

END-USER PLATFORMS > POWERSHELL

PowerShell Parameter Validation: Ensuring Valid Input for Functions

It's a good idea to validate function parameters before executing any actions. Learn these two techniques for performing parameter validation.

Brien Posey | Feb 20, 2024

PowerShell makes it easy to pass values to a function in the form of parameters. However, it's a good idea to validate parameters before executing any actions. Otherwise, unexpected values could cause a function to behave in a completely unpredictable way.

That being the case, I want to share two methods that I commonly use to validate the parameters that are passed to a function .

Related: PowerShell Modules vs. Dot Sourcing: Which Approach Is Better?

Before I get started, I want to point out that the techniques I'm about to demonstrate are only suitable for scenarios where you are passing a string value to a function and are certain that the string should contain one of several possible values. If you are working with other types of data or don't know ahead of time which values for the string are acceptable, you must resort to using other validation techniques.

#1. Validation Check Using the -Contains Operator

So, with that said, here is a very simple PowerShell script that demonstrates the first validation technique:

```
Function Display-ColorText {  
    param(  
        [String]$Color  
    )  
    $Colors = "Red","Green","Blue","Yellow","Gray"  
    if($Colors -contains $Color) {  
        Write-Host "This text is" $Color -ForegroundColor $Color  
    } else {  
        Write-Host $Color 'is not a Supported Color.'  
    }  
}  
  
$Color = Read-Host "Please type the name of a color"  
Display-ColorText $Color
```

In this script, the user is prompted to enter the name of a color, which is then sent to a function called `Display-ColorText`. Of course, the user could enter just about anything, not necessarily a valid color name. This includes misspellings and unsupported color names. To address this, the script uses a validation step to make sure the entered color name is appropriate.

Looking at the script, you'll notice that I created a variable called `$Colors` and have set it to be equal to "Red", "Green", "Blue", "Yellow", "Gray". These are the colors that the script will support. While PowerShell recognizes a broader range of color names, the script deliberately focuses on these five colors to keep things simple.

The next line of code checks if the color name entered by the user (stored in the `$Color` variable) exists in the list of supported colors defined by the `$Colors` variable. This check is based on the `-Contains` operator. If the color entered by the user is present in the list, the script outputs a line of text in the chosen color. Conversely, if the color is not supported, the user receives a message indicating their selection is unsupported. You can see the script in action in Figure 1.

 Windows PowerShell

```
PS C:\scripts> ./validate.ps1
Please type the name of a color: Red
This text is Red
PS C:\scripts> ./validate.ps1
Please type the name of a color: Green
This text is Green
PS C:\scripts> ./validate.ps1
Please type the name of a color: Blue
This text is Blue
PS C:\scripts> ./validate.ps1
Please type the name of a color: Purple
Purple is not a Supported Color.
PS C:\scripts> █
```

Figure 1. *This is what my script does.*

#2. Validation Check Using ValidateSet

Now that I have demonstrated how to perform parameter validation by using a - Contains operator, I will introduce a different technique that accomplishes more or less the same thing.

