Pode.Web 0.8.3

200 Working Examples

PowerShell module for rapid web services development
https://badgerati.github.io/Pode.Web/Tutorials

Based on text, help, tutorials, pictures and examples by Matthew Kelly Badgeratti
https://badgerati.github.io/Pode.Web

Book compiling and examples adjustements by O. Mike, January 2024

❤ Powered by oTo

Pode.Web 0.8.3

PowerShell module for rapid web services development
https://badgerati.github.io/Pode.Web/Tutorials

Completely based on text, help, tutorials, pictures and examples by Matthew Kelly Badgeratti
https://badgerati.github.io/Pode.Web/

Book compiling and examples adjustements by O. Mike, January 2024

Introduction

This is a book on ‘Pode.Web’ PowerShell module, that allows rapid development of web pages and web-services (mainly for administration purposes).

I did this because most of the examples on the page https://badgerati.github.io/Pode.Web are inconsistent due to constant changes and updates (missing lines to be complete examples, or using wrong options/switches which are not following the current versions).

This was my greatest challenge while trying to do anything useful. But, since I have never seen anything free and better on this area (ever), I have decided to make a tiny contribution (in my own way 😊).

That is why I made examples complete, the way, you can copy and paste them into your editor and run them ‘ass-they-are’.

Almost no changes to the original text were made, but most of the examples (the ones I understood 😉) were altered by adding some lines that should be present inside of every pode.web script.

I was trying to develop fast, but there are still many challenges present (bugs, limitations, unexpected time-outs, non-descriptive error messages, not executing some functions inside of some scriptBlocks, etc.). So, just forget a fast development. What has its value here, are just a few lines of code that can present your mighty application. That’s why.

But still, this is the best free HTML5, jQuery, graph.js and BootStrap framework for PowerShell! You don’t even have to know much about user-side technologies, and still you can develop a pretty advanced web apps with just a few lines of PowerShell code.

O. Mike,
January 2024

If you like this book, please consider donating:

Paypal

https://www.paypal.com/donate/?business=DY3YXT4KDJ6WW&no_recurring=0&item_name=for+Pode.Web+book&currency_code=EUR

Contents

Pode.Web Features

Features

This is a web template framework for use with the Pode PowerShell web server (v2.6.0+).

It allows you to build web pages purely with PowerShell - no HTML, CSS, or JavaScript knowledge required!

You can build charts, forms, tables, general text, tabs, login pages, etc. There's a light, dark, and terminal themes, and you can supply a custom CSS file.

To make it functional, you will also need to install a Pode module (https://badgerati.github.io/Pode/), which will be auto-installed if you perform the following as administrator:

```
install-module pode.web
```

- Like Pode, this is already cross-platform! (with support for PS5)
- Easily add pages, with different layouts and elements
- Support for authentication with a login page!
- Create line, bar, pie, and doughnut charts
- Support for forms, with all kinds of input elements
- Show toast messages on the page, or send desktop notifications
- Display data in tables, with pagination, sorting and filtering
- Use a stepper for a more controlled flow of form input
- Or, use a tabs layout for your pages!
- Show or right code via the Monaco editor (still WIP)
- Render code in code-blocks with code highlighting!
- Support for Light, Dark, Terminal, and custom themes

Libraries

The Pode.Web templates are built using Bootstrap, jQuery, Material Design Icons, Chart.js, and Highlight.js.

Installation in details

Pode.Web is a PowerShell module that works along side Pode, and it can be installed from the PowerShell Gallery, or Docker. Once installed, you can use the module in your Pode server.

Minimum Requirements

Before installing Pode.Web, the minimum requirements must be met:

- Pode v2.6.0+

Which also includes Pode's minimum requirements: * OS: * Windows * Linux * MacOS * Raspberry Pi * PowerShell: * Windows PowerShell 5+ * PowerShell (Core) 6+ * .NET Framework 4.7.2+ (For Windows PowerShell)

PowerShell Gallery

To install Pode.Web from the PowerShell Gallery, you can use the following:

```
Install-Module -Name Pode.Web
```

Docker

Like Pode, Pode.Web also has Docker images available. The images use Pode v2.8.0 on either an Ubuntu Focal image (default), an Alpine image, or an ARM32 image (for Raspberry Pis).

- To pull down the latest Pode.Web image you can do:
  ```
  # for latest
  docker pull badgerati/pode.web:latest

  # or the following for a specific version:
  docker pull badgerati/pode.web:0.8.0
  ```
- To pull down the Alpine Pode.Web image you can do:
  ```
  # for latest
  docker pull badgerati/pode.web:latest-alpine

  # or the following for a specific version:
  docker pull badgerati/pode.web:0.8.0-alpine
  ```
- To pull down the ARM32 Pode.Web image you can do:
  ```
  # for latest
  docker pull badgerati/pode.web:latest-arm32

  # or the following for a specific version:
  docker pull badgerati/pode.web:0.8.0-arm32
  ```

Once pulled, you can view here on how to use the image.

GitHub Package Registry

You can also get the Pode.Web docker image from the GitHub Package Registry! The images are the same as the ones hosted in Docker.

- To pull down the latest Pode.Web image you can do:
  ```
  # for latest
  docker pull docker.pkg.github.com/badgerati/pode.web/pode.web:latest

  # or the following for a specific version:
  docker pull docker.pkg.github.com/badgerati/pode.web/pode.web:0.8.0
  ```
- To pull down the Alpine Pode image you can do:
  ```
  # for latest
  docker pull docker.pkg.github.com/badgerati/pode.web/pode.web:latest-apline

  # or the following for a specific version:
  docker pull docker.pkg.github.com/badgerati/pode.web/pode.web:0.8.0-alpine
  ```
- To pull down the ARM32 Pode.Web image you can do:
  ```
  # for latest
  docker pull docker.pkg.github.com/badgerati/pode.web/pode.web:latest-arm32

  # or the following for a specific version:
  docker pull docker.pkg.github.com/badgerati/pode.web/pode.web:0.8.0-arm32
  ```

Once pulled, you can view here on how to use the image.

Using the Module

Once installed, you then need to import the module at the top of your Pode server's script; unlike Pode, the module's functions are not automatically exported:

```
Import-Module -Name Pode.Web

Start-PodeServer {
   Use-PodeWebTemplates -Title '<Title>'
}
```

Actually you do not need to load modules in your shell script. All needed modules will be auto-loaded by the PowerShell itself.

Basics

The first thing to note is that to use Pode.Web, you do need to import the module. But with the PowerShell version 5.1+ this is not necessary, since auto-load feature is implemented. So, what you really need to know is the following:

- you only need to install Pode.Web and Pode module will be automatically added
- only one server/script can run at a specific port at the same time on the same machine
- if you are using more advanced functions (like https), a console with admin privileges is required
- latest packages available at: https://www.powershellgallery.com/packages/Pode.web

```
Import-Module -Name Pode.Web

Start-PodeServer {
   # logic
}
```

To speed-up loading of pages, enable caching within your server.psd1 file:

```
@{
   Web = @{
      Static = @{
         Cache = @{
            Enable = $true
         }
      }
   }
}
```

Use the Templates

'Pode.Web' contains extension functions that can be used within your Pode server. To setup the templates, and start using them, you will always first need to call Use-PodeWebTemplates; this will let you define the title of your website, the default theme, and the logo/favicon:

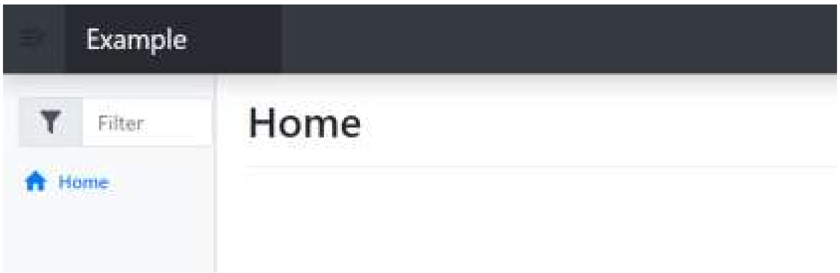
```
Import-Module -Name Pode.Web

Start-PodeServer {
   Use-PodeWebTemplates -Title 'Example' -Theme Dark
}
```

If your server uses multiple endpoints, you can set your website to be bound to only

one of them via -EndpointName:

Example000.ps1 – to access, navigate to: http://localhost:9090/

```
Import-Module -Name Pode.Web
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8080 -Protocol Http -Name User
    Add-PodeEndpoint -Address localhost -Port 8090 -Protocol Http -Name Admin

    # this will bind the site to the Admin endpoint
    Use-PodeWebTemplates -Title 'Example' -Theme Dark -EndpointName Admin
}
```



Add some Pages

Once the templates are enabled, you can start to add some pages! You can find more information on the Pages page, but in general there are 3 types of pages:

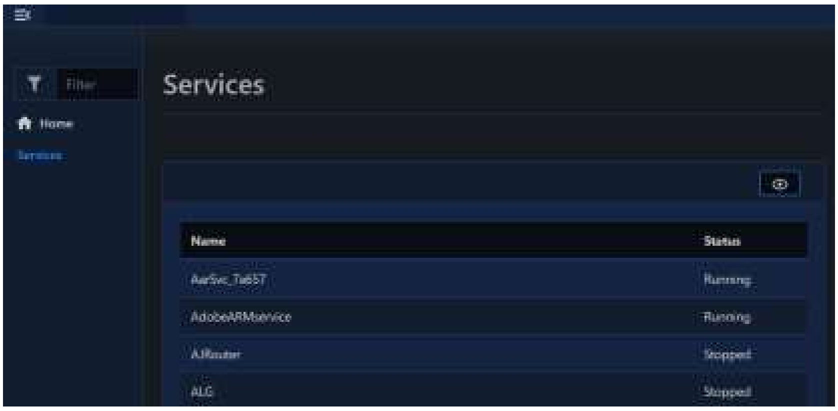
- Home
- Login
- Webpage

To just add a new page, you use Add-PodeWebPage, supplying the -Name of the page and a -ScriptBlock for defining the layout/element components on the page:

Example001.ps1

```
Start-PodeServer {
   Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
   New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
   Use-PodeWebTemplates -Title ' ' -Theme Dark
   Add-PodeWebPage -Name 'Services' -Icon 'Settings' -ScriptBlock {
    New-PodeWebCard -Content @(
        New-PodeWebTable -Name 'Services' -ScriptBlock {
            foreach ($svc in (Get-Service)) {
                [ordered]@{
                    Name   = $svc.Name
                    Status = "$($svc.Status)"
} } } ) } }
```

The above would render a new page with a table, showing all the services on the computer.



Services

Name	Status
AarSvc_Ta657	Running
AdobeARMservice	Running
AJRouter	Stopped
ALG	Stopped

Sidebar

Pages added to your site will appear in the sidebar on the left of your pages. The sidebar has a filter box at the top by default, but this can be removed via -NoPageFilter:

```
Use-PodeWebTemplates -Title 'Example' -Theme Dark -NoPageFilter
```

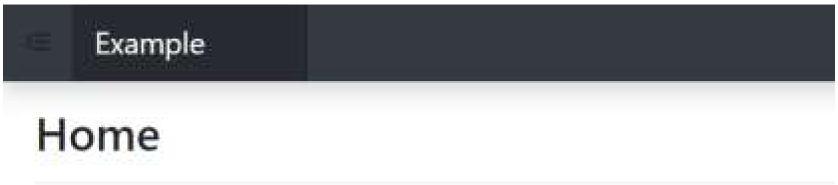
You can also force the sidebar to be hidden by default via -HideSidebar:

```
Use-PodeWebTemplates -Title 'Example' -Theme Dark -HideSidebar
```

Example002.ps1

```
Start-PodeServer {
   Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
   New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
   Use-PodeWebTemplates -Title 'Example' -Theme Dark -NoPageFilter -HideSideBar
}
```

Simple starting page:



Example

Home

Classes and Styles

Nearly every component in Pode.Web has a -CssClass and a -CssStyle parameter. These parameters allow you to add custom classes/styles onto the components, so you can control their look and functionality.

Custom Scripts/Styles

You can reference custom JavaScript and CSS files to use via Import-PodeWebJavaScript and Import-PodeWebStylesheet. Both take a relative/literal -Url to the file.

Classes

The -CssClass parameter accepts an array of strings, which are set on the class property on the parent element of approparite component. The classes by themselves don't do anything, but you can use them to build custom CSS files and import them via Import-PodeWebStylesheet; you can also use the custom classes as references in custom JavaScript files, and import these via Import-PodeWebJavaScript.

For example, the following would apply the my-custom-textbox class to a textbox:

```
New-PodeWebTextbox -Name 'Message' -CssClass 'my-custom-textbox'
```

Then you could create some /public/my-styles.css file with the following, to set the textbox's text colour to purple:

```
.my-custom-textbox {
   color: purple
}
```

and import it via: Import-PodeWebStylesheet -Url '/my-styles.css'.

or, you can create some JavaScript file at /public/my-scripts.js with an event to write to console on keyup. jQuery works here, as Pode.Web uses jQuery. Also, we have to reference the class then the input control, as the class is at the paraent level of the textbox element; this allows for more fine grained control of a component as a whole - such as a textbox's labels, divs, spans, etc.

```
$('.my-custom-textbox input').off('keyup').on('keyup', (e) => {
   console.log($(e.target).val());
})
```

and import it via: Import-PodeWebJavaScript -Url '/my-scripts.js'.

You can add/remove classes on components using the Class output actions. (This will add classes onto the component itself, not the parent).

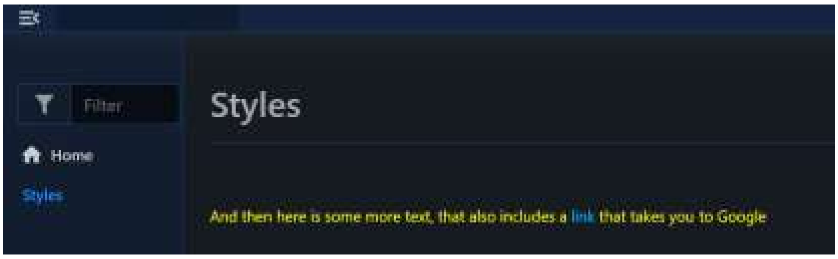
Styles

The -CssStyle parameter accepts a hashtable where the key is the name of a CSS property, with an appropriate value for that property. These styles are applied directly onto the main component.

For example, the following would display a paragraph with yellow text:

Example003.ps1

```
Start-PodeServer {
   Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
   New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
   Use-PodeWebTemplates -Title ' ' -Theme Dark
       Add-PodeWebPage -Name 'Styles' -Icon 'Settings' -ScriptBlock {
           New-PodeWebParagraph -CssStyle @{ Color = 'Yellow' } -Elements @(
               New-PodeWebText -Value 'And here is more text, including a '
               New-PodeWebLink -Value 'link' -Source 'https://google.com'
               New-PodeWebText -Value ' that takes you to Google'
) } }
```



And then here is some more text, that also includes a link that takes you to Google

You can set/remove CSS style properties on components using the Style output actions.

Events

In JavaScript you have general events that can trigger, such as onchange or onfocus. You can bind scriptblocks to these events for different components by using Register-PodeWebEvent.

For now only Element components support registering events, and not all elements support events (check an elements page to see if it supports events!).

The following general events are supported:

Name	Description
Change	Fires when the value of the component changes
Focus	Fires when the component gains focus
FocusOut	Fires when the component loses focus
Click	Fires when the component is clicked
MouseOver	Fires when the mouse moves over the component
MouseOut	Fires when the mouse moves out of the component
KeyDown	Fires when a key is pressed down on the component
KeyUp	Fires when a key is lifted up on a component

Similar to say a Button's click scriptblock, these events can run whatever logic you like, including returning output actions for Pode.Web to action against on the frontend.

You can binding the same action to multiple event types by supplying mutliple types to Register-PodeWebEvent's -Type parameter. The current event that has triggered the logic can be sourced via $EventType within the -ScriptBlock.

Example

Let's say you want to have a Select element, but not in a form. When the Select's current value is changed, you want to run a script to show a message; to achieve this you can pipe the object returned by New-PodeWebSelect into Register-PodeWebEvent:

Example004.ps1

```
Start-PodeServer {
   Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
   New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
   Use-PodeWebTemplates -Title ' ' -Theme Dark
   Add-PodeWebPage -Name 'Roles' -Icon 'Settings' -ScriptBlock {
       New-PodeWebSelect -Name 'Role' -Options @('Choose...', 'User', 'Admin', 'Operations') |
           Register-PodeWebEvent -Type Change -ScriptBlock {
              Show-PodeWebToast -Message "The value was changed: $($WebEvent.Data['Role'])"
} } }
```

If the element the event triggers for is a form input element, the value will be serialised and available in $WebEvent.Data.

Element Specific

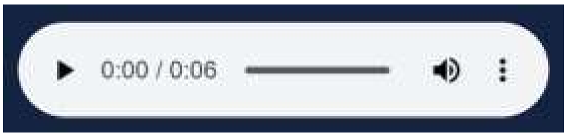
The events listed above are general events supported by almost every component. However some elements, like Audio, have their own specific events, and these can be found on the element's document page.

For example, the Audio element has a play event which can be registered as follows:

Example005.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark
   Add-PodeWebPage -Name 'Roles' -Icon 'Settings' -Layouts @(
     New-PodeWebCard -Content @(
       New-PodeWebAudio -Name 'example' -Source @(
       New-PodeWebAudioSource  -Url 'https://samplelib.com/lib/preview/mp3/sample-6s.mp3' ) |
           Register-PodeWebMediaEvent -Type Play -ScriptBlock {
                  Show-PodeWebToast -Title 'Action' -Message $EventType
} ) ) }
```

The script above with render a simple sound player with the sample sound do play.



Page Events

Pages also support events.

Events

The Login, Home and Webpages support registering the following events, and they can be registered via Register-PodeWebPageEvent:

Name	Description
Load	Fires when the page has fully loaded, including js/css/etc.
Unload	Fires when the has fully unloaded/closed
BeforeUnload	Fires just before the page is about to unload/close

To register an event for each page type:

- Login: you'll need to use -PassThru on Set-PodeWebLoginPage and pipe the result in Register-PodeWebPageEvent.
- Home: you'll need to use -PassThru on Set-PodeWebHomePage and pipe the result in Register-PodeWebPageEvent.

- Webpage: you'll need to use -PassThru on Add-PodeWebPage and pipe the result in Register-PodeWebPageEvent.

For example, if you want to show a message on a Webpage just before it closes or when is accessed. If no "$some_layouts" will be presented, it will just render something by default.

Example006.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark
 Add-PodeWebPage -Name Example -Layouts $some_layouts -PassThru |
     Register-PodeWebPageEvent -Type Load, Unload, BeforeUnload -ScriptBlock {
              Show-PodeWebToast -Message "Bye!"
    } }
```

Or on the Login page, after it's finished loading (note: you will need to use the -NoAuth switch):

Example007.ps1 – will only take **admin** for username and **admin** for password

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Enable-PodeSessionMiddleware -Duration 600 -Extend

 New-PodeAuthScheme -Form | Add-PodeAuth -Name Login -ScriptBlock {
     param($username, $password)
     if ($username -eq 'admin' -and $password -eq 'admin') {
         return @{
              User = @{
                   ID ='admin'
                   Name = 'admin'
                   Type = 'Human'
 }}}}

 Set-PodeWebLoginPage -Authentication Login -PassThru |
     Register-PodeWebPageEvent -Type Load, Unload, BeforeUnload -NoAuth -ScriptBlock {
              Show-PodeWebToast -Message "$($EventType)!"
} }
```

Icons

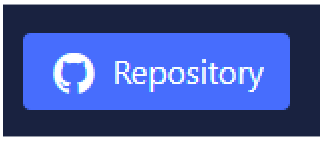
All icons rendered in Pode.Web are done using Material Design Icons. This applies to pretty much all -Icon parameters, except for the one on Set-PodeWebLoginPage which is actually the path to an image (This has been changed to -Logo, though -Icon is still an alias for now).

A list of searchable icons can be found here. When you've found the one you need, you only have to supply the part after mdi-. For example let's say you need the mdi-github icon, then you only need to supply the github part:

Example008.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark
   Add-PodeWebPage -Name 'Roles' -Icon 'Settings' -Layouts @(
       New-PodeWebCard -Content @(
       New-PodeWebButton -Name 'Repo' -Icon 'github' -Url 'https://github.com/Badgerati/Pode.Web'
) ) }
```

Which would render a button like below:



SEE MORE ICONS:
https://pictogrammers.github.io/@mdi/font/5.4.55/

https://pictogrammers.com/library/mdi/

Navigation

Web pages generated by Pode.Web all have a navigation bar (navbar) at the top. By default this is empty, but you can add your own Links, Dropdowns and Dividers to this navbar.

You create the individual components of the navbar, and then set them using Set-PodeWebNavDefault. This will set the default components for the navbar on every page. Each page created via Add-PodeWebPage has a -Navigation parameter, where you can supply custom navbar components to show instead of the default.

Links

A link on the navbar is just some text that can either take a user to another page/website, or can run a custom scriptblock within Pode.Web. To create a new link you use New-PodeWebNavLink with a -Name.

You can add multiple links, and separate them with dividers using New-PodeWebNavDivider.

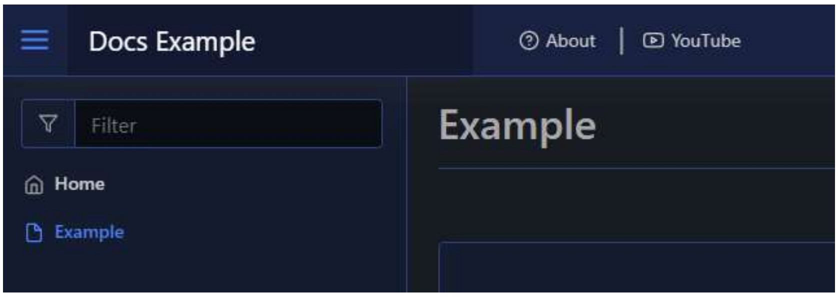
URL

The below example add a couple of simple links, and a divider. The first link takes the user to another page in Pode.Web, the other link takes the user to YouTube:

Example009.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark
 $navAbout = New-PodeWebNavLink -Name 'About' -Url '/pages/About' -Icon 'help-circle'
 $navDiv = New-PodeWebNavDivider
 $navYT = New-PodeWebNavLink -Name 'YouTube' -Url 'https://youtube.com' -Icon 'youtube'
 Set-PodeWebNavDefault -Items $navAbout, $navDiv, $navYT
 Add-PodeWebPage -Name Example
}
```

Which would look like below:



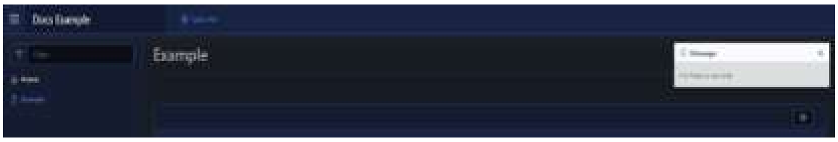
Dynamic

Whereas the next example will add a dynamic link that shows a toast message when clicked:

Example010.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 $navToast = New-PodeWebNavLink -Name 'Toast Me!' -Icon Settings -ScriptBlock {
    Show-PodeWebToast -Message "I'm from a nav link!"
 }
 Set-PodeWebNavDefault -Items $navToast

 Add-PodeWebPage -Name Example
}
```

Which would look like below:



Dropdowns

A dropdown can be created using New-PodeWebNavDropdown, and just takes a -Name and an array of -Items - which are just Links and Dividers. You can also nest another dropdown in -Items, but only one dropdown in another dropdown is supported, more do work but it gets flakey.

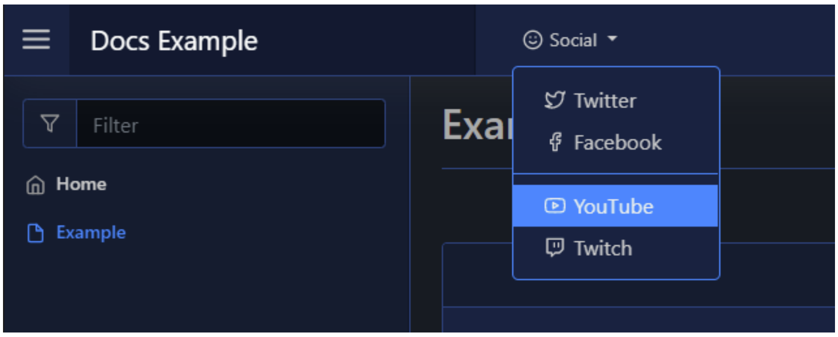
Links in dropdowns work as above, and can be normal URLs or Dynamic links.

For example, the following dropdown displays a list of social websites to the user:

Example011.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark
$navDropdown = New-PodeWebNavDropdown -Name 'Social' -Icon 'smile' -Items @(
    New-PodeWebNavLink -Name 'Twitter' -Url 'https://twitter.com' -Icon 'twitter'
    New-PodeWebNavLink -Name 'Facebook' -Url 'https://facebook.com' -Icon 'facebook'
    New-PodeWebNavDivider
    New-PodeWebNavLink -Name 'YouTube' -Url 'https://youtube.com' -Icon 'youtube'
    New-PodeWebNavLink -Name 'Twitch' -Url 'https://twitch.tv' -Icon 'twitch'
 )
Set-PodeWebNavDefault -Items $navDropdown
Add-PodeWebPage -Name Example
}
```

Which would look like below:



Pages

There are 3 different kinds of pages in Pode.Web, which are defined below. Other than the Login page, the Home and normal Webpages can be populated with custom Layout/Element components. When you add pages to your site, they appear on the sidebar for navigation - unless they are specified to be hidden from the sidebar.

Login

To enable the use of a login page, and lock your site behind authentication is simple! First, just setup sessions and define the authentication method you want via the usual Enable-PodeSessionMiddleware, New-PodeAuthScheme and Add-PodeAuth in Pode. Then, pass the authentication name into Set-PodeWebLoginPage - and that's it!

Note

Since the login page uses a form to logging a user in, the best scheme to use is Forms: New-PodeAuthScheme -Form. OAuth also works, as the login page will automatically trigger the relevant redirects.

Example012.ps1 – WARNING !!! – **no entry control below**, will take any login !!!!

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Enable-PodeSessionMiddleware -Duration 120 -Extend

 New-PodeAuthScheme -Form | Add-PodeAuth -Name Example -ScriptBlock {
     param($username, $password)
         return @{
              User = @{
                   ID ='M0R7Y302'
                 Name = 'Morty'
                 Type = 'Human'
 }}}

 Set-PodeWebLoginPage -Authentication Example

}
```

The example above is just a proof of concept, but it is far from being secure. Please see the Example007, or Example013 for better password control routine !!!

By default the Pode icon is displayed as the logo, but you can change this by using the -Logo parameter; this takes a literal or relative URL to an image file.

IIS

If you're hosting the site using IIS, and want to use Windows Authentication within IIS, then you can setup authentication in Pode.Web via Set-PodeWebAuth. This works similar to Set-PodeWebLoginPage, and sets up authentication on the pages, but it doesn't cause a login page or the sign-in/out buttons to appear. Instead, Pode.Web gets the session from IIS, and then displays the logged in user at the top - similar to how the login page would after a successful login.

```
Enable-PodeSessionMiddleware -Duration 120 -Extend
Add-PodeAuthIIS -Name Example
Set-PodeWebAuth -Authentication Example
```

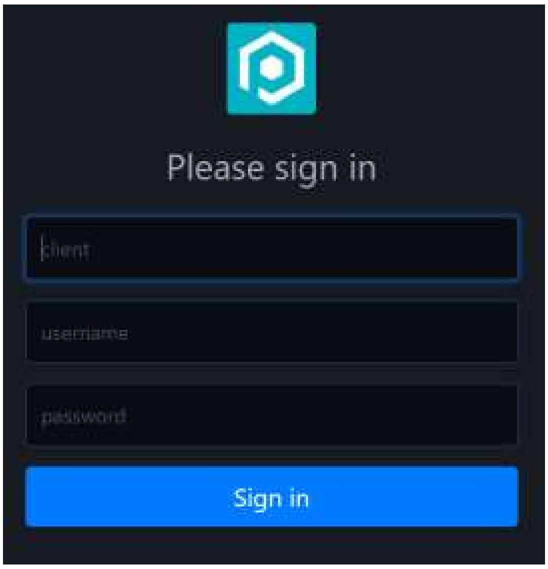
Custom Fields

By default the Login page will display a login form with Username and Password inputs. This can be overridden by supplying custom Layouts and Elements to the -Content parameter of Set-PodeWebLoginPage. Any custom content will be placed between the "Please sign in" message and the "Sign In" button.

Example013.ps1 – use 3x admin to login

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

# setup sessions
Enable-PodeSessionMiddleware -Duration 120 -Extend

# define a new custom authentication scheme, which needs a client, username, and password
$custom_scheme = New-PodeAuthScheme -Custom -ScriptBlock {
    param($opts)

    # get the client/user/password from the request's post data
    $client = $WebEvent.Data.client
    $username = $WebEvent.Data.username
    $password = $WebEvent.Data.password

    # return the data in a array, which will be passed to the validator script
    return @($client, $username, $password)
}

# now, add a new custom authentication validator using the scheme you created above
$custom_scheme | Add-PodeAuth -Name Example -ScriptBlock {
    param($client, $username, $password)

 if ($client -eq 'admin' -and $username -eq 'admin' -and $password -eq 'admin') {

    # check if the client is valid in some database
    return @{
        User = @{
            ID ='MOR7Y302'
            Name = 'Morty'
            Type = 'Human'
  } } }
    # return a user object (return $null if validation failed)
    return  @{ User = $user }
}

# set the login page to use the custom auth, and also custom login fields
Set-PodeWebLoginPage -Authentication Example -Content @(
    New-PodeWebTextbox -Type Text -Name 'client' -Id 'client' `
                       -Placeholder 'Client' -Required -AutoFocus -DynamicLabel
    New-PodeWebTextbox -Type Text -Name 'username' -Id 'username' `
                       -Placeholder 'Username' -Required -DynamicLabel
    New-PodeWebTextbox -Type Password -Name 'password' -Id 'password' `
                       -Placeholder 'Password' -Required -DynamicLabel
) }
```

This will render three fields like below:


Please sign in

client

username

password

Sign in

Home

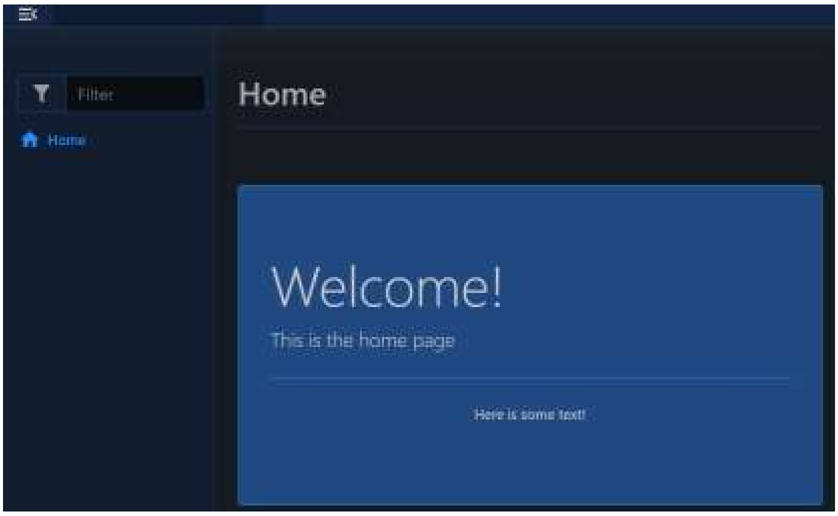
Every site is setup with a default empty home page. If you choose not to add anything to your home page, then Pode.Web will automatically redirect to the first Webpage.

To setup the home page with content, you use Set-PodeWebHomePage. At its simplest this just takes an array of -Layouts to render on the page. For example, if you wanted to add a quick Hero element to your home page:

Example014.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Set-PodeWebHomePage -Layouts @(
     New-PodeWebHero -Title 'Welcome!' -Message 'This is the home page' -Content @(
        New-PodeWebText -Value 'Here is some text!' -InParagraph -Alignment Center
) ) }
```

Which would look like below:



Home

Welcome!

This is the home page

Here is some text!

If you want to hide the title on the home page, you can also pass the -NoTitle option with the Set-PodeWebHomePage.

Webpage

By adding a page to your site, Pode.Web will add a link to it on your site's sidebar navigation. You can also group pages together so you can collapse groups of them. To add a page to your site you use Add-PodeWebPage, and you can give your page a -Name and an -Icon to display on the sidebar. Pages can either be **static** or **dynamic**.

Note

The -Icon is the name of a Material Design Icon, a list of which can be found on their website (https://pictogrammers.github.io/@mdi/font/5.4.55/).

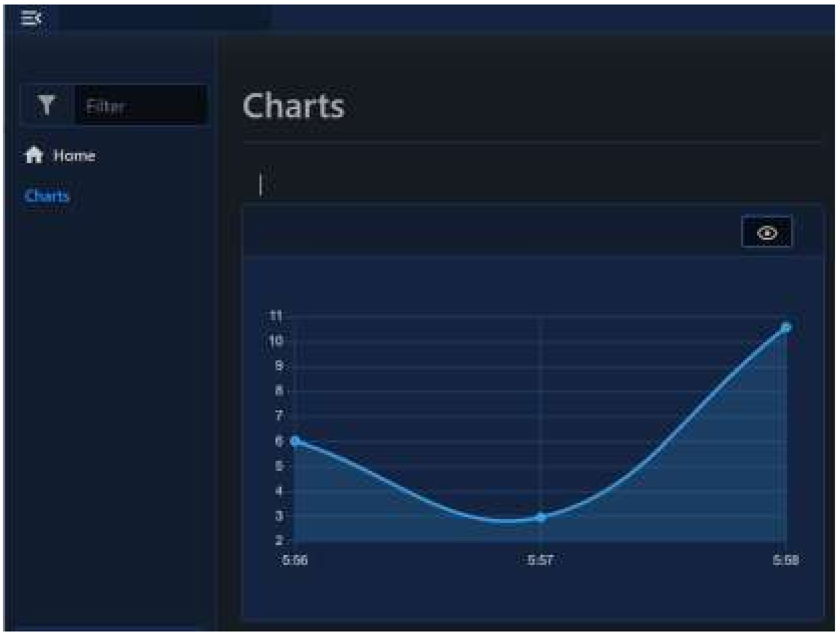
When supplyig the name, just supply the part after mdi-. For example, mdi-github should be -Icon 'github' and not -Icon 'mdi-github' !!

For example, to add a simple Charts page to your site, to show a **Windows** counter:

Example015.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Charts -Icon 'bar-chart-2' -Layouts @(
     New-PodeWebCard -Content @(
         New-PodeWebCounterChart -Counter '\Processor(_Total)\% Processor Time'
) ) }
```

If you will wait for few seconds, the chart below will be rendered:



You can split up your pages into different .ps1 files, if you do and you place them within a /pages directory, then Use-PodeWebPages will auto-load them all for you.

Link

If you just need to place a redirect link into the sidebar, then use Add-PodeWebPageLink. This works in a similar way to Add-PodeWebPage, but takes either a flat -Url to redirect to, or a -ScriptBlock that you can return output actions from - *not* layout/element components. Page links can also be grouped, like normal pages.

Flat URLs:

```
Add-PodeWebPageLink -Name Google -Url 'https://www.google.com' -Icon 'google' -NewTab
```

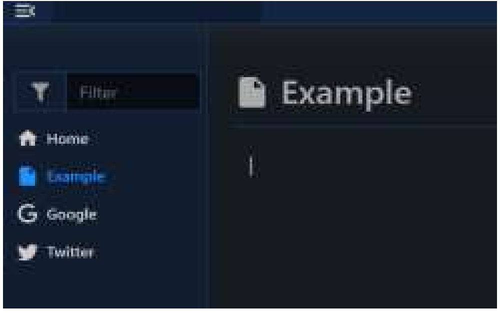
Or a dynamic link:

```
Add-PodeWebPageLink -Name Twitter -Icon Twitter -ScriptBlock {
   Move-PodeWebUrl -Url 'https://twitter.com' -NewTab
}
```

Example016.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Example

 # Static
 Add-PodeWebPageLink -Name Google -Url 'https://www.google.com' -Icon 'google' -NewTab

 # Dynamic
 Add-PodeWebPageLink -Name Twitter -Icon Twitter -ScriptBlock {
     Move-PodeWebUrl -Url 'https://twitter.com' -NewTab
} }
```

The result of the script above:



Group

You can group multiple pages together on the sidebar by using the -Group parameter on Add-PodeWebPage. This will group pages together into a collapsible container.

Help Icon

A help icon can be displayed to the right of the page's title by supplying a -HelpScriptBlock to Add-PodeWebPage. This scriptblock is used to return output actions such as: displaying a modal when the help icon is clicked; redirect the user to a help page; or any other possible actions to help a user out.

Static

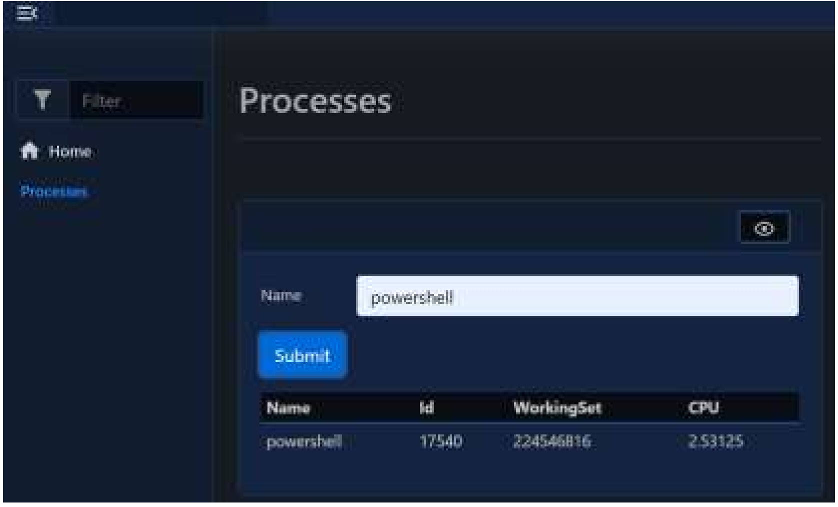
A static page is one that uses just -Layouts; this is a page that will render the same layout/element components on every page load, regardless of payload or query parameters supplied to the page.

