Windows PowerShell Get-Help on Cmdlet 'ConvertTo-Html'

PS:\>Get-HELP ConvertTo-Html -Full

NAME

ConvertTo-Html

SYNOPSIS

Converts .NET objects into HTML that can be displayed in a Web browser.

SYNTAX

ConvertTo-Html [[-Property] <System.Object[]>] [[-Head] <System.String[]>] [[-Title] <System.String>] [[-Body] <System.String[]>] [-As {Table | List}] [-CssUri

<System.Uri>] [-InputObject <System.Management.Automation.PSObject>] [-PostContent <System.String[]>] [-PreContent <System.String[]>] [<CommonParameters>]

ConvertTo-Html [[-Property] <System.Object[]>] [-As {Table | List}] [-Fragment] [-InputObject <System.Management.Automation.PSObject>] [-PostContent

<System.String[]>] [-PreContent <System.String[]>] [<CommonParameters>]

DESCRIPTION

The `ConvertTo-Html` cmdlet converts .NET objects into HTML that can be displayed in a Web browser. You can use this

cmdlet to display the output of a command in a
Web page.

You can use the parameters of `ConvertTo-Html` to select object properties, to specify a table or list format, to specify the HTML page title, to add text before and
after the object, and to return only the table or list fragment, instead of a strict DTD page.

When you submit multiple objects to `ConvertTo-Html`, PowerShell creates the table (or list) based on the properties of the first object that you submit. If the
remaining objects do not have one of the specified properties, the property value of that object is an empty cell. If the remaining objects have additional
properties, those property values are not included in the file.

PARAMETERS

-As <System.String>

Determines whether the object is formatted as a table or a list. Valid values are Table and List . The default value is Table .

The Table value generates an HTML table that resembles the PowerShell table format. The header row displays the property names. Each table row represents an
object and displays the object's values for each property.

The List value generates a two-column HTML table for each object that resembles the PowerShell list format. The first column displays the property name. The
second column displays the property value.

Required? false
Position? named
Default value None
Accept pipeline input? False
Accept wildcard characters? false

-Body <System.String[]>

Specifies the text to add after the opening `<BODY>` tag. By default, there is no text in that position.

Required? false

Position? 3

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-CssUri <System.Uri>

Specifies the Uniform Resource Identifier (URI) of the cascading style sheet (CSS) that is applied to the HTML file. The URI is included in a style sheet link in

the output.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Fragment <System.Management.Automation.SwitchParameter>

Generates only an HTML table. The `<HTML>`, `<HEAD>`, `<TITLE>`, and `<BODY>` tags are omitted.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-Head <System.String[]>

Specifies the content of the `<HEAD>` tag. The default is `<title>HTML TABLE</title>`. If you use the Head parameter, the Title parameter is ignored.

Required? false

Position? 1

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-InputObject <System.Management.Automation.PSObject>

Specifies the objects to be represented in HTML. Enter a variable that contains the objects or type a command or expression that gets the objects.

If you use this parameter to submit multiple objects, such as all of the services on a computer, `ConvertTo-Html` creates a table that displays the properties of

a collection or of an array of objects. To create a table of the individual objects, use the pipeline operator to pipe the objects to `ConvertTo-Html`.

Required? false

Position? named

Default value None

Accept pipeline input? True (ByValue)

Accept wildcard characters? false

-PostContent <System.String[]>

Specifies text to add after the closing `</TABLE>` tag. By default, there is no text in that position.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-PreContent <System.String[]>

Specifies text to add before the opening `<TABLE>` tag. By default, there is no text in that position.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Property <System.Object[]>

Includes the specified properties of the objects in the HTML. The value of the Property parameter can be a new calculated property. The calculated property can be

a script block or a hash table. Valid key-value pairs are:

- `Expression` - `<string>` or `<script block>`

- `FormatString` - `<string>`

- `Width` - `<int32>` - must be greater than `0`

- `Alignment` - value can be `Left`, `Center`, or `Right`

For more information, see about_Calculated_Properties (../Microsoft.PowerShell.Core/About/about_Calculated_Properties.md).

