Autoit Form Generator

Contents table

Disclaimer

This SOFTWARE PRODUCT is provided by El Condor "as is" and "with all faults." El Condor makes no representations or warranties of any kind concerning the safety, suitability, lack of viruses, inaccuracies, typographical errors, or other harmful components of this SOFTWARE PRODUCT. There are inherent dangers in the use of any software, and you are solely responsible for determining whether this SOFTWARE PRODUCT is compatible with your equipment and other software installed on your equipment. You are also solely responsible for the protection of your equipment and backup of your data, and El Condor will not be liable for any damages you may suffer in connection with using, modifying, or distributing this SOFTWARE PRODUCT.

You can use this **SOFTWARE PRODUCT freely, if you would you can credit me in program comment:**

El Condor – CONDOR INFORMATIQUE – Turin

Comments , suggestions and criticisms are welcomed: mail to rossati@libero.it

Conventions

Commands syntax, instructions in programming language and examples are with font `COURIER NEW`. The optional parties of syntactic explanation are contained between `[square parentheses]`, alternatives are separated by `|` and the variable parties are in *`italics`*.

1 AutoIt Form generator

The formGen package is an AutoIt (version v3.3.14.2) function, which allows to build and handle forms; it is sufficiently generalized for a wide use.

1.1 Calling Form Generator

The form is generated calling the function `formGen` which has up to seven parameters: title of the generated form, list of field's description, parent window (optional), callBack function (optional), left and top position (optional) and array of default values (optional):

```
formGen($title,$fieldDescriptions,$GUI,$cbFunction,$left,$top,$default)
```

The fields description are separated by semicolon; the default of `$GUI` (*Graphical User Interfaces*) is `-1` which means the form is generated in a separate window.

A value of `-1` for `$left` and `$top` sets the default position for the form that are `100`, `100` for external GUI and `0`, `0` for internal form.

The possible array `$default` is a bi-dimensional array whit the same format of the array returned by the function `IniReadSection`, this is useful when we are dealing with DataBase.

1.2 Data description

Every field is characterized by a list of attributes which are comma separated and in this order: *Type*, *Field Name*, *Field Label*, *Length*, *Default* value and *Extra* value(s).

In addition to the controls there can be some others information (*Pseudo types*) with different semantics that will be detailed in the paragraphs dedicated to them.

If the list starts with `//` it is a comment which ☞ must also be terminated by semicolon.

1.2.1 Type

The *Type*s are indifferent to the case.

- Buttons:
  - **B** Button;
  - **BL** Button List, every button will have a Label Field as caption;
  - **R** Radio button, a set of Radio buttons;
  - **TB** Toggle button.
- **CKB** check box;
- Combos:
  - **CMB** combo box.
  - **CMT** combo box modifiable, i.e. the user can input text or select the value from the combo.
- Text fields:
  - **C** comment;
  - **DATE** the date in the format `YYYY/MM/DD`;
  - **F, FILE** file;
  - **FOLDER, D** folder;
  - **N** integer numeric field (right aligned);
  - **S** slider is an extension of the standard control: it works also on float number range and inverted direction e.g. the start value can be greater than the end value;
  - **P** password field, the data entered are masked;
  - **T** text field, this is the default if the Type is omitted;

- **TIME** the time in the format `HH:MM:SS`;
- **X** text field handled via callback function;
- **U** not modifiable field.


the possibly commas and semicolons in default and extra field(s) must be codified respectively by `\44` and `\59` or by `mask` function.

1.2.2 Field Name

The *name* of the *field*, which is returned in a dictionary, when the form is closed, with the value associated.

1.2.3 Field Label

The *field label* or the caption of button; if it is omitted the *Field Name* is used.

1.2.4 Length

The *length*, in characters, of the field or the buttons for Type **BL**; for the others type of buttons is ignored. If the length is omitted it is substituted by 20 characters or by the length of the `default` field if it exists.

1.2.5 Default value

This value can be used for Text Fields, Combos, Check boxes, Buttons and CallBack Buttons; it is restored when the `Reset` button is clicked. It is also used to display non modifiable data (Type **U**).

1.2.6 Extra

The *Extra* field contains additional information for some types of controls.

