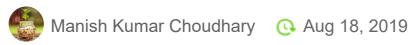






List of Most Common Regular Expressions and Their Uses



■ 163.9k ■ 16 ★ 8 :

Introduction

A regular expression is a pattern that could be matched against an input text. The following is the important list of regular expressions that we use widely in our applications.

- 1. Email id validation
- 2. URL validation
- 3. Password strength validation
- 4. Mobile number validation
- 5. String pattern validation

For using a regular expression in C# server side, we need to use the following namespace.

using System.Text.RegularExpressions;

Remember to remove / at the start and the end of the string to convert a JavaScript Regex to C#.

1. Email id validation regular expression

Regular expression:

- i. $/^w+([-+.']\w+)^*@\w+([-.]\w+)^*.\w+([-.]\w+)^*$ (Email Id)
- ii. /^([\w-\.]+@(?!gmail.com)(?!yahoo.com)(?!hotmail.com)([\w-]+\.)+[\w-]{2,4})?\$/ (free/domain specific email id)

2. URL validation regular expression

Regular expression:

- i. $/(http(s)?://)?([\w-]+\.)+[\w-]+[.com]+(/[/?%&=]*)?/$ (with or without http)
- ii. /((www\.|(http|https|ftp|news|file)+\:\/\)[.a-z0-9-]+\.[a-z0-9\\ :@=.+?,##%&~-]*[^.|\'|\# |!|\(|?|,| |>|

Build stunning real world apps using React Components











Learn more

Regular expression:

- i. / ^[a-z0-9\.@#\\$%&]+\$/ (only contains letter [a-z] digits[0-9], special characters(@#\$%&))
- ii. / ^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}\$/ (Minimum 8 characters at least 1 Alphabet and 1 Number)
- iii. / ^(?=.*[a-z])(?=.*\d)(?=.*\d)(?=.*[\$@\$!%*?&])[A-Za-z\d\$@\$!%*?&]{8,}/ (Minimum 8 characters at least 1 Uppercase Alphabet, 1 Lowercase Alphabet, 1 Number and 1 Special Character)
- iv. / ^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[\$@\$!%*?&])[A-Za-z\d\$@\$!%*?&]{8,10}/ (Minimum 8 and Maximum 10 characters at least 1 Uppercase Alphabet, 1 Lowercase Alphabet, 1 Number and 1 Special Character)
- v. / ^[a-zA-Z0-9\s]{7,16}\$/ (Minimum length 7 and Maximum length 16 Characters allowed [a-z] [A-Z] [0-9])

4. Mobile number validation regular expression

Regular expression:

- i. $/ ((+){0,1}91(s){0,1}(-){0,1}(s){0,1}){0,1}9[0-9](s){0,1}(-){0,1}(s){0,1}[1-9]{1}[0-9]{7}$/ (without +91 or 0)$
- ii. $/^(()+91-?)|0)?[0-9]{10}$/ (with or without +91 or 0)$
- iii. $(()+|00)((d{1,3})[(s-]?)?((d{10}))$ (split the number and the country code)

5. String pattern validation regular expression

Regular expression:

- i. /(?s)^((?!manish).)*\$/ (string contains manish)
- ii. \d/ (at list one digit)
- iii. $/(.)*(\d)(.)* / (contains number)$
- iv. /^\d\$/ (contains only number)
- v. /^\d{11}\$/ (contains only 11 digit number)
- vi. /^[a-zA-Z]+\$/ (contains only letter)
- vii. /^[a-zA-Z0-9]+\$/ (contains only letter and number)

Use of the regular expressions

Use the preceding regular expressions in the following ways.

In the following example, I am showing an email validation in various ways. Just replace the regular

expression and use any of the others to use another validation.

Using JavaScript

```
01.
     <script type="text/javascript">
02.
               function validateEmailId(email) {
                   var reg = regular expression above pattern
03.
04.
                   if (reg.test(email)) {
                       mesg.innerHTML = "";
05.
06.
                       return true;
07.
                   }
                   else {
08.
09.
                       mesg.style.color = "red";
                       mesg.innerHTML = "Please provide a valid email address";
10.
11.
                       return false;
12.
13.
          </script>
14.
```

Call the preceding method like.