Required? false

Position? 0

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Title <System.String>

Specifies a title for the HTML file, that is, the text that appears between the `<TITLE>` tags.

Required? false

Position? 2

Default value None

Accept pipeline input? False

Accept wildcard characters? false

<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).

INPUTS

System.Management.Automation.PSObject

You can pipe any object to this cmdlet.

OUTPUTS

System.String

This cmdlet returns an array of strings of HTML representing the converted object.

NOTES

To use this cmdlet, pipe one or more objects to the cmdlet or use the InputObject parameter to specify the object. When the input consists of multiple objects,

the output of these two methods is quite different.

- When you pipe multiple objects to a cmdlet, PowerShell sends the objects to the cmdlet one at a time. As a result, `ConvertTo-Html` creates a table that

displays the individual objects. For example, if you pipe the processes on a computer to `ConvertTo-Html`, the resulting table displays all of the processes.

- When you use the InputObject parameter to submit multiple objects, `ConvertTo-Html` receives these objects as a collection or as an array. As a result, it

creates a table that displays the array and its properties, not the items in the array. For example, if you use InputObject to submit the processes on a

computer to `ConvertTo-Html`, the resulting table displays an object array and its properties.

To comply with the XHTML Strict DTD, the `DOCTYPE` tag is modified accordingly:

`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "https://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`

------- Example 1: Create a web page to display the date -------

ConvertTo-Html -InputObject (Get-Date)

This command creates an HTML page that displays the properties of the current date. It uses the InputObject parameter to submit the results of a `Get-Date` command to

the `ConvertTo-Html` cmdlet.

-- Example 2: Create a web page to display PowerShell aliases --

Get-Alias | ConvertTo-Html | Out-File aliases.htm

Invoke-Item aliases.htm

This command creates an HTML page that lists the PowerShell aliases in the current console.

The command uses the `Get-Alias` cmdlet to get the aliases. It uses the pipeline operator (`|`) to send the aliases to the `ConvertTo-Html` cmdlet, which creates the

HTML page. The command also uses the `Out-File` cmdlet to send the HTML code to the `aliases.htm` file.

-- Example 3: Create a web page to display PowerShell events --

Get-EventLog -LogName "Windows PowerShell" | ConvertTo-Html | Out-File pslog.htm

This command creates an HTML page called `pslog.htm` that displays the events in the Windows PowerShell event log on the local computer.

It uses the `Get-EventLog` cmdlet to get the events in the Windows PowerShell log and then uses the pipeline operator (`|`) to send the events to the `ConvertTo-Html`

cmdlet. The command also uses the `Out-File` cmdlet to send the HTML code to the `pslog.htm` file.

The command also uses the `Out-File` cmdlet to send the HTML code to the `pslog.htm` file.

------ Example 4: Create a web page to display processes ------

```
Get-Process |
  ConvertTo-Html -Property Name, Path, Company -Title "Process Information" |
    Out-File proc.htm
Invoke-Item proc.htm
```

These commands create and open an HTML page that lists the name, path, and company of the processes on the local computer.

The first command uses the `Get-Process` cmdlet to get objects that represent the processes running on the computer. The command uses the pipeline operator (`|`) to

send the process objects to the `ConvertTo-Html` cmdlet.

The command uses the Property parameter to select three properties of the process objects to be included in the table. The command uses the Title parameter to specify

a title for the HTML page. The command also uses the `Out-File` cmdlet to send the resulting HTML to a file named `Proc.htm`.

The second command uses the `Invoke-Item` cmdlet to open the `Proc.htm` in the default browser.