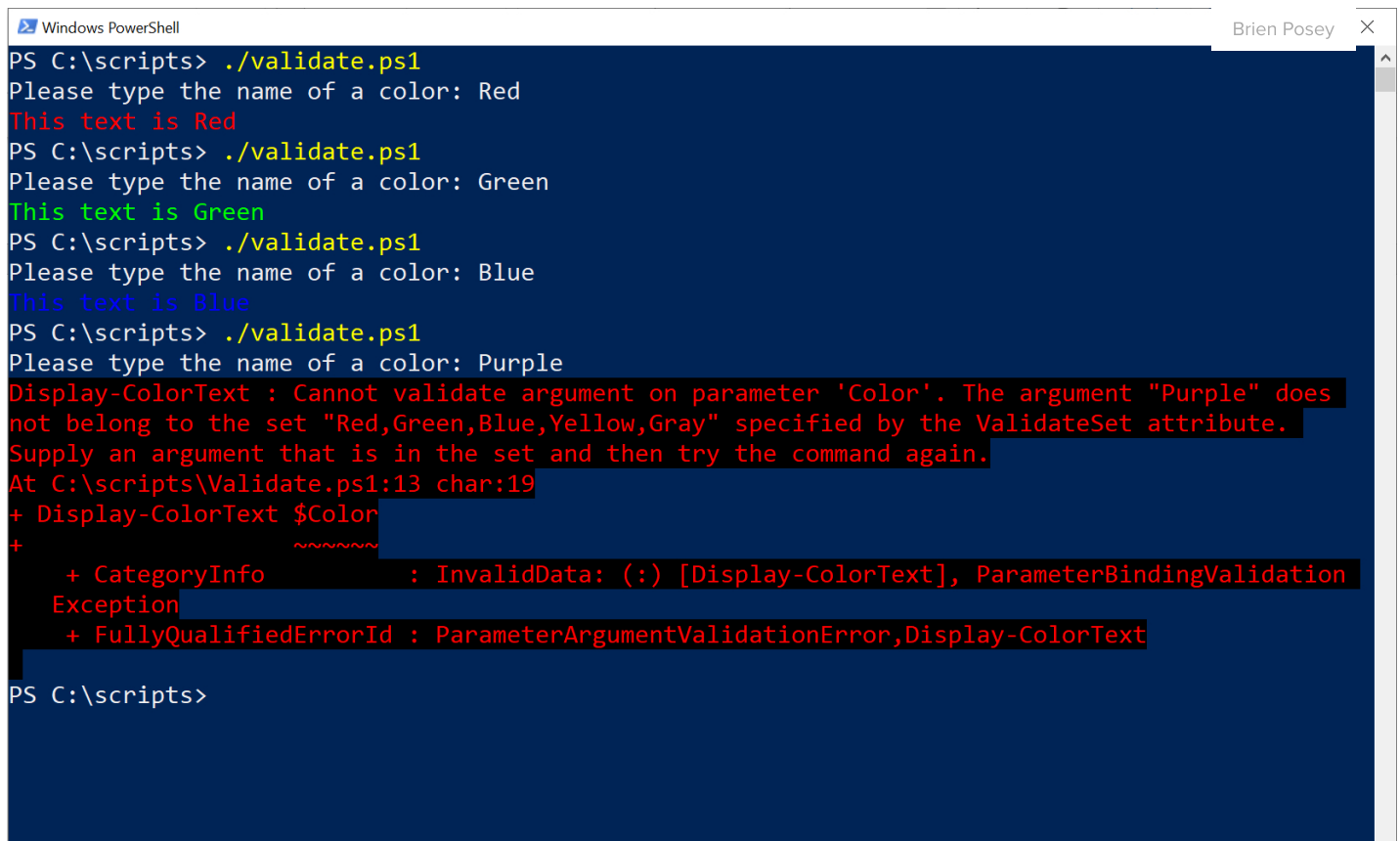
Here is the script:

```
Function Display-ColorText {
```

```
param(  
    [ValidateSet("Red","Green","Blue","Yellow","Gray")]  
    [String]$Color  
)  
  
    Write-Host "This text is" $Color -ForegroundColor $Color  
}  
  
$Color = Read-Host "Please type the name of a color"  
Display-ColorText $Color
```

In this script, `ValidateSet` is used instead of the `-Contains` operator. Notice that all acceptable input values are defined within the `Param` section. This means that you don't have to worry about writing a block of code to validate the input. The `ValidateSet` statement handles the validation process for you.

This technique works exceptionally well if the user enters a valid color. However, if the user inputs something invalid, the script spews a lengthy and convoluted error message. You can see an example of this in Figure 2.



