

PSCalendar

v2.3.2

This module contains a few functions for displaying a calendar in the PowerShell console. The primary function is based on code originally published by Lee Holmes at <http://www.leeholmes.com/blog/2008/12/03/showing-calendars-in-your-oof-messages/>. However, **v2.0.0** of this module contains a complete rewrite of the core functions.

After installing the module, you can view a local PDF version of this file by running `Show-PSCalendarHelp`.

Installation

You can install this module from the PowerShell Gallery.

```
Install-Module PSCalendar [-scope currentuser]
```

Installing this module will also install the `ThreadJob` module from the PowerShell Gallery, as that is module dependency if you want to use `Show-GuiCalendar`.

The commands in this module have been tested on PowerShell 7 both under Windows and Linux and there is no reason these commands should not work. Commands and aliases that are incompatible with non-Windows platforms are not exported.

Note: If you are upgrading to v2.0.0 or later of this module, and have older versions installed, it is recommended that you uninstall the older versions.

Get-Calendar

The commands in this module have been updated to take advantage ANSI escape sequences. The main function, `Get-Calendar`, will display the current month in the console, highlighting the current date with an ANSI escape sequence.

```
PS C:\> Get-Calendar
```

September 2020

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3

But you can also specify a calendar by month and year.

```
PS C:\> Get-Calendar -Month December -Year 2020 -HighlightDate 12/25/20,12/24/20,12/31/20,12/4/20
```

December 2020

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

In this example you can see that I specified dates to highlight. Or you can specify a range of months.

```
PS C:\> Get-Calendar -Start 1/1/2021 -End 3/1/2021
```

January 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

February 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	1	2	3	4	5	6

March 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3

The function should be culturally aware. The commands in this module that have a `-Month` parameter should autocomplete to culture-specific month names.

Show-Calendar

```
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\> $pos = [system.management.automation.host.coordinates]::new(72, 0)
PS C:\> $h = "9/4/2020","9/15/2020","9/16/2020"
PS C:\> Show-Calendar -Position $pos -HighlightDate $h
PS C:\>
```

September 2020						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3

```
#requires -modules @{ModuleName="PSCalendar";ModuleVersion="2.1.0"}
```

```

Function prompt {

    #define a buffercell fill
    $fill = [system.management.automation.host.buffercell]::new(
    ",$host.ui.RawUI.BackgroundColor,$host.ui.RawUI.BackgroundColor,"complete")

    #define a rectangle with an upper left corner X distance from the edge
    $left = $host.ui.RawUI.WindowSize.width - 42

    #need to adjust positioning based on buffer size of the console
    #is the cursor beyond the window size, ie have we scrolled down?
    if ($host.UI.RawUI.CursorPosition.Y -gt $host.UI.RawUI.WindowSize.Height) {
        $top = $host.ui.RawUI.CursorPosition.Y - $host.UI.RawUI.WindowSize.Height
    }
    else {
        $top = 0
    }

    # System.Management.Automation.Host.Rectangle new(int left, int top, int right,
    int bottom)
    $r = [System.Management.Automation.Host.Rectangle]::new($left, 0,
    $host.ui.rawui.windowSize.width,$top+10)

    #clear the area for the calendar display
    $host.ui.rawui.SetBufferContents($r,$fill)

    #show the calendar in the upper right corner of the console
    $pos = [system.management.automation.host.coordinates]::new($left,0)
    Show-Calendar -Position $pos

    "PS $($ExecutionContext.SessionState.Path.CurrentLocation)$('>' *
    ($nestedPromptLevel + 1)) ";

    # .Link
    # https://go.microsoft.com/fwlink/?LinkID=225750
    # .ExternalHelp System.Management.Automation.dll-help.xml

}

```

Assuming the width of your console is at least 120, this code should work. Otherwise, you might need to tweak the positioning. This should also work in Windows Terminal. If you add some highlighted dates using `$PSDefaultParameterValues`, then you'll have a calendar right in front of you.

```

Administrator: Windows PowerShell 5.1.18362
PS C:\> $psversiontable

Name                           Value
----                           -
PSVersion                      5.1.18362.145
PSEdition                     Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.18362.145
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1

PS C:\> get-service bits,winrm

Status  Name      DisplayName
-----
Stopped bits      Background Intelligent Transfer Ser...
Running winrm     Windows Remote Management (WS-Manag...

PS C:\>

```

Note that any command output may be truncated because of the calendar display. This prompt function works as expected when using the Windows Terminal. Function needs work to behave as expected in a traditional PowerShell console where you might have a large buffer for scrolling.

