REGULAR EXPRESSIONS

EXERCISES

home / exercises / regexp

#exercises #regexp #web

REGULAR EXPRESSIONS EXERCISES

1.

Consider the following HTML code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Regular Expressions</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form action="myscript.php">
      Username: <input type="text" pattern="" name="username"
required="required"/>
      <br/>
      <input type="submit" value="Try"/>
    </form>
  </body>
</html>
```

- Copy this code into a new *html* file.
- Notice that there's a *pattern* attribute in the *input* tag.
- This attribute, allows you to define which values for this *input* are acceptable using regular expressions.

2.

Taking advantage of regular expression restrictions defined in HTML 5, make sure the input only allows values that:

- Only have **alphanumeric** characters.
- Have a **minimum** length of 5 and a **maximum** length of 10.
- It should **start** and **end** with an **letter**.

3.

Add a new input field for the user's **real name**. Make sure it only allows **non-numeric** characters that are **not present** in the following list:

```
\|!"@#£$§%&/()=?{[]}'«»*+
```

4.

Add a new input field for the **zip code** using the format **NNNN-NNN** or just **NNNN**.

5.

Add a new input field for an **IPv4 address** that allows values like (**192.168.2.1**) but not those with **leading zeros** (**192.168.02.001**). Remember that each number should be smaller than **256**.

6.

Add a new input field for a **MAC address**. MAC addresses are composed by 6 words of 2 **hexadecimal** digits each separated by either an **hyphen**(-) or a **colon**(:). The separator should be the same for the whole field (i.e. this address should be invalid **A0:B0-C0:D0:E0:F0**. It should have been written like **A0:B0:C0:D0:E0:F0** or **A0-B0-C0-D0-E0-F0**).

7.

Add a new input field for a password (you can use a text type input to make it easier to debug). Make sure it only accepts passwords that:

- Have at least 6 characters.
- Only have **alphanumeric** characters.
- Don't have the same **initial** and **ending** character. **Tip**: use a group to capture the first character and then a negative lookbehind to test the last character.
- Only have **alphanumeric** characters or the following punctuation marks: ;.: Make sure the password has at least **2 digits** and **2 punctuation marks**. **Tip**: Use 2 lookahead assertions.

8.

Add some **Javascript** code that verifies if the **username** is included in the **password**. If it is, the submit button should not work and the password field should get a **red border** to signal that it is invalid.

Tip: Use the *classList.add* and *classList.remove* methods to add an *error* class to the password field and some CSS to draw the red border.

Alternatively you could use Javascript to change the *pattern* in the password field whenever the username changes to include a negative assertion with the username.

9.

Using the same rules, use **Javascript** to validate each field on the fly. This means that fields should be validated as the user types the data instead of only validating them upon submission of the form:

- Validate each field as soon as it loses **focus** (event *onBlur*).
- Validate each field as soon as it **changes** (event *keyUp*).
- Use the same event handler for both events.

10.

Create a *myscript.php* file and verify the format for each input at the server. Print "ok" if every field has the correct values or "error in field $field" if any field has an invalid value.

Copyright © André Restivo