--- Example 5: Create a web page to display service objects ---

```
Get-Service | ConvertTo-Html -CssUri "test.css"
```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "https://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

<head>

<title>HTML TABLE</title>

<link rel="stylesheet" type="text/css" href="test.css" />

...

This command creates an HTML page of the service objects that the `Get-Service` cmdlet returns. The command uses the CssUri parameter to specify a cascading style

sheet for the HTML page.

The CssUri parameter adds an additional `<link rel="stylesheet" type="text/css" href="test.css">` tag to the resulting HTML. The HREF attribute in the tag contains

the name of the style sheet.

--- Example 6: Create a web page to display service objects ---

Get-Service | ConvertTo-Html -As LIST | Out-File services.htm

This command creates an HTML page of the service objects that the `Get-Service` cmdlet returns. The command uses the As parameter to specify a list format. The cmdlet

`Out-File` sends the resulting HTML to the `Services.htm` file.

------ Example 7: Create a web table for the current date ------

Get-Date | ConvertTo-Html -Fragment

<table>

<colgroup>...</colgroup>

<tr><th>DisplayHint</th><th>DateTime</th><th>Date</th><th>Day</th><th>DayOfWeek</th><th>DayOfYear</th><th>Hour</th>

<th>Kind</th><th>Millisecond</th><th>Minute</th><th>Month</th><th>Second</th><th>Ticks</th><th>TimeOfDay</th><th>Year</th></tr>

<tr><td>DateTime</td><td>Monday, May 05, 2008 10:40:04 AM</td><td>5/5/2008 12:00:00

AM</td><td>5</td><td>Monday</td>

<td>126</td><td>10</td><td>Local</td><td>123</td><td>40</td><td>5</td><td>4</td><td>633455808041237213</td><td>10:40:04.12

37213</td><td>2008</td></tr>

</table>

This command uses `ConvertTo-Html` to generate an HTML table of the current date. The command uses the `Get-Date` cmdlet to get the current date. It uses a pipeline

operator (`|`) to send the results to the `ConvertTo-Html` cmdlet.

The `ConvertTo-Html` command includes the Fragment parameter, which limits the output to an HTML table. As a result, the other elements of an HTML page, such as the

`<HEAD>` and `<BODY>` tags, are omitted.

-- Example 8: Create a web page to display PowerShell events --

```
Get-EventLog -Log "Windows PowerShell" | ConvertTo-Html -Property id, level, task
```

This command uses the `Get-EventLog` cmdlet to get events from the Windows PowerShell event log.

It uses a pipeline operator (`|`) to send the events to the `ConvertTo-Html` cmdlet, which converts the events to HTML format.

The `ConvertTo-Html` command uses the Property parameter to select only the ID , Level , and Task properties of the event.

-- Example 9: Create a web page to display specified services --

```
$htmlParams = @{
  Title = "Windows Services: Server01"
  Body = Get-Date
  PreContent = "<P>Generated by Corporate IT</P>"
  PostContent = "For details, contact Corporate IT."
}
```

```
Get-Service A* |
  ConvertTo-Html @htmlParams |
    Out-File Services.htm
Invoke-Item Services.htm
```

This command creates and opens a Web page that displays the services on the computer that begin with `A`. It uses the Title , Body , PreContent , and PostContent
parameters of `ConvertTo-Html` to customize the output.

The first part of the command uses the `Get-Service` cmdlet to get the services on the computer that begin with `A`. The command uses a pipeline operator (`|`) to
send the results to the `ConvertTo-Html` cmdlet. The command also uses the `Out-File` cmdlet to send the output to the `Services.htm` file.

A semicolon (`;`) ends the first command and starts a second command, which uses the `Invoke-Item` cmdlet to open the `Services.htm` file in the default browser.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/convertto-html?view=powershell-5.1&WT.mc_id=ps-gethelp

about_Calculated_Properties

ConvertTo-Csv

ConvertTo-Json

ConvertTo-Xml

Export-Clixml

Import-Clixml