Email Address:

Using C# server side

Using a normal function:

ASP.NET

Email Address

C#

```
01.
     private bool validateEmailId(string emailId)
02.
03.
        return Regex.IsMatch
04.
05.
           emailId,
           0''^W+([-+.']^W+)*0^W+([-.]^W+)*.W+([-.]^W+)*
06.
07.
           RegexOptions.IgnoreCase
08.
        );
     }
09.
```

Call the preceding function in a button click.

```
01.
     protected void btnsubmit Click(object sender, EventArgs e)
02.
         if (validateEmailId(txtemail.Text.Trim()))
03.
04.
         {
05.
            lblmsg.Text = string.Empty;
06.
         }
         else
07.
08.
        {
            lblmsg.Text = "Please provide a valid email address";
09.
10.
            lblmsg.ForeColor = System.Drawing.Color.Red;
11.
         }
12.
```

Using RegularExpressionValidator

ASP.NET

Email Address:

Using CustomValidator

ASP.NET

Scripts

Email Address:

```
<script type="text/javascript">
01.
02.
              function validateEmailId(oSrc, args) {
03.
                  if (args.Value > 0) {
                      args.IsValid = (args.Value.match(/^([\w-\.]+)@((\[0-
04.
     9|{1,3}\.[0-9|{1,3}\.[0-9|{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}|[0-9]
     {1,3})(\]?)$/));
05.
06.
              }
07.
          </script>
         Email Address:
08.
09.
          <asp:TextBox ID="txtemail" runat="server" ></asp:TextBox>
          <asp:CustomValidator ID="CustomValidator1" runat="server"</pre>
10.
11.
              ErrorMessage="Please provide a valid email address"
              ClientValidationFunction="validateEmailId" ControlToValidate="t
12.
              Display="Dynamic" ForeColor="Red"
13.
14.
              ToolTip="Please provide a valid email address">Please provide a
```

Change the preceding regular expression in case you need to use another expression.

Use regular expression with Ling:

 $\label{eq:conditional_condition} Regex.Match(hrefValue, @"^([\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}| [0-9]{1,3})(\]?)$"). Success$

Using MVC:

Data Annotations

Suppose we have a student class like follows:

```
01. public partial class tblstudent
02. {
03.    public string Studentname { get; set; }
04.    public string Emailid { get; set; }
05. }
```

We can apply a regular expression like:

We can also extract a word or group of words from a string using a regular expression.

Suppose we want to extract a domain name and user name from an email id, then by using the following method we can do it.

Using C#:

Using JavaScript:

```
<script type="text/javascript">
01.
02.
              function validateEmailId(email) {
03.
                   var reg = /^(\w+([-+.']\w+)*)@(\w+([-.]\w+)*\.\w+
     ([-.]\w+)*)$/;
04.
05.
                   var matches = email.match(reg);
                   UserId.innerHTML ='UserName : '+ matches[1];
06.
                   Domain.innerHTML = 'Domain : '+ matches[3];
07.
08.
         </script>
09.
```

Email Address:

We need to make a group inside an expression using () characters and extract that group value using the preceding process. For checking a regular expression click here.

Summary

In this illustration you came to understand the various types of regular expressions and their uses. Here is a detailed tutorial on C# Regex class and its usage, Top C# Regex Code Examples.

Client Side Regular Expressions

Server side Regular Expressions in ASP.NET.

RECOMMENDED FREE EBOOK



Diving Into ASP.NET WebAPI

Download Now!

SIMILAR ARTICLES

Using Compiled Regular Expressions

Validation in HTML5 Using Regular Expressions

What is a Regular Expression in C#?

Regular Expressions In .NET

Formula Generator - A Regular Expression Generator Class in C# and .NET



Manish Kumar Choudhary 10P 500



Manish Kumar Choudhary is a passionate and pragmatic software engineer specializing in web application development with ASP.NET MVC, Web API, Entity Framework, NHibernet, Angular, Backbone, HTML5, and CSS. He started pro... Read more









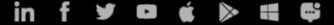




View All Comments



Type your comment here and press Enter Key (Minimum 10 characters)



About Us Contact Us Privacy Policy Terms Media Kit Sitemap Report a Bug FAQ Partners C# Tutorials Common Interview Questions Stories Consultants Ideas Certifications CSharp TV Web3 Universe Build with JavaScript Let's React DB Talks Jumpstart Blockchain Interviews.help ©2024 C# Corner. All contents are copyright of their authors.