For example, this page will always render a form to search for processes:

Example017.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Processes -Icon Activity -Layouts @(
     New-PodeWebCard -Content @(
         New-PodeWebForm -Name 'Search' -ScriptBlock {
             $p= Get-Process -Name $WebEvent.Data.Name -ErrorAction Ignore |
                     Select-Object Name, ID, WorkingSet, CPU
                             $p | Out-podewebTable
                           # $p | Out-PodeWebTextbox -Multiline -Preformat -ReadOnly
         } -Content @(
             New-PodeWebTextbox -Name 'Name'
) ) ) }
```

Example above was trying to put something to the text box. But this is not working. It is far more easier to put outpout into table.



Name: powershell

Name	Id	WorkingSet	CPU
powershell	17540	224546816	2.53125

Dynamic

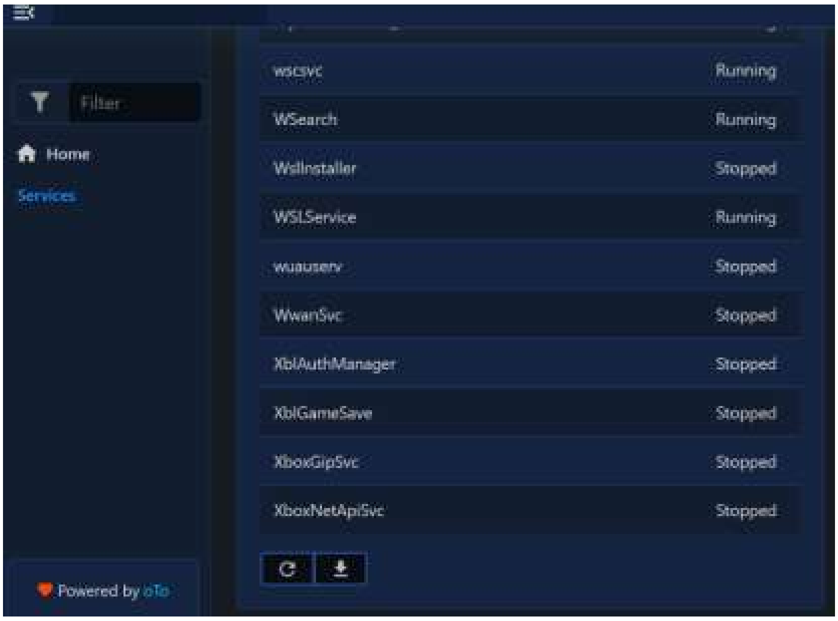
Add dynamic page uses a -ScriptBlock instead of -Layouts, the scriptblock lets you render different layout/element components depending on query/payload data in the $WebEvent. The scriptblock also has access to a $PageData object, containing information about the current page - such as Name, Group, Access, etc.

For example, the below page will render a table of services if a value query parameter is not present. Otherwise, if it is present, then a page with a code-block showing information about the service is displayed:

Example018.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Services -Icon Settings -ScriptBlock {
     $value = $WebEvent.Query['value']

     # table of services
     if ([string]::IsNullOrWhiteSpace($value)) {
         New-PodeWebCard -Content @(
             New-PodeWebTable -Name 'Services' -DataColumn Name -Click -ScriptBlock {
                 foreach ($svc in (Get-Service)) {
                     [ordered]@{
                         Name = $svc.Name
                         Status = "$($svc.Status)"
     } } } ) }
     # code-block with service info
     else {
           $svc = Get-Service -Name $value | Out-String

           New-PodeWebCard -Name "$($value) Details" -Content @(
               New-PodeWebCodeBlock -Value $svc -NoHighlight
) } } }
```

The example above will display services statuses instantly. The table is not sortable, but you can already export contents into csv. And you can also click on each service, to get more info.



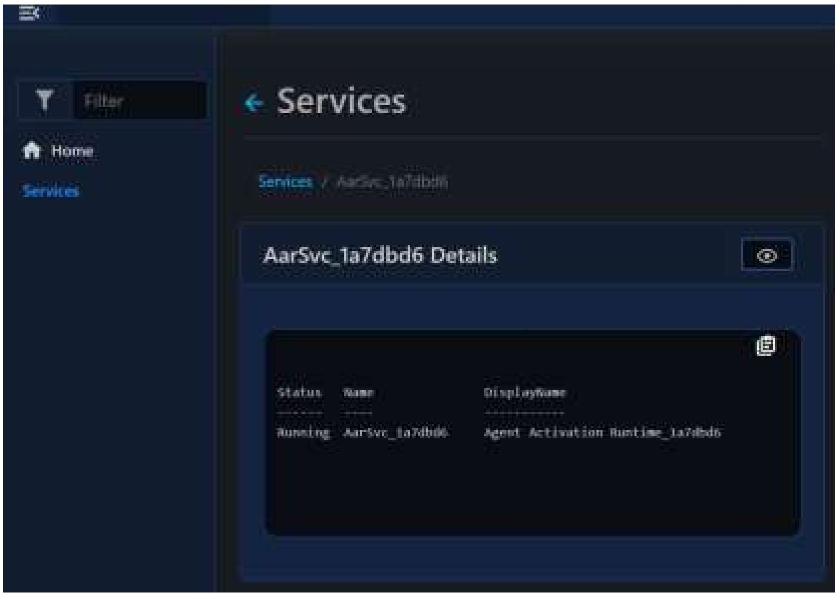
wscsvc	Running
WSearch	Running
WslInstaller	Stopped
WSLService	Running
wuauserv	Stopped
WwanSvc	Stopped
XblAuthManager	Stopped
XblGameSave	Stopped
XboxGipSvc	Stopped
XboxNetApiSvc	Stopped

You can also supply -Layouts while using -ScriptBlock. If the scriptblock returns no data, then whatever is supplied to -Layouts is treated as the default content for the page.

For example, the below is the same as the above example, but this time the table is set using -Layouts:

Example019.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

$servicesTable = New-PodeWebCard -Content @(
    New-PodeWebTable -Name 'Services' -DataColumn Name -Click -ScriptBlock {
        foreach ($svc in (Get-Service)) {
            [ordered]@{
                Name = $svc.Name
                Status = "$($svc.Status)"
            }
        }
    }
)

Add-PodeWebPage -Name Services -Icon Settings -Layouts $servicesTable -ScriptBlock {
    $value = $WebEvent.Query['value']

    # use default layouts - in this case, the services table
    if ([string]::IsNullOrWhiteSpace($value)) {
        return
    }

    # code-block with service info
    $svc = Get-Service -Name $value | Out-String

    New-PodeWebCard -Name "$($value) Details" -Content @(
        New-PodeWebCodeBlock -Value $svc -NoHighlight
) } }
```



```
Status   Name              DisplayName
------   ----              -----------
Running  AarSvc_1a7dbd6    Agent Activation Runtime_1a7dbd6
```

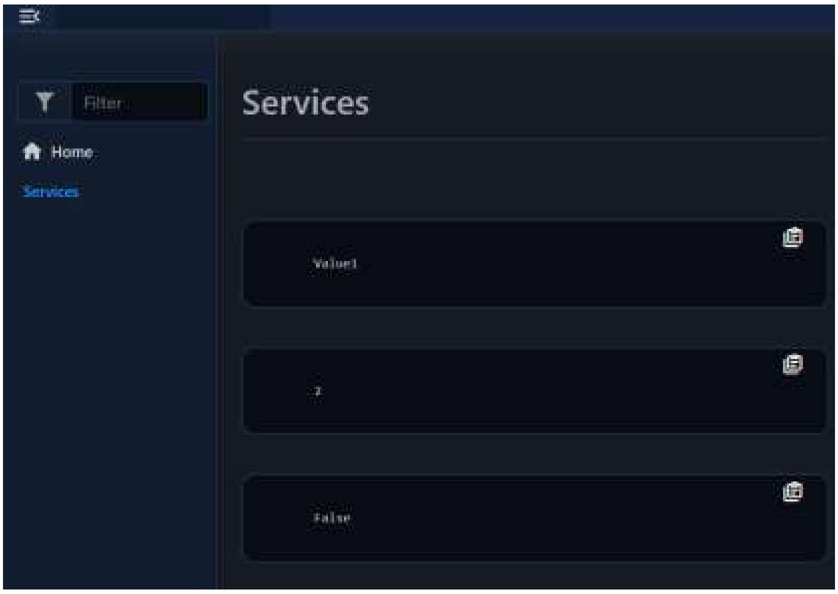
You can also pass values to the scriptblock by using the -ArgumentList parameter. This accepts an array of values/objects, and they are supplied as parameters to the scriptblock:

Example020.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark
 Add-PodeWebPage -Name Services -Icon Settings -ArgumentList 'Value1', 2, $false `
                -ScriptBlock {
     param($value1, $value2, $value3)
     # $value1 = 'Value1'
     # $value2 = 2
     # $value3 = $false
     New-PodeWebCodeBlock -Value $value1 -NoHighlight
     New-PodeWebCodeBlock -Value $value2 -NoHighlight
```

```
    New-PodeWebCodeBlock -Value $Value3 -NoHighlight
} }
```

Since this example would normally do nothing obvious, I redirected ‘positional parameters’ to the ‘PodeWebCodeBlock’, which can be created on the fly and is accepting flat-text values.



```
Value1
```

```
2
```

```
False
```

No Authentication

If you add a page when you've enabled authentication, you can set a page to be accessible without authentication by supplying the -NoAuth switch to Add-PodeWebPage.

If you do this and you add all elements/layouts dynamically (via -ScriptBlock), then there's no further action needed.

If however you're added the elements/layouts using the -Layouts parameter, then certain elements/layouts will also need their -NoAuth switches to be supplied (such as charts, for example), otherwise data/actions will fail with a 401 response.

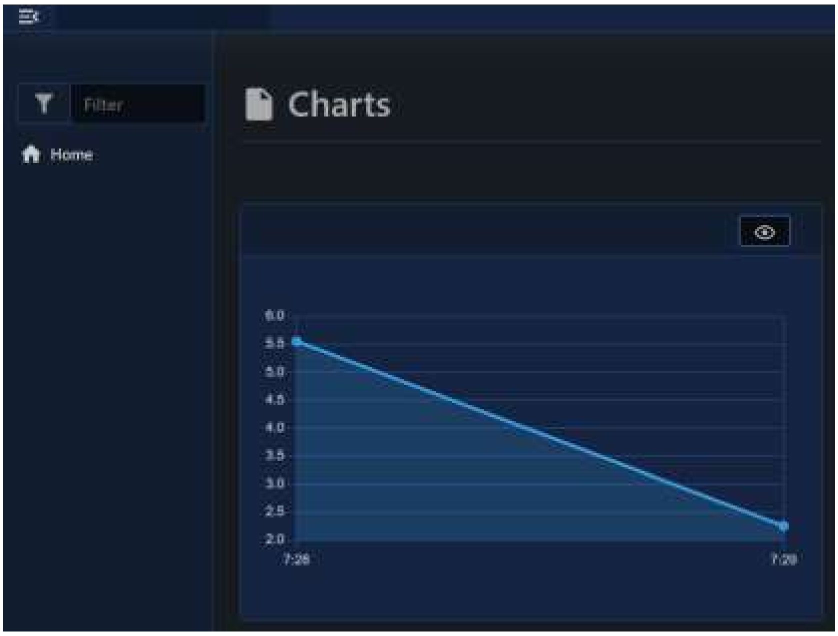
Sidebar

When you add a page by default it will show in the sidebar. You can stop pages/links from appearing in the sidebar by using the -Hide switch:

Example021.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Charts -Hide -Layouts @(
     New-PodeWebCard -Content @(
         New-PodeWebCounterChart -Counter '\Processor(_Total)\% Processor Time'
) ) }
```

Just wait for few seconds to see the result. A ‘line-chart’ will draw CPU status continuously.

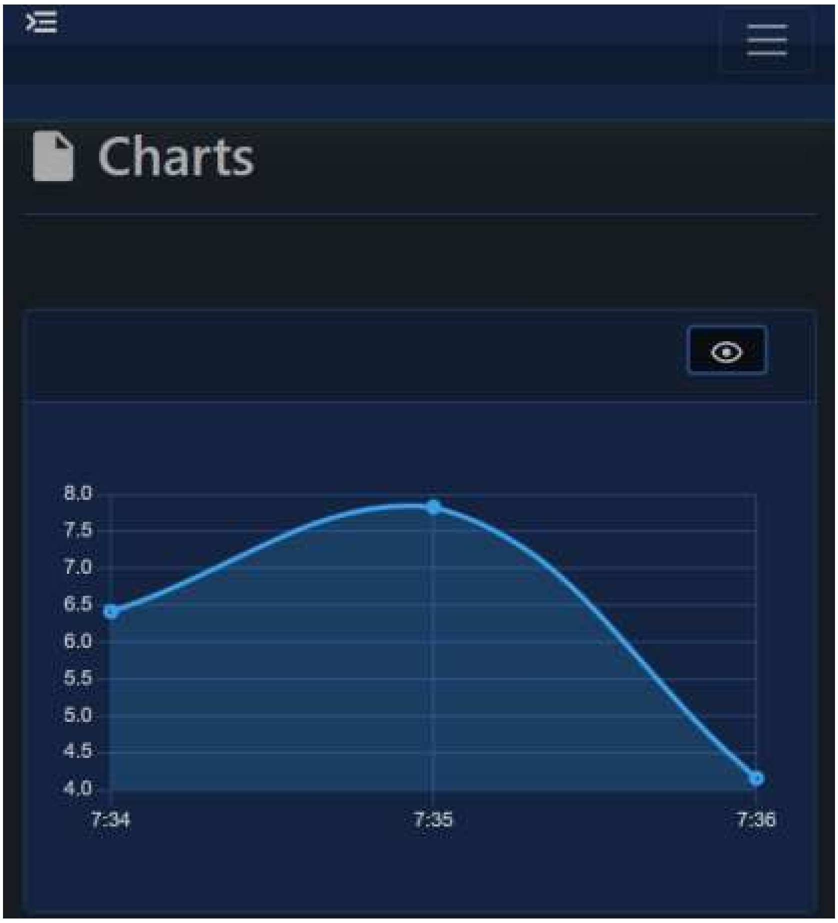


Alternatively, you can also hide the sidebar on a page by using the -NoSidebar switch; useful for dashboard pages:

Example022.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark
 Add-PodeWebPage -Name Charts -NoSidebar -Layouts @(
     New-PodeWebCard -Content @(
         New-PodeWebCounterChart -Counter '\Processor(_Total)\% Processor Time'
     )
 )
}
```

This example does the same as the previous one, except it also hides the side-bar.


Charts
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
7:34
7:35
7:36

Convert / Display Module

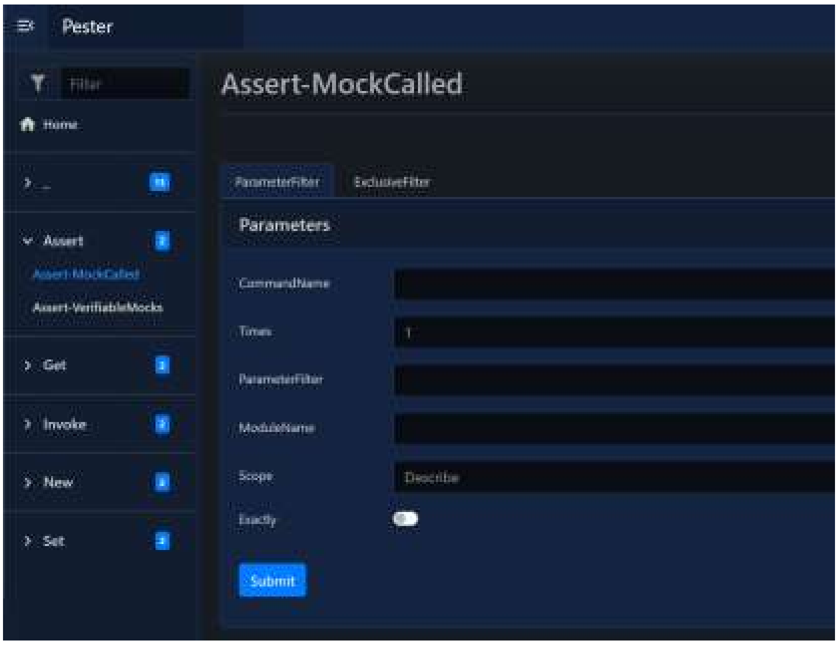
Similar to how Pode has a function to convert a Module to a REST API; Pode.Web has one that can convert a Module into Web Pages: ConvertTo-PodeWebPage!

For example, if you wanted to make a web portal for the Pester module:

Example022b.ps1

```
Start-PodeServer {
    # add a simple endpoint
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http

    # set the use of templates
    Use-PodeWebTemplates -Title 'Pester'

    # convert module to pages / forms
    ConvertTo-PodeWebPage -Module Pester -GroupVerbs
}
```

This one is really something. Just don't use it on a heavy-duty module like full Vmware.PowerCLI 😉, but you can try **Pode.web** and it will draw-out every option available. I have also tried it just on **Pode** but it ended with errors. Then I have also tried **VMware.VimAutomation.Core** and it was working. For more modules on Windows you can browae the following path:
C:\Program Files\WindowsPowerShell\Modules
or you can list all available modules
get-installedmodule 2>$null |select Name, Version



CommandName
Times 1
ParameterFilter
ModuleName
Scope Describe
Exactly

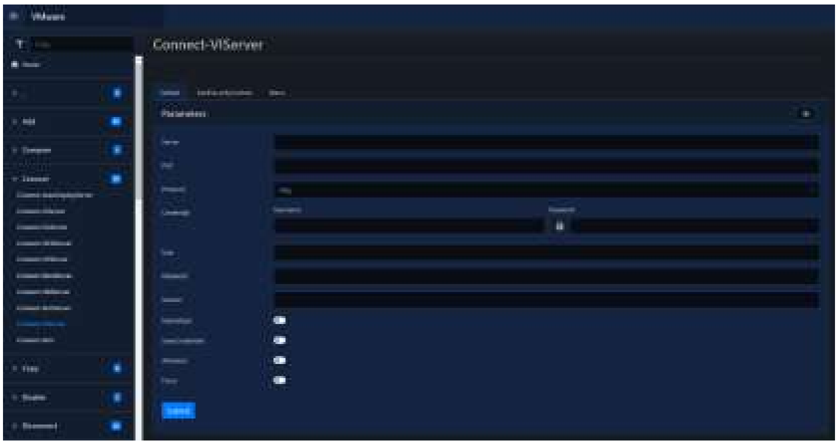
More on PESTER:
https://github.com/pester/Pester
https://pester.dev/docs/quick-start

Loading many modules from VMware PowerCLI, will join similar groups of items from different modules. You need to have **vmware.powercli** installed.

Example022c.ps1

```
Start-PodeServer {
    # add a simple endpoint
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http

    # set the use of templates
    Use-PodeWebTemplates -Title 'VMware'

    # convert module to pages

    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Core     -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.DeployAutomation       -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.ImageBuilder           -GroupVerbs

    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Cis.Core -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Common   -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.License  -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Sdk      -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Storage  -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.vds      -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Cloud    -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Hcx      -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.HorizonView -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Nsxt     -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Security -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Srm      -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.StorageUtility -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.Vmc      -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.vROps    -GroupVerbs
    ConvertTo-PodeWebPage -Module VMware.VimAutomation.WorkloadManagement  -GroupVerbs
}
```

This was done for testing purposes only. After 60 seconds on i7, 16GB RAM, 4 cores (8 logical CPUs), using aditional GPU, which was not used at all, the server has finally loaded …
You can perform the same operation just on module that you really need, to speed up the process.



Events

The Login, Home and Webpages support registering the following events, and they can be registered via Register-PodeWebPageEvent:

Name	Description
Load	Fires when the page has fully loaded, including js/css/etc.
Unload	Fires when the has fully unloaded/closed
BeforeUnload	Fires just before the page is about to unload/close

To register an event for each page type:

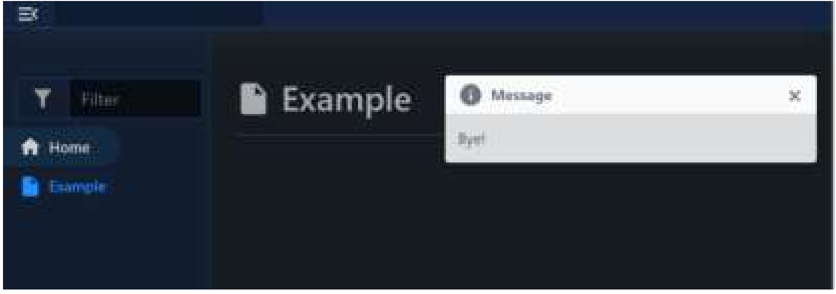
- Login: you'll need to use -PassThru on Set-PodeWebLoginPage and pipe the result in Register-PodeWebPageEvent.
- Home: you'll need to use -PassThru on Set-PodeWebHomePage and pipe the result in Register-PodeWebPageEvent.
- Webpage: you'll need to use -PassThru on Add-PodeWebPage and pipe the result in Register-PodeWebPageEvent.

For example, if you want to show a message on a Webpage just before it closes:

Example023.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Example -Layouts $some_layouts -PassThru |
     Register-PodeWebPageEvent -Type BeforeUnload -ScriptBlock {
         Show-PodeWebToast -Message "Bye!"
} }
```

While running example above, try to click links like: Home, Example, Home, to see the message displayed below.



Example

Message

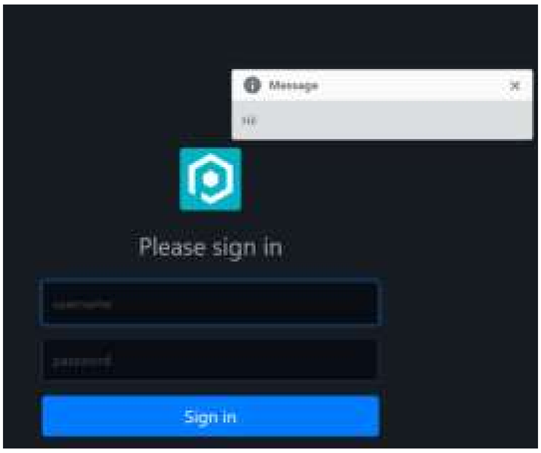
Bye!

Or on the Login page, after it's finished loading (note: you will need to use the -NoAuth switch):

Example024.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Enable-PodeSessionMiddleware -Duration 120 -Extend

 New-PodeAuthScheme -Form | Add-PodeAuth -Name Example -ScriptBlock {
     param($username, $password)
     if ($username -eq 'admin' -and $password -eq 'admin') {
         return @{
              User = @{
                   ID ='admin'
                   Name = 'admin'
 }}}}

 Set-PodeWebLoginPage -Authentication Example -PassThru |
     Register-PodeWebPageEvent -Type Load -NoAuth -ScriptBlock {
         Show-PodeWebToast -Message "Hi!"
} }
```

You will see the ‘toast’ message at load and after logout.



Message

Hi!

Pode.Web Security

Security

In Pode v2.6.0 support was added for HTTP security headers to be automatically added to requests, such as: Access Control, Cross-Origin, Content Security Policy, and more.

Pode.Web uses this feature to automatically set some default headers on requests, to make your site more secure.

Options

To set which security type to use, you can optionally specify a type via the `-Security` parameter on `Use-PodeWebTemplates`. The valid values are: None, Default, Simple, and Strict.

```
Use-PodeWebTemplates -Title 'Test' -Theme Dark -Security Default
```

In the case of the Default, Simple and Strict types: Pode.Web uses the inbuilt security types within Pode (simple for default), and adds some extra essential default Content Security Policy rules to allow Pode.Web to function:

```
* script-src: self, unsafe-inline
* style-src: self, unsafe-inline
* image-src: self, data
```

The `unsafe-inline` values for scripts and styles are required due to Pode.Web templates currently have inline JavaScript and CSS - this could change in the future, and the values removed.

None

This type is pretty self-explanatory, if specified Pode.Web will call `Remove-PodeSecurity` to remove all security headers.

Default

This type is the default that Pode.Web uses when no `-Security` is supplied. Under the hood this type uses the Simple security type within Pode, plus some extras:

- The default-src, script-src, style-src, and media-src for Content Security Policy are extended with `http` and `https`, to allow content to be retrieved externally
- The Cross-Origin headers are removed
- The essentials above

Simple

This is just the Simple security type within Pode, plus the essentials mentioned above.

Strict

This is just the Strict security type within Pode, plus the essentials mentioned above.

Content Not Loading

If you're using the Simple or Strict types, and you find that media isn't loading, then you likely need to add extra Content Security Policy rules. In the Default type, http/https is added to prevent this from occurring, so the same should work also:

```
Add-PodeSecurityContentSecurityPolicy `
    -Default 'http', 'https' `
    -Style 'http', 'https' `
    -Scripts 'http', 'https' `
    -Image 'http', 'https'
```

However, if you want to control it more granularly, then you'll need to specify the URLs for media appropriately. For example, if you were loading audio from https://samplelib.com then you'd need to add:

```
Add-PodeSecurityContentSecurityPolicy -Media 'https://samplelib.com'
```

The same also applies to styles and scripts as well.

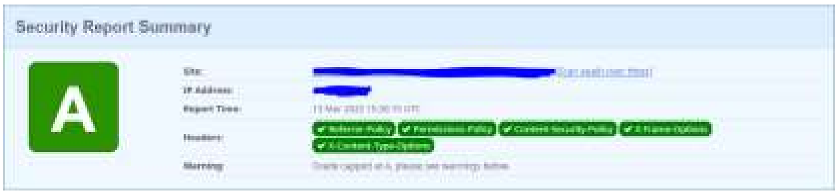
Misc

HSTS

If you need to enable HSTS for your site, you can do so via supplying the -UseHSTS switch on Use-PodeWebTemplates.

Rating

Using securityheaders.com on a Pode.Web site hosted publicly with the -Security set as Default, an A rating is achieved:



Security Report Summary

A

Site: [redacted] (Scan again over http)

IP Address: [redacted]

Report Time: 13 Mar 2022 15:30:15 UTC

Headers: ✔ Referrer-Policy ✔ Permissions-Policy ✔ Content-Security-Policy ✔ X-Frame-Options ✔ X-Content-Type-Options

Warning: Grade capped at A, please see warnings below.

https://www.securityheaders.com

Sizes

Some -Width, -Height, and -Size parameters for components accept either plain number or a pure CSS value.

When a plain number is supplied these parameters will append a default unit, such as px or %, and use that for the CSS width/height. In some cases, if 0 is supplied then the default value will be auto. For components that support this, they will have a relevant "Size" section in their docs page.

For example, an Image element has a default unit of px for its width/height. If you wanted to add a image of 100px by 100px you could do:

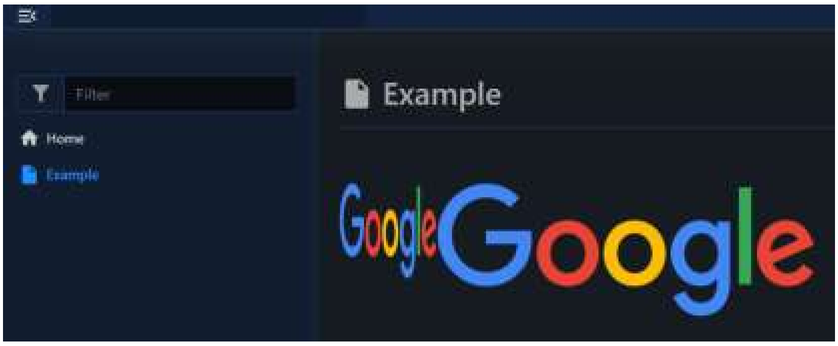
```
New-PodeWebImage -Source '/url/to/image.png' -Width 100 -Height 100
```

and Pode.Web will append the default px unit. If instead you wanted a image at 25% and 10em:

```
New-PodeWebImage -Source '/url/to/image.png' -Width '25%' -Height '10em'
```

Example025.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Example -ScriptBlock {
     New-PodeWebImage -Width 100 -Height 100        `
        -Source 'https://www.google.com/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png'
     New-PodeWebImage -Width '25%' -Height '10em'  `
        -Source 'https://www.google.com/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png'
} }
```



Themes

Pode.Web comes with some inbuilt themes (including a dark theme!), plus the ability to use custom themes. Themes are set using `Use-PodeWebTemplates`.

Inbuilt

Pode.Web has 4 inbuilt themes:

- Light
- Dark
- Terminal (black/green)
- Auto (uses the user's system theme)

Path on windows:

C:\Program Files\WindowsPowerShell\Modules\Pode.Web\0.8.3\Templates\Public\styles

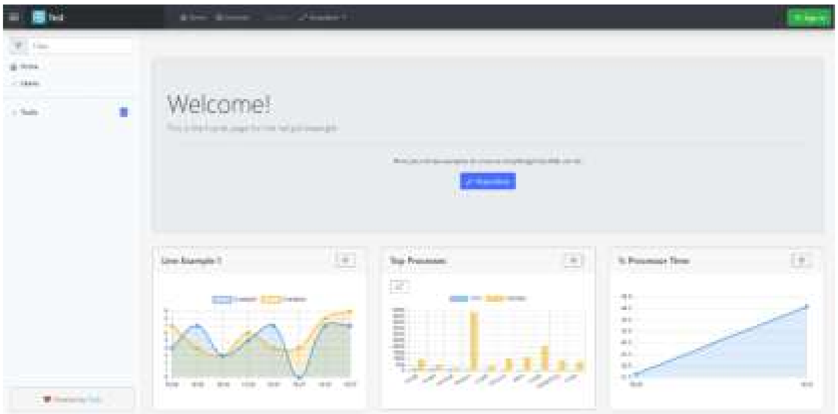
Building styles for jQuery-UI online:

https://jqueryui.com/themeroller

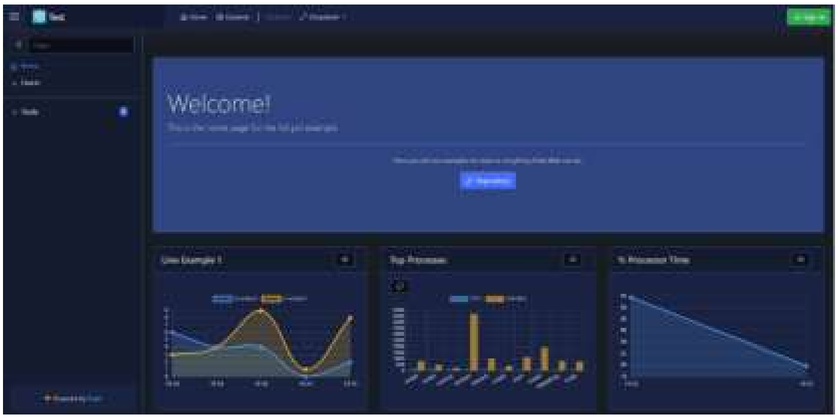
For example, to use the `Terminal` theme:

```
Use-PodeWebTemplates -Title 'Example' -Theme Terminal
```

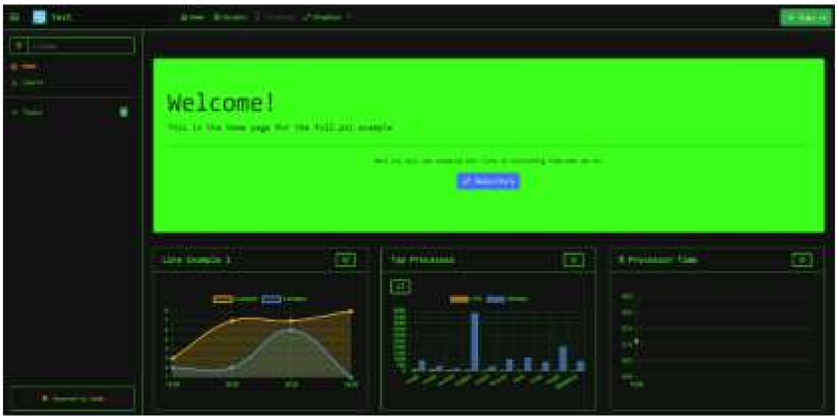
Light



Dark



Terminal



Custom

And there is also a Custom theme option, which will let you add your own custom themes via Add-PodeWebCustomTheme, which you can supply a URL for the custom CSS:

```
Use-PodeWebTemplates -Title 'Example' -Theme Custom
Add-PodeWebCustomTheme -Name 'Custom1' -Url 'https://example.com/custom-theme.css'
```

The first custom theme you add is the default theme.

Set the Theme

You can override the default theme on a per user basis, by either setting the pode.web.theme cookie on the frontend, or by setting a Theme property in the user's authentication object:

```
New-PodeAuthScheme -Form | Add-PodeAuth -Name Example -ScriptBlock {
  param($username, $password)
  return @{
    User = @{
      Theme = 'Light'
    }
  }
}
```

The theme that gets used is defined in the following order:

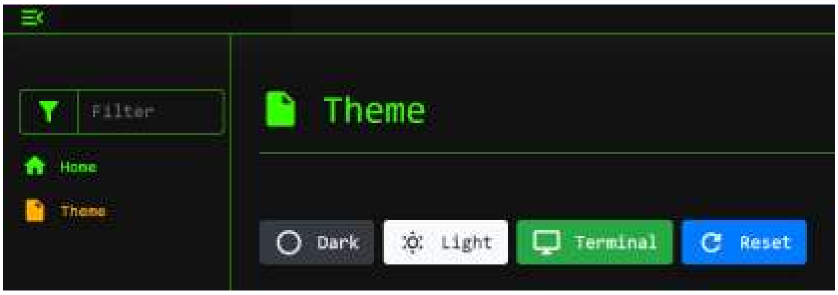
1. Cookie
2. User Authentication
3. Server Default

You can also use the Update-PodeWebTheme and Reset-PodeWebTheme output actions.

Example026.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Theme -ScriptBlock {
     New-PodeWebButton -Name 'Dark' -Icon 'moon-new' -Colour Dark `
                       -ScriptBlock {
                        Update-PodeWebTheme -Name Dark }
     New-PodeWebButton -Name 'Light' -Icon 'weather-sunny' -Colour Light `
                       -ScriptBlock {
                        Update-PodeWebTheme -Name Light }
     New-PodeWebButton -Name 'Terminal' -Icon 'monitor' -Colour Green `
                       -ScriptBlock {
                        Update-PodeWebTheme -Name Terminal }
     New-PodeWebButton -Name 'Reset' -Icon 'refresh'
                       -ScriptBlock {
                        Reset-PodeWebTheme }
} }
```

You will see four buttons running the example above (One for each possible theme and also button to reset theme to predefined). If you will close the server and run it again, the last theme set, will continue to persist even if you close or reload the browser.



Pode.Web Elements

Alert

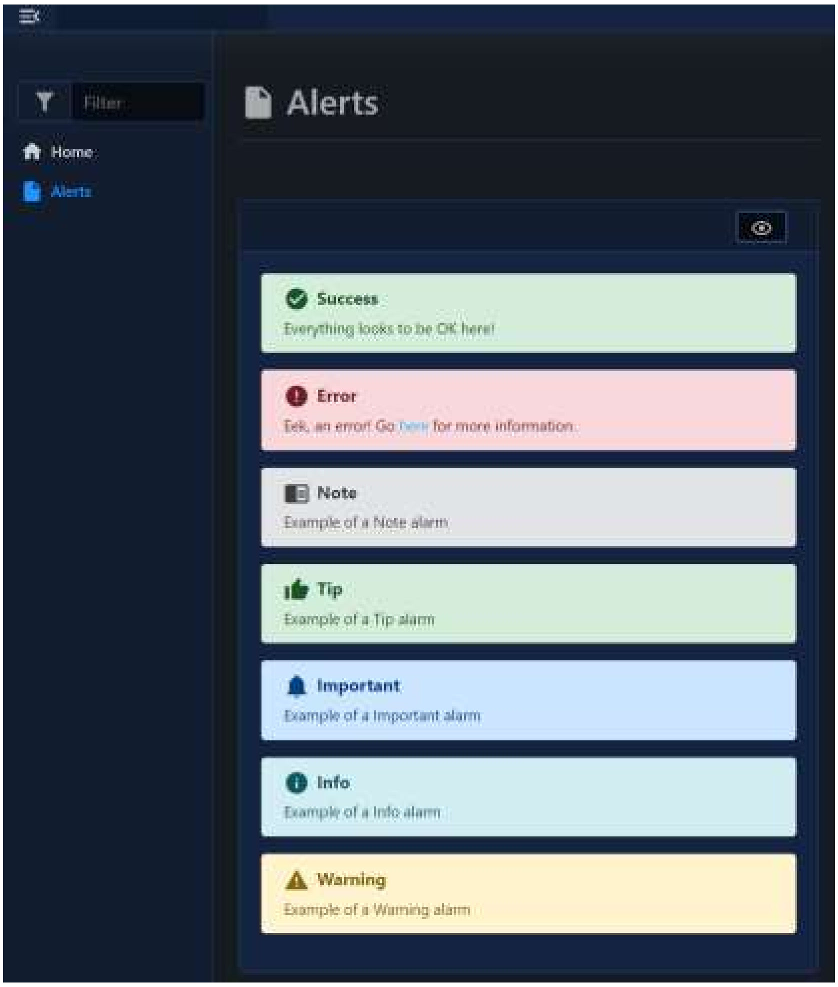
Support	
Events	No

An alert is a colour block containing information text; alerts can for warning, errors, tips, etcs. To add an alert you use New-PodeWebAlert, and supply either a -Value or -Content:

Example027.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Alerts -ScriptBlock {
     New-PodeWebCard -Content @(
         New-PodeWebAlert -Value 'Everything looks to be OK here!' -Type Success
         New-PodeWebAlert -Type Error -Content @(
             New-PodeWebText -Value 'Eek, an error! Go'
             New-PodeWebLink -Value 'here' -Source 'https://google.com'
             New-PodeWebText -Value 'for more information.'
         )
         New-PodeWebAlert -Value 'Example of a Note alarm' -Type Note
         New-PodeWebAlert -Value 'Example of a Tip alarm' -Type Tip
         New-PodeWebAlert -Value 'Example of a Important alarm' -Type Important
         New-PodeWebAlert -Value 'Example of a Info alarm' -Type Info
         New-PodeWebAlert -Value 'Example of a Warning alarm' -Type Warning
) } }
```

Which will look like below:



Alerts

Success

Everything looks to be OK here!

Error

Eek, an error! Go here for more information.

Note

Example of a Note alarm

Tip

Example of a Tip alarm

Important

Example of a Important alarm

Info

Example of a Info alarm

Warning

Example of a Warning alarm

Audio

Support	
Events	Yes

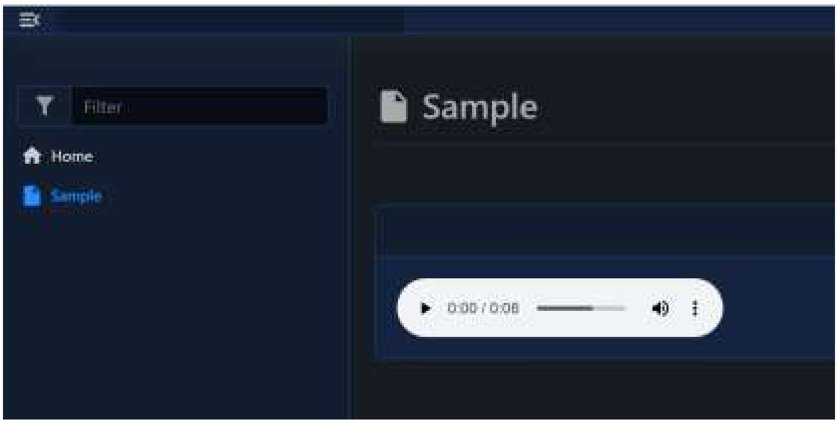
To play audio clips on your site you can use New-PodeWebAudio. The audio can be set to auto-play and loop, and can also accept multiple sources and tracks.

The sources specified can be done so via -Source and New-PodeWebAudioSource. Sources can only be of type MP3, OGG and WAV, and at least one source must be specified:

Example028.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Sample -ScriptBlock {
     New-PodeWebCard -Content @(
         New-PodeWebAudio -Name 'example' `-Source @(
             New-PodeWebAudioSource -Url `
                 'https://samplelib.com/lib/preview/mp3/sample-6s.mp3'
) ) } }
```

Which looks like below:



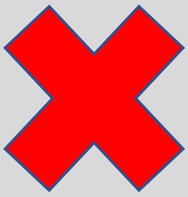
See the Example005.ps1 to see, how to implement the same using -views option.