1.2.7 Summary by type

Type	Length	Default	Extra
BL			An item list separated by `|`, a second extra is a possibly name of CallBack function
C	Ignored	The comment	
F,D		if omitted is `@DocumentsCommonDir`	Possible File Filter
CKB		`1` = checked	Possible Description at right of check box
CMB, **CMT**		Value of the couple key, value	An item list separated by `|`
S		Initial value	Start and end value, default is `0 100`
B	Ignored	Value returned	Possible CallBack function
X			CallBack function
`DATE`		Date in the form `YYYY/MM/DD`, if omitted is the actual date	
`TIME`		Time in the form `HH:MM:SS`, if omitted is the actual time	

1.2.7.1 Buttons

Buttons can be used both for take different actions on closing form both for show user caption instead of default `Ok`. The caption can contains `&` in order to be selectable by `ALT + char` key.

A CallBack Button, is a Button where the *extra* field contains the name of the function that must be called, it is useful for retrieve data without closing the form. When clicked, formGen call the function indicated passing the dictionary of data, without destroying the form.

A toggle button (**TB**) can have `On/Off` or `Start/Stop` value, the Extra is a possible CallBack function.

The second extra field of the **BL** type can be a name of a CallBack function, in this case the form is not exited; this can be used to insert one button after a set of fields:

```
...
& "F,File,File to guard,40,C:\citazioni.txt;" _
& "D,Folder,Folder to save,30,C:\Dropbox\backup;" _
& "P,Password,Password,20,,;" _
& "Control,Password,pattern=.{8\44};" _
& "BL,Ok,&Guard,,,Run and start guard,guard;" _
& "C,separator,-;" _
...
```

1.2.7.2 Check box

For checked box insert into *default* field `1`.

The *extra* field can contain a possibly description at right of the check box.

The value returned of check box is `1` if checked else is `4`.

1.2.7.3 Comment

The comment is the **Label** field, the **length** field is ignored, if the **Label** field is one character, this is repeated:

```
C,Field1,*
```

1.2.7.4 Date and Time

DATE is in the form `YYYY/MM/DD`, if omitted is the actual date; **TIME is** in the form `HH:MM:SS`, if omitted is the actual time.

1.2.7.5 Extended

The type X control is a text associated with a button that allows to invoke a function for process the request, for example take a data from a list tree. The function, whose name is in the *extra* field, is called with the handle of the text field and if it return a value different from an empty string, this replaces the previous value.

1.2.7.6 File

The initial Folder (and possibly file name) is in *default* field; if omitted is `@DocumentsCommonDir)`; the *extra* field can contains a file filter e.g. `Images (*.jpg;*.bmp)|Videos (*.avi;*.mpg).`

```
Example: File,MedFile,Media files,25,,Images (*.jpg\59*.bmp)|Videos
(*.avi\59*.mpg)
```

☞ note the mask of semicolons.

1.2.7.7 Folder

The initial Folder is in *default* field; if omitted is `@DocumentsCommonDir)`.

1.2.7.8 Radio buttons

The *length* is the length of the single view; the *Extra* field contains the item list separated by `,`. For get a key instead the description, the item must have the form: `key=value`.

The *default* value can be the key or the value:

```
Rdb,Status,,20,Single,M=Married|S=Single|W=Widow
```

It is possible to have more than one set of radio radio buttons.

1.2.7.9 Slider

The *length* is the length of the unmodifiable text which shows the value of slider.

The *extra* field can contains the start and end values in the form `start end`, e.g. `-5 5`; if omitted the range is `0 100`, if only one value is present, the default value for the second is `100`; the result can have decimals depending on the difference from `start` and `end` value, see the table at right.

`start` can be greater of `end` e.g.:

```
Slider,,15,S,-3,10 -10
```

abs(start - end)	n. decimals
> 99	0
<100 and > 10	1
<10 and > 1	2
<1 and > 0.1	3
...	...

1.2.7.10 Combo boxes and Lists

CMB type is a combo box, **CMT** is a combo box with text associated for insert a value not present in the list, The *extra* field contains the item list separated by `,` (see description in Radio button).

For combos or button List if you would return a code associated to the description, the item has the form: *`key=value`* (without spaces).

Example:

```
Unit,Measure,5,CMT,Kilos,|Kg=Kilos|Lt=Liters|Mc=Cubic Meters|Wh=Watt/hour
```

If the form contains only one combo box the `Ok` , `Reset` and `Cancel` buttons will not be generated.

```
$records = "CMB,Font,Font name,20,,Arial|Courier New|Times New Roman|Verdana"
$parms = formGen("KWIC Index",$records,-1,"handleEvents",100,100)
```

1.2.7.11 Text fields

In the numeric fields (type **N**) for real numbers the extra field must contains `Sn.m` where `S` (or `s`) is optional and is for accept a sign, *`n`* and *`m`* are respectively the digits for integer part and for the decimal part; this is controlled on exit of the control and before submission.

