwY", "XxY", "XyY"e "XzY".

Resolvendo o problema da acentuação



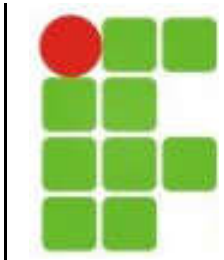
- Referência → Padrão Unicode: \u**(hex)**

00C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
00D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
00E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
00F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

- Busca por qualquer letra, incluindo as da tabela acima:

/[a-zA-Z\u00C0-\u00FF]/

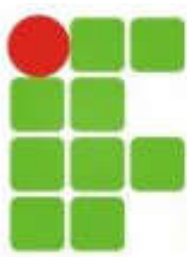
Excluindo uma gama de caracteres



- Além de buscar uma gama de caracteres com expressões regulares, é possível excluir uma gama de caracteres;
- O padrão será combinado quando qualquer caractere, exceto os dispostos no padrão, for encontrado;
- Isto pode ser realizado com o metacaratere **^**, agora como primeiro caractere dentro dos colchetes **[** e **]**.

/[^caracteres não desejados]/

Excluindo uma gama de caracteres



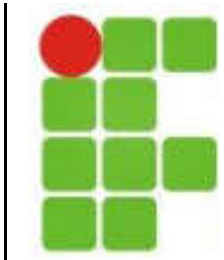
- Busca por qualquer caractere que não seja uma vogal não acentuada:

/[^aeiou]/ → “b”, “X”, “A”, “Ç”, “3”...

- Busca por qualquer caractere fora do alfabeto, que não seja acentuada:

/[^A-Za-z]/ → “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “Á”, “ç”, “õ”, “à”...

Quantificadores

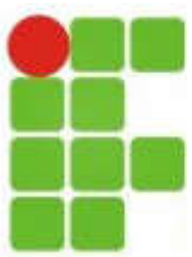


- Até agora, fazemos uma expressão regular para encontrar apenas uma ocorrência, sendo ela obrigatória e única;

- Mas e se quisermos encontrar uma quantidade de combinações do padrão utilizado?

- Isto pode ser realizado através dos quantificadores, que permitem controlar com precisão a quantidade de combinações a ser considerada válida.

Quantificadores

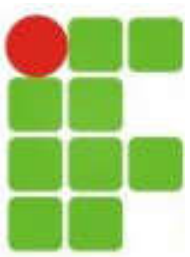


- Os quantificadores devem ser colocados após o padrão desejado, simbolizado por **n**:

Quantificador	Descrição
n+	Combina qualquer string com ao menos um **n**.
n*	Combina qualquer string com 0 ou mais ocorrências de **n.**
n?	Combina qualquer string com 0 ou 1 ocorrência de **n**.
n{X}	Combina qualquer string com exatamente **X** ocorrências de **n.**
n{X,Y}	Combina qualquer string com entre **X** e **Y** ocorrências de **n**.
n{X,}	Combina qualquer string com ao menos X ocorrências de **n**.

- Onde $\{X,Y \in N \mid X < Y\}$

Exemplos com Quantificadores



- Busca por um ou mais dígitos, seguido ou não de letras:

```
/\d+[A-Za-z]*/
```

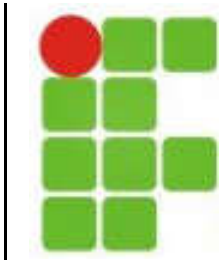
- Busca por um campo com senha que contenha mais de 8 caracteres não acentuados:

```
/^\w{8,}$/
```

- Busca por um campo com uma placa de veículo registrado no Brasil:

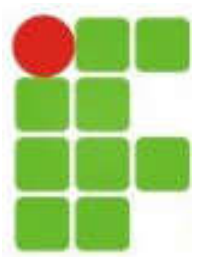
```
/^[A-Z]{3}-\d{4}$/
```

Agrupamento de padrões



- Com quantificadores, abriu-se a possibilidade de combinar repetições de padrões;
- Porém, isto pode ser feito apenas para um caractere, metacaractere ou uma gama/faixa de caracteres;
- E que quiséssemos que um padrão fosse opcional ou obrigatória, dentro do todo?
- Ou se quiséssemos que houvesse múltiplas alternativas de padrões, i. e., que um de múltiplos padrões fosse suficiente para haver a combinação.

Agrupamento de padrões



- Para fazer o agrupamento de padrões, utiliza-se os metacaracteres **(** e **)**. Entre os parênteses o padrão desejado.

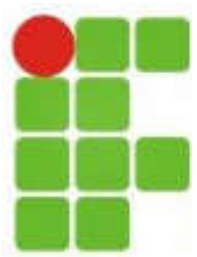
```
/(padrão)/
```

- Como este padrão é agrupado, pode-se utilizar um quantificador para o mesmo:

```
/(padrão)?/
```

- Este exemplo indica que o padrão é opcional.

Alternância de padrões



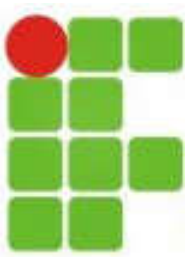
- Dentro do agrupamento, é possível que haja múltiplos padrões, onde qualquer um deles é suficiente para realizar uma combinação;

- Para isto, utiliza-se o metacaractere **|** (barra vertical), separando os padrões, como um OU lógico:

```
/(padrão1|padrão2)/
```

- Neste caso, haja padrão1 ou padrão2 presentes na string, a combinação será realizada.

Exemplos com Agrupadores



- Busca por abc múltiplas vezes:

```
/(abc)+/
```

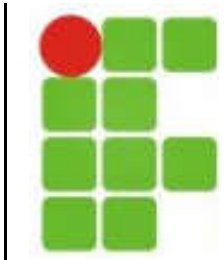
- Busca por um endereço na web simples, seja ele .com ou .com.br:

```
/www\.[a-z]\.com(\.br)?/i
```

- Busca pela palavra jovem ou seu sinônimo:

```
/(jovem|mancebo)/
```

Conclusão



- Com as ferramentas aprendidas aqui, podemos buscar padrões em qualquer string, das mais variadas maneiras possíveis;
- Isto será ótimo em conjunção com a verificação de formulários, onde a personalização será total;
- Aqui foi mostrado somente uma pequena parte do poder das expressões regulares. Estude e se aprofunde no assunto.
- Exemplos em **1-regex.html**