Any optional tracks you wish to specify can be done via -Track and New-PodeWebMediaTrack. Tracks can only be of type VTT, and can be used for subtitles, captions, metadata, etc.

The -Language is mandatory if the track's -Type is subtitles, and should be a 2-letter language code.

Example029.ps1 (no subtitles will appear here !!!)

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Sample -ScriptBlock {

 New-PodeWebCard -Content @(
     New-PodeWebAudio -Name 'example' -Source @(
         New-PodeWebAudioSource -Url `
             'https://samplelib.com/lib/preview/mp3/sample-6s.mp3'
     ) `
 -Track @(
          New-PodeWebMediaTrack -Url `
               '/english.vtt' -Language 'en' `
                              -Title 'English' `
                              -Type 'subtitles' -Default
         )
 )

 }
}
```



Size

The -Width of an audio element has the default unit of %. If 0 is specified then 20% is used instead. Any custom value such as 100px can be used, but if a plain number is used then % is appended.

Events

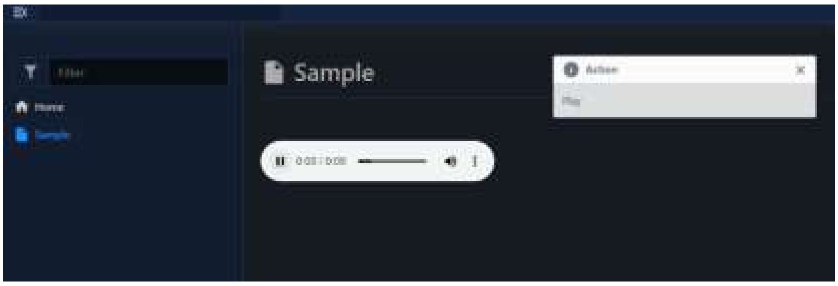
The following specific events are supported by the Audio element, and can be registered via Register-PodeWebMediaEvent:

Name	Description
CanPlay	Fires when the browser is ready to play the audio
Ended	Fires when the audio has finished playing, unless looping
Pause	Fires when the audio is paused
Play	Fires when the audio is played, or un-paused

Example030.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Sample -ScriptBlock {

     New-PodeWebAudio -Name 'example' -Source @(
         New-PodeWebAudioSource -Url `
         'https://samplelib.com/lib/preview/mp3/sample-6s.mp3'
     ) |
        Register-PodeWebMediaEvent -Type Play -ScriptBlock {
                 Show-PodeWebToast -Title 'Action' -Message $EventType
} } }
```

This one will show a Toast message while pressing start.



Action

Play

Badge

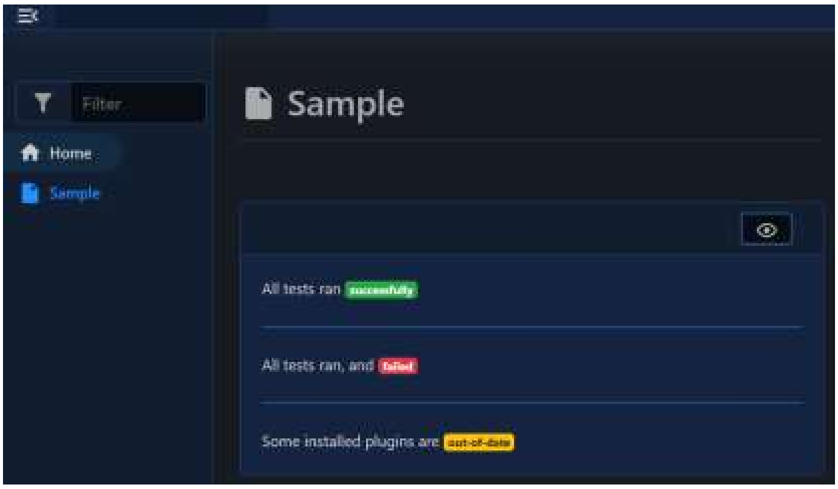
Support	
Events	Yes

A Badge is just normal text, but has a styled/coloured background. You can add a badge to your page using New-PodeWebBadge:

Example031.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Sample -ScriptBlock {
     New-PodeWebCard -Content @(
         New-PodeWebText -Value 'All tests ran'
         New-PodeWebBadge -Value 'successfully' -Colour Green
         New-PodeWebLine
         New-PodeWebText -Value 'All tests ran, and'
         New-PodeWebBadge -Value 'failed' -Colour Red
         New-PodeWebLine
         New-PodeWebText -Value 'Some installed plugins are'
         New-PodeWebBadge -Value 'out-of-date' -Colour Yellow
) } }
```

Which looks like below:



All tests ran successfully

All tests ran, and failed

Some installed plugins are out-of-date

Button

Support	
Events	No

To display a button on your page you use New-PodeWebButton; a button can either be dynamic and run custom logic via a -ScriptBlock, or it can redirect a user to a -Url.

Dynamic

A dynamic button is one that takes a custom -ScriptBlock, and when clicked will invoke that logic. You can run whatever you like, including output actions for Pode.Web to action against.

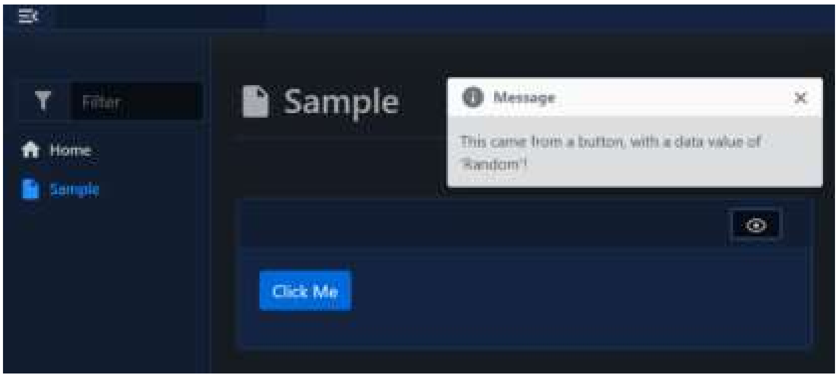
When using a dynamic button you can also supply a -DataValue, which is a way of supplying a special value/identity when the button is clicked. If supplied, this value is available in your scriptblock via $WebEvent.Data['Value'].

For example, the below button, when clicked, will display a toast message on the page:

Example032.ps1

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Sample -ScriptBlock {
     New-PodeWebCard -Content @(
         New-PodeWebButton -Name 'Click Me' `
                           -DataValue 'Random' -ScriptBlock {
             Show-PodeWebToast -Message `
                  "This came from a button, with a data value of `
                  '$($WebEvent.Data['Value'])'!"
} ) } }
```

Which will render page like below:



This came from a button, with a data value of 'Random'!

You can also pass values to the scriptblock by using the -ArgumentList parameter. This accepts an array of values/objects, and they are supplied as parameters to the scriptblock:

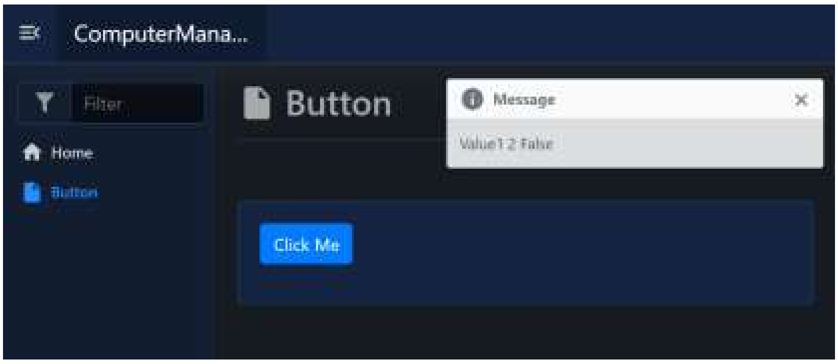
Example033.ps1– will not work as expected.

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Sample `
                -ScriptBlock {New-PodeWebCard -Content @(
                              New-PodeWebTextBox -name 'aa' -value $a
                              New-PodeWebTextBox -name 'bb' -value $b
                              New-PodeWebTextBox -name 'cc' -value $c
                              New-PodeWebButton `
                                 -Name 'Click Me' `
                                 -ArgumentList 'Value1', 2, 3 `
                                 -ScriptBlock {
                                               param($value1, $value2, $value3)
                                               $global:a=$value1
                                               $global:b=$value2
                                               $global:c=$value2
                                               Update-PodeWebTextBox -name 'aa' -value $a
                                               Update-PodeWebTextBox -name 'bb' -value $b
                                               Update-PodeWebTextBox -name 'cc' -value $c
                                               $a=0;$b=0;$c=0
                                              }
                              )
                             }
}
```

Example034.ps1

```
Start-PodeServer {
    # add a simple endpoint
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http

    # set the use of templates
    Use-PodeWebTemplates -Title 'ComputerManagement'

    Add-PodeWebPage -name Button -ScriptBlock {
        New-PodeWebContainer -Content @(
            New-PodeWebButton -Name 'Click Me' `
                              -ArgumentList 'Value1', 2, $false `
                              -ScriptBlock {
                param($value1, $value2, $value3)

                # $value1 = 'Value1'
                # $value2 = 2
                # $value3 = $false

               # New-PodeWebCodeBlock -Value $Value1 # This will not work !!!
               Show-PodeWebToast -Message "$value1 $value2 $value3"

} ) } }
```

Since I was unable to work with TextBox or CodeBlock element here, I used the Toast element, which works fine.



Message

Value1 2 False

URL

To have a button that simply redirects to another URL, all you have to do is supply -Url:

```
New-PodeWebCard -Content @(
   New-PodeWebButton -Name 'Repository' -Icon Link -Url 'https://github.com/Badgerati/Pode.Web'
)
```

New Tab

To open the URL in a new tab, supply the -NewTab switch:

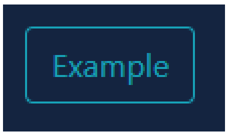
```
New-PodeWebButton -Name 'Repository' -Icon Link -Url 'https://github.com/Badgerati/Pode.Web' -NewTab
```

Outlined

By default a button will be displayed as a block colour, but you can show a button as an outline by using the -Outline switch:

```
New-PodeWebButton -Name 'Example' -Colour Cyan -Outline -ScriptBlock {
   # logic
}
```

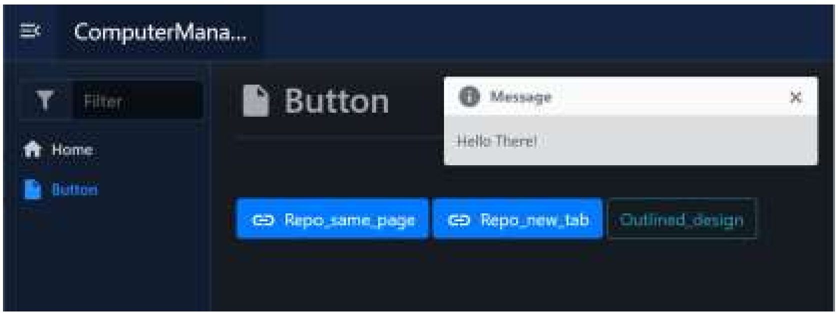
Which looks like below:



Example035.ps1

```
Start-PodeServer {
    # add a simple endpoint
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http

    # set the use of templates
    Use-PodeWebTemplates -Title 'ComputerManagement'

    Add-PodeWebPage -name Button -ScriptBlock {

        New-PodeWebButton -Name 'Repo_same_page' `
           -Icon Link -Url 'https://github.com/Badgerati/Pode.Web'
        New-PodeWebButton -Name 'Repo_new_tab' `
           -Icon Link -Url 'https://github.com/Badgerati/Pode.Web' -NewTab

        New-PodeWebButton -Name 'Outlined_design' -Colour Cyan -Outline -ScriptBlock {
            Show-PodeWebToast -Message 'Hello There!'
} } }
```

The script above will render page like this:



Message

Hello There!

Charts

Support	
Events	No

You can display data rendered as a chart by using New-PodeWebChart, and the following chart types are supported:

- Line (default)
- Bar
- Pie
- Doughnut

A chart gets its data from a supplied -ScriptBlock, more information below, and you can set a -MaxItems to be shown, and whether new data points should -Append to the chart. You can also -AutoRefresh a chart, to fetch new data every minute.

Data

To supply data to be rendered on a chart, you have to supply a -ScritpBlock which returns the appropriate data in the correct format; fortunately there's ConvertTo-PodeWebChartData to help with this format.

You can pass values to the scriptblock by using the -ArgumentList parameter. This accepts an array of values/objects, and they are supplied as parameters to the scriptblock:

```
New-PodeWebChart -Name 'Example Chart' -Type Line -ArgumentList 'Value1', 2, $false -ScriptBlock {
    param($value1, $value2, $value3)

    # $value1 = 'Value1'
    # $value2 = 2
    # $value3 = $false
}
```

Raw

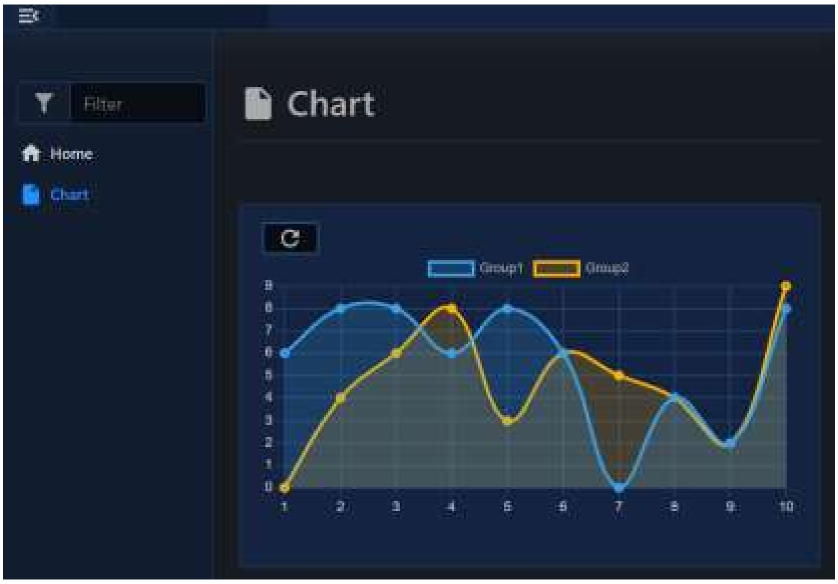
Before showing the ConvertTo function, the data format needed is as follows: the returned value has be an array of hashtables, with each hashtable requires a Key property, and a Values property that's an array of further hashtables. The Key is the value used on the X-axis, and the Values is an array of data points used on the Y-axis. The Values hashtables also has to contain a Key and a Value property - the Key here is the dataset group name, and the Value is the value on the Y-axis.

For example, the below will show a line chart consisting of 10 data points across 2 groups: Group1/Group2:

Example036.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {

    New-PodeWebContainer -Content @(
        New-PodeWebChart -Name 'Example Chart' -Type Line -ScriptBlock {
            return (1..10 | ForEach-Object {
              @{
                Key = $_ # x-axis value
                Values = @(
                    @{
                        Key = 'Group1'
                        Value = (Get-Random -Maximum 10) # y-axis value
                    }
                    @{
                        Key = 'Group2'
                        Value = (Get-Random -Maximum 10) # y-axis value
} ) } } ) } ) } }
```

which will render a chart that looks like below:



If you click the refresh button in the top-left corner, the -ScriptBlock will be re-invoked, and the chart updated.

ConvertTo-PodeWebChartData

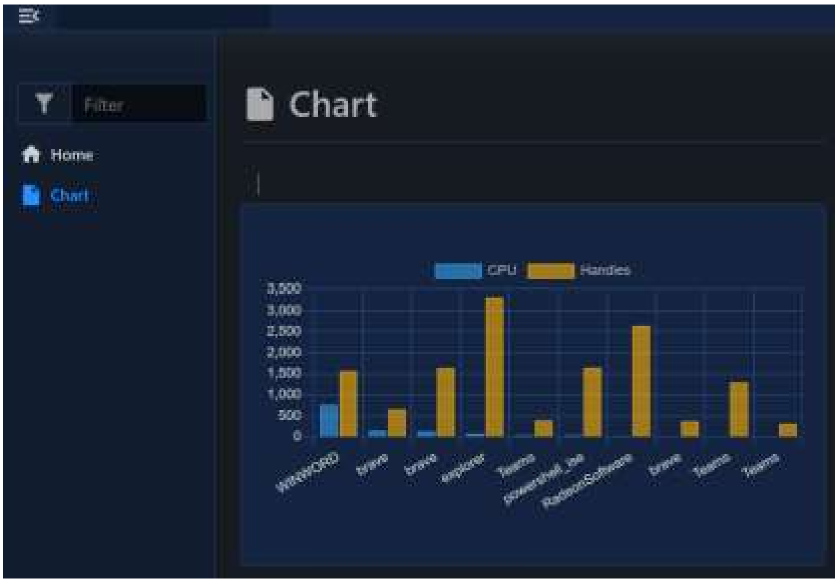
The ConvertTo-PodeWebChartData helps to simplify the above a raw format, by letting you convert data at the end of a pipeline. The function takes a -LabelProperty which is the name of a property in the input that should be used for the X-axis, and then a -DatasetProperty with is property names for Y-axis values.

For example, let's say we want to display the top 10 processes using the most CPU. We want to display the process name, and its CPU and Memory usage, *and* we want it to auto-refresh every minute:

Example037.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {

    New-PodeWebContainer -Content @(
    New-PodeWebChart -Name 'Top Processes' -Type Bar -AutoRefresh -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 10 |
            ConvertTo-PodeWebChartData -LabelProperty ProcessName -DatasetProperty
CPU,Handles
    }
    )
  }
 }
```

which renders a chart that looks like below:



Colours

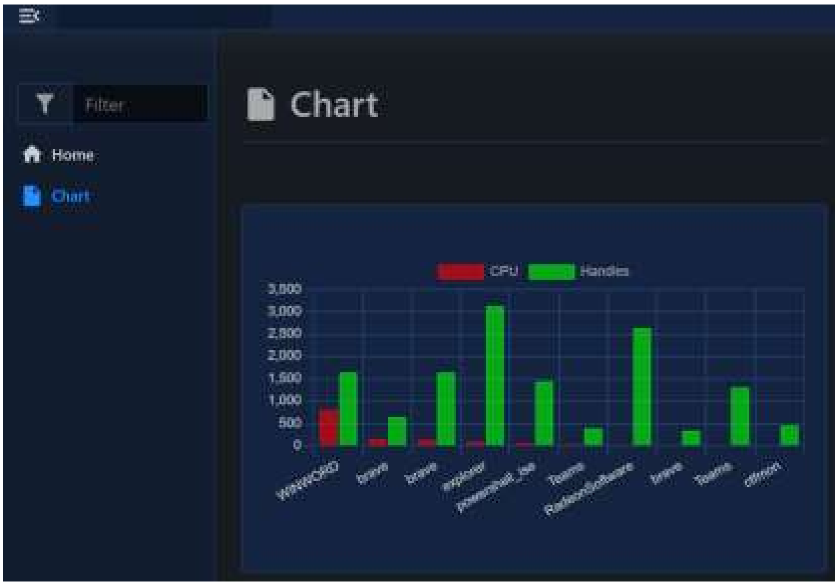
A chart has 15 default colours that they can be rendered using. These colours can be customised if required, by suppling an array of hex colour codes to the -Colours parameter.

For example, to render the below bar chart with reg/green bars, you could use:

Example038.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {

    New-PodeWebContainer -Content @(
        New-PodeWebChart -Name 'Top Processes' -Type Bar `
                         -AutoRefresh -Colours '#ff0000', '#00ff00' -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
                 Select-Object -First 10 |
                        ConvertTo-PodeWebChartData -LabelProperty ProcessName `
                                                   -DatasetProperty CPU, Handles
} ) } }
```

This will change bar graph colors.



Append

Now let's say we want -AutoRefresh a chart every minute, and we want it to display the current CPU usage, but only for the last 15 minutes. To do this, you have to supply -Append and any data returned from the -ScriptBlock will be appended to the chart instead. To restrict the data points to 15 minutes, we can supply -MaxItems 15.

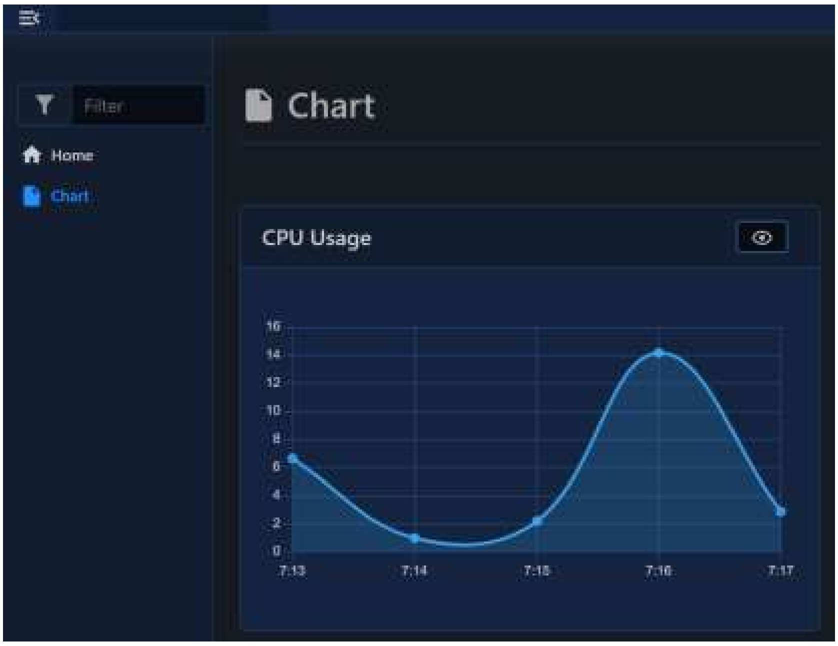
It would also be nice to have the X-axis labels be timestamps, and Pode.Web can automatically show a data points time stamp by using the -TimeLabels switch.

Appending data to chart example

Example039.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {

    New-PodeWebChart -Name 'CPU Usage' `
                     -Type Line -AutoRefresh -Append `
                     -TimeLabels -MaxItems 15 -AsCard `
                     -ScriptBlock {
                            return @{
                              Values = ((Get-Counter -Counter `
                                       '\Processor(_Total)\% Processor Time' `
                                       -SampleInterval 1 -MaxSamples `
                                       2).CounterSamples.CookedValue | `
                                       Measure-Object -Average).Average
} } } }
```

which renders a chart that looks like below:



First Load

The scriptblock has access to the $WebEvent, and when a chart requests its -ScriptBlock for the first time, $WebEvent.Data.FirstLoad will have a value of '1'. This will let you preload data on your chart on its first load, and then just return 1 data point thereafter to be appended.

Windows Counters

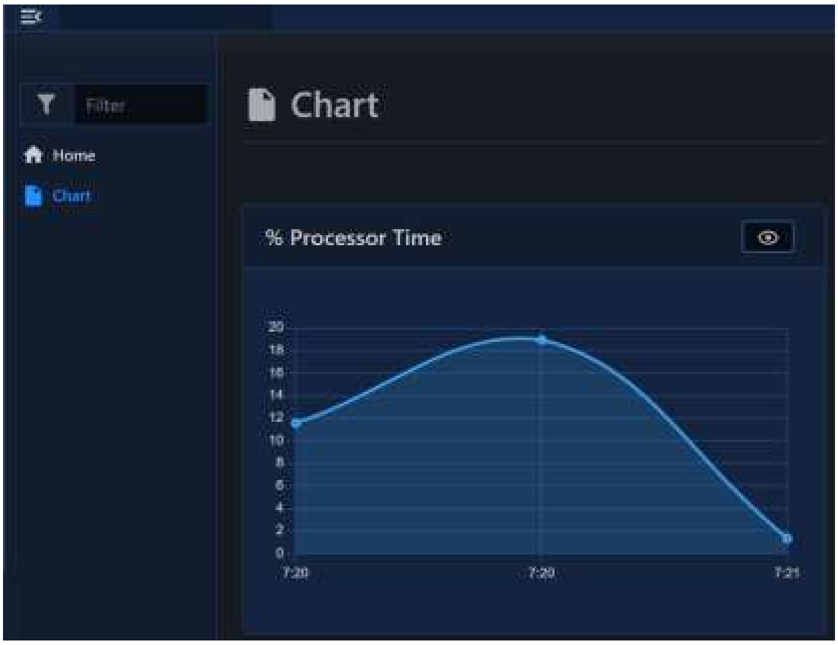
To display line charts for Windows Performance Counters more easily, Pode.Web has an inbuilt New-PodeWebCounterChart. When used, this automatically sets up an auto-refreshing, timestamped, appending line chart for you; all you have to do is supply the name of the counter, and it will show a chart for up to the last 30mins.

For example, taking the above CPU example, this would be it using the inbuilt helper:

Example040.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {

    New-PodeWebCounterChart -Counter '\Processor(_Total)\% Processor Time' -AsCard

} }
```

which renders a chart that looks like below:



Size

The -Height of a chart has the default unit of px. If 0 is specified then auto is used instead. Any custom value such as 10% can be used, but if a plain number is used then px is appended.

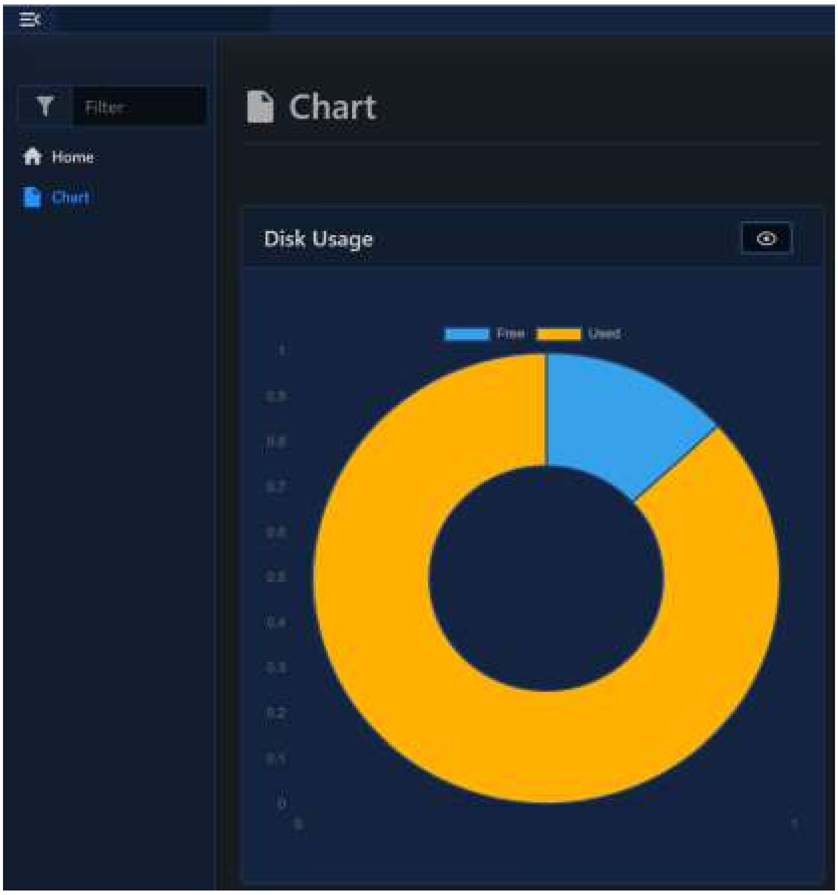
Pie Chart / Doughnut Chart

Here is used a concept of a “pre-defined” script-block. The $Doughnut variable, defined as:
$Doughnut = { some-logic-here }
is actually a pure text, passed to the script-block of the New-PodeWebChart, where it will be evaluated. This is why you cannot pass pre-defined values. You need to insert formulas into **$Doughnut** instead, which will be evaluated in New-PodeWebChart. This requires some practice and good nerves, but the result is impressive.

Example041.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {
      #$size = (Get-Volume|where {$_.DriveLetter -eq 'c'}).size
      #$free = (Get-Volume|where {$_.DriveLetter -eq 'c'}).SizeRemaining
      #$used = $size - $free

      $Doughnut = { @{ Key = 'Free' ; Values = @( @{ Value = (Get-Volume|where `
                     {$_.DriveLetter -eq 'c'}).SizeRemaining } ) }
                    @{ Key = 'Used' ; Values = @( @{ Value = ((Get-Volume|where `
                     {$_.DriveLetter -eq 'c'}).size)-((Get-Volume|where `
                     {$_.DriveLetter -eq 'c'}).SizeRemaining) } ) } }
      New-PodeWebGrid -Cells @(
                   New-PodeWebCell -Content @( New-PodeWebChart -AutoRefresh `
                                  -Name 'Disk Usage' -Type Doughnut -AsCard `
                                  -ScriptBlock $Doughnut  )
) } }
```

This script will render a ‘simple’ doughnut chart. Simple (with this ugly syntax)? Yes! You can make it even more complicated 😉. Also see: **New-PodeWebCell** to shrink sizes …



Checkbox

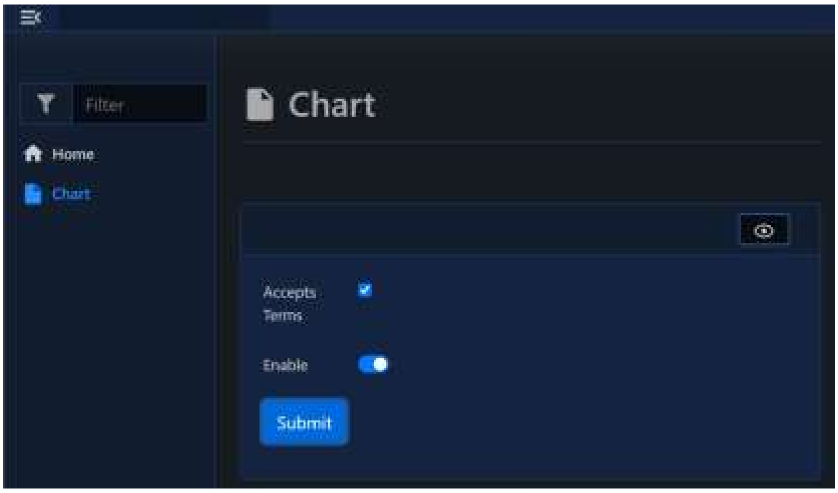
Support	
Events	Yes

The Checkbox element is a form input element, and can be added using New-PodeWebCheckbox. This will add a checkbox to your form, and you can render with checkbox as a switch using -AsSwitch:

Example042.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $accept = $WebEvent.Data['Accept Terms']
        $enable = $WebEvent.Data['Enable']
    } -Content @(
        New-PodeWebCheckbox -Name 'Accepts Terms'
        New-PodeWebCheckbox -Name 'Enable' -Checked -AsSwitch
) ) } }
```

When using singular checkboxes like above, the value in $WebEvent will be true or false strings. Since there is no aditional logic to handle web-requests here, pretty much nothing will happen so far.

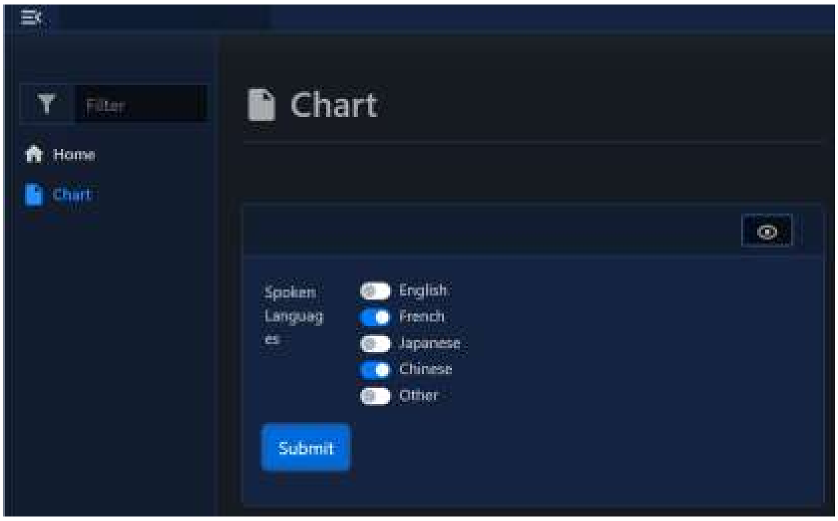


You can also setup a checkbox to have multiple options like below; in this case, the value will be a comma separated list of the selected options:

Example043.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {

    New-PodeWebCard -Content @(
        New-PodeWebForm -Name 'Example' -ScriptBlock {
            $langs = $WebEvent.Data['Spoken Languages']
        } -Content @(
            New-PodeWebCheckbox -Name 'Spoken Languages' `
        -Options 'English', 'French', 'Japanese', 'Chinese', 'Other' -AsSwitch
) ) } }
```

Which looks like below:



Inline

You can render this element inline with other non-form elements by using the -NoForm switch. This will remove the form layout, and render the element more cleanly when used outside of a form.

Display Name

By default the label displays the -Name of the element. You can change the value displayed by also supplying an optional -DisplayName value; this value is purely visual, when the user submits the form the value of the element is still retrieved using the -Name from $WebEvent.Data.

Display Options

By default the options displayed are from the -Options parameter. Like the Name, you can change the values displayed by supplying the optional -DisplayOptions - values in the array should be in the same order as the values in -Options. These values are, like the Display Name, purely visual, and when the form is submitted the server receives the original values from -Options.

Code

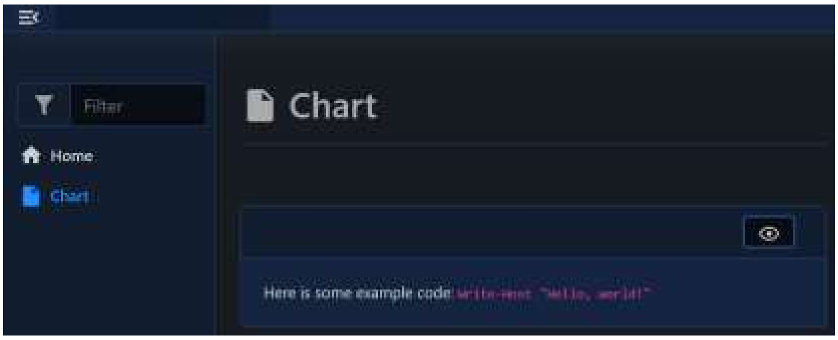
Support	
Events	No

The code element will render pre-formatted text as <code>value</code>. To create a code element you use New-PodeWebCode:

Example044.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {

    New-PodeWebCard -Content @(
        New-PodeWebText -Value 'Here is some example code: '
        New-PodeWebCode -Value 'Write-Host "Hello, world!"'
) } }
```

Which looks like below:



Here is some example code Write-Host "Hello, world!"

Code Block

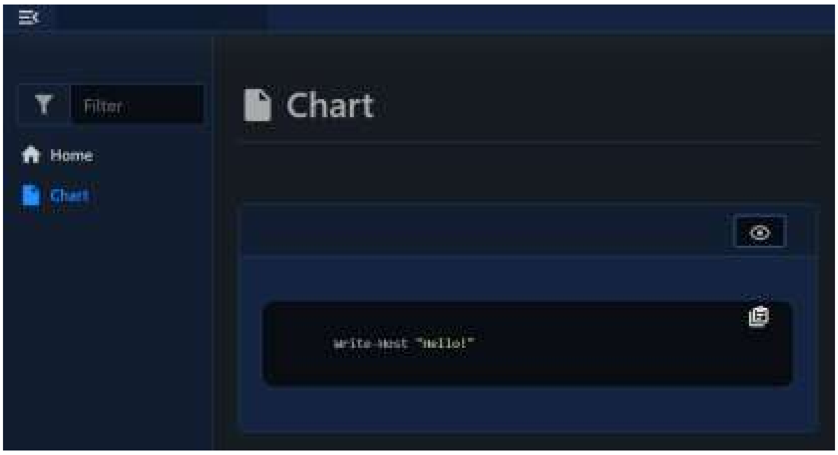
Support	
Events	No

The code block element will render pre-formatted text as <pre>value</pre>. To create a code block element you use New-PodeWebCodeBlock, and you can specify the highlighting language via -Language - the default is plain text.

Example045.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Chart -ScriptBlock {
        New-PodeWebCard -Content @(
            New-PodeWebCodeBlock -Value 'Write-Host "Hello!"' -Language PowerShell
) } }
```

Which looks like below:



```
Write-Host "Hello!"
```

Code Editor

Support	
Events	Yes

The code editor, which is done using MicroSoft's Monaco Editor, is currently still a WIP but functional. You can add a code editor to your page via New-PodeWebCodeEditor, and specify the language is editor is for via -Language.

This is the largest part of Pode.Web. It is a javascript editor (10MB of code), actually a pretty good one, just perhaps for web-apps a little bit over-blown 😉. You can locate it on windows path (if Pode.Web is installed):

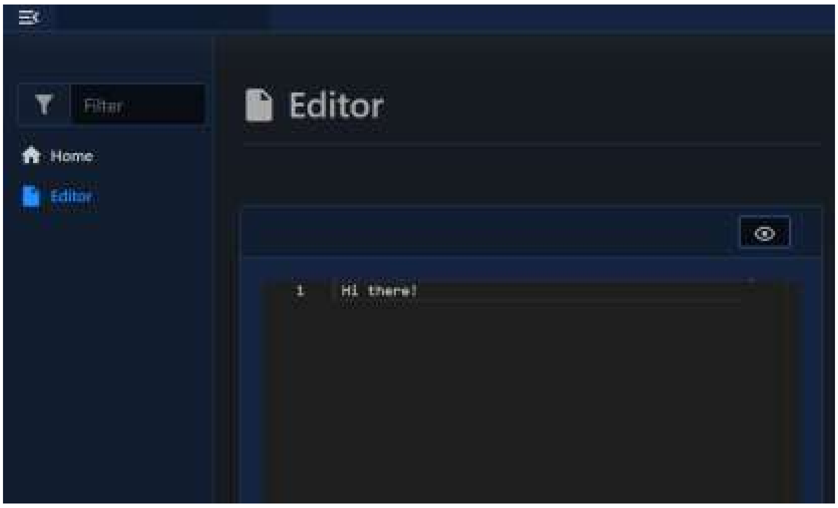
C:\Program Files\WindowsPowerShell\Modules\Pode.Web\0.8.3\Templates\Public\min-maps\vs\editor

To display the editor with some initial content you can supply -Value:

Example046.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Editor -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebCodeEditor -Name 'Editor' -Language 'powershell' -value ' Hi there!'
) } }
```

Which looks like below:



```
Hi there!
```

If you are resizing windows, there is no auto-sizing. You need to reload it.

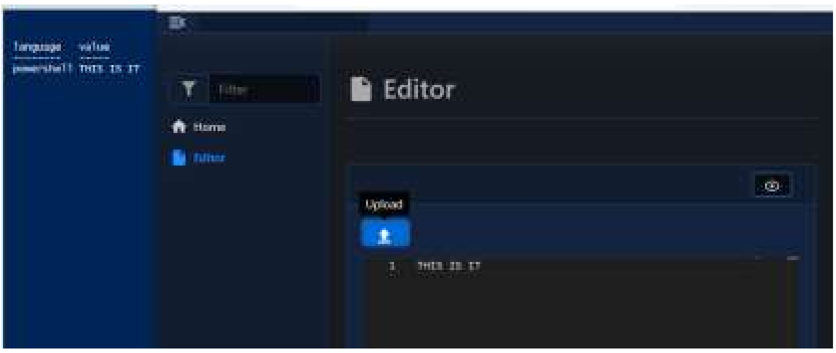
Upload files with Code Editor

You can also supply an optional -Upload scriptblock, this will show a button at the top of the editor which, when clicked, will pass the editor's current value to the scriptblock. This will allow you to save the value:

Example047.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Editor -ScriptBlock {

    New-PodeWebCard -Content @(
        New-PodeWebCodeEditor -Name 'Editor' -Language 'powershell' -Upload {
            $WebEvent.Data | Out-Default
} ) } }
```

The Upload button will only return last line from the editor to the terminal. But the uploaded variable is actually a full content of your editor, so you can upload contents to the server this way.