1.3 Pseudo types

1.3.1 After

By defaults the buttons are inserted at the bottom of the form, the pseudo type **After** tell instead where insert a button, the syntax is:

```
After,buttonName,controlName
```

1.3.2 Check or Control

`Control` is used to perform controls on the data. The structure for this command is:

```
[Check|Control],controlName,type[=value][,type[=value]...
```

The possibly `control`(s) are:

Type	Value	Note
`required`	none	controlled by pattern `^\s*$`
`min`	numeric value	
`max`	numeric value	
`mail`	none	controlled by pattern `^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$`
`pattern`	a regular expression	the possibly commas and semicolons must be codified respectively by `\44` and `\59`.

Example (valid if only letters, numbers or - and _, starts with letter or number):

```
$form = "T,Folder,,30;Check,Folder,pattern=^[a-zA-Z0-9]+[a-zA-Z0-9_-]*$"
$aReturn = formGen("Patient Folder",$form,-1)
```

1.3.3 Menu and Submenu

Those *pseudo types* add a menu and sub menu to the form:

```
Menu,name,label
SubMenu,name,label,callbackFunction
```

☞ The `SubMenu` must be after a relative `Menu` item.

The *`callbackFunction`* is called with the data of form; the field `fg_button` contains the name of the sub menu.

☞ in the submenu is not possible create a form (my be when Autoit will supports objects).

1.3.4 Required

This *pseudo type* is used for control if fields has been entered:

```
required,controlName[,controlName[, ...]]
```

1.3.5 Timer

```
timer,name,interval,callbackFunction[,Off]
```

By default the timer is active, unless it is indicated in the fourth value Off (case insensitive).
After `interval` milliseconds the `callbackFunction` function receive the control with the data of form; the field `fg_button` contains The `name` of the timer.

☞ the `timer`(s) are always active, when in `Off` status the callBack function is not invoked.

1.3.6 Tooltip

Tooltip is useful for show short information about a field when the cursor is over the control:

```
tooltip,controlName,tooltip_text
```

1.4 Exiting and values returned

Exit from formGen is by the generated buttons (except the `Reset` button), the *close* window button and the `Escape` key; window *close* button, `Escape` key and `Cancel` button are equivalents.

The function returns a dictionary where the value of the control can be accessed through his name; it contains also the item `fg_button` whose value is the name of the button clicked; if the form is canceled is returned only the item `fg_button` with value `Cancel`.

The value returned for buttons is the value indicated in the *default* field or the *label field*; for Button List (**BL**) is the value associated that is contained in the list in the *extra* field.

1.5 CallBacks

There are some possibilities to interact with the form using the general callbacks or the one associated to the types **X** and **B**, **BL** or **TB**. All functions are called with dictionary of data and the values contained in the form are accessible via its ID (see paragraph 2.3 Retrieve control ID).

formGen works in polling mode that is with `GUIOnEventMode=0;` this means which the events of the parent window aren't handled; if you need to handle also these events, you must use a CallBack function.

The CallBack function is useful for handle controls external at the generated form, e.g. some menu items, and / or for interact with the form depending on the user choice.

1.5.1 General Callback

The possible fourth parameter of `formGen` function is the name of function for handle GUI events: it receive the ID of the control as parameter. The function is called at the start with parameters equal -1, at the close with parameters equal -3 and -2 when data are requested (typically in case of callBacks function, otherwise with the ID of control that has generated the event.

