PowerShell

About PowerShell

PowerShell is a powerful command-line shell and scripting language developed by Microsoft. It is designed to help users automate tasks, manage systems, and perform administrative tasks efficiently. Built on the .NET framework, **PowerShell** is both a ***command-line tool*** and a ***scripting environment***, making it a versatile tool for programmers and system administrators.

> "PowerShell is great because we had a series of rockstar engineers add their awesomeness to the project."
>
> *Source: Interview with Jeffrey Snover, PowerShell Inventor*

Why Learn PowerShell?

PowerShell is an essential tool for:

1. Automating repetitive tasks.
2. Managing and configuring systems.
3. Working with files, processes, and services.
4. Interacting with APIs and web services.
5. Writing scripts to solve complex problems.

PowerShell Features and Terms

- **Cmdlets**: Lightweight commands that perform specific functions.
- **Scripting**: Write and execute scripts, series of commands, to automate tasks.
- **Pipeline**: Chain commands together to pass data between them.
- **Remote Management**: Execute commands on remote systems.
- **Modules**: Extend functionality with additional cmdlets and scripts.
- **Extensibility**: Create custom functions, modules, and scripts.
- **Cross-Platform**: PowerShell Core runs on Windows, macOS, and Linux.

PowerShell History

PowerShell Version Comparison

Version	Release Year	Key Features
1.0	2006	Initial release Windows XP SP2 and Windows Server 2003.
2.0	2009	Introduced remote management and modules.
5.1	2016	Last Windows-only version
6.0	2018	Known as PowerShell Core 6.0. First cross-platform version.
7.0	2020	Unified Windows PowerShell and PowerShell Core, offering a modern, cross-platform experience.

Version Summary

- **Windows PowerShell**: Versions 1.0 to 5.1 (Windows-only).
- **PowerShell Core**: Versions 6.0 and above (cross-platform).

PowerShell for Beginners

Basic Commands to Get Started

- `Get-Command`: Lists all available commands.
- `Get-Help`: Provides help information for commands.
- `Get-Process`: Displays all running processes.
- `Stop-Process`: Stops a specific process.
- `Set-Location`: Changes the current directory (like `cd` in Command Prompt).
- `New-Item`: Creates a new file or directory.

Example Script

Here's a simple script to list all files in a directory and display their sizes:

