03

[JavaScript] Uso de expressões regulares

Olá, chegou a hora de praticarmos o que aprendemos neste curso com outras linguagens do mercado. No caso do JavaScript, podemos usar o próprio console do navegador para testar nossas expressões.

Abra o console do seu navegador favorito (eu uso Chrome, e você?) e declare a seguinte variável:

```
var target = "11a22b33c";
```

Declaramos a variável `target` que é o alvo, ou seja, o conteúdo no qual aplicaremos a expressão regular que vimos no vídeo.

Em JavaScript, podemos declarar uma expressão regular de suas maneiras. A primeira forma consiste em criarmos uma instância de `RegExp` :

Declare-a em seu console:

```
var exp = new RegExp('(\\d\\d)(\\w)', 'g');
```

Veja que `RegExp` recebe uma string, mas o que a torna verbosa é que precisamos escapar cada `\` colocando um barra extra! Além disso, o segundo parâmetro indica que queremos todas as ocorrências encontradas da nossa expressão, não apenas a primeira que encontrar.

Podemos usar a forma literal, menos verbosa, Nela,colocamos nossa expressão entre `/` :

```
exp = /(\d\d)(\w)/g;
```

Veja que dessa forma não foi necessário colocar, por exemplo, `\\d` , apenas `\d` e nem `\\w` , apenas `w` . E para testar nossa expressão?

```
exp.test(target);
```

Veja que uma expressão regular criada possui o método `test` que recebe o alvo no qual ela será aplicada. Ela retornará `true` apenas se o alvo seguir o padrão da expressão.

Ótimo, mas se quisermos obter como resultado as partes do alvo, que atendem à nossa expressão regular? Nesse caso, usamos o método `exec` :

```
exp.exec(target);
```

Ela imprimirá no console:

```
["11a", "11", "a"]
```

É um array, no qual o primeiro item é o `match`, ou seja, a parte do nosso alvo que condiz com nossa expressão. Contudo, precisamos executar mais uma vez nossa expressão para que ele encontre o próximo `match`. Precisamos fazer até que o resultado final seja `null`, ou seja, quando não houver mais `match`:

```
exp.exec(target);
    ["22b", "22", "b"]
exp.exec(target);
    ["33c", "33", "c"]
exp.exec(target);
    null
```

Excelente. Contudo, você deve estar se perguntando o que são os outros dois elementos do array. O primeiro já sabemos, que é a parte do target que atendeu nossa expressão regular. Os demais parâmetros equivalem ao padrão que colocamos para cada `()` da nossa expressão. Usamos dois `()`, se tivéssemos usando cinco, teríamos no lugar de dois itens, cinco itens.