For handle Events of controls external to formGen, they must be declared `Global`; the script below is a sample for access a menu:

```
#Include <Array.au3>
#include <WindowsConstants.au3>
#include "formGen.au3"
$parentWindow = GUICreate("Parent Window", 450, 350, 100, 100)
$helpMenu = GUICtrlCreateMenu("?")
Global $infoItem = GUICtrlCreateMenuItem("&Info", $helpMenu)
Global $seeItems = GUICtrlCreateMenuItem("&See Data", $helpMenu)
AutoItSetOption("GUIResizeMode",$GUI_DOCKALL)
$records = "CMB,Name,,12,Beta,||K1=Alfa|K2=Beta|K3=Delta" _
             & ";BL,Nome2,Ok,7,,K1=Alfa|K2=Beta|K3=Delta;Nome3,ClickMe,7,B,,Gamma"
$resp = formGen("Form handler sample",$records,-1,"checkEvents")
_ArrayDisplay($resp,"Data")
Do
```

```
      $msg = GUIGetMsg()
      checkEvents($msg)
Until $msg = $GUI_EVENT_CLOSE
Func checkEvents($msg)        ; call back
      If $msg = $infoItem Then
            MsgBox(0, "Credits", "Condor Informatique - Turin" _
                        & @CRLF  & "Form Handler Sample" _
                        & @CRLF  & "Program developed with AutoIt3")
      EndIf
EndFunc
```

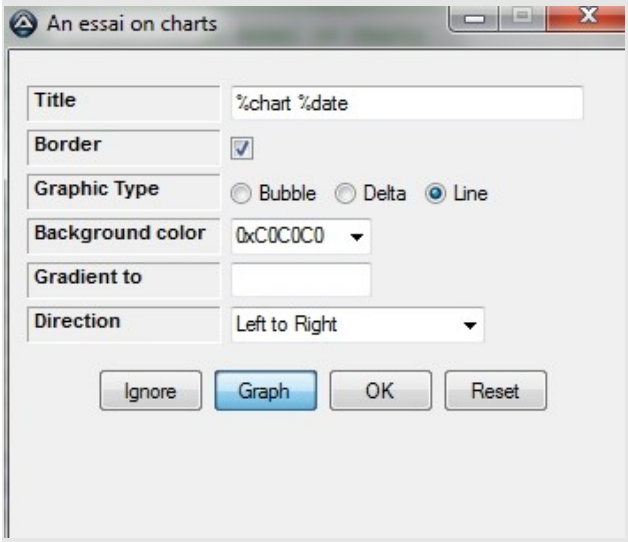
1.5.2 Callback of X control type

The extra field of **X** control type must contain a name of function to be called, this function receive the name of the field and the return the data is inserted in the control text.

1.5.3 Callback of buttons

The extra field of the button can contain a name of function to be called, this function receive the dictionary of the data which also contains the handle of the window containing the form (`fg_winhandle`).

1.6 Example




```
T,Title,,25,%chart %date;
CKB,Border,,10,1;
R,Type,Graphic Type,10,Line,Bubble|
Delta|Line;
B,Graph,,10,Graph,makeGraph;
CMT,BColor,Background
color,10,0xC0C0C0,0xC0C0C0|0xc080c0|
0xa0a0a0;
T,BColorG,Gradient to,10;
CMT,Direction,,18,Left to Right,0=Left
to Right|45=Bottom Left to Top Right|
90=Bottom to Top|135=Top Left to Bottom
Right
```

1.7 Errors

If a type is unrecognized the control is substituted by a comment (type C) showing an alert.

1.8 Remarks

1.8.1 Automatic buttons generation

formGen inserts the `Ok` button, the `Cancel` button and the `Reset` button, this is function of the controls contained in the form:

- the `Cancel` button is always present,
- there is no `Ok`, `Cancel` and `Reset` button if the only control is the a combo box (**CMB** type),
- the `Reset` button is present if there are data fields (e.g. Type F, D, N, P, R, CKB, CMB, CMT, S),
- the `Ok` button is present if there are not buttons (type **B**, **BC** or **BL)**, in other words these replace the button `Ok`.

1.8.2 Data presentation

The data are presented in the order they appear in the parameters list, except for the buttons of Type **B** and **BC** that appear together to buttons inserted by formGen at the bottom of the form.

For widget of Type Text (**T**, **F**, **D**, **U**, **P**) if the length exceed 50 the widget is multi lined.

The order of buttons at the bottom of the form is:
`Ok` button or BC and B buttons, `Reset` button, `Cancel` button.

2 Functions

2.1 Retrieve control name

This function `widgetName($ID)` retrieve the name of control from the ID of control; it is useful for get the value of control.