```
language   value
--------   -----
powershell THIS IS IT
```

```
THIS IS IT
```

Comment Block

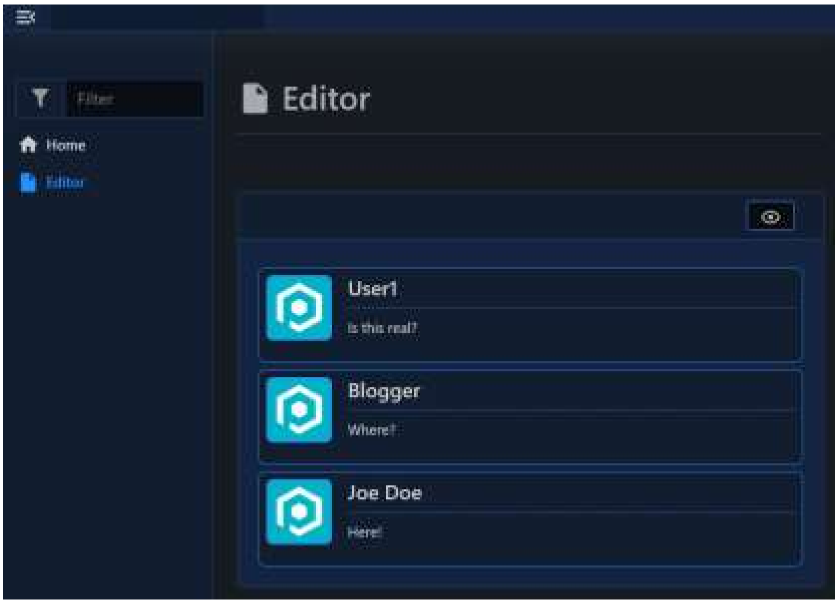
Support	
Events	No

A Comment Block lets you display messages/comments on your page, showing a user's name, message and an icon/avatar. You can add comment blocks using New-PodeWebComment:

Example048.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Editor -ScriptBlock {

    New-PodeWebCard -Content @(
        New-PodeWebComment `
           -Username 'User1' -Message 'Is this real?' `
           -Icon '/pode.web/images/icon.png'
        New-PodeWebComment `
           -Username ‘Blogger' -Message 'Where?' `
           -Icon '/pode.web/images/icon.png'
        New-PodeWebComment `
           -Username 'Joe Doe' -Message 'Here!' `
           -Icon '/pode.web/images/icon.png'
) } }
```

Which looks like below:



User1

Is this real?

Blogger

Where?

Joe Doe

Here!

Credentials

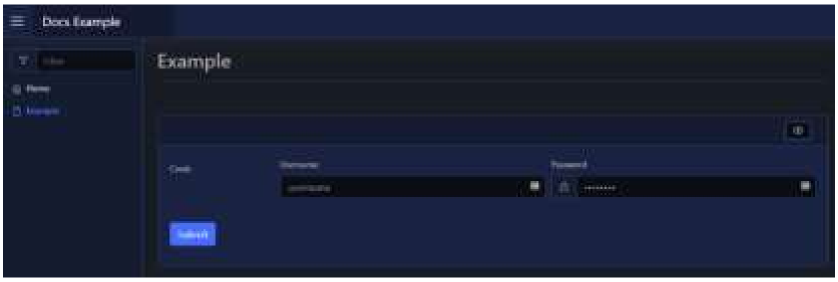
Support	
Events	Yes

The Credentials element is a form input element, and can be added using New-PodeWebCredential. This will automatically add a username and password input fields to your form:

Example049.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Credentials' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $username = $WebEvent.Data['Creds_Username']
        $password = $WebEvent.Data['Creds_Password']
        echo $username |out-default
        echo $password |out-default
    } -Content @(
        New-PodeWebCredential -Name 'Creds'
) ) } }
```

Which looks like below. This example will echo username and password to the console (for demo purposes only).

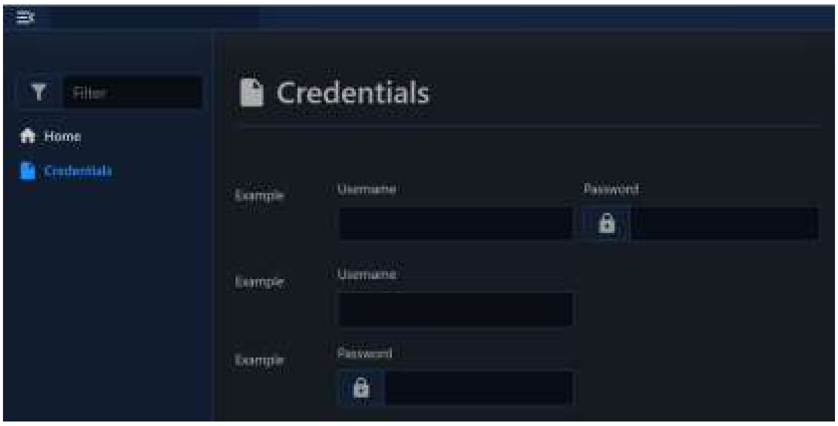


Type

By default both the Username and Password fields are displayed, but you can control which ones are displayed by using the -Type parameter:

Example050.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Credentials' -ScriptBlock {

    # both (this is the default)
    New-PodeWebCredential -Name 'Example' -Type Username, Password

    # just username
    New-PodeWebCredential -Name 'Example' -Type Username

    # just password
    New-PodeWebCredential -Name 'Example' -Type Password
} }
```



Display Name

By default the label displays the -Name of the element. You can change the value displayed by also supplying an optional -DisplayName value; this value is purely visual, when the user submits the form the value of the element is still retrieved using the -Name from $WebEvent.Data.

DateTime

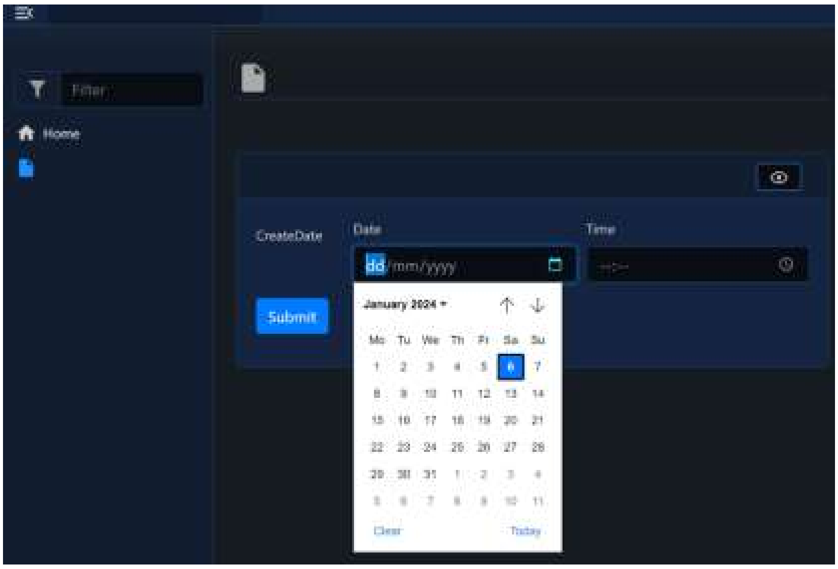
Support	
Events	Yes

The DateTime element is a form input element, and can be added using New-PodeWebDateTime. This will automatically add a date and time input fields to your form:

Example051.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $date = $WebEvent.Data['CreateDate_Date']
        $time = $WebEvent.Data['CreateDate_Time']
    } -Content @(
        New-PodeWebDateTime -Name 'CreateDate'
) ) } }
```

Which looks like below:

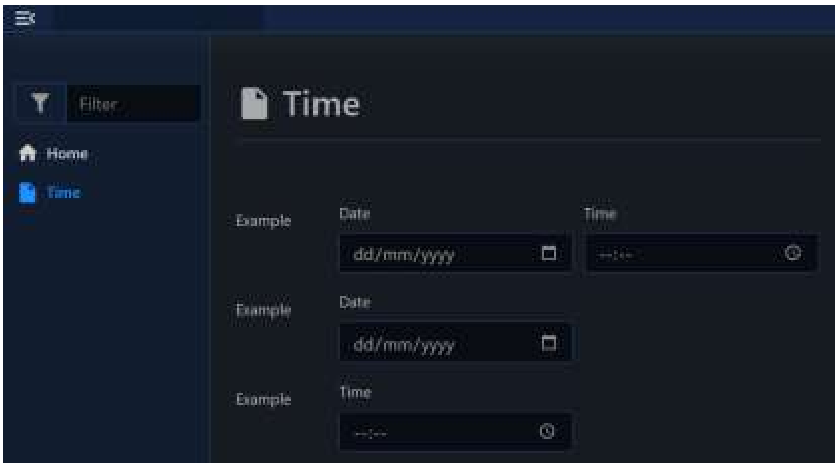


Type

By default both the Date and Time fields are displayed, but you can control which ones are displayed by using the -Type parameter:

Example052.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Time' -ScriptBlock {

    # both (this is the default)
    New-PodeWebDateTime -Name 'Example' -Type Date, Time

    # just date
    New-PodeWebDateTime -Name 'Example' -Type Date

    # just time
    New-PodeWebDateTime -Name 'Example' -Type Time
} }
```



Display Name

By default the label displays the -Name of the element. You can change the value displayed by also supplying an optional -DisplayName value; this value is purely visual, when the user submits the form the value of the element is still retrieved using the -Name from $WebEvent.Data.

File Stream

Support	
Events	Yes

A file stream element is a readonly textarea that will stream the contents of a file from the server - usually a text/log file from the /public directory. To add a file streaming element to your page you can use New-PodeWebFileStream.

The simplest file stream just needs a -Url being supplied; this URL should be a relative/literal URL path to a static text file.

Important

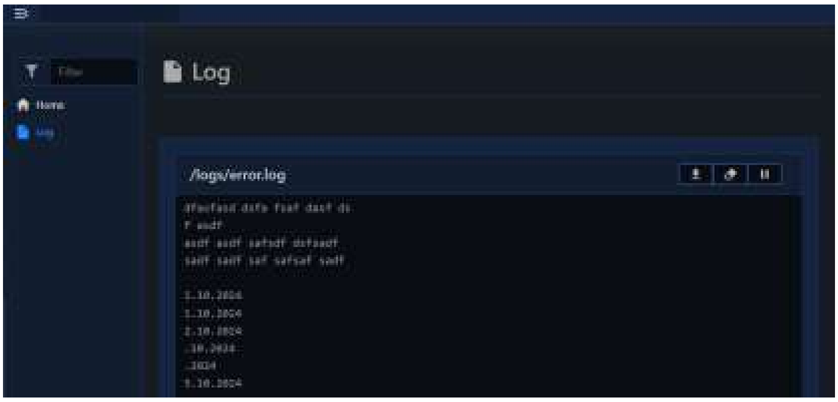
The server the file is being streamed from must support the Range HTTP header - Pode already supports this.

For example, the following will stream a log file from the public/logs/error.log file. The **public** directory is in the same directory with the **Example053.ps1** file.

Example053.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Log -ScriptBlock {

    New-PodeWebContainer -Content @(
    New-PodeWebFileStream -Url '/logs/error.log'
) } }
```

Which looks like below:



/logs/error.log

```
dfacfasd dsfa fsaf dasf ds
f asdf
asdf asdf safsdf dsfasdf
sadf sadf saf safsaf sadf

1.10.2024
1.10.2024
2.10.2024
.10.2024
.2024
5.10.2024
```

You can control the height and refresh interval of the element via the -Height and -Interval parameters. The interval is specified as a number of seconds - the default is 10secs.

Each file stream element renders with a header which shows the file being streamed; you can hide the header using the -NoHeader switch.

BUG: If there is a file, that has reverse entries (last at the top of the file), this will display it all wrong. But if the file gets filled from top to bottom (latest entries at the bottom of the file), this works like a charm.

Connection

If the connection (or any error) occurs while streaming the file, then the header (or just the border if the header is hidden) will turn red and the streaming will stop.

File Upload

Support	
Events	No

The File Upload element is a form input element, and can be created using New-PodeWebFileUpload. It allows users to upload files from your page forms. It will copy file to the full system path specified. The file to upload cannot exceed 100MB !

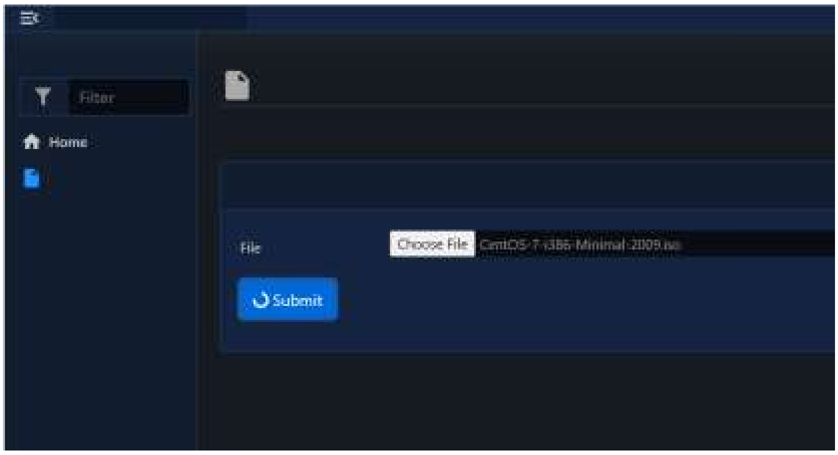
To overcome this limit, set the pode server settings in file Server.psd1

```
@{
    Server = @{
        # Reload server when files changes, useful for development.
        FileMonitor = @{
            Enable    = $true
            ShowFiles = $true
            Exclude   = @('.git\*', 'Storage\*', 'logs\*')
        }
        Request     = @{
            # Request timeout (seconds), default is 30.
            Timeout   = 600
            # Default limit is 100MB,
            # Pode supports up to `[Int]::MaxValue` (1.99GB),
            # Pode.Kestrel support higher limits.
            BodySize = 1.99GB
        }
    }
}
```

Example054.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        Save-PodeRequestFile -Key 'File' -Path 'C:\temp\pode'
    } -Content @(
        New-PodeWebFileUpload -Name 'File'
) ) } }
# https://raw.githubusercontent.com/ili101/PWT/main/Server.Example.psd1
```

The example above will render page like this:



Inline

You can render this element inline with other non-form elements by using the -NoForm switch. This will remove the form layout, and render the element more cleanly when used outside of a form.

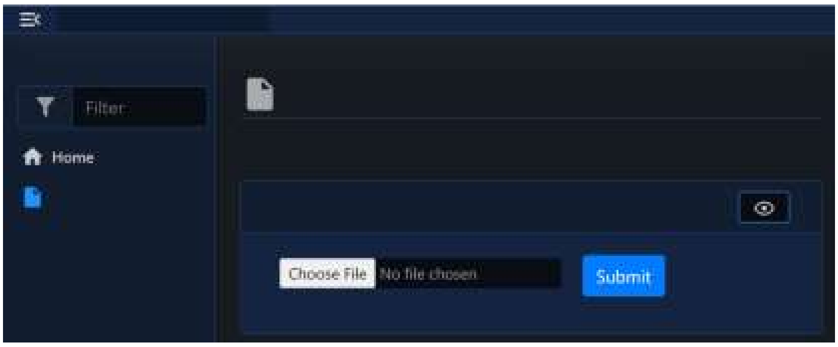
Accept

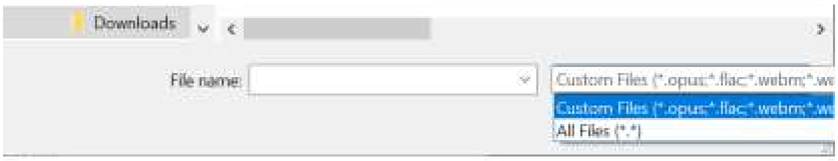
By default the file upload dialog will accept every file type, but you can filter which files are accepted via the -Accept parameter. This accepts an array of file types/extensions such as:

Example055.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        Save-PodeRequestFile -Key 'File' -Path 'c:\dnl'
    } -Content @(
        New-PodeWebFileUpload -Name 'File' -NoForm -Accept '.png', 'audio/*'
) ) } }
```

which will accept any .png file, and all sound files. But this will only be a default selection. The "All Files" option while selecting files is still available.




Downloads
File name:
Custom Files (*.opus;*.flac;*.webm;*.w
Custom Files (*.opus;*.flac;*.webm;*.w
All Files (*.*)

Display Name

By default the label displays the -Name of the element. You can change the value displayed by also supplying an optional -DisplayName value; this value is purely visual, when the user submits the form the value of the element is still retrieved using the -Name from $WebEvent.Data.

Form

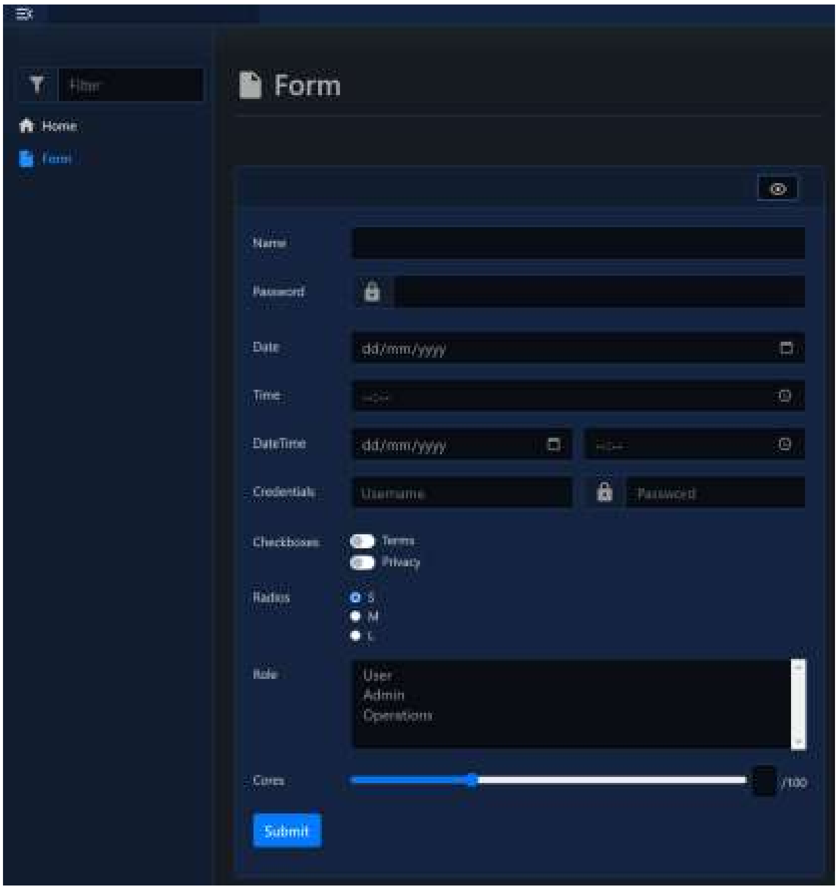
Support	
Events	No

A form is kind of like a layout, but is an element that contains other elements. It automatically wraps the -Content as a <form> and adds a submit button to the bottom. When clicked, the form is serialised and sent to the -ScriptBlock. To add a form to you page use New-PodeWebForm along with other form elements:

Example056.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Form -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $WebEvent.Data | Out-Default
    } -Content @(
        New-PodeWebTextbox -Name 'Name' -AutoComplete {
            return @('billy', 'bobby', 'alice', 'john', 'sarah', 'matt', 'zack', 'henry')
        }
        New-PodeWebTextbox -Name 'Password' -Type Password -PrependIcon Lock
        New-PodeWebTextbox -Name 'Date' -Type Date
        New-PodeWebTextbox -Name 'Time' -Type Time
        New-PodeWebDateTime -Name 'DateTime' -NoLabels
        New-PodeWebCredential -Name 'Credentials' -NoLabels
        New-PodeWebCheckbox -Name 'Checkboxes' -Options @('Terms', 'Privacy') -AsSwitch
        New-PodeWebRadio -Name 'Radios' -Options @('S', 'M', 'L')
        New-PodeWebSelect -Name 'Role' -Options @('User', 'Admin', 'Operations') -Multiple
        New-PodeWebRange -Name 'Cores' -Value 30 -ShowValue
) ) } }
```

Which will render the following form:



Note

If you have multiple forms on one page, make sure the Name/IDs are unique, including the Name/IDs of all form input elements as well.

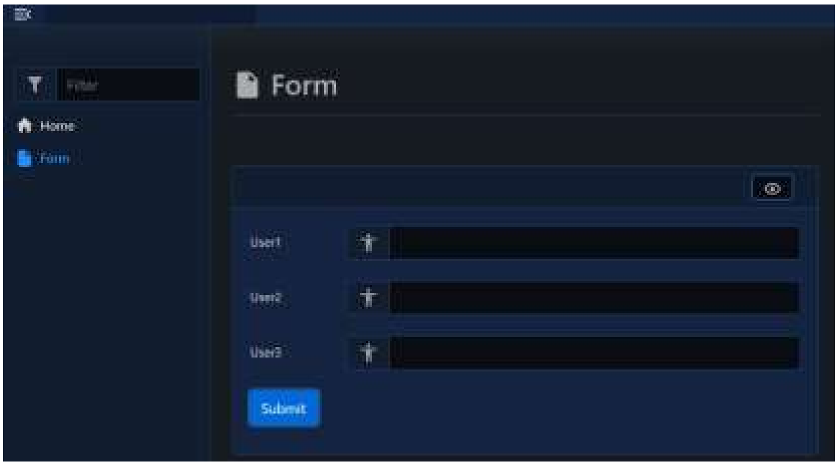
You can pass values to the scriptblock by using the -ArgumentList parameter. This accepts an array of values/objects, and they are supplied as parameters to the scriptblock:

```
New-PodeWebForm -Name 'Example' -ArgumentList 'Value1', 2, $false -ScriptBlock {
   param($value1, $value2, $value3)

   # $value1 = 'Value1'
   # $value2 = 2
   # $value3 = $false
} -Content @()
```

Example057.ps1 – this is not working !!!

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Form -ScriptBlock {

    New-PodeWebCard -Content @(

  New-PodeWebForm -Name 'Example' -ArgumentList "Tom", "Tim", "Tina" -ScriptBlock {
          param($value1, $value2, $value3)
          $WebEvent.Data | Out-Default
          Update-PodeWebTextbox -Name User1 -Value $value1
          Update-PodeWebTextbox -Name User2 -Value $value2
```

```
        Update-PodeWebTextbox -Name User3 -Value $value3
    } -Content @(
        New-PodeWebTextbox -Name User1 -PrependIcon Human
        New-PodeWebTextbox -Name User2 -PrependIcon Human
        New-PodeWebTextbox -Name User3 -PrependIcon Human
) ) } }
```



The text fields here should be populated, but they are not!

Elements

The available form elements in Pode.Web are:

- Checkbox
- Credentials
- DateTime
- FileUpload
- Hidden
- MinMax
- Range
- Radio
- Range
- Select
- Textbox

Method/Action

The default method for forms is Post, and the action is the internal route created for the form.

You can change these values by using the -Method and -Action parameters. The method can only be Get or Post, and the action must be a valid URL.

Reset

You can reset all form inputs by either using the Reset-PodeWebForm output action, or by using -ShowReset switch on New-PodeWebForm to display an optional reset button.

Header

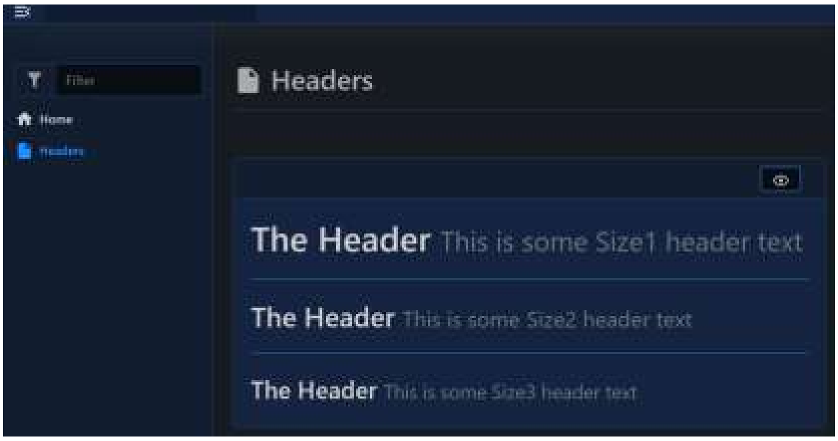
Support	
Events	No

You can render header titles to your page by using New-PodeWebHeader. This will show a title in various header sizes (h1 - h6):

Example058.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Headers -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebHeader -Value 'The Header' -Size 1 `
                      -Secondary 'This is some Size1 header text'
    New-PodeWebLine
    New-PodeWebHeader -Value 'The Header' -Size 2 `
                      -Secondary 'This is some Size2 header text'
    New-PodeWebLine
    New-PodeWebHeader -Value 'The Header' -Size 3 `
                      -Secondary 'This is some Size3 header text'
) } }
```

Which looks like below:



Headers

The Header This is some Size1 header text

The Header This is some Size2 header text

The Header This is some Size3 header text

Hidden

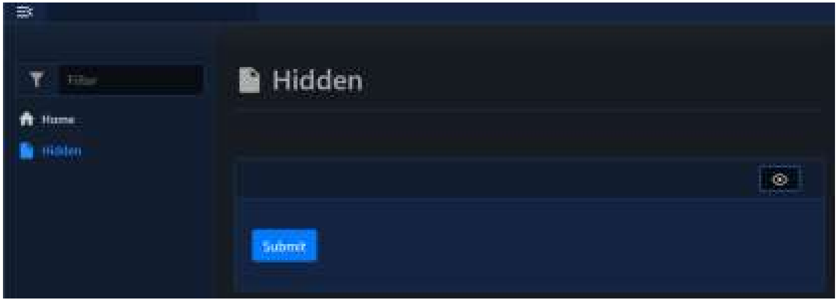
Support	
Events	No

The Hidden element is a form input element, and can be added using New-PodeWebHidden. It allows you to add hidden values/elements to your forms:

Example059.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Hidden -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $date = $WebEvent.Data['HiddenDate']
    } -Content @(
        New-PodeWebHidden -Name 'HiddenDate' -Value ([datetime]::Now.ToString())
) ) } }
```

No special output here:



iFrame

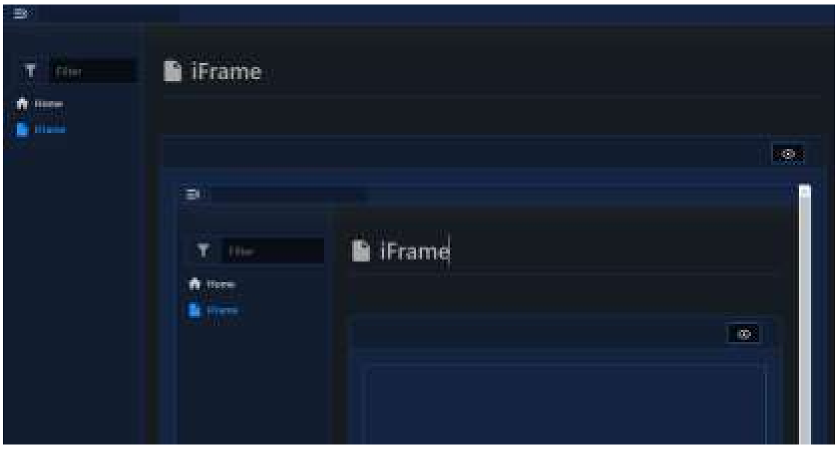
Support	
Events	No

The iFrame element lets you embed other websites into your pages via New-PodeWebIFrame, and you just need to supply the -Url to embed:

Example060.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name iFrame -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebIFrame -Url '/'

) } }
```

Which looks like a nested page (see the picture).



Icon

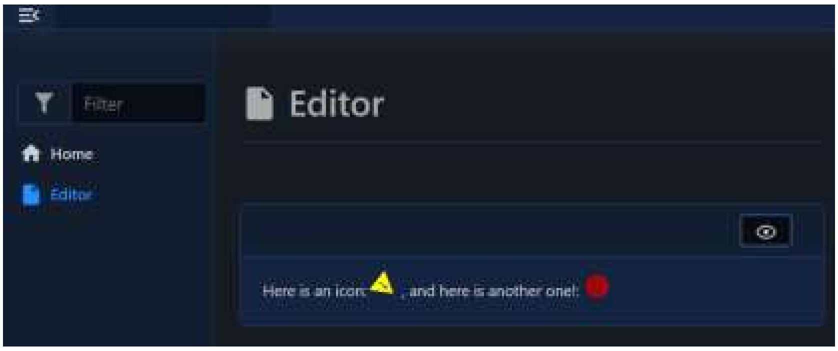
Support	
Events	Yes

The icon element will render a Material Design Icon to your page. To create an icon element you use New-PodeWebIcon, and supply the name of a Material Design Icon using -Name; you can also change the icon colour via -Colour which can be a typical name (red/green/etc) or a hex value (#333333). https://pictogrammers.com/library/mdi/

Example061.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Editor -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebText -Value 'Here is an icon: '
    New-PodeWebIcon -Name 'alert' -Colour 'yellow' -Spin
    New-PodeWebText -Value ', and here is another one!: '
    New-PodeWebIcon -Name 'emoticon-happy' -Colour '#bb0000'
) } }
```

Which rotate triangle like below:



Here is an icon: , and here is another one!

Actions

Flip

You can flip an icon by passing Horizontal or Vertical to the -Flip parameter. You cannot supply both flip and rotate together.

Rotate

You can rotate an icon by a fixed number of degrees by supplying a value, 45-315 in 45 degree increments, to the -Rotate parameter. You cannot supply both flip and rotate together.

Spin

You can make an icon spin by supplying the -Spin switch.

Image

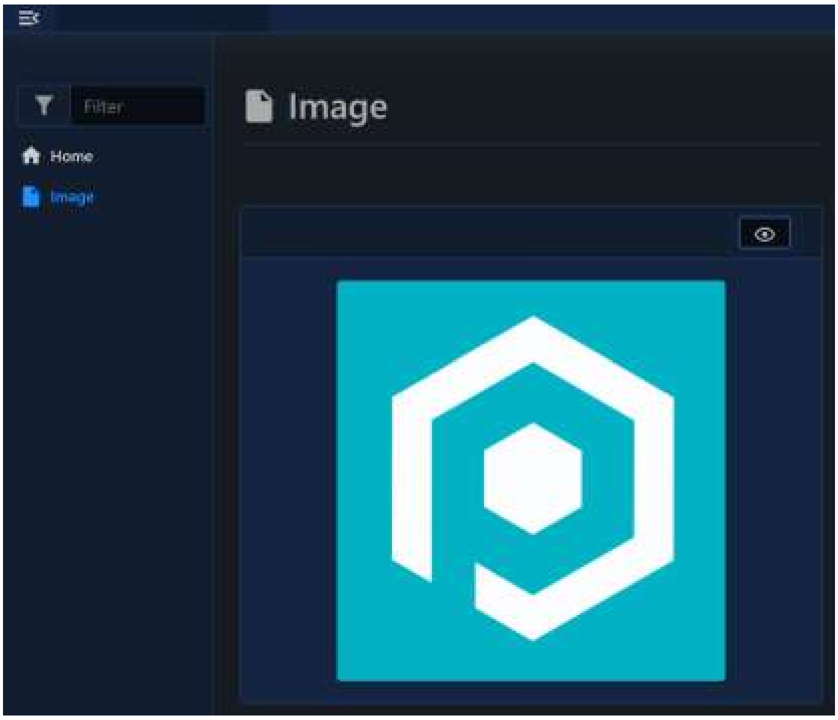
Support	
Events	Yes

This will render an image onto your page, using New-PodeWebImage. You need to supply a -Source URL to the image you wish to display:

Example062.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Image -ScriptBlock {
    New-PodeWebCard -Content @(
    New-PodeWebImage -Source '/pode.web/images/icon.png' -Title 'Pode' -Alignment Center
) } }
```

Which looks like below:



Size

The -Width and -Height of an image have the default unit of px. If 0 is specified then auto is used instead. Any custom value such as 10% can be used, but if a plain number is used then px is appended.

Line

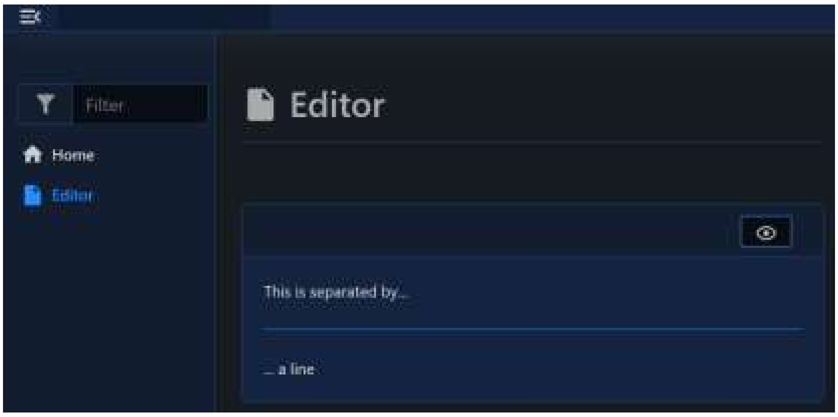
Support	
Events	No

This will render a line (<hr>) to your page, using New-PodeWebLine:

Example063.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Editor -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebText -Value 'This is separated by...'
    New-PodeWebLine
    New-PodeWebText -Value '... a horizontal line'
) } }
```

Which looks like below:



This is separated by...

... a line

Link

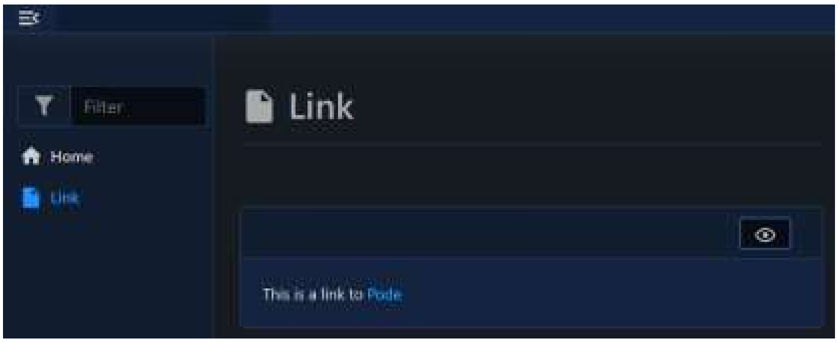
Support	
Events	Yes

This will render a hyperlink (`<a>`) to your page, using `New-PodeWebLink`. You need to supply a `-Source` (the href), and a `-Value` to show to the user:

Example064.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Link -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebText -Value 'This is a link to '
    New-PodeWebLink -Source 'https://github.com/Badgerati/Pode' -Value 'Pode'
) } }
```

Which looks like below:



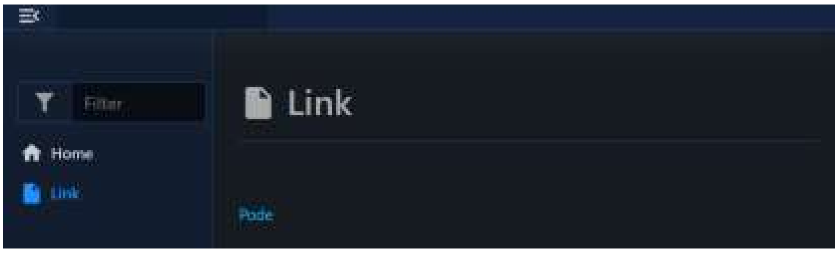
This is a link to Pode

New Tab

To open the link in a new tab, supply the `-NewTab` switch:

Example065.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Link -ScriptBlock {

    New-PodeWebLink -Source 'https://github.com/Badgerati/Pode' -Value 'Pode' -NewTab

} }
```



List

Support	
Events	No

Pode.Web lets you display lists of items, either bullet pointed or numbered, using New-PodeWebList. You need to supply an array of -Items or -Values, and then the -Numbered flag for numbered lists.

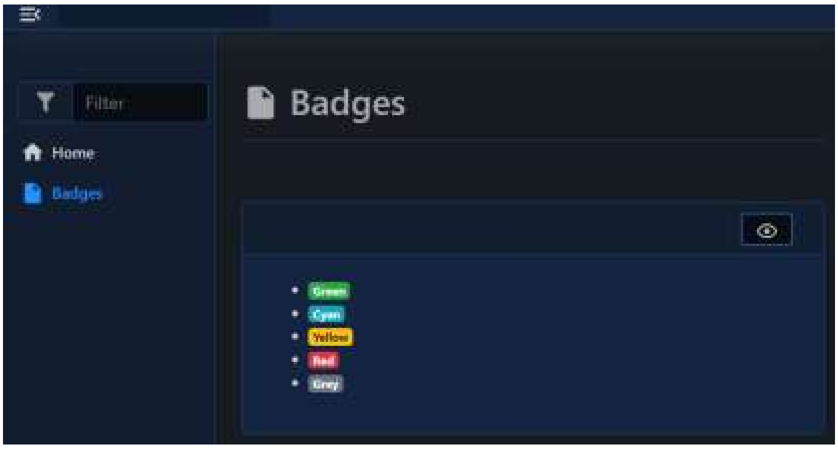
Items

The -Items parameter takes an array of ListItem elements (created via New-PodeWebListItem). The ListItem element accepts an array of -Content, which can be either layouts or elements:

Example066.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Badges -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebList -Items @(
        New-PodeWebListItem -Content @(
            New-PodeWebBadge -Colour Green -Value 'Green'
        )
        New-PodeWebListItem -Content @(
            New-PodeWebBadge -Colour Cyan -Value 'Cyan'
        )
        New-PodeWebListItem -Content @(
            New-PodeWebBadge -Colour Yellow -Value 'Yellow'
        )
        New-PodeWebListItem -Content @(
            New-PodeWebBadge -Colour Red -Value 'Red'
        )
        New-PodeWebListItem -Content @(
            New-PodeWebBadge -Colour Grey -Value 'Grey'
) ) ) } }
```

Which will render some badgers in list:



