### 1. Username Regular Expression Pattern

```
 ^[a-z0-9_-]{3,15}$
```

```
^                   # Start of the line
  [a-z0-9_-]        # Match characters and symbols in the list, a-z, 0-9 ,
underscore , hyphen
            {3,15}  # Length at least 3 characters and maximum length of 15
$                   # End of the line
```

### 2. Password Regular Expression Pattern

```
((?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%]).{6,20})
```

```
(                       # Start of group
  (?=.*\d)              #   must contains one digit from 0-9
  (?=.*[a-z])           #   must contains one lowercase characters
  (?=.*[A-Z])           #   must contains one uppercase characters
  (?=.*[@#$%])          #   must contains one special symbols in the list "@#$%"
              .         #     match anything with previous condition checking
                {6,20}  #       length at least 6 characters and maximum of 20
)                       # End of group
```

### 3. Hexadecimal Color Code Regular Expression Pattern

```
^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$
```

```
^                #start of the line
 #               #  must constains a "#" symbols
 (               #  start of group #1
  [A-Fa-f0-9]{6} #    any strings in the list, with length of 6
  |              #    ..or
  [A-Fa-f0-9]{3} #    any strings in the list, with length of 3
 )               #  end of group #1
$                #end of the line
```

### 4. Email Regular Expression Pattern

```
^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-z0-9]+
(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$
```

```
^                       #start of the line
  [_A-Za-z0-9-]+        #  must start with string in the bracket [ ], must
contains one or more (+)
  (                     #  start of group #1
    \\.[_A-Za-z0-9-]+   #     follow by a dot "." and string in the bracket [ ],
must contains one or more (+)
  )*                    #  end of group #1, this group is optional (*)
    @                   #     must contains a "@" symbol
     [A-Za-z0-9]+       #        follow by string in the bracket [ ], must
contains one or more (+)
      (                 #          start of group #2 - first level TLD checking
       \\.[A-Za-z0-9]+  #            follow by a dot "." and string in the
bracket [ ], must contains one or more (+)
      )*                #          end of group #2, this group is optional (*)
```

```
      (                 #          start of group #3 - second level TLD checking
       \\.[A-Za-z]{2,}  #            follow by a dot "." and string in the
bracket [ ], with minimum length of 2
      )                 #          end of group #3
$                       #end of the line
```

### 5. Image File Extension Regular Expression Pattern

```
([^\s]+(\.(?i)(jpg|png|gif|bmp))$)
```

```
(                       #Start of the group #1
 [^\s]+                 #  must contains one or more anything (except white
space)
       (                #    start of the group #2
         \.             #       follow by a dot "."
         (?i)           #       ignore the case sensitive checking
             (          #         start of the group #3
              jpg       #           contains characters "jpg"
              |         #           ..or
              png       #           contains characters "png"
              |         #           ..or
              gif       #           contains characters "gif"
              |         #           ..or
              bmp       #           contains characters "bmp"
             )          #         end of the group #3
       )                #     end of the group #2
  $                     #  end of the string
)                       #end of the group #1
```

### 6. IP Address Regular Expression Pattern

```
^([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.
([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])$
```

```
^               #start of the line
 (              #  start of group #1
   [01]?\\d\\d? #    Can be one or two digits. If three digits appear, it must
start either 0 or 1
                #    e.g ([0-9], [0-9][0-9],[0-1][0-9][0-9])
    |           #    ...or
   2[0-4]\\d    #    start with 2, follow by 0-4 and end with any digit (2[0-4]
[0-9])
    |           #    ...or
   25[0-5]      #    start with 2, follow by 5 and end with 0-5 (25[0-5])
 )              #  end of group #2
  \.            #  follow by a dot "."
....            # repeat with 3 time (3x)
$               #end of the line
```

### 7. Time Format Regular Expression Pattern

### Time in 12-Hour Format Regular Expression Pattern

```
(1[012]|[1-9]):[0-5][0-9](\\s)?(?i)(am|pm)
```

```
(                           #start of group #1
 1[012]                     #  start with 10, 11, 12
 |                          #  or
 [1-9]                      #  start with 1,2,...9
```

```
)                                 #end of group #1
 :                                #    follow by a semi colon (:)
  [0-5][0-9]                      #   follow by 0..5 and 0..9, which means 00 to
59
            (\\s)?                #        follow by a white space (optional)
                  (?i)            #          next checking is case insensitive
                      (am|pm)     #            follow by am or pm
```

### Time in 24-Hour Format Regular Expression Pattern

```
([01]?[0-9]|2[0-3]):[0-5][0-9]
```

```
(                                 #start of group #1
 [01]?[0-9]                       #  start with 0-9,1-9,00-09,10-19
 |                                #  or
 2[0-3]                           #  start with 20-23
)                                 #end of group #1
 :                                #  follow by a semi colon (:)
  [0-5][0-9]                      #    follow by 0..5 and 0..9, which means 00 to
59
```

### 8. Date Format (dd/mm/yyyy) Regular Expression Pattern

```
(0?[1-9]|[12][0-9]|3[01])/(0?[1-9]|1[012])/((19|20)\\d\\d)
```

```
(                         #start of group #1
 0?[1-9]                  #  01-09 or 1-9
 |                        #  ..or
 [12][0-9]                #  10-19 or 20-29
 |                        #  ..or
 3[01]                    #  30, 31
)                         #end of group #1
  /                       #  follow by a "/"
   (                      #    start of group #2
    0?[1-9]               #       01-09 or 1-9
    |                     #       ..or
    1[012]                #       10,11,12
    )                     #    end of group #2
     /                    #       follow by a "/"
      (                   #         start of group #3
       (19|20)\\d\\d      #           19[0-9][0-9] or 20[0-9][0-9]
       )                  #         end of group #3
```

### 9. HTML tag Regular Expression Pattern

```
<("[^"]*"|'[^']*'|[^'">])*>
```

```
<               #start with opening tag "<"
 (              #   start of group #1
   "[^"]*"      #       only two double quotes are allow - "string"
   |            #       ..or
   '[^']*'      #       only two single quotes are allow - 'string'
   |            #       ..or
   [^'">]       #       cant contains one single quotes, double quotes and ">"
 )              #   end of group #1
 *              # 0 or more
>               #end with closing tag ">"
```

## 10. HTML links Regular Expression Pattern

### HTML A tag Regular Expression Pattern

```
(?i)<a([^>]+)>(.+?)</a>
```

```
(                #start of group #1
 ?i              #  all checking are case insensive
)                #end of group #1
<a               #start with "<a"
  (              #  start of group #2
    [^>]+        #     anything except (">"), at least one character
   )             #  end of group #2
  >              #     follow by ">"
    (.+?)        #       match anything
         </a>    #         end with "</a>
```

### Extract HTML link Regular Expression Pattern

```
\s*(?i)href\s*=\s*(\"([^"]*\")|'[^']*'|([^'">\s]+));
```

```
\s*                         #can start with whitespace
  (?i)                      # all checking are case insensive
     href                   #  follow by "href" word
        \s*=\s*             #   allows spaces on either side of the equal sign,
              (             #    start of group #1
               "([^"]*")    #      only two double quotes are allow - "string"
               |            #      ..or
               '[^']*'      #      only two single quotes are allow - 'string'
               |            #      ..or
               ([^'">]+)    #     cant contains one single / double quotes and
">"
              )             #    end of group #1
```