2.2 Retrieve control value

This function `GetCtrlData($controlData)` retrieve the value of control:

```
Func checkEvents($msg)        ; callback function
   if $msg <= 0 Then Return
   $name = widgetName($msg)
   if $name = "FileName" Then
        msgbox(0,"",GetCtrlData($fhaRecords[$fhDictCRef.Item($name)]))
   EndIf
EndFunc
```

The `id` returned for *field* or *folder* control is the `id` of button (whose name is `fg_name`); for retrieve the value we can use this fragment:

```
$name = $fhDictWID.item($id)
if $name = "fg_Folder" Then
 $value = GUICtrlRead(widgetID("Folder"))
 ...
EndIf
```

2.3 Retrieve control ID

The function `widgetID($fieldname)` return the control ID of widget:

```
If GUICtrlRead(widgetID("Agree")) = 1 Then
```

2.4 Replace data on combo box

This function permits to change the data on combo box:

```
replaceCombo($ctrlName,$data)
```

☞ `$data` must start with the separator character | in order to replace the previous data; the possibly default value is not affected.

2.5 Mask commas and semicolons

If `label` or `default` or `extra` contains commas or semicolons, the function `mask($data)` return a string with that characters encoded; comma is substituted by `\44` and semicolon by `\59`.

2.6 Data for `.ini` files

The function `getIniFormat(dictionay)` returns an array that can be used for `IniWriteSection` function, in order to save the choices for later reuse.

2.7 Manage Timer functions

```
fg_TimerStatus($timer, $status = "")     ;==> set or get the timer status
```

If `$status` is empty or is not provided, the function returns the status of `$timer`, otherwise the status is set (to `On` or `Off`). ☞ `Start` can be a synonym of `On` and `Stop` of `Off`.

```
fg_TimerInterval($timer, $interval = 0)
```

If `$interval` is `0` or is not provided, the function returns the interval otherwise it sets the new interval.

3 Utilities

3.1 Check parameters

The script `formGenCheck.au3` contains the `fhCheck($item)` function which can be used for (some) checking of the correctness of the data description; this is useful when the form is dynamically built.

3.2 Sand Box

The script `sbformGen.au3` help to understand, build parameters and test formGen/formGen.

It contains also `cbFunction` a function for test the callbacks of controls type **BC** and **X**.

4 Technical notes

4.1 GUI controls

The radio buttons have a hidden text for store the value of the radio button checked.

4.2 Data array

`$fhaRecords` is the data array i.e. an array of arrays which contain normalized data description plus 3 items:

- 6 index: handle to label control,
- 7 index: number of decimals for slide,
- 8 index: handle to label that contains the value of the slide.

4.2.1 Scripting Dictionary

From w3schools.com

Property	Description
`CompareMode`	Sets or returns the comparison mode for comparing keys in a Dictionary object
`Count`	Returns the number of key/item pairs in a Dictionary object
`Item`	Sets or returns the value of an item in a Dictionary object
`Key`	Sets a new key value for an existing key value in a Dictionary object

Method	Description
`Add`	Adds a new key/item pair to a Dictionary object
`Exists`	Returns a Boolean value that indicates whether a specified key exists in the Dictionary object
`Items`	Returns an array of all the items in a Dictionary object
`Keys`	Returns an array of all the keys in a Dictionary object
`Remove`	Removes one specified key/item pair from the Dictionary object
`RemoveAll`	Removes all the key/item pairs in the Dictionary object

4.3 Dictionaries

The dictionaries are a COM object used to create a key value access for handle internal activities.

4.3.1 $fhDictCRef

Key	Value	
`Field Name`	`Index on $fhaRecords`	`All widgets`
`Field Name & key name`	`Index on $fhaRecords`	`For Radio button, Lists and button list`

fg_*FieldName*	Index on $fhaRecords	For **F**ile, Fol**D**er and e**X**tended

4.3.2 $fhAfterDict

This global dictionary contains the After references.

4.3.3 $fhControlsDict

This dictionary contains the required controls on fields.

Key	Value
Field Name	$control_1, \ldots, control_n$

4.3.4 $fhDict

The global dictionary `$fhDict` is used for store the values for list, buttons list and radio buttons:

Key	Value
namevalue	key *value*

4.3.5 $fhDictW

Is a global Dictionary which for all Widgets contains:

Key	Value
Field Name	Control ID

4.3.6 $fhDictWID

This dictionary contains for all ID the correspondent name field, normally is one to one; for File, Directory and Extended type there is also a ID of the button associated.