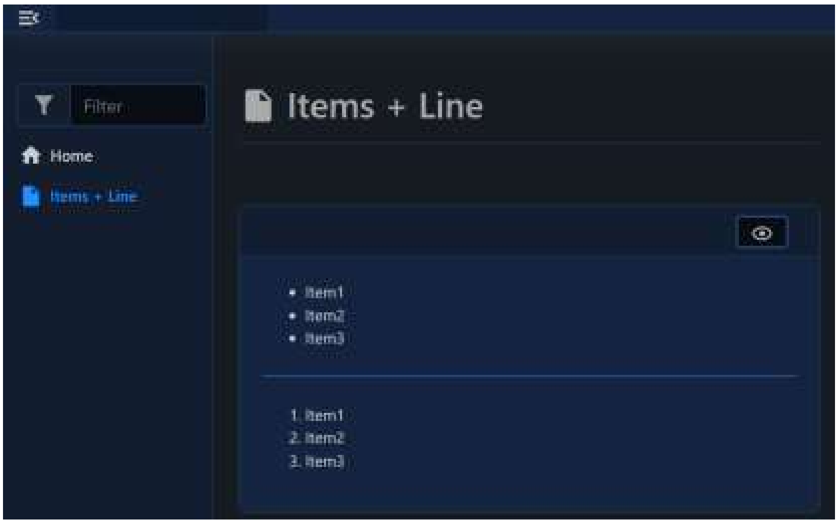
Values

The -Values parameter takes an array of strings, and renders them as a normal list:

Example067.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Items + Line' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebList -Values 'Item1', 'Item2', 'Item3'
    New-PodeWebLine
    New-PodeWebList -Values 'Item1', 'Item2', 'Item3' -Numbered
) } }
```

Which looks like below:



- Item1
- Item2
- Item3

1. Item1
2. Item2
3. Item3

MinMax

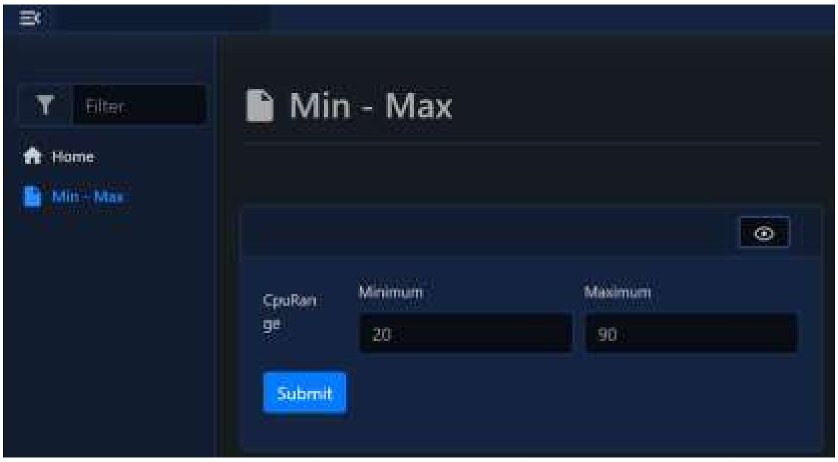
Support	
Events	Yes

The MinMax element is a form input element, and can be added using New-PodeWebMinMax. This will add a pair of number type textboxes to your form to allow input of minimum/maximum values, you can set preset values using -MinValue and -MaxValue:

Example068.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Min - Max' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $min = $WebEvent.Data['CpuRange_Min']
        $max = $WebEvent.Data['CpuRange_Max']
    } -Content @(
        New-PodeWebMinMax -Name 'CpuRange' -MinValue 20 -MaxValue 90
) ) } }
```

Which looks like below:



CpuRange

Minimum: 20

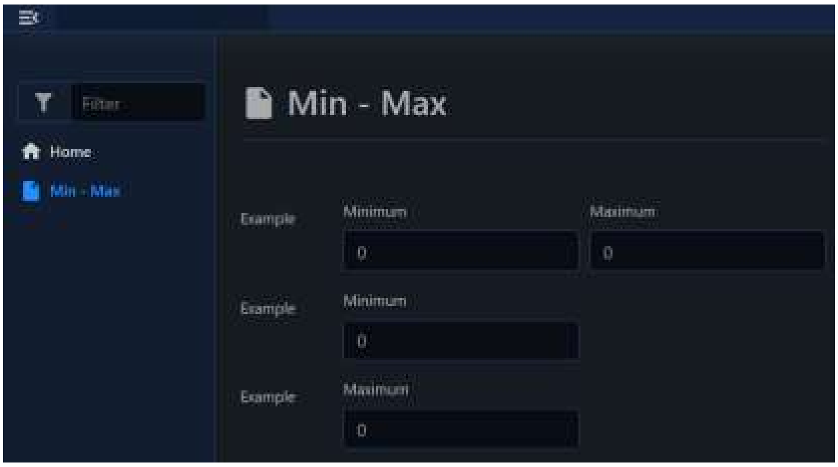
Maximum: 90

Type

By default both the Min and Max fields are displayed, but you can control which ones are displayed by using the -Type parameter:

Example069.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Min - Max' -ScriptBlock {

    # both (this is the default)
    New-PodeWebMinMax -Name 'Example' -Type Min, Max

    # just min
    New-PodeWebMinMax -Name 'Example' -Type Min

    # just max
    New-PodeWebMinMax -Name 'Example' -Type Max
} }
```



Display Name

By default the label displays the -Name of the element. You can change the value displayed by also supplying an optional -DisplayName value; this value is purely visual, when the user submits the form the value of the element is still retrieved using the -Name from $WebEvent.Data.

Paragraph

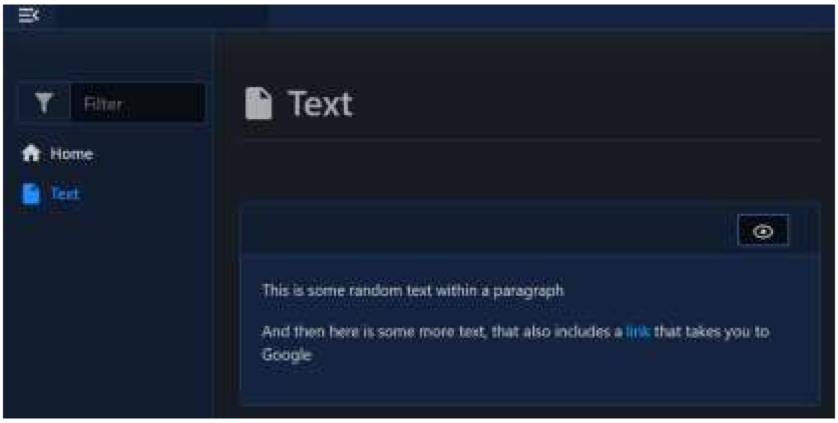
Support	
Events	No

You can display a -Value, or other -Elements, within a paragraph block using New-PodeWebParagraph. This lets you separate blocks of text/elements neatly on a page:

Example070.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Text -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebParagraph -Value 'This is some random text within a paragraph'
    New-PodeWebParagraph -Elements @(
        New-PodeWebText -Value 'And then here is some more text, that also includes a '
        New-PodeWebLink -Value 'link' -Source 'https://google.com'
        New-PodeWebText -Value ' that takes you to Google'
) ) } }
```

Which looks like below:



This is some random text within a paragraph

And then here is some more text, that also includes a link that takes you to Google

Progress Bar

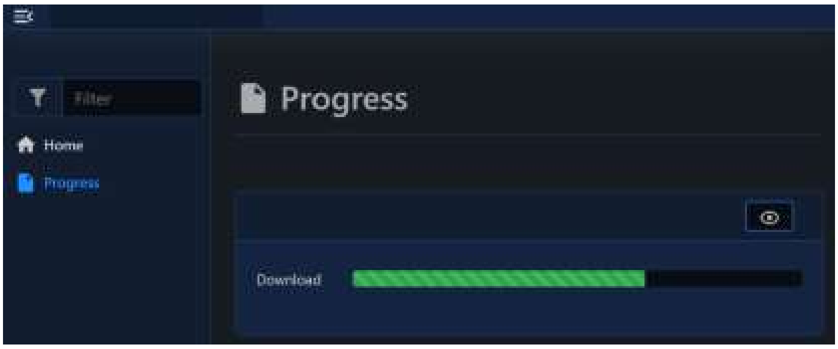
Support	
Events	Yes

You can show a progress bar on your page via New-PodeWebProgress. The default min/max values are 0-100, but can be changed via -Min and -Max. You can specify the current value via -Value, and alter the -Colour. You can also make the progress bar -Striped and/or -Animated (only really works with a striped progress bar!):

Example071.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Progress -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebProgress -Name 'Download' -Value 65 -Colour Green -Striped -Animated

) } }
```

Which looks like below:



Quote

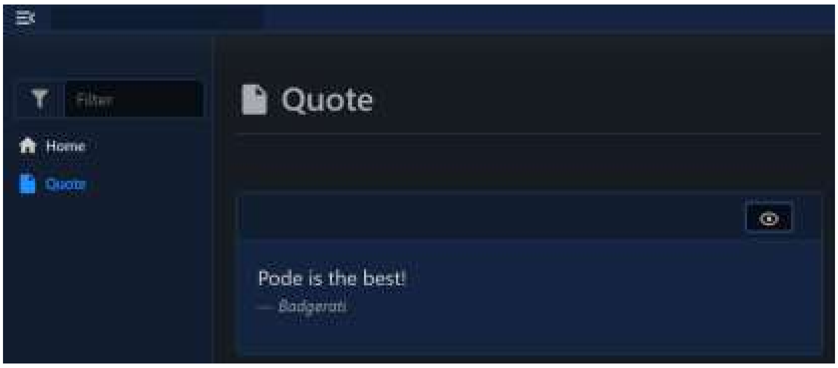
Support	
Events	No

You can render a quote to your page by using New-PodeWebQuote. This will show a quoted message (-Value), with optional -Source, to your page:

Example072.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Quote -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebQuote -Value 'Pode is the best!' -Source 'Badgerati'
) } }
```

Which looks like below:



Pode is the best!
— Badgerati

Radio

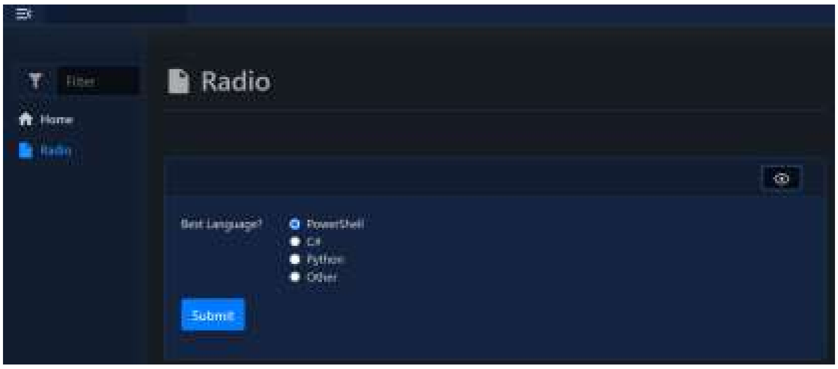
Support	
Events	Yes

The Radio element is a form input element, and can be added using New-PodeWebRadio. This will add a series of radio buttons to your form:

Example073.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Radio -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $bestLang = $WebEvent.Data['Best Language?']
    } -Content @(
        New-PodeWebRadio -Name 'Best Language?' -Options 'PowerShell', 'C#', 'Python',
'Other'
    )
    )
   }
}
```

Which looks like below:



Inline

You can render this element inline with other non-form elements by using the -NoForm switch. This will remove the form layout, and render the element more cleanly when used outside of a form.

Display Name

By default the label displays the -Name of the element. You can change the value displayed by also supplying an optional -DisplayName value; this value is purely visual, when the user submits the form the value of the element is still retrieved using the -Name from $WebEvent.Data.

Display Options

By default the options displayed are from the -Options parameter. Like the Name, you can change the values displayed by supplying the optional -DisplayOptions - values in the array should be in the same order as the values in -Options. These values are, like the Display Name, purely visual, and when the form is submitted the server receives the original values from -Options.

Range

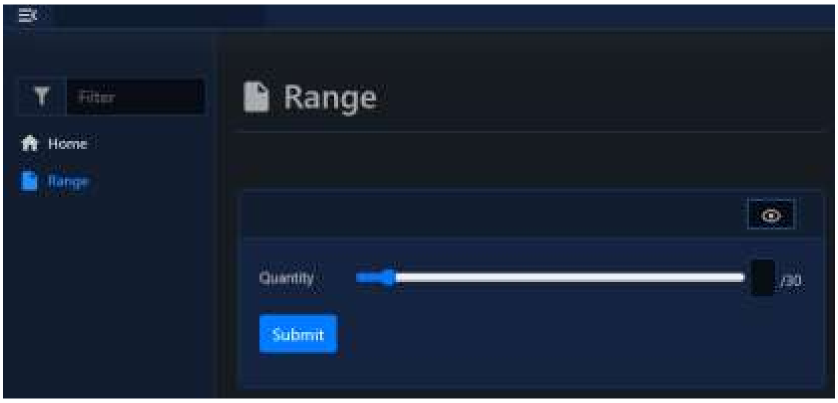
Support	
Events	Yes

The Range element is a form input element, and can be added using New-PodeWebRange. This will add a range slider to your form, with default min/max values of 0-100, but these can be altered via -Min and -Max. You can set a default -Value, and show the currently selected value via -ShowValue:

Example074.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Range -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $quantity = $WebEvent.Data['Quantity']
    } -Content @(
        New-PodeWebRange -Name 'Quantity' -Max 30 -Value 1 -ShowValue
) ) } }
```

Which looks like below:



Inline

You can render this element inline with other non-form elements by using the -NoForm switch. This will remove the form layout, and render the element more cleanly when used outside of a form.

Display Name

By default the label displays the -Name of the element. You can change the value displayed by also supplying an optional -DisplayName value; this value is purely visual, when the user submits the form the value of the element is still retrieved using the -Name from $WebEvent.Data.

Raw

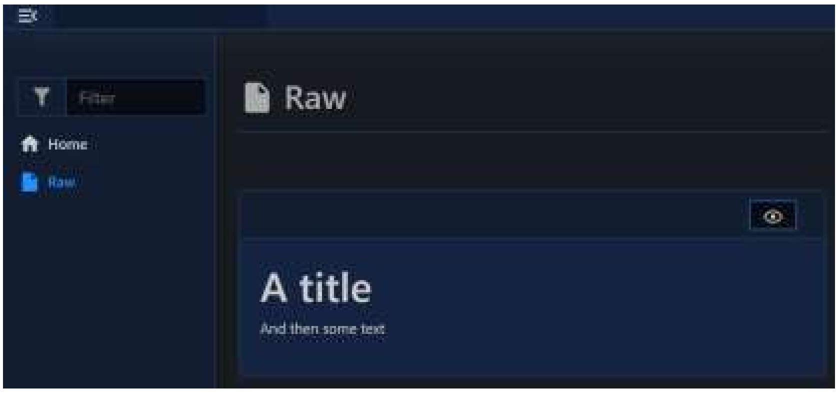
Support	
Events	No

This will render raw HTML to your page, using New-PodeWebRaw:

Example075.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Raw -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebRaw -Value '<h1>A title</h1><p>And then some text</p>'
) } }
```

Which looks like below:



A title

And then some text

Select

Support	
Events	Yes

The Select element is a form input element, and can be added using New-PodeWebSelect. This will add a dropdown select menu to your form, allowing the user to select an entry; to allow multiple entries to be selected you can pass -Multiple, and to specify a pre-selected value you can use -SelectedValue.

Options

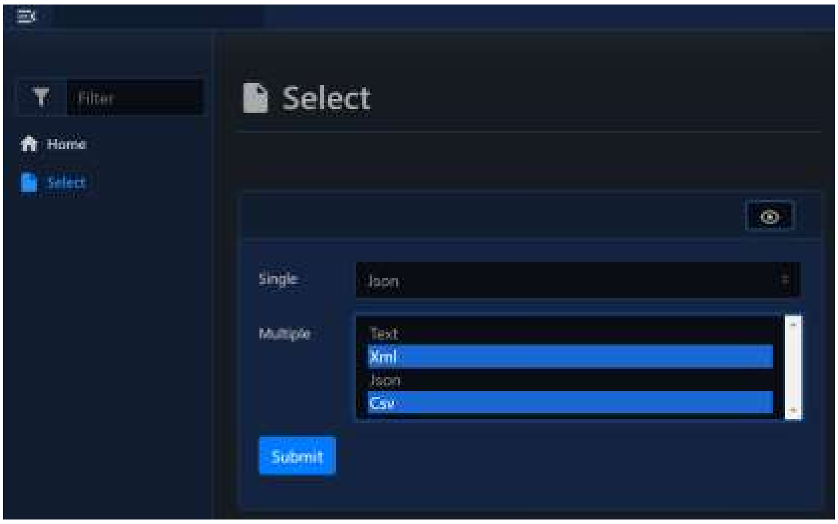
To create a Select element with pre-defined options, you can use the -Options parameter:

Example076.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Select -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $single = $WebEvent.Data['Single']
        $multiple = $WebEvent.Data['Multiple']
    } -Content @(
   New-PodeWebSelect -Name 'Single' -Options 'Text', 'Xml', 'Json', 'Csv' -SelectedValue 'Json'
    New-PodeWebSelect -Name 'Multiple' -Options 'Text', 'Xml', 'Json', 'Csv' -Multiple
) ) } }
```

Note

When using -Multiple, the values will be sent back in a comma separated list Which looks like below:



Single: Json

Multiple: Text, Xml (selected), Json, Csv (selected)

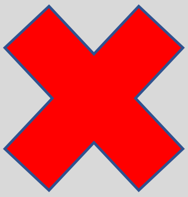
Dynamic

You can build a Select element's options dynamically by using the -ScriptBlock parameter. This will allow you to retrieve the options from elsewhere and use them as options instead.

You can either return an array of raw values, or pipe the options into, and return, Update-PodeWebSelect. The following will both build a Select element with 10 random numbers as the options:

Example077.ps1 – This approach does not work !!!

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Select -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        New-PodeWebSelect -Name 'Random1' -ScriptBlock {
            return @(foreach ($i in (1..10)) {
                Get-Random -Minimum 1 -Maximum 10
            })
        }

        New-PodeWebSelect -Name 'Random2' -ScriptBlock {
            $options = @(foreach ($i in (1..10)) {
                Get-Random -Minimum 1 -Maximum 10
            })

            $options | Update-PodeWebSelect -Id $ElementData.Id
} } ) } }
```

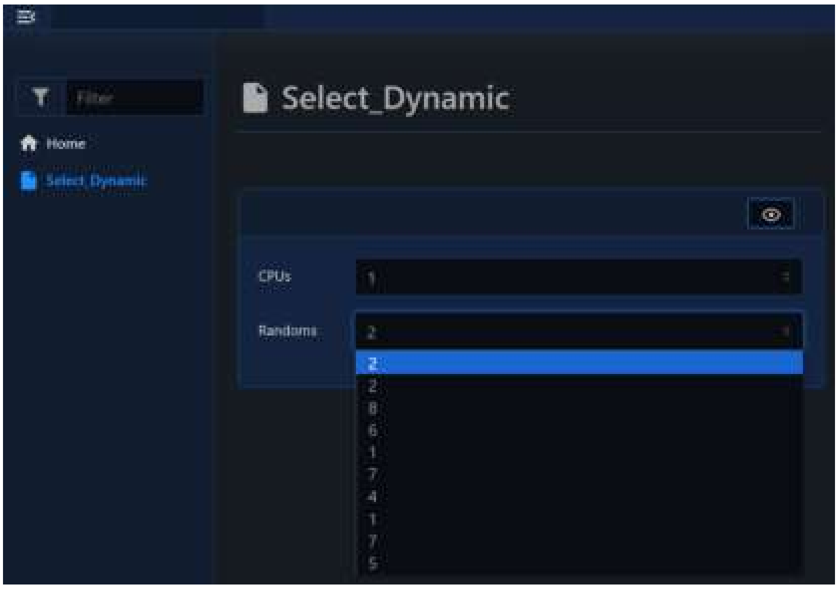


This example is all wrong! It will ask for Pode-WebForm content parameters.

To genereate values for dropdown select box, this would be the way to do it:

Example077b.ps1 – this one works fine

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Select_Dynamic -ScriptBlock {

    New-PodeWebCard -Content @(
        New-PodeWebSelect  -Name 'CPUs' `
                           -Required -ScriptBlock { $(1..8) }
        New-PodeWebSelect  -Name 'Randoms' `
                           -Required -ScriptBlock { foreach ($i in (1..10)) `
                            { Get-Random -Minimum 1 -Maximum 10 } }
) } }
```



Multiple

You can render a multiple select element, where more than one option can be selected, by using the -Multiple switch. By default only the first 4 options are shown, this can be altered using the -Size parameter.

Inline

You can render this element inline with other non-form elements by using the -NoForm switch. This will remove the form layout, and render the element more cleanly when used outside of a form.

Display Name

By default the label displays the -Name of the element. You can change the value displayed by also supplying an optional -DisplayName value; this value is purely visual, when the user submits the form the value of the element is still retrieved using the -Name from $WebEvent.Data.

Display Options

By default the options displayed are from the -Options parameter. Like the Name, you can change the values displayed by supplying the optional -DisplayOptions - values in the array should be in the same order as the values in -Options. These values are, like the Display Name, purely visual, and when the form is submitted the server receives the original values from -Options.

Spinner

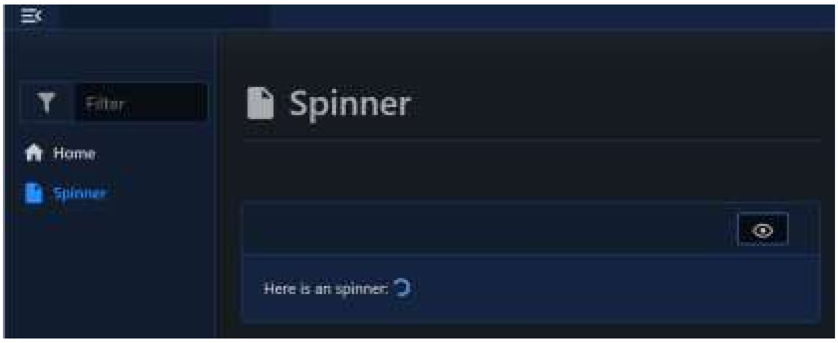
Support	
Events	No

The spinner element will render a progress spinner to your page. To create a spinner element you use New-PodeWebSpinner, you can also change the spinner colour via -Colour which can be a known name (red/green/etc) or a hex value (#333333).

Example078.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Spinner -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebText -Value 'Here is an spinner: '
    New-PodeWebSpinner -Colour 'cornflowerblue'
) } }
```

Which looks like below:



Here is an spinner:

Table

Support	
Events	No

You can display data rendered in a table by using New-PodeWebTable, and you can also render certain other elements within a table such as:

- Buttons
- Badges
- Spinners
- Icons
- Links

A table gets its data from a supplied -ScriptBlock, more information below, and you can also -AutoRefresh a table to fetch new data every minute. Tables also support being sorted, paginated, clickable and filtered.

Data

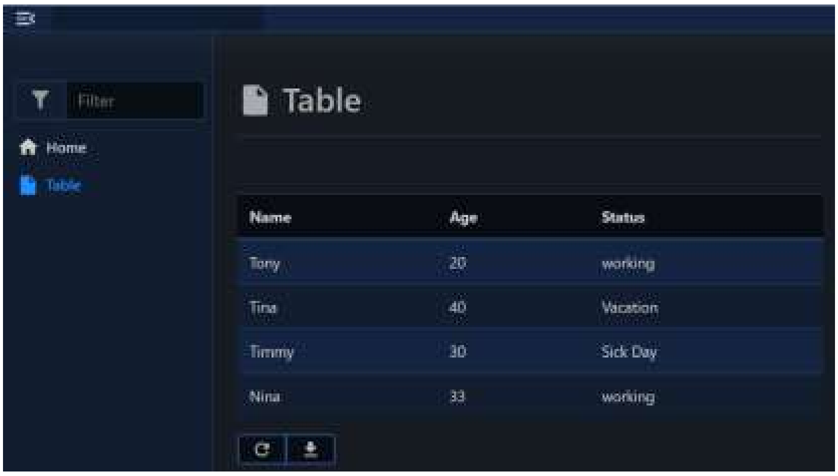
To supply data to be rendered in a table, you have to supply a -ScritpBlock which returns the appropriate data in the correct format. You can also supply other elements to be rendered within the table, within the data that's returned. You can pass values to the scriptblock by using the -ArgumentList parameter. This accepts an array of values/objects, and they are supplied as parameters to the scriptblock:

```
New-PodeWebTable -Name 'Example' -ArgumentList 'Value1', 2, $false -ScriptBlock {
  param($value1, $value2, $value3)
  # $value1 = 'Value1'
  # $value2 = 2
  # $value3 = $false
}
```

Example079.ps1 (how to process parameters)

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Table -ScriptBlock {

    New-PodeWebTable -Name 'Example' -ArgumentList 'Nina', '33', 'working' -ScriptBlock {
    param($value1, $value2, $value3)

          function statuses {
                   [pscustomobject]@{ name='Tony' ;age='20'   ;status='working'  }
                   [pscustomobject]@{ name='Tina' ;age='40'   ;status='Vacation' }
                   [pscustomobject]@{ name='Timmy';age='30'   ;status='Sick Day' }
                   [pscustomobject]@{ name=$value1;age=$value2;status=$value3    }
          }

          foreach ($hash in statuses ) {
          [ordered]@{
                     Name   = $hash.Name
                     Age    = $hash.Age
                     Status = $hash.Status
} } } } }
```

This will render a nice table with the customized values in the last row.



Table

Name	Age	Status
Tony	20	working
Tina	40	Vacation
Timmy	30	Sick Day
Nina	33	working

Raw

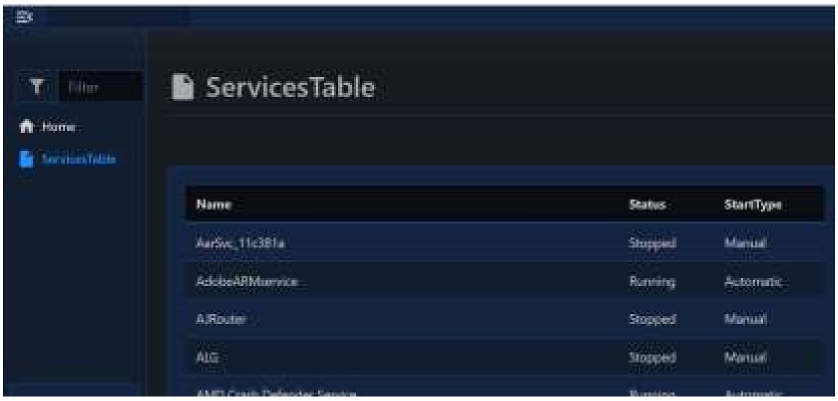
The data format to be returned from a table's -ScriptBlock is simple, it's purely just Key:Value in an ordered hashtable/pscustomobject.

The following example renders a table for services on a computer, displaying the Name, Status and StartTypes of the services:

Example080.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ServicesTable -ScriptBlock {

    New-PodeWebContainer -Content @(
    New-PodeWebTable -Name 'Services' -ScriptBlock {
        foreach ($svc in (Get-Service)) {
            [ordered]@{
                Name      = $svc.Name
                Status    = "$($svc.Status)"
                StartType = "$($svc.StartType)"
} } } ) } }
```

which renders a table that looks like below:



ServicesTable

Name	Status	StartType
AarSvc_11c381a	Stopped	Manual
AdobeARMservice	Running	Automatic
AJRouter	Stopped	Manual
ALG	Stopped	Manual
AMD Crash Defender Service	Running	Automatic

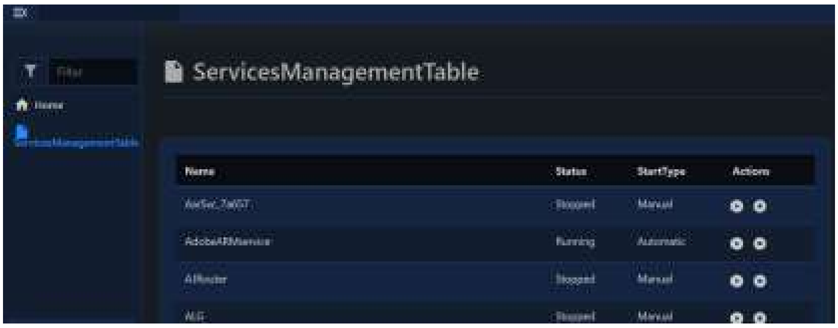
Elements

Extending on the raw example above, you can also render certain elements within a table. Let's say you want two buttons in the table, one to start the service, and one to stop the service; to do this, we just have to use New-PodeWebButton within the returned hashtable. So that the button's scriptblock knows which services we which to stop/start, we'll need to supply -DataColumn Name to the table; when a button within the table is clicked, the value of the Name column in that button's row will be available via $WebEvent.Data.Value in the button's scriptblock.

Example081.ps1 - ISSUES

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ServicesManagementTable -ScriptBlock {

    New-PodeWebContainer -Content @(
    New-PodeWebTable -Name 'Services' -DataColumn Name -ScriptBlock {
        foreach ($svc in (Get-Service)) {
            [ordered]@{
                Name      = $svc.Name
                Status    = "$($svc.Status)"
                StartType = "$($svc.StartType)"
                Actions   = @(
      New-PodeWebButton -Name 'Start' -Icon 'Play-Circle' -IconOnly -ScriptBlock {
          Start-Service -Name $WebEvent.Data.Value -Force | Out-Null
          Show-PodeWebToast -Message "$($WebEvent.Data.Value) started"
          Sync-PodeWebTable -Id $ElementData.Parent.ID
          $($WebEvent.Data.Value) | out-default
      }
      New-PodeWebButton -Name 'Stop' -Icon 'Stop-Circle' -IconOnly -ScriptBlock {
          Stop-Service -Name $WebEvent.Query['value'] -Force | Out-Null
          Show-PodeWebToast -Message "$($WebEvent.Data.Value) stopped"
          Sync-PodeWebTable -Id $ElementData.Parent.ID
          $($WebEvent.Data.Value) | out-default
} ) } } } ) } }
```

which renders a table that looks like below (but not very functional). I managed to run a stop function on some services but no start. If stop is running. start should too. But it does not!



ServicesManagementTable

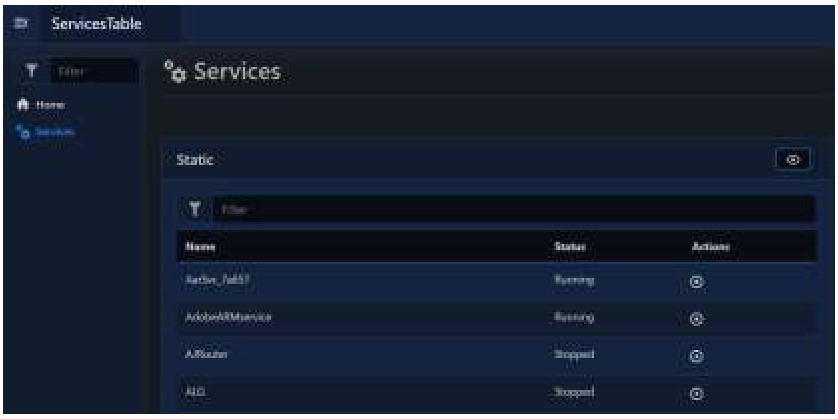
Name	Status	StartType	Actions
AarSvc_7a0G7	Stopped	Manual	● ●
AdobeARMservice	Running	Automatic	● ●
AJRouter	Stopped	Manual	● ●
ALG	Stopped	Manual	● ●

A better version of the script above, that works:

Example081b.ps1 – a working services starter example

```
Start-PodeServer -StatusPageExceptions Show {
    Add-PodeEndpoint -Address * -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title 'ServicesTable'  -Theme Dark

    $table = New-PodeWebTable -Name 'Static' `
                              -DataColumn Name `
                              -AsCard -Filter -SimpleSort -Click -Paginate `
                              -ScriptBlock {
        $stopBtn = New-PodeWebButton -Name 'Stop' `
                                     -Icon 'stop-circle-outline' `
                                     -IconOnly `
                                     -ScriptBlock {
            Stop-Service -Name $WebEvent.Data.Value -Force | Out-Null
            Show-PodeWebToast -Message "$($WebEvent.Data.Value) stopped"
            Sync-PodeWebTable -Id $ElementData.Parent.ID
        }

        $startBtn = New-PodeWebButton -Name 'Start' `
                                      -Icon 'play-circle-outline' `
                                      -IconOnly `
                                      -ScriptBlock {
            Start-Service -Name $WebEvent.Data.Value | Out-Null
            Show-PodeWebToast -Message "$($WebEvent.Data.Value) started"
            Sync-PodeWebTable -Id $ElementData.Parent.ID
        }

        $filter = "*$($WebEvent.Data.Filter)*"
        foreach ($svc in (Get-Service)) {
            if ($svc.Name -inotlike $filter) {
                continue
            }
                             $btns = @{}
            if ($svc.Status -ieq 'running') {
                $btns += $stopBtn
            }
            else {
                $btns += $startBtn
            }

            [ordered]@{
                Name = $svc.Name
                Status = "$($svc.Status)"
                Actions = $btns
            }
        }
    }
    Add-PodeWebPage -Name Services -Icon 'cogs' -Layouts $table  -ScriptBlock {
        $name = $WebEvent.Query['value']
        if ([string]::IsNullOrWhiteSpace($name)) {
            return
        }
        $svc = Get-Service -Name $name | Out-String

        New-PodeWebCard -Name "$($name) Details" -Content @(
            New-PodeWebCodeBlock -Value $svc -NoHighlight
        )
    }
    $form = New-PodeWebForm -Name 'Search' -AsCard -ScriptBlock {
        if ($WebEvent.Data.Name.Length -le 3) {
            Out-PodeWebValidation -Name 'Name' `
                                  -Message 'Name must be greater than 3 characters'
            return
        }
        Get-Process -Name $WebEvent.Data.Name -ErrorAction Ignore |
            Select-Object Name, ID, WorkingSet, CPU |
            New-PodeWebTextbox -Name 'Output' -Multiline -Preformat -ReadOnly |
            Out-PodeWebElement
    } -Content @(
        New-PodeWebTextbox -Name 'Name'
    )
}
```

This will only render a single button for each line. The button will change according to service status.



Name	Status	Actions
AarSvc_7a857	Running	⊙
AdobeARMservice	Running	⊙
AJRouter	Stopped	⊙
ALG	Stopped	⊙

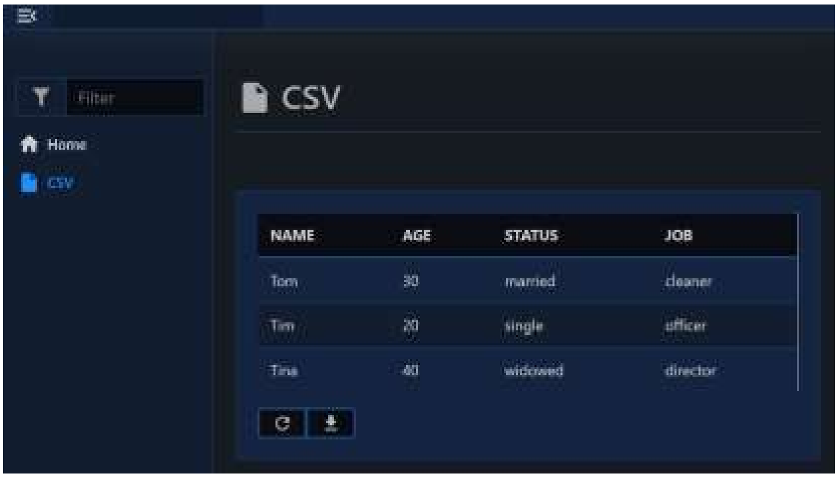
CSV (create a table from comma-separated-values)

There's also the option to render a table straight from a CSV file. If you supply the path to a CSV file via -CsvFilePath, then the table will be built using that file:

Example082.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name CSV -ScriptBlock {

    New-PodeWebContainer -Content @(
    New-PodeWebTable -Name 'Users' -DataColumn UserId -CsvFilePath './users.csv'
    )
   }
}

<#

put a csv file into same directory with  the script
a file should be of the following form:

NAME, AGE,  STATUS, JOB
 Tom,  30, married, cleaner
 Tim,  20,  single, officer
Tina,  40, widowed, director

#>
```

This will render a page like this:



NAME	AGE	STATUS	JOB
Tom	30	married	cleaner
Tim	20	single	officer
Tina	40	widowed	director

Options

Compact

If you have a lot of data that needs to be displayed, and you need to see more on the screen without scrolling, you can supply the -Compact switch to New-PodeWebTable. This will remove extra padding space on rows, to help show more data than normal.

Click

You can set a table's rows to be clickable by passing -Click. This by default will set it so that when a row is clicked the page is reloaded, and the -DataColumn value for that row will be set in the query string as ?value=<value> - available in $WebEvent.Query.Value.

You can set a dynamic click action by supplying the -ClickScriptBlock parameter. Now when a row is clicked the scriptblock is called instead, and the -DataColumn value will now be available via $WebEvent.Data.Value within the scriptblock. The scriptblock expects the normal output actions to be returned.

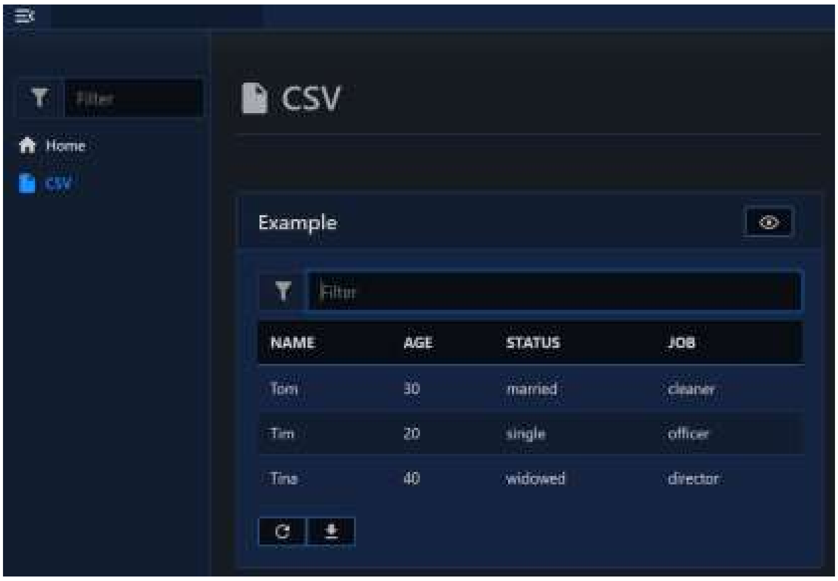
Any values specified to -ArgumentList will also be passed to the -ClickScriptBlock as well.

Filter

You can set a table to be filterable by passing the -Filter switch; this will cause a textbox to be rendered above the table. Any value typed into this textbox will, after a small delay, re-call the table's -ScriptBlock and the filter value will be available in $WebEvent.Data.Filter. You can then return the filtered data and the table will be reloaded:

Example083.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name CSV-filter -ScriptBlock {

    New-PodeWebTable -Name 'Example' -Filter -AsCard -ScriptBlock {
    # load a csv file
    $filePath = Join-Path (Get-PodeServerPath) './users.csv'
    $data = Import-Csv -Path $filePath

    # apply filter if present
    $filter = $WebEvent.Data.Filter
    if (![string]::IsNullOrWhiteSpace($filter)) {
        $filter = "*$($filter)*"
  $data = @($data | Where-Object { ($_.psobject.properties.value -ilike $filter).length -gt 0
})
    }
    # update table
    return $data
} } }
```



NAME	AGE	STATUS	JOB
Tom	30	married	cleaner
Tim	20	single	officer
Tina	40	widowed	director

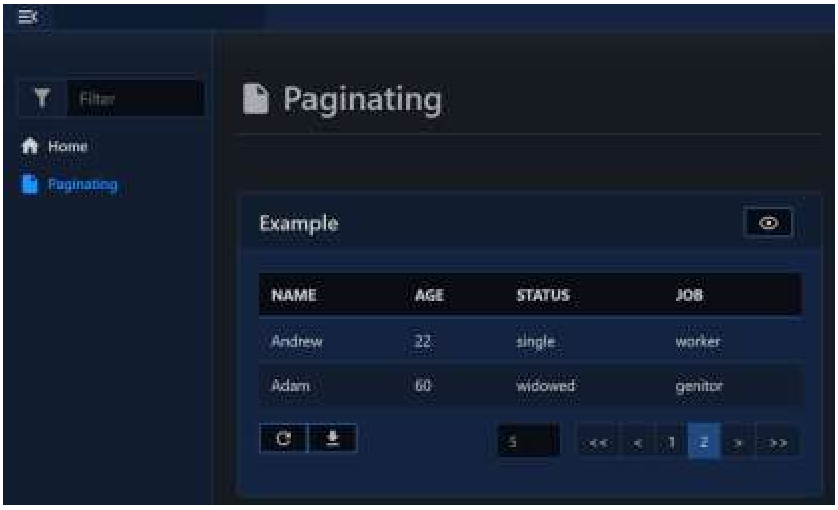
There's also a -SimpleFilter switch available, if you supply this instead of -Filter then a textbox is still displayed, however the filter is done directly via javascript and only applied to the current page. (useful for small tables displaying all data).