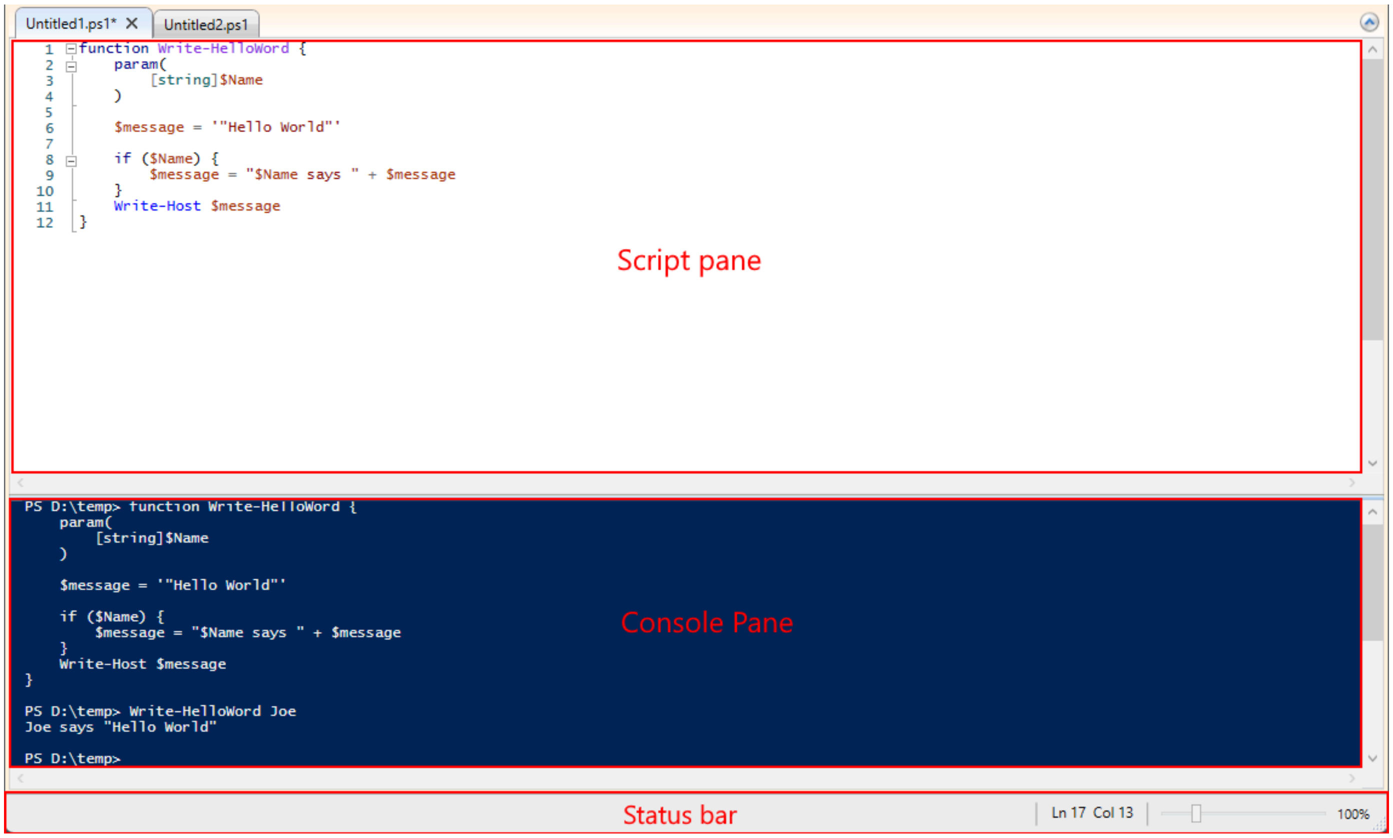
```powershell
# Get all files in the current directory
$files = Get-ChildItem

# Display file names and sizes
foreach ($file in $files) {
    Write-Output "$($file.Name) - $($file.Length) bytes"
}
```

Editors for PowerShell Scripting

When writing and debugging PowerShell scripts, using a dedicated editor can greatly enhance your productivity. Here are two popular options:

Windows PowerShell Integrated Scripting Environment (ISE)

- **What it is**: A built-in editor for Windows PowerShell (versions 1.0 to 5.1).
- **Features**:
    - Syntax highlighting.
    - Debugging tools *(breakpoints, step-through execution)*.
    - Integrated console for testing scripts.
    - Multi-tab interface for working with multiple scripts.
- **Best for**: Beginners and users working on older versions of PowerShell.



