REGEX &

Javascript

Pattern Matching: RegEX

Javascript supports pattern matching using regular expressions. Javascript uses regular expressions of the Perl programming language:

```
/pattern/modifiers;
```

Modifiers:

Modifier	Description
g	Perform a global match (find all matches rather than stopping after the first match)
i	Perform case-insensitive matching
m	Perform multiline matching

Quantifiers:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{X}	Matches any string that contains a sequence of X n's
n{X,Y}	Matches any string that contains a sequence of X to Y n's
n{X,}	Matches any string that contains a sequence of at least X n's
n$	Matches any string with *n* at the end of it
^n	Matches any string with *n* at the beginning of it
?=n	Matches any string that is followed by a specific string *n* *"beforaea".match(/[a-z]*(?=ae)/g) : "befor"*
?!n	Matches any string that is not followed by a specific string *n*

Predefined character classes:

Class name	Semantics
\d	[0-9]
\D	[^0-9] not a digit
\w	[a-zA-Z0-9]
\W	[^a-zA-Z0-9]
\s	[\r\t\n\f] : A white-space character: a space or a carriage return or a tab or a new line …
\S	[^ \r\t\n\f]

Examples:

Pattern	Description
`[abc]`	Matches with any of the characters a or b or c
`[0-9]`	One digit from 0 to 9
`[0-9]{6}`	Six digits from 0 to 9
`[0-9]{13,16}`	Pattern for credit cards: strings match if they have between 13 to 16 digits
`[a-z]`	One lower case character between a and z
`[a-zA-Z]`	Any upper case or lower case character, from a to z or from A to Z
`[a-zA-Z0-9]`	Any alphanumeric character
`[a-zA-Z0-9]+`	Alphanumeric string of at least one character
`(x|y)`	Matches with pattern x or y
`/\d.\d\d/`	A digit followed by a dot then two digits.
`/\D\d\D/`	A single digit.

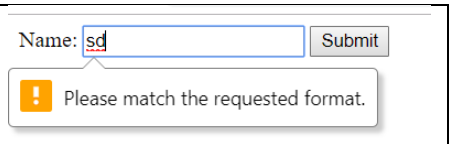
Many ready-to-use patterns can be found at: http://html5pattern.com/

Example: RegEx in HTML5 attribute

Note: in this example, the browser do not inform the user about the accepted format. So you need to add text in front of the field or use javascript validation. Placeholders can be used to inform the user about the pattern but they disappear as soon as the field gets the input focus.

Notice also that an empty string matches also the pattern, so we need to add the "required" constraint to avoid empty strings.

```
<form name="formA" method="post">
   Name: <input type="text" name="name_var"
        pattern="[a-z]{5}">
   <input type="submit" value="Submit">
</form>
```



1. RegEx with Javascript

To combine the power of regular expressions with the freedom of javascript, we can use the RegExp javascript object or the String object:

If you write: `var a = /[a-z]/i;` You obtain a regular expression object RegExp.

The **test**() method is a RegExp expression method. It searches a string for a pattern, and returns true or false, depending on the result.

The **exec**() method tests for a match in a string.This method returns the matched text if it finds a match, otherwise it returns null.

The **string.match()** is a built-in function in JavaScript used to search a string for a match against a regular expression and returns the matches as an array.

REGEX & Javascript

Examples with RegExp object:

javascript	output
`var reg = /abc/;` `console.log(reg.test("before"));`	false
`var reg = /[abc]/;` `console.log(reg.test("before"));`	true
`console.log(/[a-z]{2}[0-9]/.test("he8"));`	true
`console.log(/[a-z]{2}[0-9]/.test(""));`	false
`console.log(/is(?= all)/.test("Is this all ? It is."));`	true
`/(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,}/.test("2sg4dH");`	true
`/(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,}/.test("2s4dH");`	false

Examples with the String object:

javascript	output
`"123456".match(/[^1-4]/g)`	["5", "6"]
`"Is this all there is".search(/is(?= all)/g)`	5 : Position of the first result

Javascript Example:

First, we have the html form:

```
<form action="" onsubmit="return validate()">
  Phone number: <input type="text" id="phone-number"/> <br>
  Postal code:   <input type="text" id="postal-code"/><br>
  <input type="submit" />
</form>
```

Then, we have the validation of the phone number:

```
function validate(){
  var phoneNumber = document.getElementById('phone-number').value;
  var postalCode = document.getElementById('postal-code').value;
  var phoneRGEX = /^[(]{0,1}[0-9]{3}[)]{0,1}[-\s\.]{0,1}[0-9]{3}[-\s\.]{0,1}[0-9]{4}$/;
  var postalRGEX = /^[A-Z]{1,2}[0-9]{1,2} ?[0-9][A-Z]{2}$/i;
  var phoneResult = phoneRGEX.test(phoneNumber);
  var postalResult = postalRGEX.test(postalCode);

  if(!phoneResult){
        alert("The phone number is invalid");
        return false;
  }
  if(!postalResult){
        alert("The postal code is invalid");
        return false;
  }
}
```

Explanation of the RegEx:

The RegEx is between slashes. Then the "^" sign matches with the beginning of the string.

[(]{0,1} : Means that we may have an opening parenthesis.

REGEX & Javascript

[0-9]{3} : Than we have here 3 digits

[)]{0,1} : Than we may have a closing parenthesis.

[-\s\.] : matches a hyphen(-) space or a dot (.)

Accepted inputs examples:
(541) 754-3010
or

541-754-3010
or with spaces

a. More Regular Expression Checks

Date

```
/^\d{2}\/\d{2}\/\d{4}$/
```

This simple regular expression just checks for 2 digits / 2 digits / 4 digits date format
If you want more complex, tight validation for mm/dd/yyyy format, here is the format

```
/^(0[1-9]|1[0-2])\/(0[1-9]|1\d|2\d|3[01])\/(19|20)\d{2}$/
```

URL

A URL of the format http(s)://website/page can be validated with this regular expression:

```
(www\.)?https?:\/\/[a-zA-Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,6}
```

Any Alpha Numeric String

If you want to allow only alphanumeric characters, use this regular expression:

```
/^[a-zA-Z0-9]*$/
```

Decimal Numbers

For decimal numbers with one decimal point, the regular expression is:

```
/^[0-9]+\.?[0-9]*$/
```

END