PowerShell Get-Help on command 'Write-Host'

PS C:\Users\wahid> Get-Help Write-Host

```
NAME
    Write-Host

SYNOPSIS
    Writes customized output to a host.


SYNTAX
    Write-Host [[-Object] <System.Object>] [-BackgroundColor {Black | DarkBlue |
    DarkGreen | DarkCyan | DarkRed | DarkMagenta | DarkYellow | Gray | DarkGray |
    Blue | Green | Cyan | Red | Magenta | Yellow | White}] [-ForegroundColor
    {Black | DarkBlue | DarkGreen | DarkCyan | DarkRed | DarkMagenta | DarkYellow
    | Gray | DarkGray | Blue | Green | Cyan | Red | Magenta | Yellow | White}]
    [-NoNewline] [-Separator <System.Object>] [<CommonParameters>]


DESCRIPTION
    The `Write-Host` cmdlet's primary purpose is to produce
    for-(host)-display-only output, such as printing colored text like when
    prompting the user for input in conjunction with Read-Host (Read-Host.md).
    `Write-Host` uses the [ToString()](/dotnet/api/system.object.tostring)method
```

to write the output. By contrast, to output data to the pipeline, use
Write-Output (Write-Output.md)or implicit output.

You can specify the color of text by using the `ForegroundColor` parameter, and you can specify the background color by using the `BackgroundColor` parameter. The Separator parameter lets you specify a string to use to separate displayed objects. The particular result depends on the program that is hosting PowerShell.

> [!NOTE] > Starting in Windows PowerShell 5.0, `Write-Host` is a wrapper for `Write-Information` This allows > you to use `Write-Host` to emit output to the information stream. This enables the capture or > suppression of data written using `Write-Host` while preserving backwards compatibility. > > The `$InformationPreference` preference variable and `InformationAction` common parameter do not > affect `Write-Host` messages. The exception to this rule is `-InformationAction Ignore`, which > effectively suppresses `Write-Host` output. (see "Example 5")

PARAMETERS

-BackgroundColor <System.ConsoleColor>

Specifies the background color. There is no default. The acceptable values for this parameter are:

- `Black`

- `DarkBlue`

- `DarkGreen`

- `DarkCyan`

- `DarkRed`

- `DarkMagenta`

- `DarkYellow`

- `Gray`

- `DarkGray`

- `Blue`

- `Green`

- `Cyan`

- `Red`

- `Magenta`

- `Yellow`

- `White`

-ForegroundColor <System.ConsoleColor>

Specifies the text color. There is no default. The acceptable values for this parameter are:

- `Black`

- `DarkBlue`

- `DarkGreen`

- `DarkCyan`

- `DarkRed`

- `DarkMagenta`

- `DarkYellow`

- `Gray`

- `DarkGray`

- `Blue`

- `Green`

- `Cyan`

- `Red`

- `Magenta`

- `Yellow`

- `White`

-NoNewline <System.Management.Automation.SwitchParameter>

The string representations of the input objects are concatenated to form the output. No spaces or newlines are inserted between the output strings. No newline is added after the last output string.

-Object <System.Object>

Objects to display in the host.

-Separator <System.Object>

    Specifies a separator string to insert between objects displayed by the
    host.

<CommonParameters>

    This cmdlet supports the common parameters: Verbose, Debug,
    ErrorAction, ErrorVariable, WarningAction, WarningVariable,
    OutBuffer, PipelineVariable, and OutVariable. For more information, see
    about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

-- Example 1: Write to the console without adding a new line --

```
Write-Host "no newline test " -NoNewline
Write-Host "second string"
```

```
no newline test second string
```

This command displays the string 'no newline test' with the `NoNewline`
parameter.

A second string is written, but it ends up on the same line as the first due
to the absence of a newline separating the strings.

--- Example 2: Write to the console and include a separator ---

```
Write-Host (2,4,6,8,10,12) -Separator ", +2= "
```

```
2, +2= 4, +2= 6, +2= 8, +2= 10, +2= 12
```

This command displays the even numbers from two through twelve. The Separator
parameter is used to add the string `, +2= ` (comma, space, `+`, `2`, `=`,
space).

-- Example 3: Write with different text and background colors --

```
Write-Host (2,4,6,8,10,12) -Separator ", -> " -ForegroundColor DarkGreen
-BackgroundColor White
```

```
2, -> 4, -> 6, -> 8, -> 10, -> 12
```

This command displays the even numbers from two through twelve. It uses the `ForegroundColor` parameter to output dark green text and the `BackgroundColor` parameter to display a white background.

-- Example 4: Write with different text and background colors --

```
Write-Host "Red on white text." -ForegroundColor red -BackgroundColor white
```

```
Red on white text.
```

This command displays the string "Red on white text." The text is red, as defined by the `ForegroundColor` parameter. The background is white, as defined by the `BackgroundColor` parameter.

---------- Example 5: Suppress output from Write-Host ----------

```
# The following two statements can be used to effectively suppress output from
Write-Host
Write-Host "I won't print" -InformationAction Ignore
Write-Host "I won't print" 6> $null
```

These commands effectively suppress output of the `Write-Host` cmdlet. The first one uses the `InformationAction` parameter with the `Ignore` Value to suppress output to the information stream. The second example redirects the information stream of the command to the `$null` variable and thereby suppresses it. For more information, see about_Output_Streams (../Microsoft.PowerShell.Core/About/about_Output_Streams.md).

REMARKS

To see the examples, type: "get-help Write-Host -examples".

For more information, type: "get-help Write-Host -detailed".

For technical information, type: "get-help Write-Host -full".

For online help, type: "get-help Write-Host -online"