Paginate

You can set a table to support paging by passing -Paginate. This will auto-paginate the table data into pages of 20 items, which can also be configured via -PageSize:

Example084.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Paginating -ScriptBlock {

    New-PodeWebTable -Name 'Example' `
                     -Paginate -PageSize 5 `
                     -AsCard -ScriptBlock {
    # load the file
    $filePath = Join-Path (Get-PodeServerPath) './users.csv'
    $data = Import-Csv -Path $filePath

    # update table (Pode.Web will auto-paginate for you)
    return $data
} } }
```



NAME	AGE	STATUS	JOB
Andrew	22	single	worker
Adam	60	widowed	genitor

You can take control of the paging yourself, useful for querying databases, by using the values from $WebEvent.Data.PageIndex and $WebEvent.Data.PageSize. Once you have the pre-paged data, you will need to directly pass this into Update-PodeWebTable along with the -PageIndex and the -TotalItemCount:

Example085.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Table -ScriptBlock {

    New-PodeWebTable -Name 'Dynamic Users' -Paginate `
                     -AsCard -ScriptBlock {
    # load the file
    $filePath = Join-Path (Get-PodeServerPath) './users.csv'
    $data = Import-Csv -Path $filePath
    # apply paging
    $totalCount = $data.Length
    $pageIndex = [int]$WebEvent.Data.PageIndex
    $pageSize = [int]$WebEvent.Data.PageSize
    $data = $data[(($pageIndex - 1) * $pageSize) .. (($pageIndex * $pageSize) - 1)]

    # update table
    $data | Update-PodeWebTable -Name 'Dynamic Users' `
                                -PageIndex $pageIndex `
                                -TotalItemCount $totalCount
} } }
```

Important

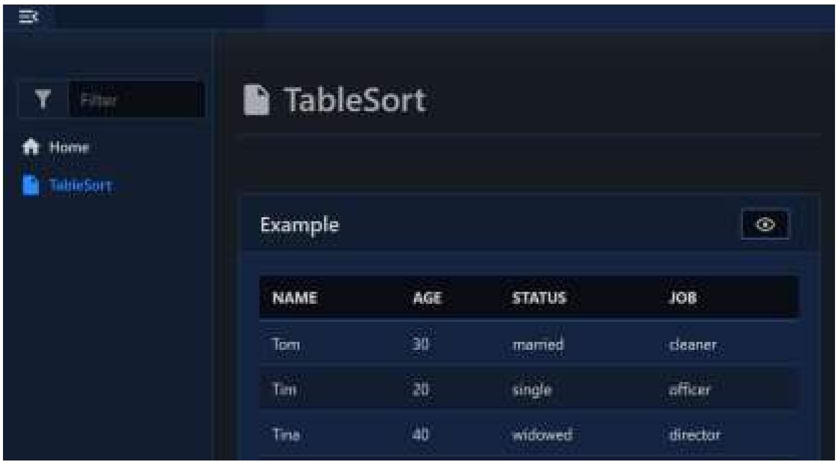
If you don't pass the data into the Update-PodeWebTable output action, then Pode.Web will do this automatically and use the auto-paging - which won't have the desired results!

Sort

You can set a table to be sortable by passing the -Sort switch. When passed then clicking a table's headers will re-call the table's -ScriptBlock; the name of the column to be sorted, as well as the direction (asc/desc), will be available in $WebEvent.Data.SortColumn and $WebEvent.Data.SortDirection. You can then return the sorted data and the table will be reloaded:

Example086.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name TableSort -ScriptBlock {

    New-PodeWebTable -Name 'Example' -Sort -AsCard `
                     -Paginate -Pagesize 10 -ScriptBlock {
    # load the file
    $filePath = Join-Path (Get-PodeServerPath) './users.csv'
    $data = Import-Csv -Path $filePath

    # apply sorting
    $sortColumn = $WebEvent.Data.SortColumn
    if (![string]::IsNullOrWhiteSpace($sortColumn)) {
        $descending = ($WebEvent.Data.SortDirection -ieq 'desc')
        $data = @($data | `
        Sort-Object -Property { $_.$sortColumn } -Descending:$descending)
    }

    # update table
    return $data
} } }
```



NAME	AGE	STATUS	JOB
Tom	30	married	cleaner
Tim	20	single	officer
Tina	40	widowed	director

There's also a -SimpleSort switch available, if you supply this instead of -Sort then sorting is done directly via javascript and only applied to the current page. (useful for small tables displaying all data).

Columns

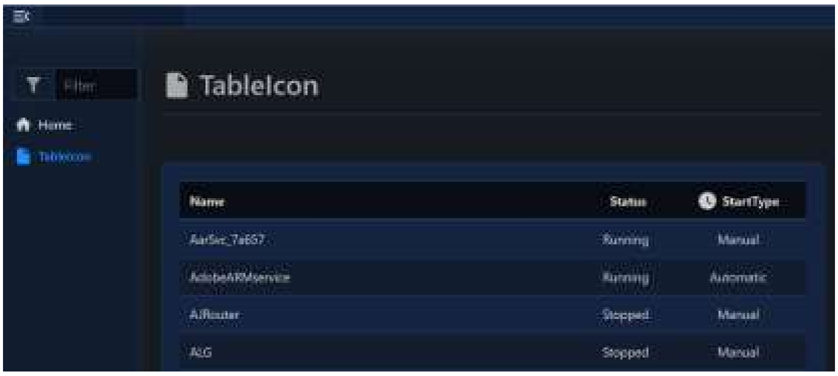
You can set how certain columns in a table behave, such as: width, alignment, display name, and header icon. You can do this via Initialize-PodeWebTableColumn, and by passing these columns into -Columns of New-PodeWebTable.

For example, using the services examples above, you can centrally align the Status/StartType columns and give them icons:

Example087.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name TableIcon -ScriptBlock {

    New-PodeWebContainer -Content @(
    New-PodeWebTable -Name 'Services' -ScriptBlock {
        foreach ($svc in (Get-Service)) {
            [ordered]@{
                Name      = $svc.Name
                Status    = "$($svc.Status)"
                StartType = "$($svc.StartType)"
            }
        }
    } `
    -Columns @(
        Initialize-PodeWebTableColumn -Key Status -Alignment Center -Icon Activity
        Initialize-PodeWebTableColumn -Key StartType -Alignment Center -Icon Clock
) ) } }
```

which renders a table that looks like below:



TableIcon

Name	Status	StartType
AarSvc_7a657	Running	Manual
AdobeARMservice	Running	Automatic
AJRouter	Stopped	Manual
ALG	Stopped	Manual

You can also use the -Hide switch so a particular column isn't displayed on the page.

Width

The -Width of a table column has the default unit of %. If 0 is specified then auto is used instead. Any custom value such as 100px can be used, but if a plain number is used then % is appended.

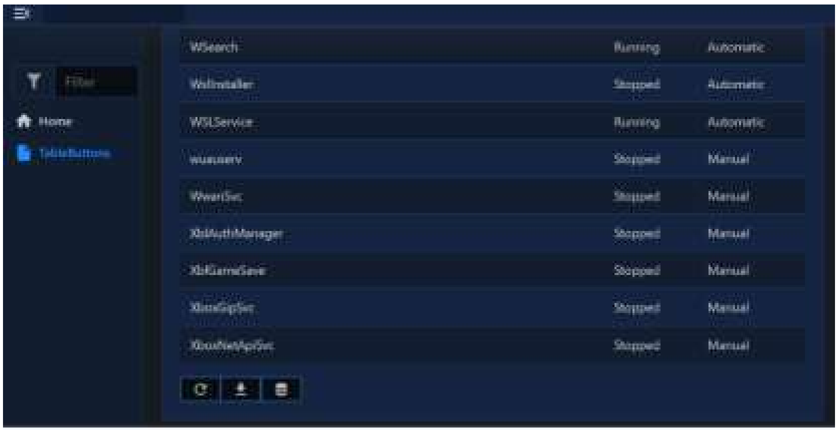
Buttons

At the bottom of a table, there are usually two buttons on the left: Refresh and Export. You can add more buttons to the end of the table by piping a new table into Add-PodeWebTableButton:

Example088.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name TableButtons -ScriptBlock {

    $table = New-PodeWebTable -Name 'Services' -ScriptBlock {
    foreach ($svc in (Get-Service)) {
        [ordered]@{
            Name      = $svc.Name
            Status    = "$($svc.Status)"
            StartType = "$($svc.StartType)"
        }
    }
    }
    $table | Add-PodeWebTableButton -Name 'Excel' -Icon Database -ScriptBlock {
        $WebEvent.Data | Export-Csv -Path $path -NoTypeInformation
    }
    New-PodeWebContainer -Content $table
   }
}
```

The script above will render aditional button at the end of the table



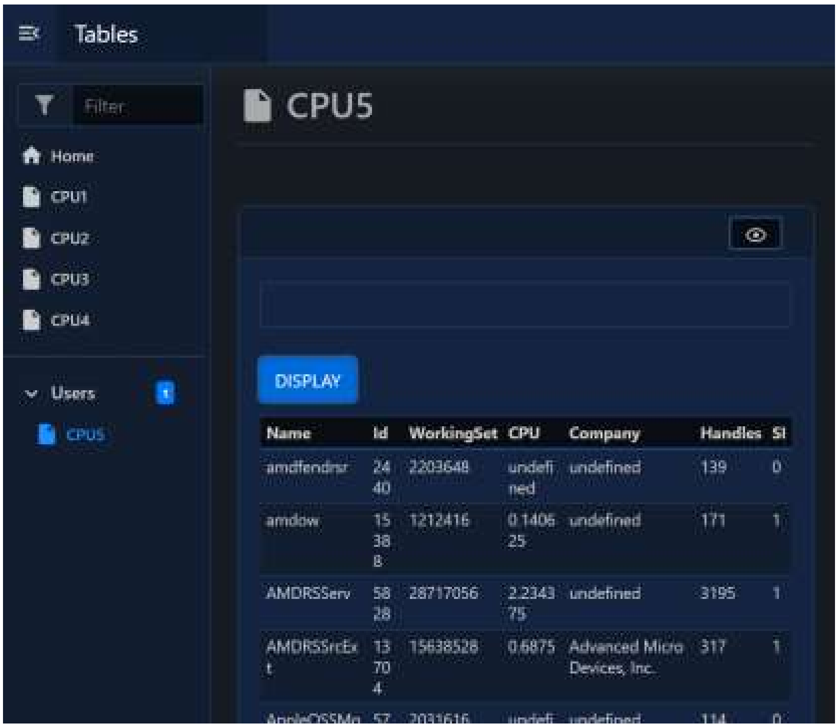
WSearch	Running	Automatic
WslInstaller	Stopped	Automatic
WSLService	Running	Automatic
wuauserv	Stopped	Manual
WwanSvc	Stopped	Manual
XblAuthManager	Stopped	Manual
XblGameSave	Stopped	Manual
XboxGipSvc	Stopped	Manual
XboxNetApiSvc	Stopped	Manual

Example088b.ps1 (5 tables techniques)

```
function PS_func(){
         Get-Process|select Name, ID, WorkingSet, CPU, Company, Handles, Id, SI
         }

Start-PodeServer -Threads 2 {
    Add-PodeEndpoint -Address localhost -Port 8080 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title 'Tables' -Theme Dark

##### METHOD1 - using Layouts
      $table1 = New-PodeWebTable -Name ' ' -SimpleFilter `
                                          -SimpleSort `
                                          -Compact `
                                          -AsCard `
                                          -ScriptBlock `
             { PS_func }
                  Add-PodeWebPage -Name 'CPU1' -Layouts $table1

##### METHOD2 - using Dynamic ScriptBlock
      Add-PodeWebPage -Name 'CPU2' -ScriptBlock `
          { New-PodeWebTable -Name ' ' -SimpleFilter `
                                       -SimpleSort `
                                       -Compact `
                                       -AsCard `
                                       -ScriptBlock `
          { PS_func } }


##### METHOD3 - using Functions + Invoke-expression
    # (useful to create menus in a loop)
      $e="Add-PodeWebPage  -Name 'CPU3' -ScriptBlock {
          New-PodeWebTable -Name T3 -SimpleFilter -SimpleSort ``
                           -Compact -AsCard -ScriptBlock { PS_func } } "
      Invoke-Expression $e

##### METHOD4 - using Functions + Invoke-expression
    # + confirmation button + Views/Contents
      $e="Add-PodeWebPage -Name 'CPU4' -ScriptBlock {
          New-PodeWebForm  -Name `"PSF`" -SubmitText `"DISPLAY`" ``
                                         -AsCard  -ScriptBlock {
              update-PodeWebTable -name `"PST`" -Data @(PS_func)
              }  -Content @(
                  New-PodeWebContainer -Content @(
                  New-PodeWebText -Value `" `")
                  New-PodeWebTable -Name `"PST`" -SimpleFilter -SimpleSort -Compact
                  New-PodeWebParagraph -value `" `"
                  ) } "
      Invoke-Expression $e

##### METHOD5 - simplified table with confirmation and in separated menu
      $e=echo "Add-PodeWebPage -Name `"CPU5`" -Group 'Users' ``
                               -scriptblock {
                                New-PodeWebForm -Name `" `" ``
                               -SubmitText `"DISPLAY`" -AsCard ``
                               -ScriptBlock { PS_func|ft|Out-Default
                                              PS_func|Out-PodeWebTable} ``
                               -Content @(New-PodeWebContainer ``
                               -Content @(New-PodeWebText -Value `" `"))}"
```

```
        Invoke-Expression $e
}
```

In this example you will find 5 different way to build a table from data.



Name	Id	WorkingSet	CPU	Company	Handles	SI
amdfendrsr	2440	2203648	undefined	undefined	139	0
amdow	153888	1212416	0.140625	undefined	171	1
AMDRSServ	5828	28717056	2.234375	undefined	3195	1
AMDRSSrcExt	13704	15638528	0.6875	Advanced Micro Devices, Inc.	317	1
AppleOSSMg	57	2031616	undefi	undefined	114	0

Text

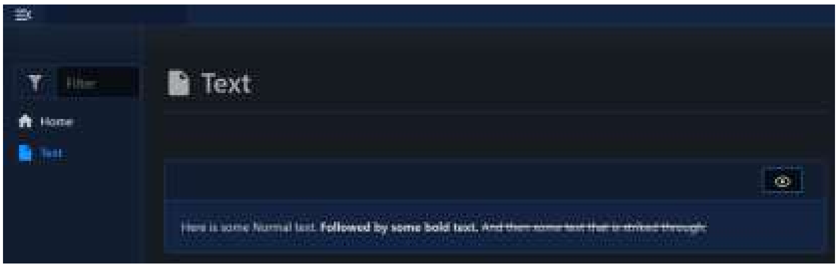
Support	
Events	No

You can render different types of text/typography to your page by using New-PodeWebText. You can specify the -Value to display, and then a custom -Style to render the text; such as Normal, Bold, Italic, etc. (default is Normal):

Example089.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Text -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebText -Value 'Here is some Normal text.'
    New-PodeWebText -Value 'Followed by some bold text.' -Style Bold
    New-PodeWebText -Value 'And then some text that is striked through.' -Style StrikeThrough
) } }
```

Which looks like below:



Here is some Normal text. **Followed by some bold text.** ~~And then some text that is striked through.~~

Pronunciation

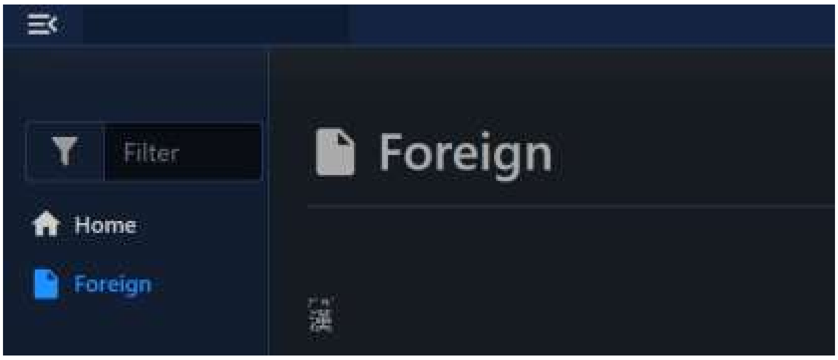
You can add small pronunciation text above displayed text by using -Pronunciation:

Example090.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Foreign -ScriptBlock {

    New-PodeWebText -Value '漢' -Pronunciation 'ㄏㄢˋ'

} }
```

(Note: for the above to render properly, use PowerShell 6+)

This is how it looks like on PowerShell 5.x



Foreign

漢

Textbox

Support	
Events	Yes

A textbox element is a form input element; you can render a textbox, single and multiline, to your page using New-PodeWebTextbox.

A textbox by default is a normal plain single lined textbox, however you can customise its -Type to Email/Password/etc. To change the textbox to be multilined you ca supply -Multiline.

Textboxes also allow you to specify -AutoComplete values.

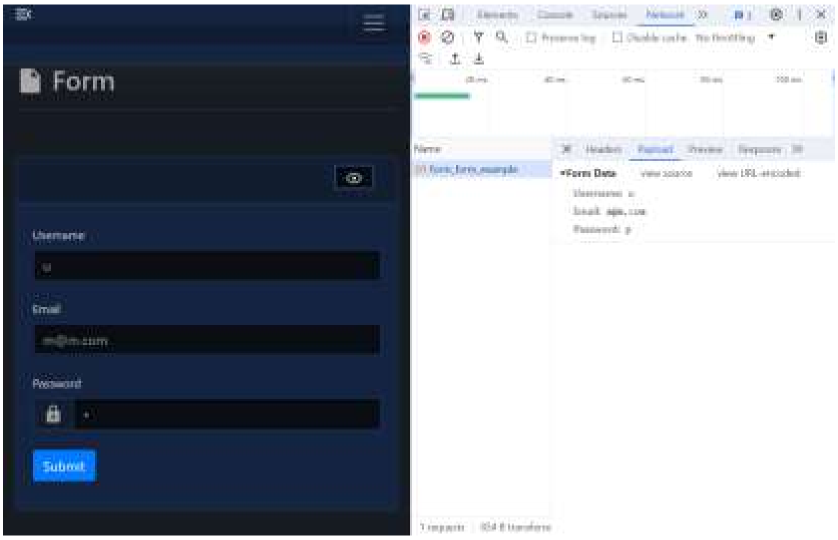
Single

A default textbox is just a simple single lined textbox. You can change the type to Email/Password/etc using the -Type parameter:

Example091.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Form -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $username = $WebEvent.Data['Username']
        $email = $WebEvent.Data['Email']
        $password = $WebEvent.Data['Password']
        (echo "$username, $email, $password") | out-default
    } -Content @(
        New-PodeWebTextbox -Name 'Username'
        New-PodeWebTextbox -Name 'Email' -Type Email
        New-PodeWebTextbox -Name 'Password' -Type Password -PrependIcon Lock
) ) } }
```

Which looks like below (press F12 for analysis of Payload data):



Form

Username

u

Email

m@m.com

Password

Submit

Form Data view source view URL-encoded

Username: u

Email: m@m.com

Password: p

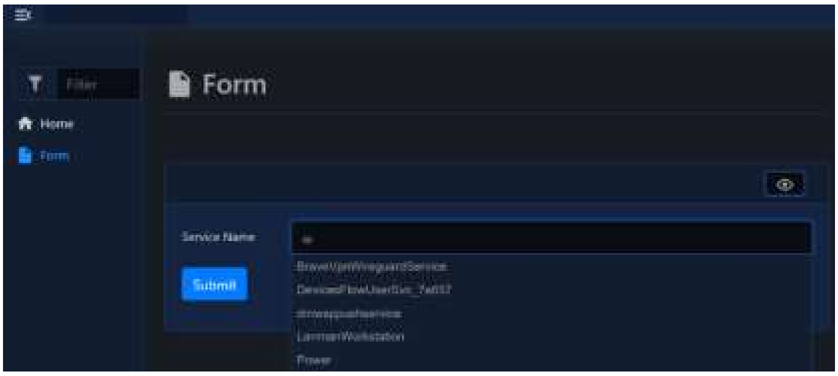
AutoComplete

For a single textbox, you can supply autocomplete values via a scriptblock passed to -AutoComplete. This scriptblock should return an array of strings, and will be called once when the textbox is initially loaded:

Example092.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name FormAuto -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $svcName = $WebEvent.Data['Service Name']
    } -Content @(
        New-PodeWebTextbox -Name 'Service Name' -AutoComplete {
            return @(Get-Service).Name
} ) ) } }
```

Which looks like below:



Form

Service Name

w

BraveVpnWireguardService

DevicesFlowUserSvc_7a657

dmwappushservice

LanmanWorkstation

Power

Submit

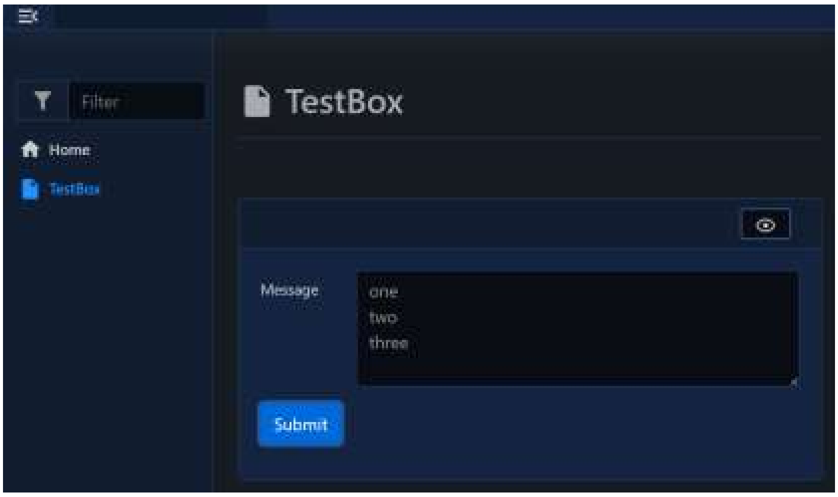
Multiline

A mutlilined textbox can be displayed by passing -Multiline. You cannot change the type of this textbox, it will always allow freestyle text:

Example093.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name TestBox -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $message = $webEvent.Data['Message']
        $message|out-default
    } -Content @(
        New-PodeWebTextbox -Name 'Message' -Multiline
) ) } }
```

Which looks like below:



By default it shows the first 4 lines of text, this can be altered using the -Size parameter.

Inline

You can render this element inline with other non-form elements by using the -NoForm switch. This will remove the form layout, and render the element more cleanly when used outside of a form.

Display Name

By default the label displays the -Name of the element. You can change the value displayed by also supplying an optional -DisplayName value; this value is purely visual, when the user submits the form the value of the element is still retrieved using the -Name from $WebEvent.Data.

Size

The -Width of a textbox has the default unit of %. If 0 is specified then auto is used instead. Any custom value such as 100px can be used, but if a plain number is used then % is appended.

The -Height of the textbox is how many lines are displayed when the textbox is multilined.

Tile

Support	
Events	No

A tile is a small coloured container, that contains either a static value or more elements. There purpose is to display quick informational data like: CPU, counters, charts, etc.

To add a tile you use New-PodeWebTile, and supply a -Name and either a -ScriptBlock or -Content.

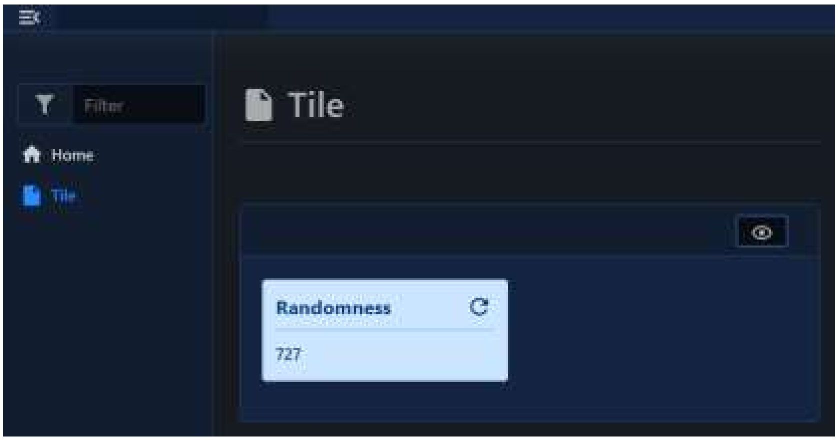
Value

The simplest tile is one that shows a flat value. This value should be returned from a -ScriptBlock, for example to show an random number:

Example094.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Tile -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebTile -Name 'Randomness' -ScriptBlock {
        return (Get-Random -Minimum 0 -Maximum 1000)
} ) } }
```

Which looks like below:



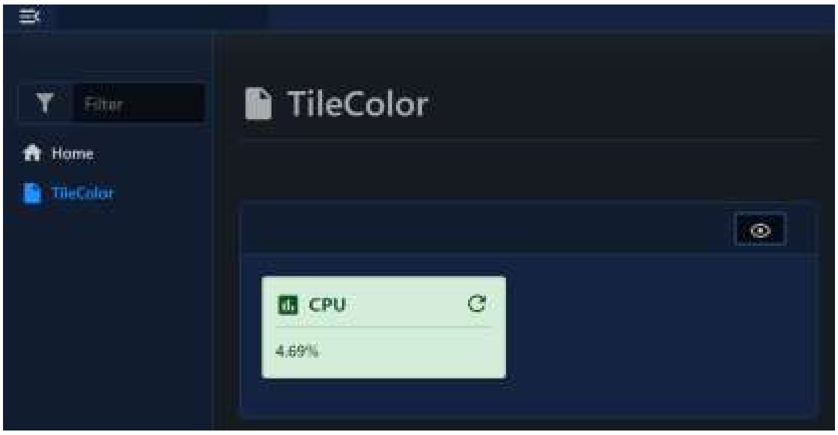
If you click the refresh icon, the scriptblock will be re-called, and the value updated.

Or, if you want to display the current CPU but change the colour if it goes above 50%, then you can use Update-PodeWebTile instead:

Example095.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name TileColor -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebTile -Name 'CPU' -Icon 'chart-box' -ScriptBlock {
        $cpu = ((Get-Counter -Counter '\Processor(_Total)\% Processor Time' `
        -SampleInterval 1 -MaxSamples 2).CounterSamples.CookedValue |
         Measure-Object -Average).Average

        $colour = 'green'
        if ($cpu -gt 90) {
            $colour = 'red'
        }
        elseif ($cpu -gt 50) {
            $colour = 'yellow'
        }

        $cpu = [System.Math]::Round($cpu, 2)
        "$($cpu)%" | Update-PodeWebTile -ID $ElementData.ID -Colour $colour
} ) } }
```

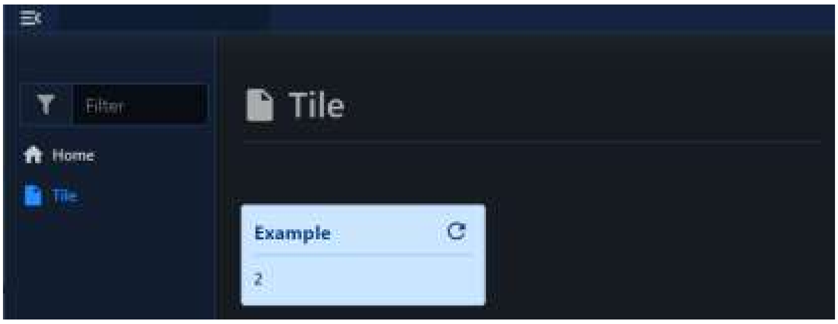
Which looks like below:



You can pass values to the scriptblock by using the -ArgumentList parameter. This accepts an array of values/objects, and they are supplied as parameters to the scriptblock:

Example096.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Tile -ScriptBlock {

    New-PodeWebTile -Name 'Example' -ArgumentList 'Value1', 2, $false -ScriptBlock {
    param($value1, $value2, $value3)

    # $value1 = 'Value1'
    # $value2 = 2
    # $value3 = $false

    "$value2" | Update-PodeWebTile -ID $ElementData.ID
} } }
```



Elements

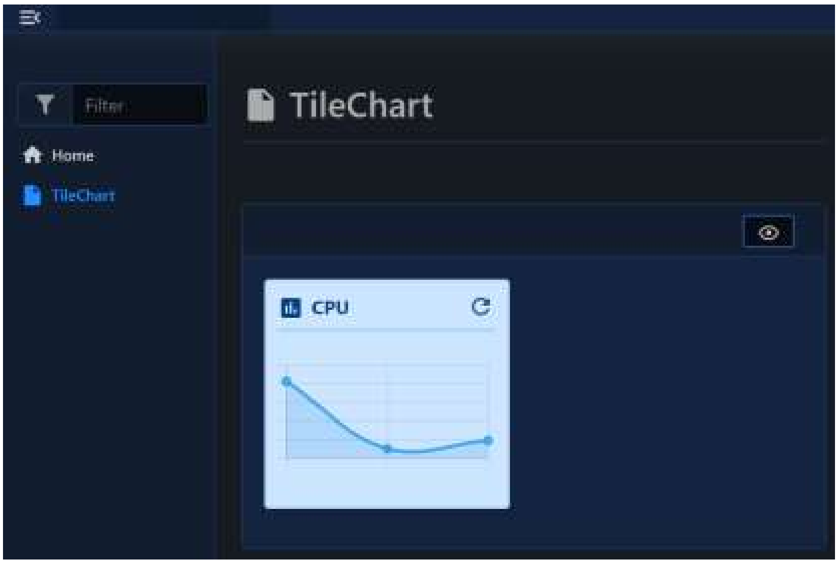
You can also display other elements within a tile, such as a chart. To display elements, add them via the -Content parameter.

For example, the following will display a tile showing the current CPU of your machine, and will auto-refresh every minute:

Example097.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name TileChart -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebTile -Name 'CPU' -Icon 'chart-box' -Content @(
        New-PodeWebCounterChart -Counter `
        '\Processor(_Total)\% Processor Time' -MaxItems 10
) ) } }
```

Which looks like below:



Clickable

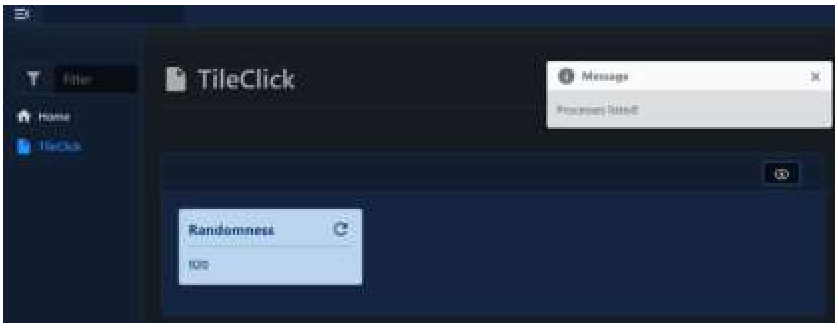
You can make a whole tile clickable by supplying a -ClickScriptBlock. When clicked the scriptblock will be called, and you can use any output action within the scriptblock.

For example, the following will show a tile with a random number, but when clicked it will display a toast message:

Example098.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name TileClick -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebTile -Name 'Randomness' -ScriptBlock {
        return (Get-Random -Minimum 0 -Maximum 1000)
    } `
    -ClickScriptBlock {
        Show-PodeWebToast -Message 'Processes listed!'
        get-process|out-default
} ) } }
```

When clicked, this will display Toast message and proccesses at the console.



Processes listed!

Refresh

By default, all tiles will show a "refresh" icon at the top-right corner, and when clicked the tile's data will be refreshed. You can hide the refresh icon by using -NoRefresh.

To set a tile to automatically refresh once a minute, you can supply the -AutoRefresh switch.

Display

Tiles will be displayed inline, and to display one on a new line you can supply the -NewLine switch.

If you want to display the tiles more neatly in a line, it's recommended to display them using a Grid.

Timer

Support	
Events	No

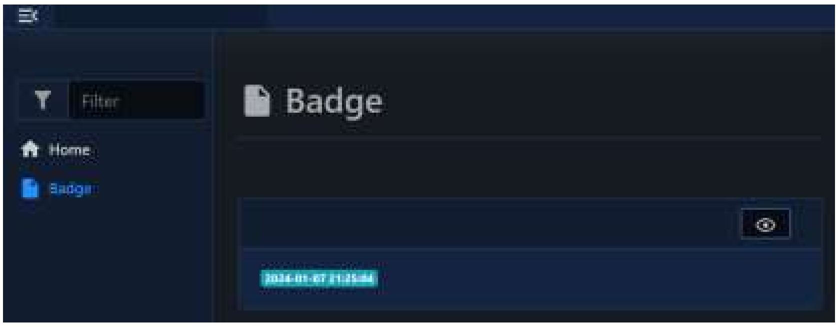
A timer is a non-visible element, it sets up a javascript timer in the background that periodically (60s) invokes logic. You can add a timer using New-PodeWebTimer, and they're mostly used with the outputs function to alter the page.

The below example sets up a timer that will update the badge's value and colour every 10 seconds:

Example099.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Badge -ScriptBlock {

    New-PodeWebTimer -Interval 10 -ScriptBlock {
        $rand = Get-Random -Minimum 0 -Maximum 3
        $colour = (@('Green', 'Yellow', 'Cyan'))[$rand]
        Update-PodeWebBadge -Id 'bdg_example' `
              -Value ([datetime]::Now.ToString('yyyy-MM-dd HH:mm:ss')) `
              -Colour $colour
    }

New-PodeWebCard -Content @(
    New-PodeWebBadge -Id 'bdg_example' `
       -Value ([datetime]::Now.ToString('yyyy-MM-dd HH:mm:ss')) `
       -Colour Cyan)

} }
```

This will display clock in the badge, updated every 10 seconds:

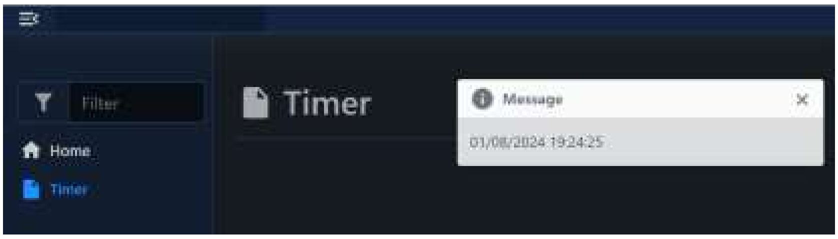


You can pass values to the scriptblock by using the -ArgumentList parameter. This accepts an array of values/objects, and they are supplied as parameters to the scriptblock:

Example100.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name Timer -ScriptBlock {

     New-PodeWebTimer -Interval 5 -ArgumentList 'Value1', 2, $(date) -ScriptBlock {
     param($value1, $value2, $value3)
     # $value1 = 'Value1'
     # $value2 = 2
     # $value3 = $false
     get-process|out-default
     Show-PodeWebToast -Message $value3
    }
   }
}
```

This one will print processes every 10 seconds and wil also display a toast message with the current date. (it wil not react under 10 seconds)



01/08/2024 19:24:25

Video

Support	
Events	Yes

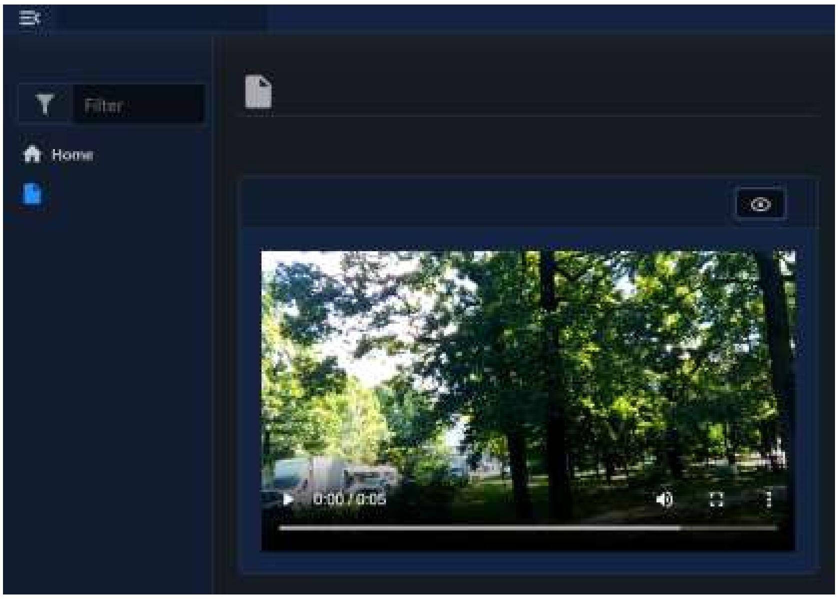
To play video clips on your site you can use 10 seconds. The video can be set to autoplay and loop, and can also accept multiple sources and tracks.

The sources specified can be done so via -Source and New-PodeWebVideoSource. Sources can only be of type MP4, OGG and WebM, and at least one source must be specified. An optional thumbnail Url can also be supplied, and this will be the image used for the video before it is played:

Example101.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebVideo -Name 'video' -Width 100 `
       -Thumbnail 'https://samplelib.com/lib/preview/mp4/sample-5s.jpg' -Source @(
        New-PodeWebVideoSource -Url 'https://samplelib.com/lib/preview/mp4/sample-5s.mp4'
    )
    )
} }
```

Which looks like below:



Any optional tracks you wish to specify can be done via -Track and New-PodeWebMediaTrack. Tracks can only be of type VTT, and can be used for subtitles, captions, metadata, etc.

The -Language is mandatory if the track's -Type is subtitles, and should be a 2-letter language code.

I was unable to put subtitles to work.