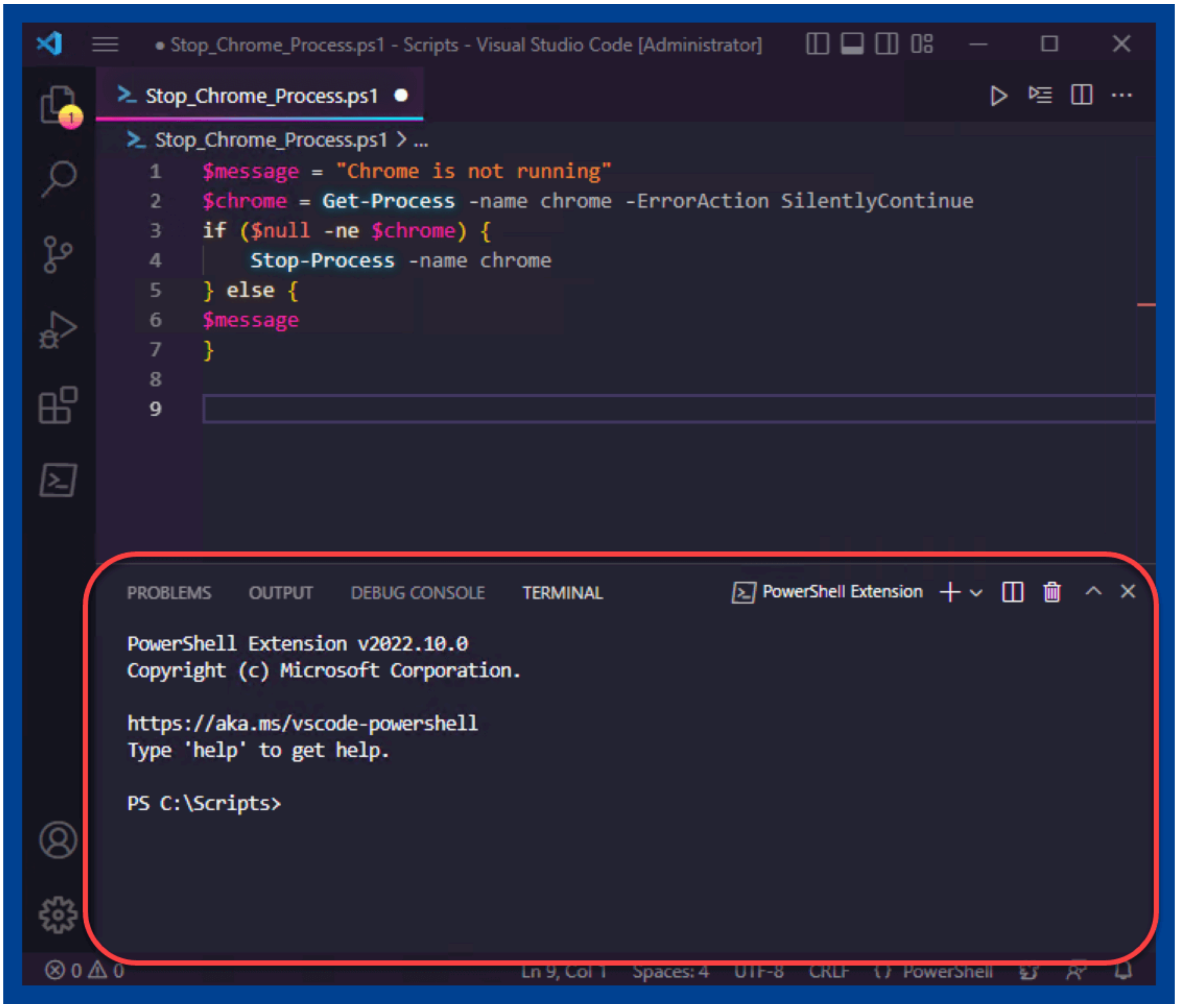
```
function Write-HelloWord {
    param(
        [string]$Name
    )

    $message = '"Hello World"'

    if ($Name) {
        $message = "$Name says " + $message
    }
    Write-Host $message
}
```

```
PS D:\temp> function Write-HelloWord {
    param(
        [string]$Name
    )

    $message = '"Hello World"'

    if ($Name) {
        $message = "$Name says " + $message
    }
    Write-Host $message
}

PS D:\temp> Write-HelloWord Joe
Joe says "Hello World"

PS D:\temp>
```

Visual Studio Code (VS Code)

- **What it is**: A free, open-source, cross-platform code editor by Microsoft.
- **Features**:
    - Syntax highlighting and IntelliSense for PowerShell.
    - Integrated terminal for running scripts.
    - Extensions for additional functionality *(e.g., PowerShell extension)*.
    - Support for **Git** and other version control systems.
    - Cross-platform *(Windows, macOS, Linux)*.

- **Best for**: Modern PowerShell development, especially with PowerShell 7 and cross-platform scripting.



```
$message = "Chrome is not running"
$chrome = Get-Process -name chrome -ErrorAction SilentlyContinue
if ($null -ne $chrome) {
    Stop-Process -name chrome
} else {
$message
}
```

```
PowerShell Extension v2022.10.0
Copyright (c) Microsoft Corporation.

https://aka.ms/vscode-powershell
Type 'help' to get help.

PS C:\Scripts>
```