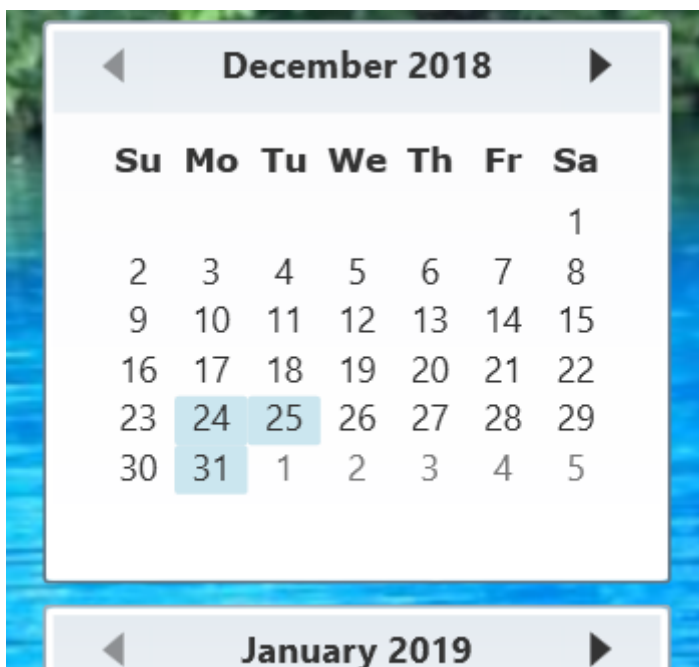
Show-GUICalendar

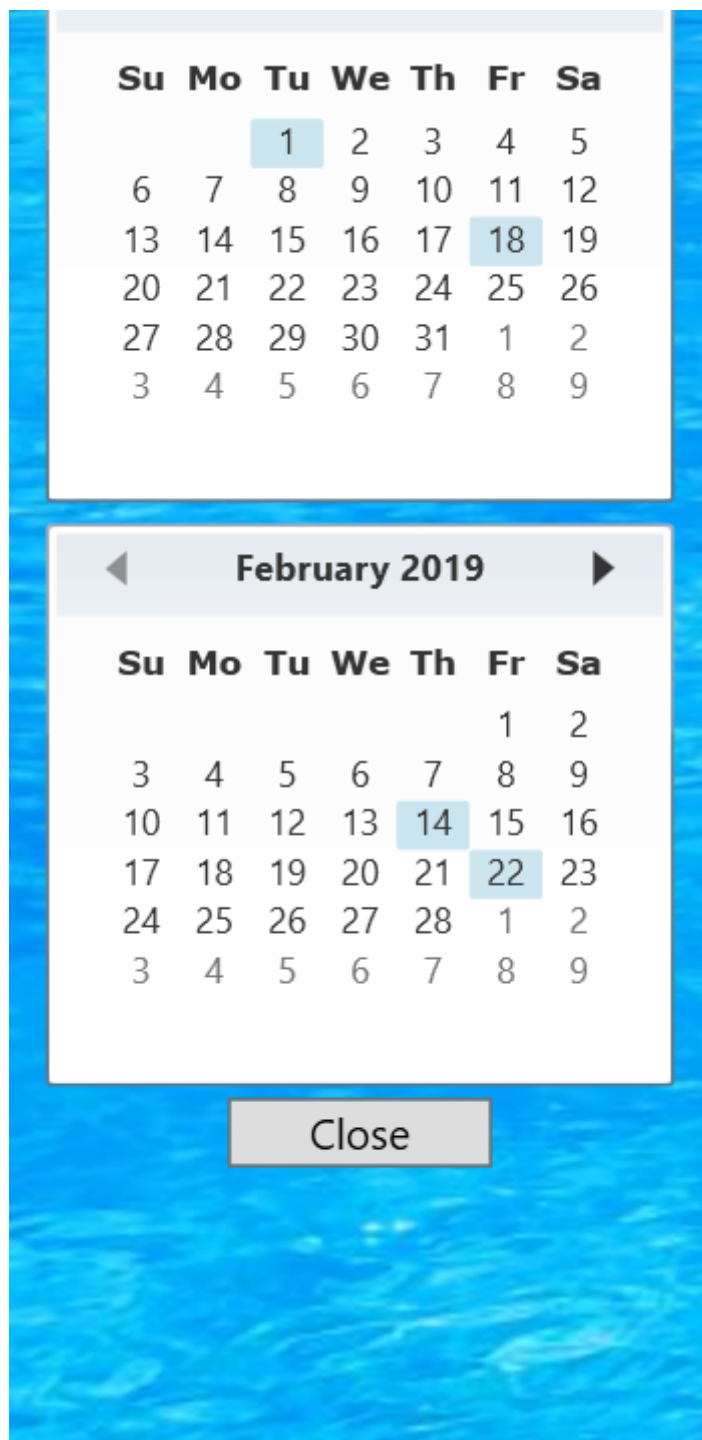
Finally, you can display a graphical calendar using a Windows Presentation Foundation (WPF) based script. The function runs the calendar-related code in a runspace so it does not block your prompt. You can display up to 3 months and specify dates to highlight.

```

PS C:\> Show-GuiCalendar 12/2018 2/2019 -highlight
12/24/18,12/25/18,12/31/18,1/1/19,1/18/19,2/14/19,2/22/19