Example102.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
     New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebVideo -Name 'example' -Thumbnail
'https://samplelib.com/lib/preview/mp4/sample-5s.jpg' `
    -Source @(
        New-PodeWebVideoSource -Url 'https://samplelib.com/lib/preview/mp4/sample-5s.mp4'
    ) `
    -Track @(
        New-PodeWebMediaSource -Url './english.vtt' -Language 'en' `
                               -Title 'English' -Type 'subtitles' -Default
        #New-PodeWebMediaTrack -Url $(Join-Path (Get-PodeServerPath) './English.vtt') `
        #                      -Language 'en' `
        #                      -Title 'English' -Type 'subtitles' -Default
    )
    )
} }
```

Size

The -Width and -Height of a video have the default unit of %. If 0 is specified then 20% is used for the width, and 15% for the height instead. Any custom value such as 100px can be used, but if a plain number is used then % is appended.

Events

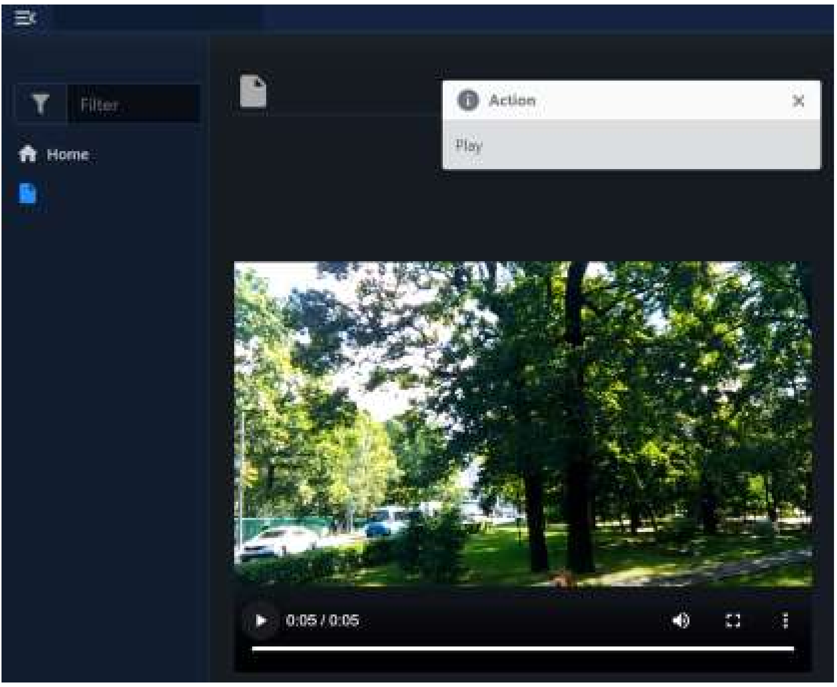
The following specific events are supported by the Video element, and can be registered via Register-PodeWebMediaEvent:

Name	Description
CanPlay	Fires when the browser is ready to play the video
Ended	Fires when the video has finished playing, unless looping
Pause	Fires when the video is paused
Play	Fires when the video is played, or un-paused

Example103.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebVideo -Name 'example' -Width 100 -height 100 `
       -Thumbnail 'https://samplelib.com/lib/preview/mp4/sample-5s.jpg' -Source @(
        New-PodeWebVideoSource -Url 'https://samplelib.com/lib/preview/mp4/sample-5s.mp4'
    ) |
    Register-PodeWebMediaEvent -Type Play -ScriptBlock {
             Show-PodeWebToast -Title 'Action' -Message $EventType
    }
} }
```

a



Pode.Web Layouts

Accordion

An accordion layout is an array of accordion bellows with content. They are displayed in a collapsible manor, with the first bellow being expanded. When another bellow is expanded, all other bellows are collapsed.

The bellows take an array of components via -Content, that can be either other layouts or raw elements.

Usage

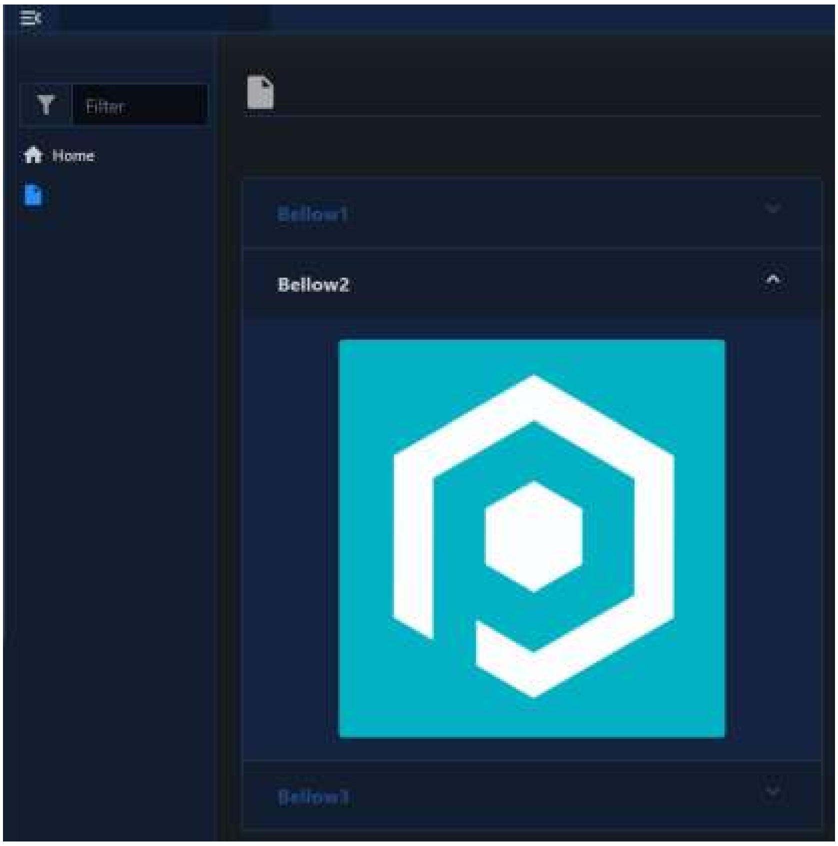
To create an accordion layout you use New-PodeWebAccordion, and supply it an array of -Bellows using New-PodeWebBellow. The bellows themselves accept an array of other elements/layouts as -Content.

For example, the following renders an accordion with 3 bellows each containing an image:

Example104.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebAccordion -Bellows @(
    New-PodeWebBellow -Name Bellow1 -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebBellow -Name Bellow2 -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebBellow -Name Bellow3 -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    )
} }
```

Which would look like below:

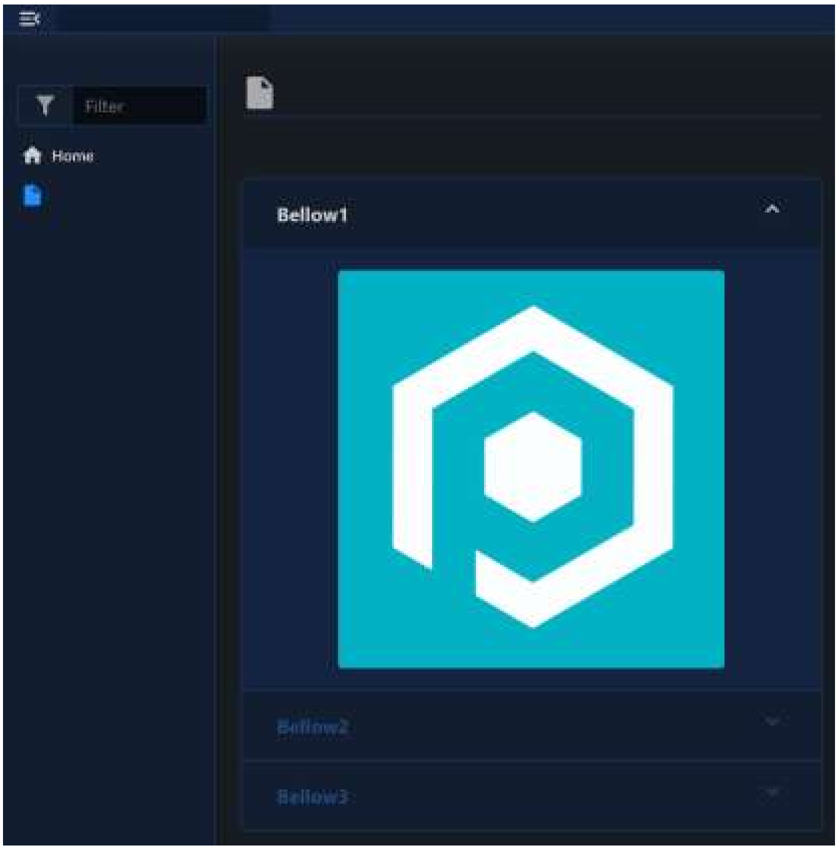


Cycling

You can render accordions that automatically cycle through their bellows every X seconds, by using the -Cycle switch and -CycleInterval parameter. The default interval is every 15secs:

Example105.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebAccordion -Cycle -Bellows @(
    New-PodeWebBellow -Name Bellow1 -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebBellow -Name Bellow2 -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebBellow -Name Bellow3 -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
) ) } }
```


Filter
Home
Bellow1
Bellow2
Bellow3

Breadcrumbs

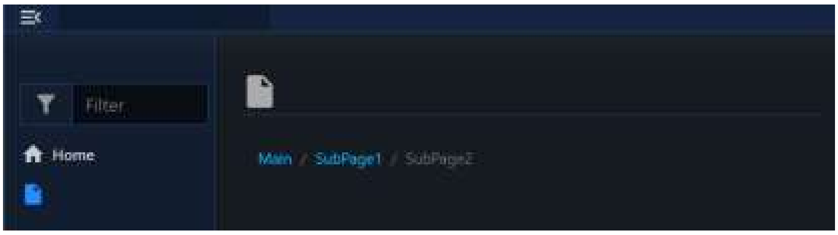
You can control optional breadcrumbs that appear at the top of each page using Set-PodeWebBreadcrumb and New-PodeWebBreadcrumbItem. By default, pages will show breadcrumbs if the appropriate query string parameters are set (base and value).

Or, you can control the breadcrumbs yourself, by creating breadcrumb items and then setting them on the page. Only one item can be active, and each item can have a URL that clicking the item takes it back to:

Example106.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    Set-PodeWebBreadcrumb -Items @(
    New-PodeWebBreadcrumbItem -Name 'Main' -Url '/pages/PageName'
    New-PodeWebBreadcrumbItem -Name 'SubPage1' -Url '/pages/PageName?value=stuff1'
    New-PodeWebBreadcrumbItem -Name 'SubPage2' -Url '/pages/PageName?value=stuff2' -Active
) } }
```

Which looks like below:



Card

A card is a layout that renders with an optional title, and can be collapsed by the end-user.

A card takes an array of components via -Content, that can be either other layouts or raw elements.

Usage

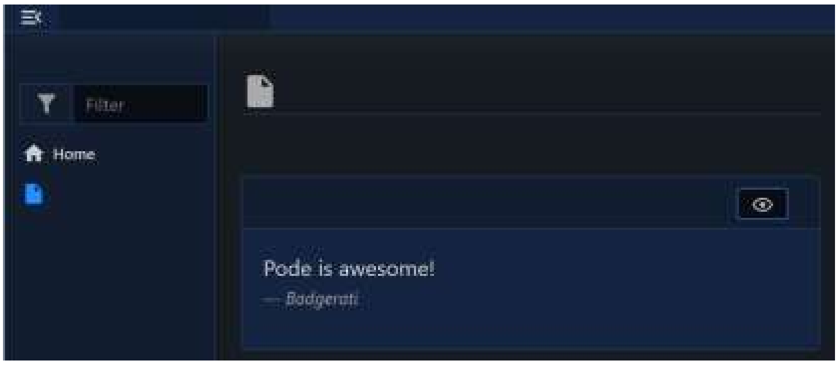
To create a card layout you use New-PodeWebCard, and supply it an array of -Content.

For example, the below renders a card with a quote:

Example107.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebQuote -Value 'Pode is awesome!' -Source 'Badgerati'
) } }
```

Which would look like below:



Pode is awesome!

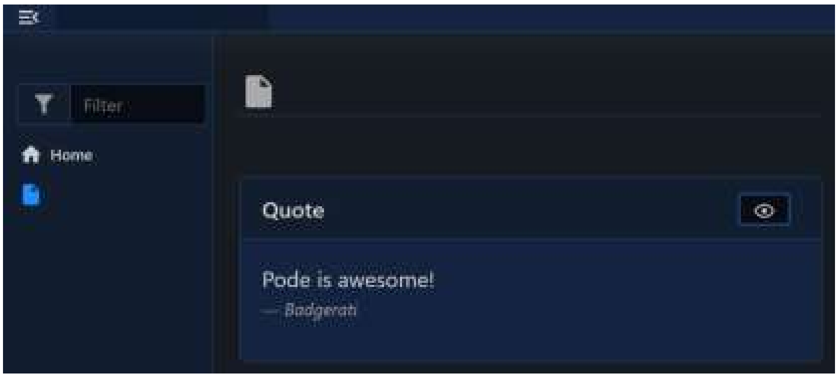
— Badgerati

Or with a title:

Example108.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Name 'Quote' -Content @(
    New-PodeWebQuote -Value 'Pode is awesome!' -Source 'Badgerati'
) } }
```

Which would look like below:



Pode is awesome!

— *Badgerati*

Carousel

A carousel layout is an array of slides with content, each slide can also have a title and a message. The slides will periodically move between one-another, with arrows on either side to manually move to the next slide.

The slides take an array of components via -Content, that can be either other layouts or raw elements.

Usage

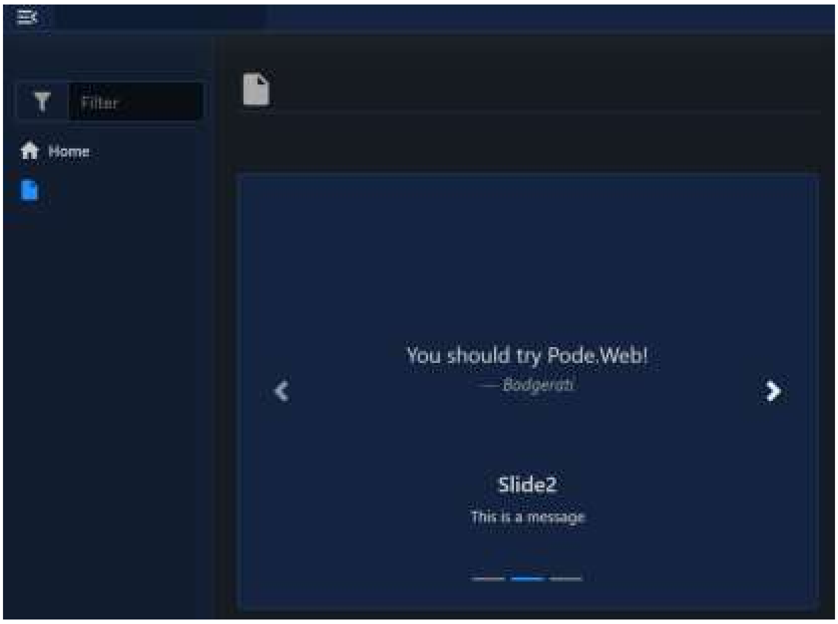
To create a carousel layout you use New-PodeWebCarousel, and supply it an array of -Slides using New-PodeWebSlide. The slides themselves accept an array of -Content.

For example, the below renders a carousel with 3 slides each containing an image, a title, and a message:

Example109.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCarousel -Slides @(
    New-PodeWebSlide -Title Slide1 -Message 'This is a message' -Content @(
        New-PodeWebContainer -Nobackground -Content @(
            New-PodeWebQuote -Value 'Pode is awesome!' -Source 'Badgerati' -Alignment Center
        )
    )
    New-PodeWebSlide -Title Slide2 -Message 'This is a message' -Content @(
     New-PodeWebContainer -Nobackground -Content @(
     New-PodeWebQuote -Value 'You should try Pode.Web!' -Source 'Badgerati' -Alignment Center
        )
    )
    New-PodeWebSlide -Title Slide3 -Message 'This is a message' -Content @(
        New-PodeWebContainer -Nobackground -Content @(
            New-PodeWebQuote -Value 'PowerShell rocks!' -Source 'Badgerati' -Alignment Center
) ) ) } }
```

Which would look like below:



You should try Pode.Web!

— Badgerati

Slide2

This is a message

Container

A container is similar to a card layout, but it has not title, nor can it be collapsed. It's a way of group multiple elements together, with the option of making the background of the container transparent.

A container takes an array of components via -Content, that can be either other layouts or raw elements.

Usage

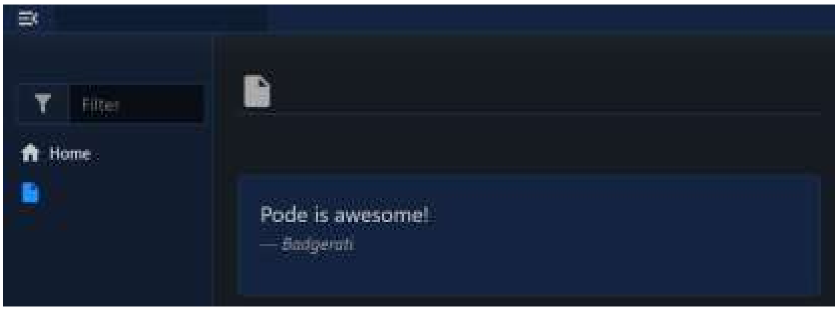
To create a container layout you use New-PodeWebContainer, and supply it an array of -Content.

For example, the below renders a container with a quote:

Example110.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -Content @(
    New-PodeWebQuote -Value 'Pode is awesome!' -Source 'Badgerati'
    )
} }
```

Which would look like below:

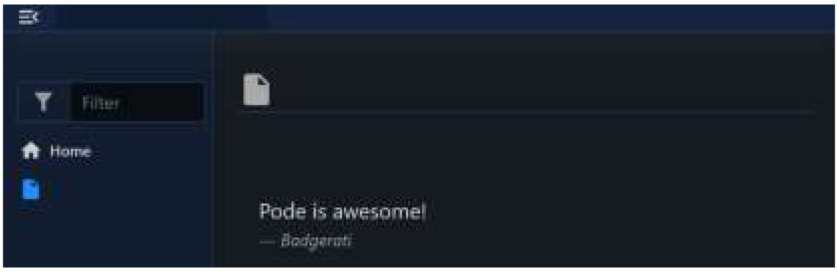


Or with no background:

Example111.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebQuote -Value 'Pode is awesome!' -Source 'Badgerati'
    )
} }
```

Which would look like below:



Pode is awesome!
— Badgerati

Grid

A grid layout is an array of cells with content, equally spaced in size, that can be either horizontal or vertical in orientation.

The cells take an array of components via -Content, that can be either other layouts or raw elements.

Usage

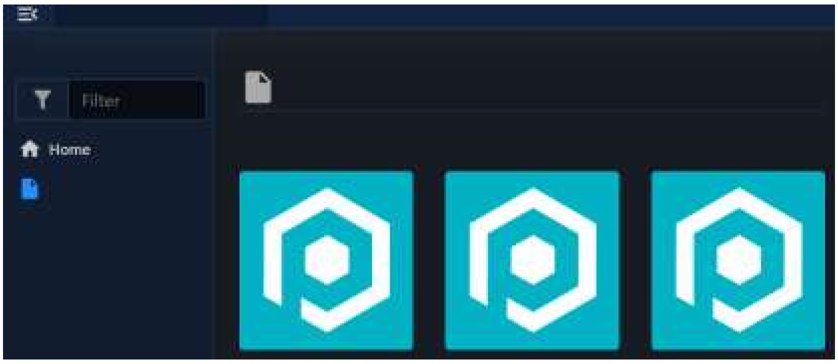
To create a grid you use New-PodeWebGrid, and supply it an array of -Cells using New-PodeWebCell. The cells themselves accept an array of -Content.

For example, the below renders a 3 celled horizontal grid of centered images:

Example112.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebGrid -Cells @(
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
) ) } }
```

Which would look like below:



A grid also has an optional -Width parameter, and when this parameter isn't supplied the cells are all placed into one horizontal row. However, if you supply a -Width then this limits the number of cells that can be rendered on a row. For example if you pass 7 cells with a width of 3, then you'll end up with 3 rows: 2 rows of 3 cells and 1 row of 1 cell - the last row is padded to match match the width of the other rows.

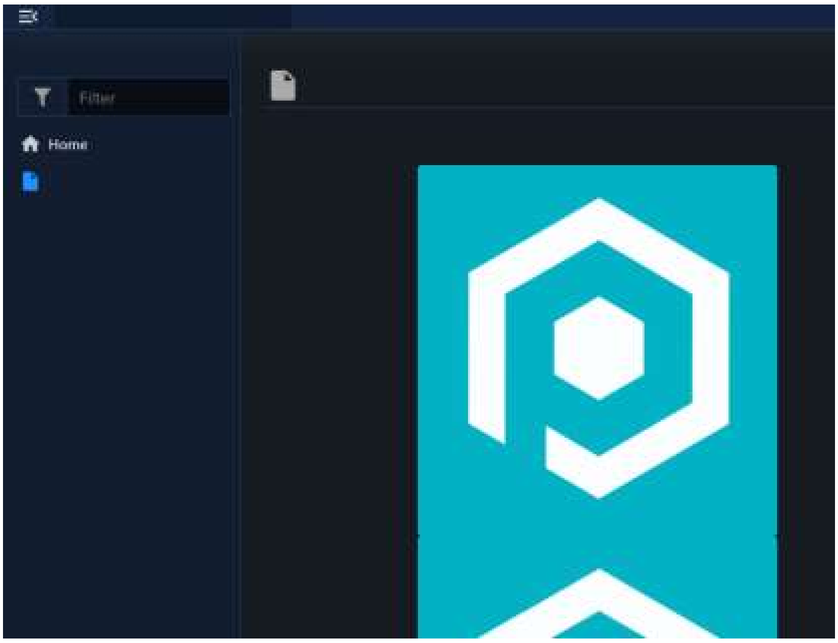
Vertical

You can render the cells of a grid vertically by either supplying -Width 1 or by using the -Vertical switch on New-PodeWebGrid:

Example113.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebGrid -Vertical -Cells @(
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
) ) } }
```

Which would look like below:



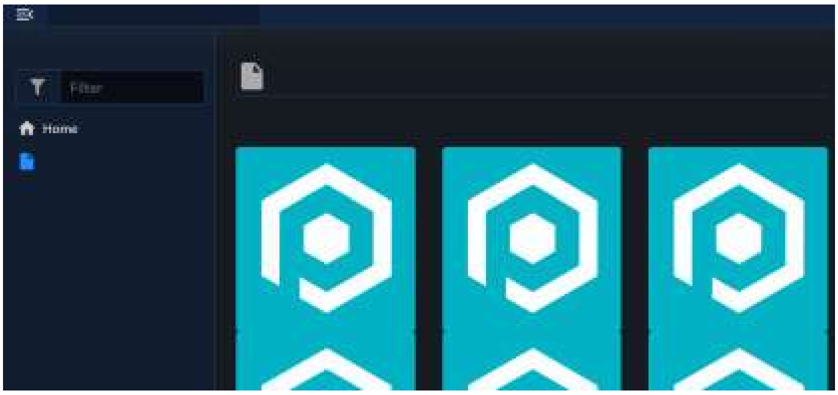
Grids in Grids

You can put grids within grids to render a multi-dimensional grid/cell layout. For example, to create a 3x3 grid of cells with images:

Example114.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebGrid -Width 3 -Cells @(
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
) ) } }
```

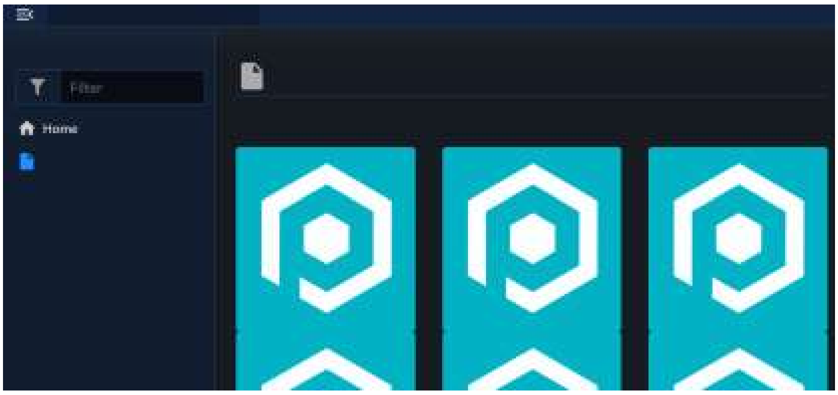
Which would look like below:



The above is useful if you want pure control over the grid layout. However, the following would also produce a 3x3 grid by just using the -Width parameter, and supplying all the cells to one grid:

Example115.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebGrid -Width 3 -Cells @(
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
) ) } }
```



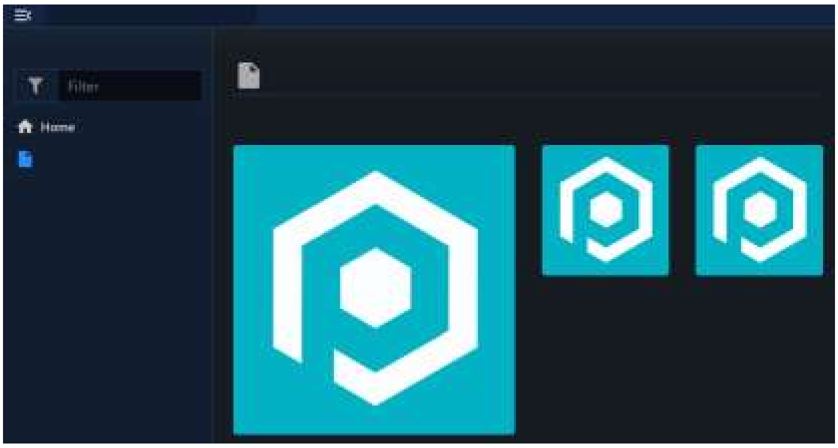
Cell Width

You can optionally specify the width of a cell within a grid, by using the -Width parameter on New-PodeWebCell. A grid is split up into 12 segments, and the -Width parameter lets you specify a value between 1 and 12. A value of 12 means the cells takes up the full width of the grid, where as 6 would be half the width.

You can also supply the value as a percentage as well; 50% being 6 segments, etc.

Example116.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebGrid -Cells @(
    New-PodeWebCell -Width 6 -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Width '25%' -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
    )
    New-PodeWebCell -Width 3 -Content @(
        New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
) ) } }
```



The above example would display a grid, with 1 cell occupying 50% of the width, and the last 2 just 25% each.

Hero

A hero is a layout that renders with a title, message, and extra optional content. They're the big display messages that are normally seen at the top of websites with buttons like "Download Now!" etc.

A hero can take an array of components via -Content, that can be either other layouts or raw elements.

Usage

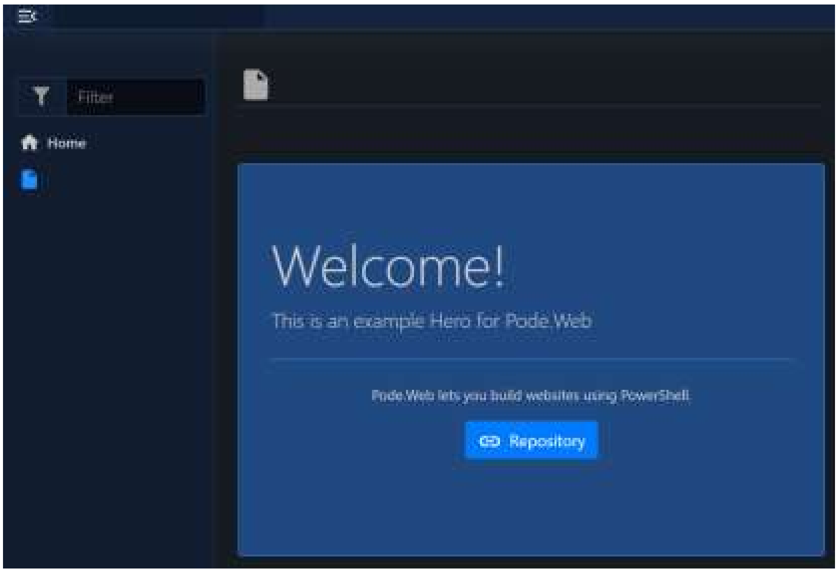
To create a hero layout you use New-PodeWebHero, and supply it a -Title, -Message and an optional array of -Content.

For example, the below renders a hero with a title, message, and content for further text and a button:

Example117.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebHero -Title 'Welcome!' `
                    -Message 'This is an example Hero for Pode.Web' -Content @( `
    New-PodeWebText -Value 'Pode.Web lets you build websites using PowerShell.' `
                    -InParagraph -Alignment Center
    New-PodeWebParagraph -Alignment Center -Elements @( `
        New-PodeWebButton -Name 'Repository' -Icon Link `
           -Url 'https://github.com/Badgerati/Pode.Web'
) ) } }
```

Which would look like below:



Welcome!

This is an example Hero for Pode.Web

Pode.Web lets you build websites using PowerShell

Modal

A modal is a layout that renders on top of all other content on your web page - such as prompts to comfirm information before performing an action, or a quick edit dialog.

A modal takes an array of components via -Content, that can be either other layouts or raw elements.

Usage

To create a modal layout you use New-PodeWebModal, supplying a -Name and any -Content. The modal will then be on the page, but will be hidden until required. You can also supply a -ScriptBlock with logic to be invoked when the modal's Submit button is clicked.

To show the modal, you can use the output action Show-PodeWebModal, and supply the -Name of the modal to show. You can populate elements within the modal just prior to it being shown by using -Actions; this takes more output actions to invoke such as Update-PodeWebCheckbox and Out-PodeWebTextbox, which reference elements within the modal.

On Show-PodeWebModal you can also supply a -DataValue, such as a UserId, Service Name, etc., and this value will be supplied on Submit via $WebEvent.Data['Value'] in New-PodeWebModal's scriptblock. This can be useful if showing a modal from a table using -DataColumn.

There's also Hide-PodeWebModal, which, well, hides a modal!

Confirmation

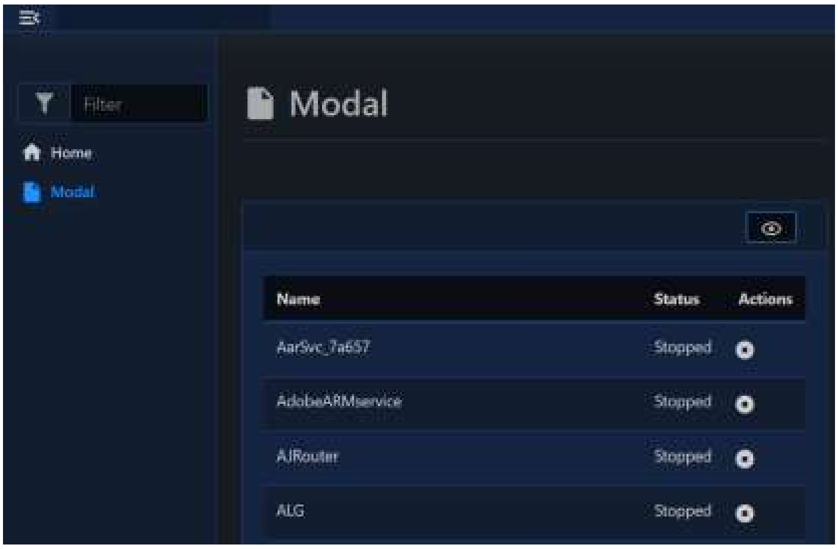
The example below renders a table of services on the current computer. When the "stop" button/icon is clicked, the name of the service on that row (the -DataColumn) is passed to the button, and in-turn that supplies the service name to Show-PodeWebModal. The modal asks if the user wants to stop the service, and on Submit it will stop the service and hide the modal. You need to run this script from the administrator console!

Example118.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Modal' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebTable -Name 'Services' -Click -DataColumn Name -ScriptBlock {
       $stopBtn = New-PodeWebButton -Name 'Stop' -Icon 'Stop-Circle' -IconOnly -ScriptBlock {
          Show-PodeWebModal -Name 'Stop Service' -DataValue $WebEvent.Data.Value -Actions @(
                Update-PodeWebText -Id 'svc_name' -Value $WebEvent.Data.Value
          )
```

```
        }
        foreach ($svc in (Get-Service)) {
            [ordered]@{
                Name = $svc.Name
                Status = "$($svc.Status)"
               #Actions = @($stopBtn)
                Actions =   $stopBtn
            }
        }
    }
    )
    New-PodeWebModal -Name 'Stop Service' -Content @(
    New-PodeWebText -Value 'Are you sure you want to stop '
    New-PodeWebText -Id 'svc_name'
    New-PodeWebText -Value '?'
    ) -ScriptBlock {
    Stop-Service -Name $WebEvent.Data.Value -Force | Out-default
    Show-PodeWebToast -Message "$($WebEvent.Data.Value) stopped"
    Sync-PodeWebTable -Name 'Services'
    Hide-PodeWebModal

} } }
```

Which would look like below:



Name	Status	Actions
AarSvc_7a657	Stopped	●
AdobeARMservice	Stopped	●
AJRouter	Stopped	●
ALG	Stopped	●

Form Input

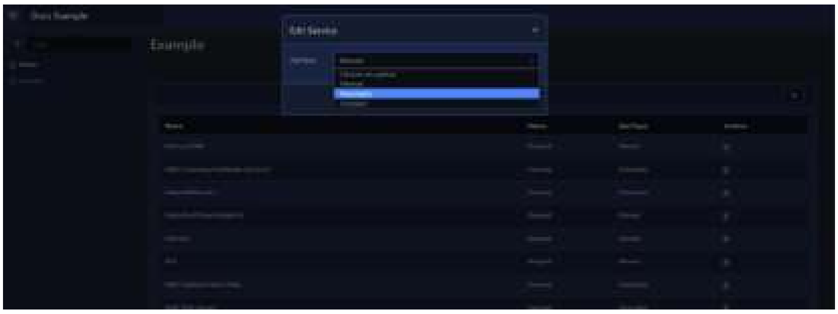
To use a form in your modal, you need to supply -AsForm to New-PodeWebModal.

The example below again renders a table of services on the current computer. But this time there's an edit button to alter the Start-Up type of a service. When the "edit" button/icon is clicked, the name of the service on that row (the -DataColumn) is passed to the button, and in-turn that supplies the service name to Show-PodeWebModal. The modal then shows a form with a select input to change the service's Start-Up type:

Example119.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'TableActions' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebTable -Name 'Services' -DataColumn Name -ScriptBlock {
        $editBtn = New-PodeWebButton -Name 'Edit' -Icon 'Edit' -IconOnly -ScriptBlock {
            $svc = Get-Service -Name $WebEvent.Data.Value
            Show-PodeWebModal -Name 'Edit Service' -DataValue $WebEvent.Data.Value `
                -Actions @(
                           Set-PodeWebSelect -Name 'StartType' -Value $svc.StartType
            )
        }
        foreach ($svc in (Get-Service)) {
            [ordered]@{
                Name = $svc.Name
                Status = "$($svc.Status)"
                StartType = "$($svc.StartType)"
                Actions = @($editBtn)
              # Actions = $editBtn
            }
        }
    }
    )
    New-PodeWebModal -Name 'Edit Service' -AsForm -Content @(
    New-PodeWebSelect -Name 'StartType' -Options Manual, Automatic, Disabled
    ) -ScriptBlock {
  Get-Service -Name $WebEvent.Data.Value | Set-Service -StartType $WebEvent.Data.StartType |
  Out-Null
    Show-PodeWebToast -Message "$($WebEvent.Data.Value) editted"
    Sync-PodeWebTable -Name 'Services'
    Hide-PodeWebModal

} } }
```

Which should look like below. But obviously you need to use a higher version (than v5) of PowerShell.



Method/Action

The default method for forms is Post, and the action is the internal route created for the form.

You can change these values by using the -Method and -Action parameters. The method can only be Get or Post, and the action must be a valid URL.