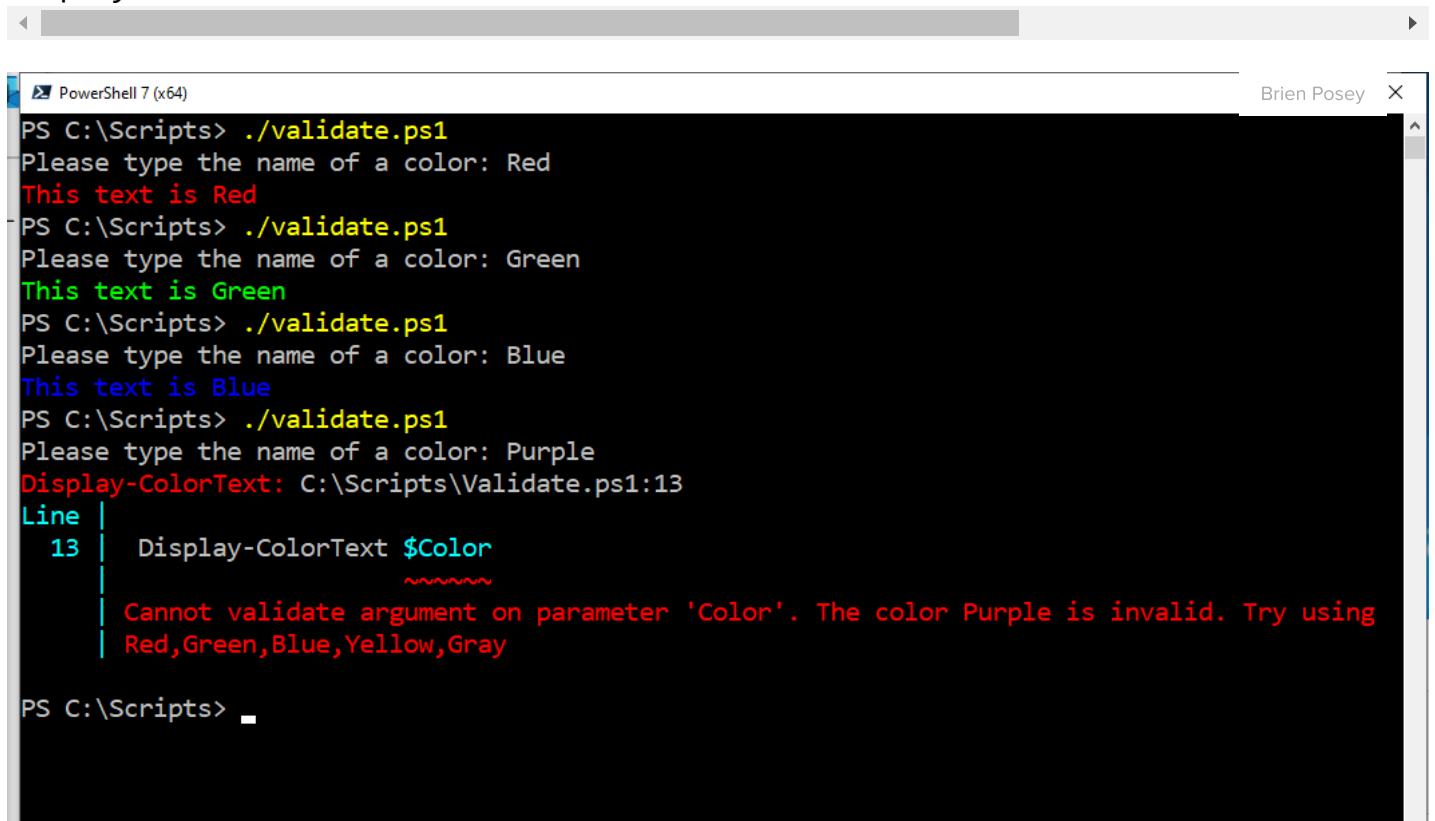
```
Windows PowerShell Brien Posey X  
PS C:\scripts> ./validate.ps1  
Please type the name of a color: Red  
This text is Red  
PS C:\scripts> ./validate.ps1  
Please type the name of a color: Green  
This text is Green  
PS C:\scripts> ./validate.ps1  
Please type the name of a color: Blue  
This text is Blue  
PS C:\scripts> ./validate.ps1  
Please type the name of a color: Purple  
Display-ColorText : Cannot validate argument on parameter 'Color'. The argument "Purple" does  
not belong to the set "Red,Green,Blue,Yellow,Gray" specified by the ValidateSet attribute.  
Supply an argument that is in the set and then try the command again.  
At C:\scripts\Validate.ps1:13 char:19  
+ Display-ColorText $Color  
+ ~~~~~  
+ CategoryInfo          : InvalidData: (:) [Display-ColorText], ParameterBindingValidation  
Exception  
+ FullyQualifiedErrorId : ParameterArgumentValidationError,Display-ColorText  
PS C:\scripts>
```

Figure 2. *Using ValidateSet works well until the user enters something invalid.*

There is a way to handle errors more gracefully, but error handling requires you to be running PowerShell 7 or higher.

Take a look at the script below: It's identical to the previous script but adds an error message to the validation set. The cool thing about this error message is that it tells the user what values can be used as acceptable input.

```
Function Display-ColorText {  
  
    param(  
        [ValidateSet("Red", "Green", "Blue", "Yellow", "Gray", ErrorMessage="The color {  
        [String]$Color  
        )  
  
        Write-Host "This text is" $Color -ForegroundColor $Color  
    }  
  
$Color = Read-Host "Please type the name of a color"  
Display-ColorText $Color
```

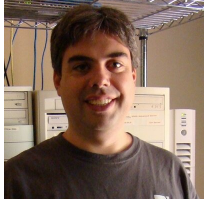


The screenshot shows a PowerShell 7 terminal window with the following content:

```
PS C:\Scripts> ./validate.ps1  
Please type the name of a color: Red  
This text is Red  
PS C:\Scripts> ./validate.ps1  
Please type the name of a color: Green  
This text is Green  
PS C:\Scripts> ./validate.ps1  
Please type the name of a color: Blue  
This text is Blue  
PS C:\Scripts> ./validate.ps1  
Please type the name of a color: Purple  
Display-ColorText: C:\Scripts\Validate.ps1:13  
Line |  
  13 | Display-ColorText $Color  
     | ~~~~~  
     | Cannot validate argument on parameter 'Color'. The color Purple is invalid. Try using  
     | Red,Green,Blue,Yellow,Gray  
PS C:\Scripts> █
```

Figure 3. *PowerShell 7 greatly improves error handling.*

About the author



Brien Posey is a bestselling technology author, speaker, and 21x Microsoft MVP. In addition to his ongoing work in IT, Posey has trained as a commercial astronaut candidate in preparation to fly on a mission to study polar mesospheric clouds from space.

Source URL: <https://www.itprotoday.com/powershell/powershell-parameter-validation-ensuring-valid-input-functions>