```

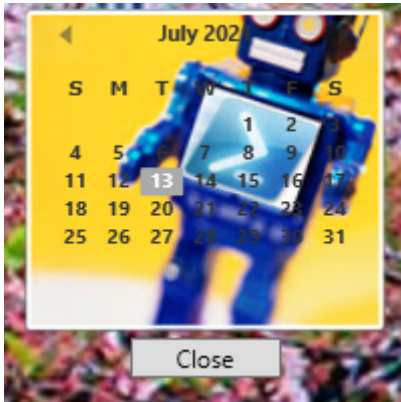




The calendar form is transparent. But you should be able to click on it to drag it around your screen. You can also use the `+` and `-` keys to increase or decrease the calendar's opacity. Be aware that if you close the PowerShell session that launched the calendar, the calendar too will close.

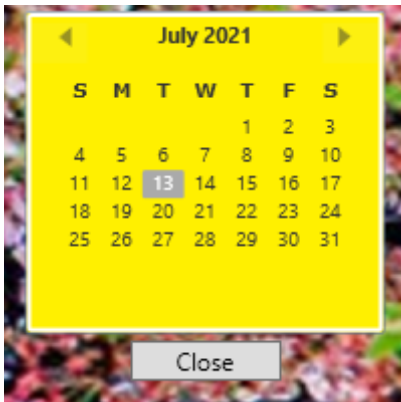
Beginning with module version 2.2.0 you can also customize the calendar background with an image:

```
Show-GuiCalendar -BackgroundImage D:\images\blue-robot-ps-thumb.jpg -Stretch
UniformToFill -FontWeight Bold
```



Or you can specify a color. You can specify a WPF brush color like Cornsilk or Wheat, or use a color code like #FFF000:

```
Show-GuiCalendar -BackgroundColor "#FFF000"
```



On Windows platforms, the `-BackgroundColor` parameter will autocomplete the available brush colors.

Get-NCalendar

The Linux world has an `ncal` command which displays the month in a vertical fashion. `Get-NCalendar` and its alias `ncal` work in a similar manner. The default is for the current month and year.


```
PS C:\> ncal
          July 2021
Sun       4 11 18 25
Mon       5 12 19 26
Tue       6 13 20 27
Wed       7 14 21 28
Thu      1  8 15 22 29
Fri      2  9 16 23 30
Sat      3 10 17 24 31
```

The current date will be highlighted unless you use `-HideHighlight`. You must use the full month name, although there is tab completion.

```
PS C:\> ncal January 2022
          January 2022
Sun    2  9 16 23 30
Mon    3 10 17 24 31
Tue     4 11 18 25
Wed     5 12 19 26
Thu     6 13 20 27
Fri     7 14 21 28
Sat    1  8 15 22 29
```

This command does not support date highlighting. See below.

Get-MonthName

This simple command will list the full month names for the current culture.

```
PS C:\> Get-MonthName
January
February
March
April
May
June
July
```

```

August
September
October
November
December

```

You might use this to build a larger `nca1` listing.

```
PS C:\> Get-MonthName | Select-Object -first 3 | Get-NCalendar -Year 2022
```

```
January 2022
```

```

Sun   2  9 16 23 30
Mon   3 10 17 24 31
Tue    4 11 18 25
Wed    5 12 19 26
Thu    6 13 20 27
Fri    7 14 21 28
Sat    1  8 15 22 29

```

```
February 2022
```

```

Sun    6 13 20 27
Mon    7 14 21 28
Tue    1  8 15 22
Wed    2  9 16 23
Thu    3 10 17 24
Fri    4 11 18 25
Sat    5 12 19 26