4.3.7 $fg_TimersDict

Key	Value
Timer Name	*Delay*, *CallBack*, *Timer ID*, [On|Off]
Timer ID	Timer Name

4.3.8 $fhToolTipsDict

Key	Value
Field Name	Text for tooltip

4.4 Personalization

Space between controls	$deltaY	
Left margin	$left	
Top margin	$top	
Edit size	$editSize	50
Button width	$buttonWidth	60

5 History

0.2.7	Text multi line when length is greater 50, comma and semicolon accepted in label, default and item list, comments added.
0.3.0	Added Radio Button, added CallBack Button
0.4.0	Red label when error,

	added Comment field, added Slider control
`0.4.1`	Code improvement, some errors corrected
`0.4.2`	Error on `Mask` and `deMask` functions (poor choice of implementation of the `StringReplace` function), return the name of the button pressed (not yet for the Ignore button), aesthetic improvements
`0.5.x`	`Escape` key added, handled default values for type **F** and **D**, added title for form in the father GUI, added type **X** widget, added type **DATE** and **TIME** widgets, correct error on combo when reset button is clicked.
`0.5.4`	Unrecognized widget is substituted by a comment (type C) showing an alert, if the comment is one character, this is repeated to fill a line. The mask of comma and semicolon are changed to `\44` and `\59` respectively.
	Major modification
`0.6.0`	Changed the parameter list: the type is the first parameter. The data are returned in a Scripting Dictionary. Added the pseudo types `control`, `required`, `after` The data list has changed from `key,value|...` to `key=value|...`
`0.6.4.x`	Some bugs amended. Set Focus on first text control
`0.7.2`	Added pseudo type `Menu` and `SubMenu` The data returned contain the window ID Added pseudo type `Timer` Added the support of CallBack function to **BL** type In the **File** type if there is a default value, the starting folder is set from this
`07.3`	Support of toggle button

6 My be in future

- Add Hidden Fields,
- handle CR and LF for Type **U**,
- some personalization e.g. event handling.

7 Annexes

7.1 Introduction to regular expressions

A regular expression is a string of characters used to search, check, extract part of text in a text; it has a cryptic syntax and here there is a sketch with a few examples.
The regular expression can be prefixed by modifiers such as **(?i)** to ignore the case.
The expression is formed with the characters to search in the text and control characters, among the latter there is a \ said *escape* used to introduce the control characters or categories of characters:

- **\ escape character**, for special characters (for example asterisk) or categories of characters:
  - **\w** any alphabetical and numerical character, **\W** any non alphabetical and numerical character,
  - **\s** *white space* namely. tabulation, line feed, form feed, carriage return, and space,
  - **\d** any numeric digits, **\D** any non digit,
- **.** any character,
- **quantifiers**, they apply to the character(s) that precede:
  - ***** zero or more characters
  - **+** one or more characters
  - **?** zero or one character (means possibly)
  - **{*n*}, {*n*,}** and **{*n*,*m*}** respective exactly *n* characters, almost *n* characters and from *n* to *m* characters.

(...) what is between parentheses is memorized,
?=*pattern* checks if *pattern* exists,
[a-z] any letter from a to z included,
[*a*|*b*] a or b,
\b word boundary,
$ (at the bottom),
^ (at start).

7.1.1 Examples

^\s*$	Empty set or white spaces
aa+	Find a sequence of two or more a, like aa, aaa,....
(\w+)\s+(\w+)\s+(\w+)	Find and memorize three words
(\-[a-z])	Find and memorize minus followed by one alphabetic character
.(jpg|jpeg)$	Controls file type jpg or jpeg
^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$	Control of mail address
^\d+$	Only integers
((?=.*\d)(?=.*[a-z]+)(?=.*[\W]).{6,12})	(?=.*\d) almost a digit from 0-9 (?=.*[a-z]) almost one lowercase character (?=.*[\W]+) almost one special character . match anything with previous condition checking {6,12} length at least 8 characters and maximum 12
^[-+]?\d{1,2}(\.\d{1,2})?$	**Numeric values** [-+]? the sign is possible \d{1,2} one or two digits (\.\d{1,2})? It is possible to have a decimal point followed by one or two digits
(?=.*\d)(?=.*[A-Z])(?=.*[a-z]).{6,12};	At most one digit, one capital letter, one minuscule and from 6 to 12 characters

7.1 Some Unicode symbols

edit	\270E
delete	\2718
save	\2714
email	\2709
cross	\2716
dollar	\0024
euro	\20AC
pound	\00A3
cent	\20B5