```

```
March 2022
```

```

Sun    6 13 20 27
Mon    7 14 21 28
Tue    1  8 15 22 29
Wed    2  9 16 23 30
Thu    3 10 17 24 31
Fri    4 11 18 25
Sat    5 12 19 26

```

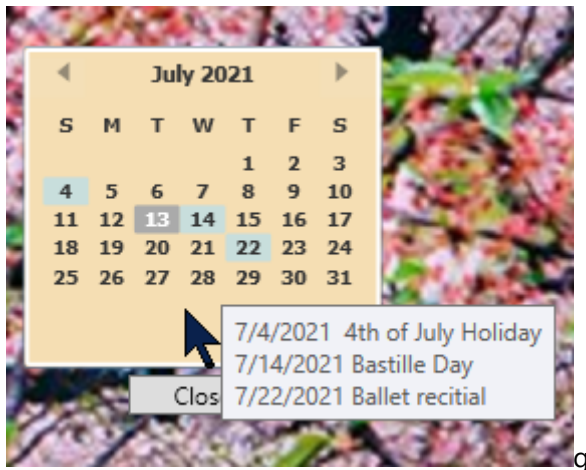
Highlight Dates with Notes

Beginning with v2.2.0, in addition to specifying an array of dates to highlight, you can also use a hashtable. The key should be the highlight date, and the value a brief description.

```
$h = @{"7/4/2021"="4th of July Holiday";"7/14/2021"="Bastille
Day";"7/22/2021"="Ballet recital"}
```

```
Show-GuiCalendar -Start 7/1/2021 -HighlightDate $h -BackgroundColor wheat -
FontWeight Bold -Font Tahoma
```

When you pass a hashtable, you will get a tooltip popup when you hover the mouse over the month.



This function requires the WPF-related assemblies. It should work in Windows PowerShell and PowerShell 7. You will receive a warning if any incompatibility is detected.

Customizing the Calendar Appearance

Beginning with v2.0.0 of this module, ANSI escape sequences used to format the calendar are stored in module-scoped hashtable. You can use [Get-PSCalendarConfiguration](#) to view the current settings.

```
PS C:\> Get-PSCalendarConfiguration

Title       : `e[38;5;3m
Dayofweek   : `e[1;4;36m
Today       : `e[91m
Highlight   : `e[92m
```

The output will show you the escape sequence appropriate for your PowerShell version. If you want to change a setting, you can use:

[Set-PSCalendarConfiguration](#)

You need to include the escape character but you do not need to include the closing escape sequence.

```
PS C:\> Set-PSCalendarConfiguration -title "`e[48;5;57m"
PS C:\> Get-PSCalendarConfiguration
```

```
Title       : `e[48;5;57m
DayofWeek   : `e[1;4;36m
Today       : `e[91m
Highlight   : `e[92m
```

```
PS C:\> Get-Calendar
```

September 2020

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3

This change lasts for the duration of your PowerShell session. If you want to make it more permanent, you will need to add the commands to your PowerShell profile script.

A Note on Culture

I've tried very hard to make the commands respect culture. Most commands now that string values to represent dates which are then treated as dates internally. For this reason, it is important that you follow the culture-specific short date format that you get from running this command:

```
(Get-Culture).datetimeformat.ShortDatePattern
```

In Windows PowerShell, all of the commands appear to respect culture settings. However, when running in PowerShell 7 there appears to be a bug in .NET Core and how it returns culture information for some cultures, specifically the first day of the week. If you run `Get-Calendar` or `Show-Calendar` and the week begins on the wrong day, use the `FirstDay` parameter to override the detected .NET values with the correct one.

```
PS C:\> Get-Calendar august -firstday Monday -highlight 1/8/2021,15,8,2021
```

August 2021

Mon	Tue	Wed	Thu	Fri	Sat	Sun
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15

```
16    17    18    19    20    21    22
23    24    25    26    27    28    29
30    31     1     2     3     4
```

For example, if you are running under the `en-AU` culture, you would need to use this syntax.

Potential Issues

I have tried to make this module culture-aware. Testing across cultures is not an easy process. If you encounter a problem and are not running PowerShell under the `EN-US` culture, run the calendar command you are trying to use with `-Verbose` and post the results in a new issue. Or if you have both Windows PowerShell and PowerShell 7 installed, try the same command in both versions.

Last Updated 2021-11-02 18:25:51Z