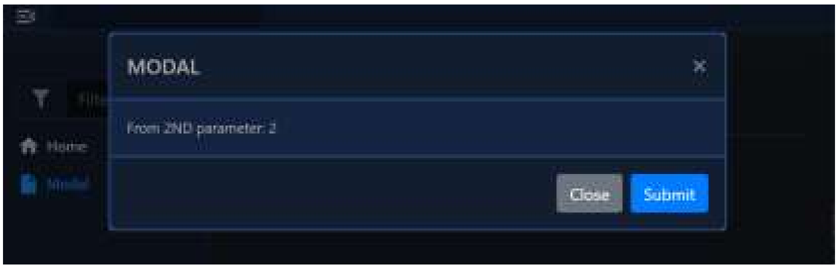
Arguments

You can pass values to the scriptblock by using the -ArgumentList parameter. This accepts an array of values/objects, and they are supplied as parameters to the scriptblock:

Example120.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Modal' -ScriptBlock {

    New-PodeWebModal -Name 'MODAL' -Content @(
        New-PodeWebText -id 'TT' -Value "2ND"
        ) -ArgumentList 'Value1', 2, $false -ScriptBlock {
        param($value1, $value2, $value3)

        # $value1 = 'Value1'
        $b = $value2
        # $value3 = $false


        Update-PodeWebText -Id 'TT' -value "From 2ND parameter: $value2"
        Hide-PodeWebModal
    }

    New-PodeWebModal -Name 'MODAL' -Content @(
        New-PodeWebText -Value "1ST: $value2"
    ) -ScriptBlock { Hide-PodeWebModal }

    New-PodeWebButton -Name 'Modal' -Icon 'Stop-Circle' -ScriptBlock {
        Show-PodeWebModal -Name 'MODAL'
} } }
```

If you press SUBMIT or CLOSE below, the 2nd positional parameter will be processed.



From 2ND parameter: 2

Steps

A steps layout is an array of steps with content. You can use them to step through multiple parts of a setup process. The steps take an array of components via -Content, that can be either other layouts or raw elements.

Usage

To create a steps layout you use New-PodeWebSteps, and supply it an array of -Steps using New-PodeWebStep. The New-PodeWebSteps,also takes a -ScriptBlock, this is the final scriptblock that is invoked after every other step's optional -ScriptBlock; the one where any main logic should be performed.

Note

If you have multiple steps layouts on one page, make sure the Name/IDs are unique, including the Name/IDs of all form input elements as well.

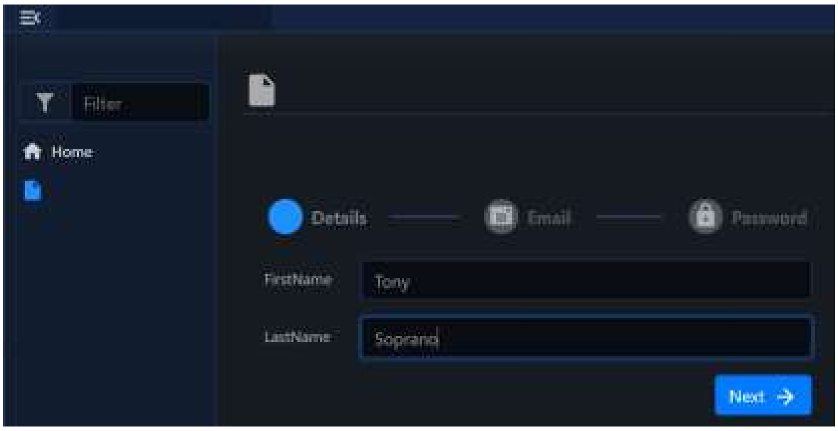
Each step you create via New-PodeWebStep has a -Name, -Content, an optional -ScriptBlock. This scriptblock lets you run validation, or other logic, on a per step basis. If any Out-PodeWebValidation is used, then the step will be prevented from moving forwards.

For example, the below renders a layout with 3 steps to setup a very basic user. The email/password perform validation in their steps, with the user being created in the main final scriptblock:

Example121.ps1

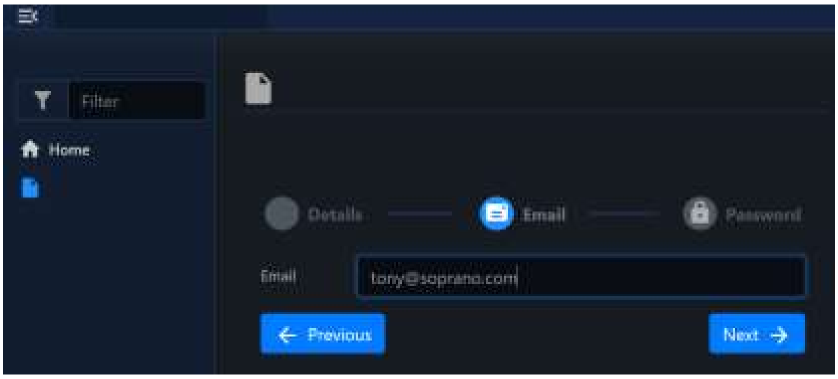
```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebSteps -Name 'AddUser' -Steps @(
    New-PodeWebStep -Name 'Details' -Icon 'User-Plus' -Content @(
        New-PodeWebTextbox -Name 'FirstName'
        New-PodeWebTextbox -Name 'LastName'
    )
    New-PodeWebStep -Name 'Email' -Icon 'Mail' -Content @(
        New-PodeWebTextbox -Name 'Email'
    ) -ScriptBlock {
        if ($WebEvent.Data['Email'] -inotlike '*@*') {
            Out-PodeWebValidation -Name 'Email' -Message 'The email supplied is invalid'
        }
    }
    New-PodeWebStep -Name 'Password' -Icon 'Lock' -Content @(
        New-PodeWebTextbox -Name 'Password' -Type Password
    ) -ScriptBlock {
        if ($WebEvent.Data['Password'].Length -lt 8) {
            Out-PodeWebValidation -Name 'Password' -Message `
               'Password should be 8+ characters'
        }
    }
    ) -ScriptBlock {
       Show-PodeWebToast -Message `
       "User created: $($WebEvent.Data['FirstName']) $($WebEvent.Data['LastName'])"
} } }
```

Which would look like below:

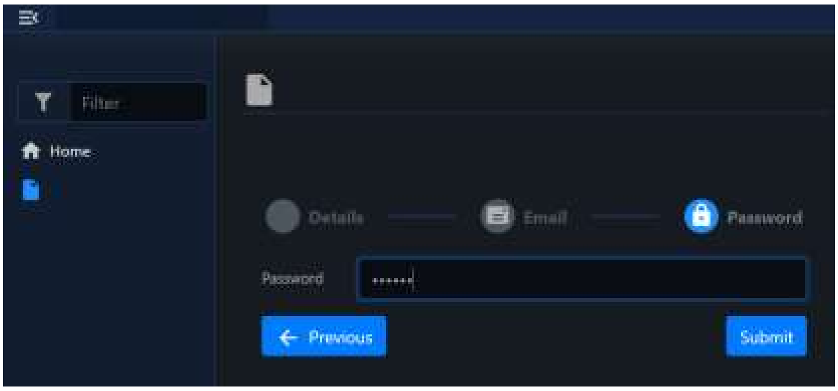


FirstName Tony

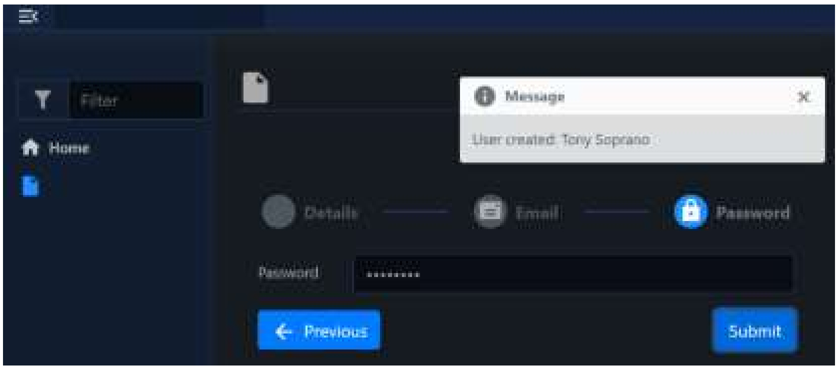
LastName Soprano



Email tony@soprano.com



Password ••••••



Message

User created: Tony Soprano

Password ••••••••

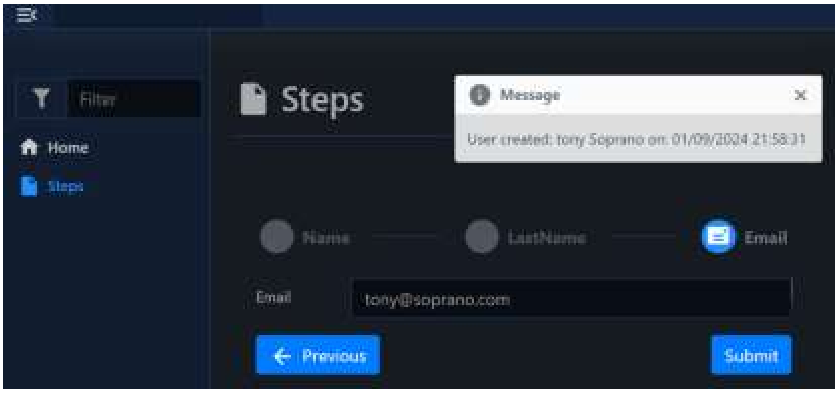
Arguments

You can pass values to the scriptblock by using the -ArgumentList parameter. This accepts an array of values/objects, and they are supplied as parameters to the scriptblock:

Example122.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Steps' -ScriptBlock {

    New-PodeWebSteps -Name 'AddUser' -Steps @(

    New-PodeWebStep -Name 'Name' -Icon 'User-Plus' -Content @(
        New-PodeWebTextbox -Name 'FirstName'
    )
    New-PodeWebStep -Name 'LastName' -Icon 'User-Plus' -Content @(
        New-PodeWebTextbox -Name 'LastName'
    )
    New-PodeWebStep -Name 'Email' -Icon 'Mail' -Content @(
        New-PodeWebTextbox -Name 'Email'
    )
    ) -ArgumentList 'Value1', "$(get-date)", $false `
      -ScriptBlock {
       param($value1, $value2, $value3)
       Show-PodeWebToast -Message `
     "User created: $($WebEvent.Data['FirstName']) $($WebEvent.Data['LastName']) on: $value2"
    }

} }
```

This example will show the Date (2nd parameter) in the Toast message:



User created: tony Soprano on: 01/09/2024 21:58:31

Email: tony@soprano.com

Tabs

A tabs layout is an array of tabs with content.

The tabs take an array of layouts, that can only be other layouts and *not* raw elements.

Usage

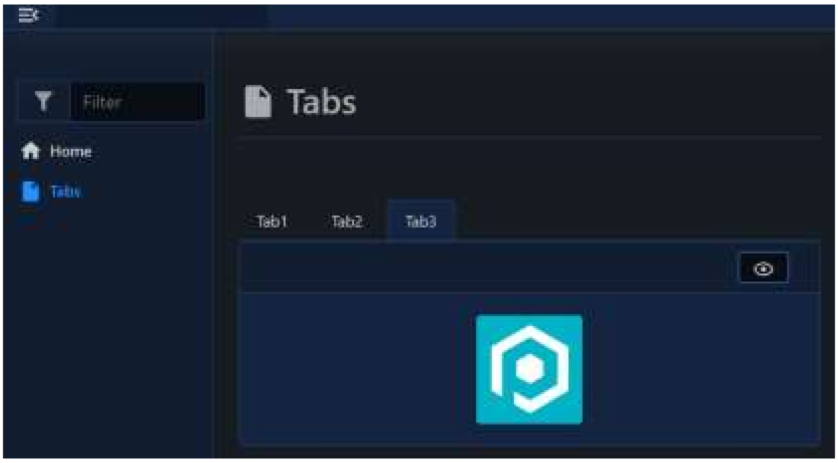
To create a tabs layout you use New-PodeWebTabs, and supply it an array of -Tabs using New-PodeWebTab. The tabs themselves accept an array of other -Layouts.

For example, the below renders a layout with 3 tabs each containing an image:

Example123.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Tabs' -ScriptBlock {

    New-PodeWebTabs -Tabs @(
    New-PodeWebTab -Name Tab1 -Layouts @(
        New-PodeWebCard -Content @(
            New-PodeWebImage -Source '/pode.web/images/icon.png' -Alignment Center
        )
    )
    New-PodeWebTab -Name Tab2 -Layouts @(
        New-PodeWebCard -Content @(
           New-PodeWebImage -Source '/pode.web/images/icon.png' -Height 200 -Alignment Center
        )
    )
    New-PodeWebTab -Name Tab3 -Layouts @(
        New-PodeWebCard -Content @(
           New-PodeWebImage -Source '/pode.web/images/icon.png' -Height 100 -Alignment Center
        )
    )
    )
} }
```

Which would look like below:

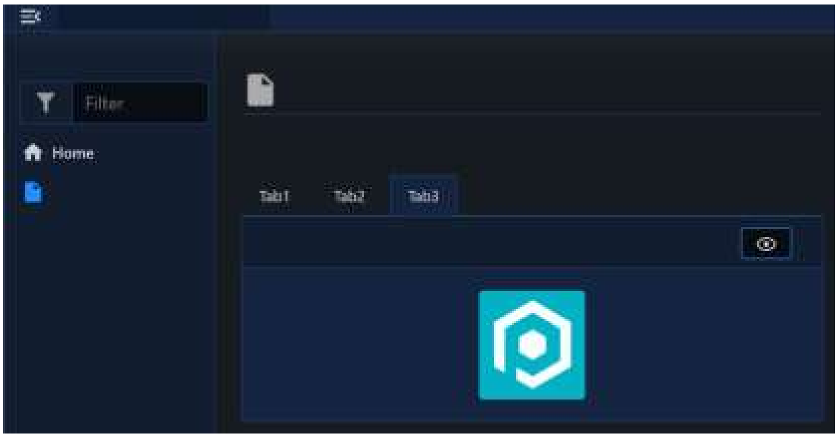


Cycling Tabs

You can render tabs that automatically cycle through themselves every X seconds, by using -Cycle and -CycleInterval. The default interval is every 15secs:

Example124.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebTabs -Cycle -Tabs @(
    New-PodeWebTab -Name Tab1 -Layouts @(
        New-PodeWebCard -Content @(
           New-PodeWebImage -Source '/pode.web/images/icon.png' -height 200 -Alignment Center
        )
    )
    New-PodeWebTab -Name Tab2 -Layouts @(
        New-PodeWebCard -Content @(
           New-PodeWebImage -Source '/pode.web/images/icon.png' -height 150 -Alignment Center
        )
    )
    New-PodeWebTab -Name Tab3 -Layouts @(
        New-PodeWebCard -Content @(
           New-PodeWebImage -Source '/pode.web/images/icon.png' -height 100 -Alignment Center
) ) ) } }
```



Outputs

Accordion

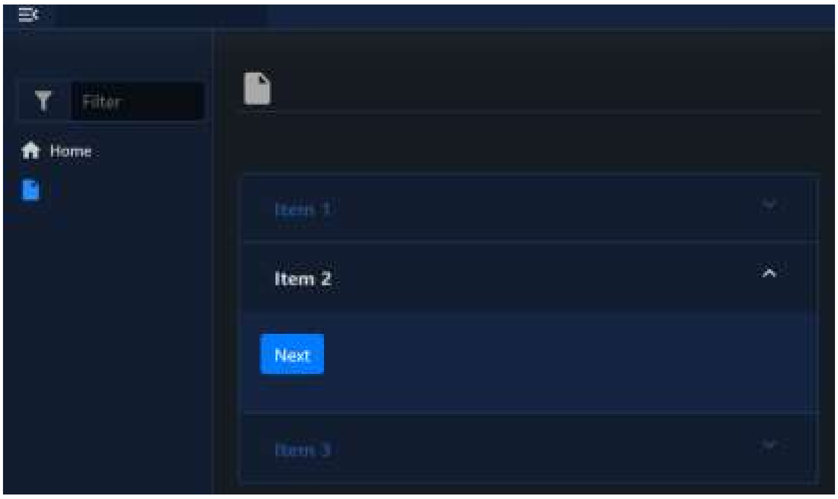
This page details the output actions available to control an Accordion layout.

Move

You can change the current active accordion bellow by using Move-PodeWebAccordion. This will make the specified bellow become the active one, and collapse the others.

Example125.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebAccordion -Bellows @(
    New-PodeWebBellow -Name 'Item 1' -Content @(
        New-PodeWebButton -Name 'Next' -Id 'next_1' -ScriptBlock {
            Move-PodeWebAccordion -Name 'Item 2'
        }
    )
    New-PodeWebBellow -Name 'Item 2' -Content @(
        New-PodeWebButton -Name 'Next' -Id 'next_2' -ScriptBlock {
            Move-PodeWebAccordion -Name 'Item 3'
        }
    )
    New-PodeWebBellow -Name 'Item 3' -Content @(
        New-PodeWebButton -Name 'Next' -Id 'next_3' -ScriptBlock {
            Move-PodeWebAccordion -Name 'Item 1'
} ) ) } }
```



Audio

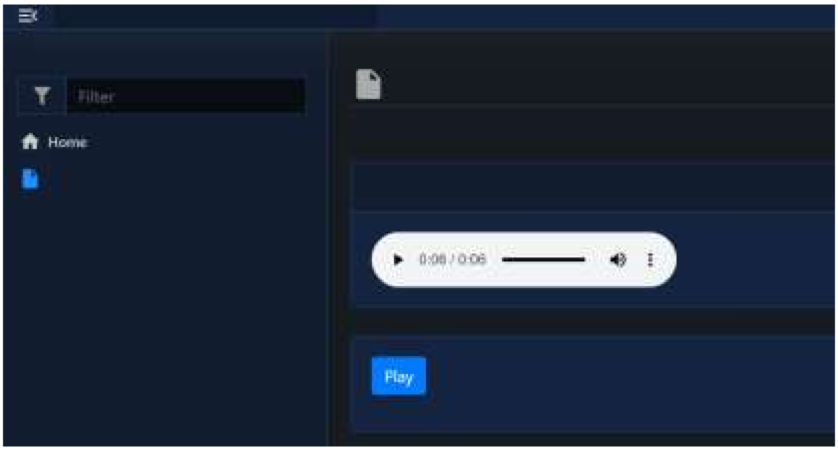
This page details the output actions available to Audio.

Start

To play audio that's currently stopped/paused, you can use Start-PodeWebAudio:

Example126.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebAudio -Name 'example' -Source @(
        New-PodeWebAudioSource -Url 'https://samplelib.com/lib/preview/mp3/sample-6s.mp3'
    ) )

    New-PodeWebContainer -Content @(
    New-PodeWebButton -Name 'Play' -ScriptBlock {
        Start-PodeWebAudio -Name 'example'
} ) } }
```

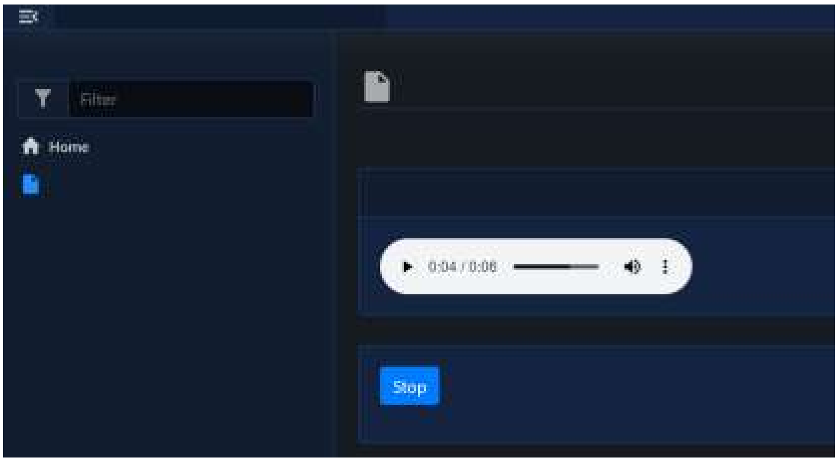


Stop

To pause audio that's currently playing, you can use Stop-PodeWebAudio:

Example127.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebAudio -Name 'example' -Source @(
        New-PodeWebAudioSource -Url 'https://samplelib.com/lib/preview/mp3/sample-6s.mp3'
    ) )

    New-PodeWebContainer -Content @(
    New-PodeWebButton -Name 'Stop' -ScriptBlock {
        Stop-PodeWebAudio -Name 'example'
} ) } }
```

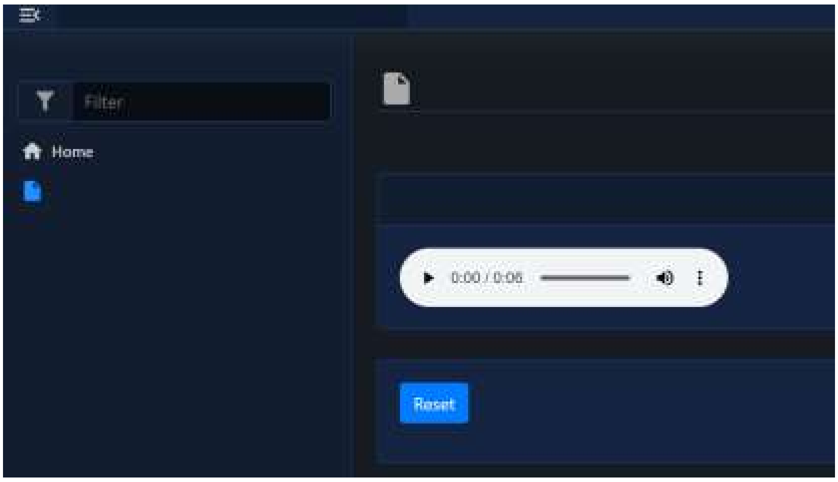


Reset

To reload an audio element, and also reset the audio back to the start, you can use Reset-PodeWebAudio:

Example128.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebAudio -Name 'example' -Source @(
        New-PodeWebAudioSource -Url 'https://samplelib.com/lib/preview/mp3/sample-6s.mp3'
    ) )

    New-PodeWebContainer -Content @(
    New-PodeWebButton -Name 'Reset' -ScriptBlock {
        Reset-PodeWebAudio -Name 'example'
} ) } }
```

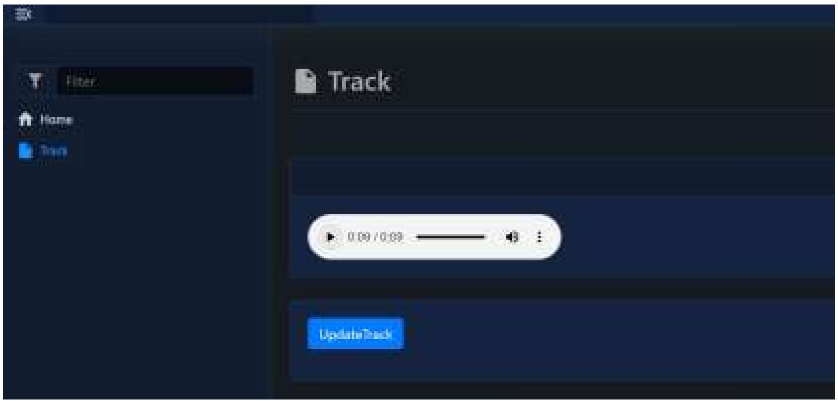


Update

To update the sources/tracks of an audio element, you can use Update-PodeWebAudio. This will clear all current sources/tracks, add the new ones, and then reload the element:

Example129.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Track' -ScriptBlock {

    New-PodeWebCard -Content @(
        New-PodeWebAudio -Name 'example1' -Source @(
        New-PodeWebAudioSource -Url 'https://samplelib.com/lib/preview/mp3/sample-6s.mp3'
        )
    )
    New-PodeWebContainer -Content @(
        New-PodeWebButton -Name 'UpdateTrack' -ScriptBlock {
            Update-PodeWebAudio -Name 'example1' -Source @(
            New-PodeWebAudioSource -Url 'https://samplelib.com/lib/preview/mp3/sample-9s.mp3'
) } ) } }
```



Badge

This page details the output actions available to Badges.

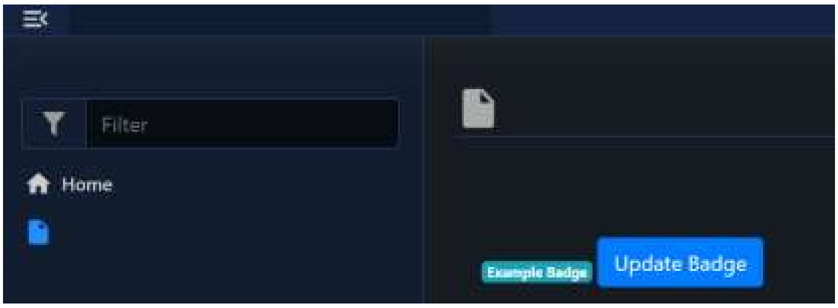
Update

To update the value or the colour of a badge on the page, you can use Update-PodeWebBadge:

Example130.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebBadge -Id 'example_bdg' -Value 'Example Badge' -Colour Green

    New-PodeWebButton -Name 'Update Badge' -ScriptBlock {
        $rand = Get-Random -Minimum 0 -Maximum 3
        $colour = (@('Green', 'Yellow', 'Cyan'))[$rand]
        Update-PodeWebBadge -Id 'example_bdg' -Colour $colour
} ) } }
```

This example will change the badge color on every click.



Breadcrumbs

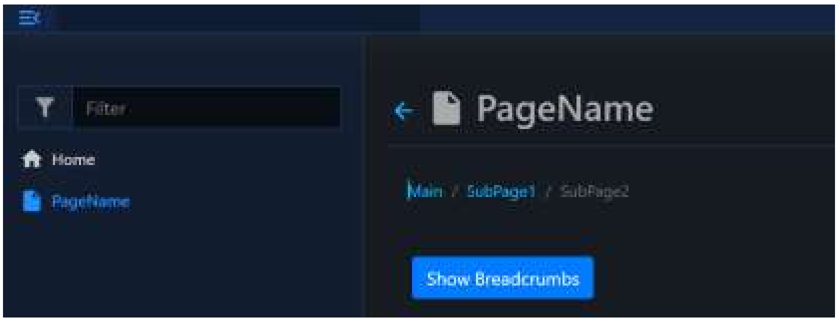
This page details the output actions available to Breadcrumbs.

Out

To create/update the breacrumbs that appear at the top of the page, you can use Out-PodeWebBreadcrumb. This works in a similar fashion to Set-PodeWebBreadcrumb, and also uses New-PodeWebBreadcrumbItem as well:

Example131.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'PageName' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Show Breadcrumbs' -ScriptBlock {
      Out-PodeWebBreadcrumb -Items @(
      New-PodeWebBreadcrumbItem -Name 'Main' -Url '/pages/PageName'
      New-PodeWebBreadcrumbItem -Name 'SubPage1' -Url '/pages/PageName?value=stuff1'
      New-PodeWebBreadcrumbItem -Name 'SubPage2' -Url '/pages/PageName?value=stuff2' -Active
) } ) } }
```



Button

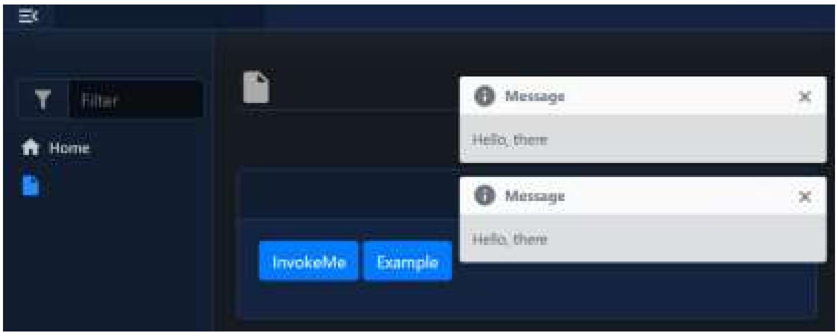
This page details the output actions available to Buttons.

Invoke

To invoke/click a button on the page, you can use Invoke-PodeWebButton:

Example132.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebButton -Name 'InvokeMe' -ScriptBlock {
        Show-PodeWebToast -Message 'Hello, there'
    }
    New-PodeWebButton -Name 'Example' -ScriptBlock {
        Invoke-PodeWebButton -Name 'InvokeMe'
} ) } }
```



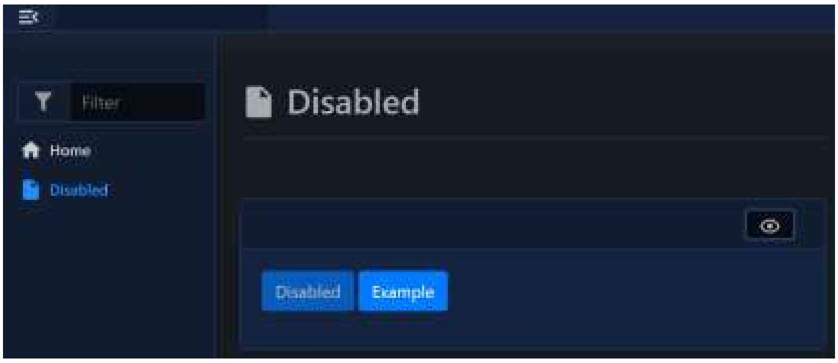
Enable

To enable a disabled button on the page, you can use Enable-PodeWebButton:

Example133.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Disabled' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebButton -Name 'Disabled' -Disabled -ScriptBlock {
        Show-PodeWebToast -Message 'Hello, there'
    }
    New-PodeWebButton -Name 'Example' -ScriptBlock {
        Enable-PodeWebButton -Name 'Disabled'
} ) } }
```

Click the ‘Example’ button below, to enable the ‘Disabled’ button.



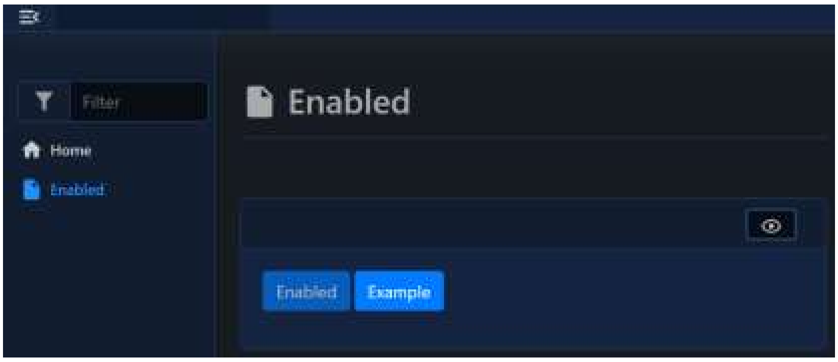
Disable

To disable a enabled button on the page, you can use Disable-PodeWebButton:

Example134.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Enabled' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebButton -Name 'Enabled' -ScriptBlock {
        Show-PodeWebToast -Message 'Hello, there'
    }

    New-PodeWebButton -Name 'Example' -ScriptBlock {
        Disable-PodeWebButton -Name 'Enabled'
} ) } }
```

To disable the enabled button on the page, click an ‘Example’ button on example below.



Update

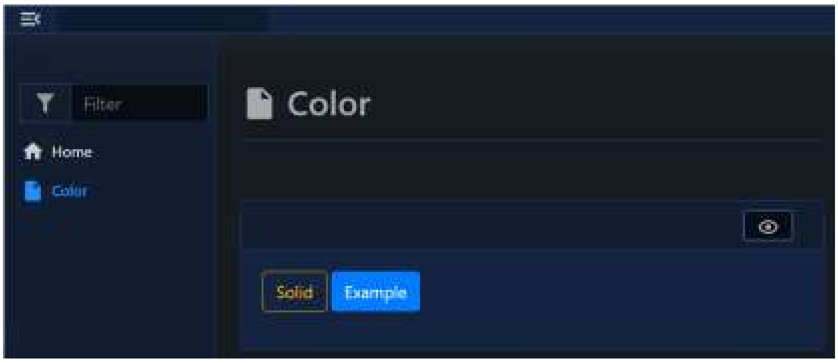
You can update a button's Icon, DisplayName, Colour, and Size using Update-PodeWebButton. For example, just change a solid button to be yellow and outlined:

Example135.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Color' -ScriptBlock {
```

```
    New-PodeWebCard -Content @(
    New-PodeWebButton -Name 'Solid' -Colour Green -ScriptBlock {
        Show-PodeWebToast -Message 'Hello, there'
    }
    New-PodeWebButton -Name 'Example' -ScriptBlock {
        Update-PodeWebButton -Name 'Solid' -Colour Yellow -ColourState Outline
} ) } }
```

To change the button on the page, try this 'Example' button.

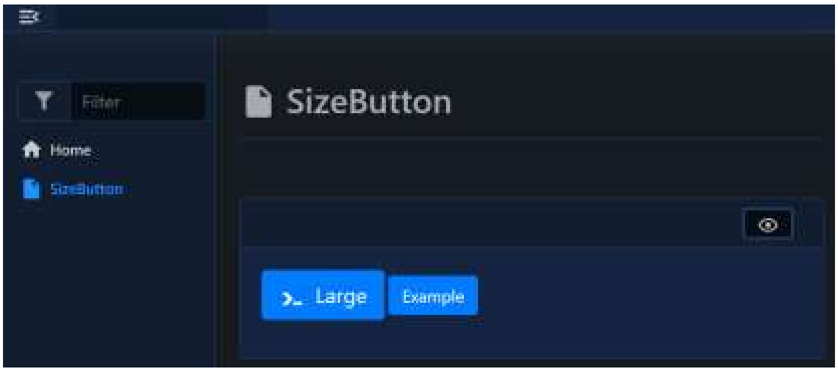


Or to change the Icon and Size of a button:

Example136.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'SizeButton' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebButton -Name 'Large' -Icon 'console-line' -Size Large -ScriptBlock {
        Show-PodeWebToast -Message 'Hello, there'
    }
    New-PodeWebButton -Name 'Example' -ScriptBlock {
        Update-PodeWebButton -Name 'Large' -Size Small -Icon 'safety-goggles'
} ) } }
```

Change the button-size example:



The -ColourState and -SizeState have default values of Unchanged. They map to -Outline and -FullWidth of a button's switches, so they can be toggled in a stateful manor.

Charts

This page details the output actions available to Charts.

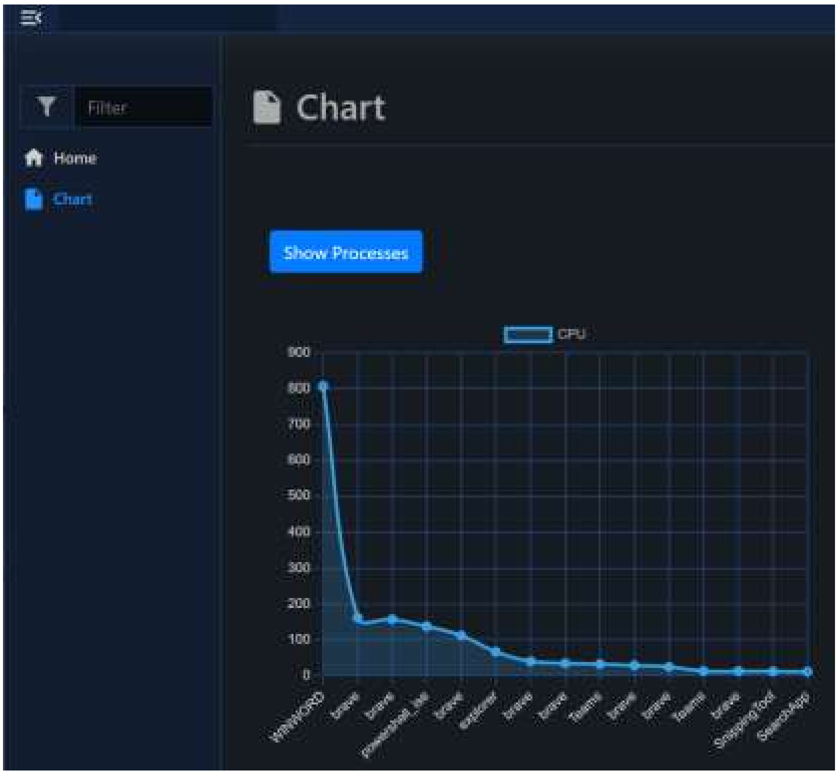
Out

To create a new chart, usually appended beneath the sending element, you can use Out-PodeWebChart. The -Data supplied can either raw or from ConvertTo-PodeWebChartData:

Example137.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Chart' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Show Processes' -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 |
            ConvertTo-PodeWebChartData -LabelProperty ProcessName -DatasetProperty CPU |
            Out-PodeWebChart -Type Line
} ) } }
```

This page details the output actions available to Charts.

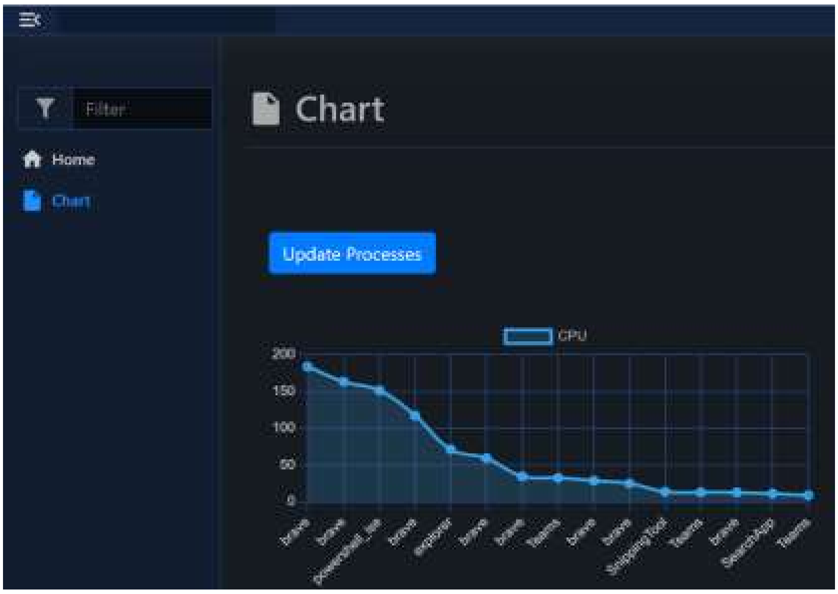


Update

To update the data points of a chart on the page, you can use Update-PodeWebChart. The -Data supplied can either raw or from ConvertTo-PodeWebChartData:

Example138.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Chart' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Update Processes' -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 |
            ConvertTo-PodeWebChartData -LabelProperty ProcessName -DatasetProperty CPU |
            Update-PodeWebChart -Name 'Processes'
    }
    New-PodeWebChart -Name 'Processes' -Type Line -NoRefresh -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 |
            ConvertTo-PodeWebChartData -LabelProperty ProcessName -DatasetProperty CPU
} ) } }
```



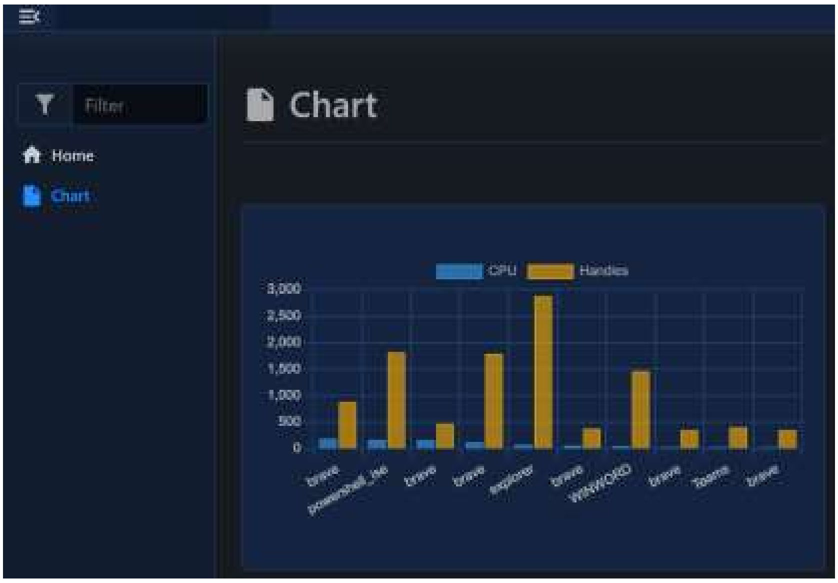
ConvertTo

The ConvertTo-PodeWebChartData simplifies using the raw format, by letting you convert data at the end of a pipeline. The function takes a -LabelProperty which is the name of a property in the input that should be used for the X-axis, and then a -DatasetProperty with is property names for Y-axis values.

For example, let's say we want to display the top 10 processes using the most CPU. We want to display the process name (x-axis), and its CPU and Memory usage (y-axis):

Example139.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Chart' -ScriptBlock {

    New-PodeWebContainer -Content @(
    New-PodeWebChart -Name 'Top Processes' -Type Bar -AutoRefresh -ScriptBlock {
        Get-Process |
          Sort-Object -Property CPU -Descending |
          Select-Object -First 10 |
          ConvertTo-PodeWebChartData -LabelProperty ProcessName -DatasetProperty CPU,Handles
} ) } }
```

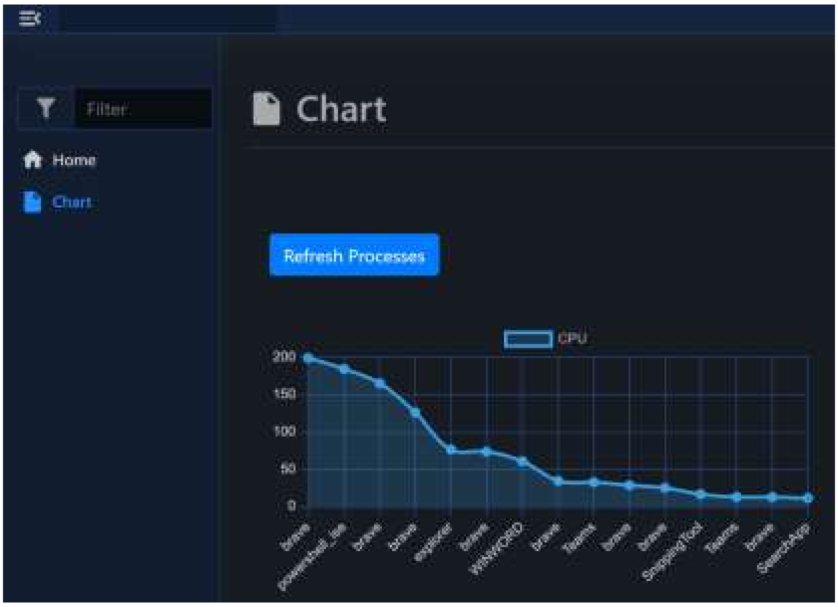


Sync

To force a chart to refresh its data you can use Sync-PodeWebChart:

Example140.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Chart' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Refresh Processes' -ScriptBlock {
        Sync-PodeWebChart -Name 'Processes'
    }

    New-PodeWebChart -Name 'Processes' -Type Line -NoRefresh -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 |
            ConvertTo-PodeWebChartData -LabelProperty ProcessName -DatasetProperty CPU
} ) } }
```

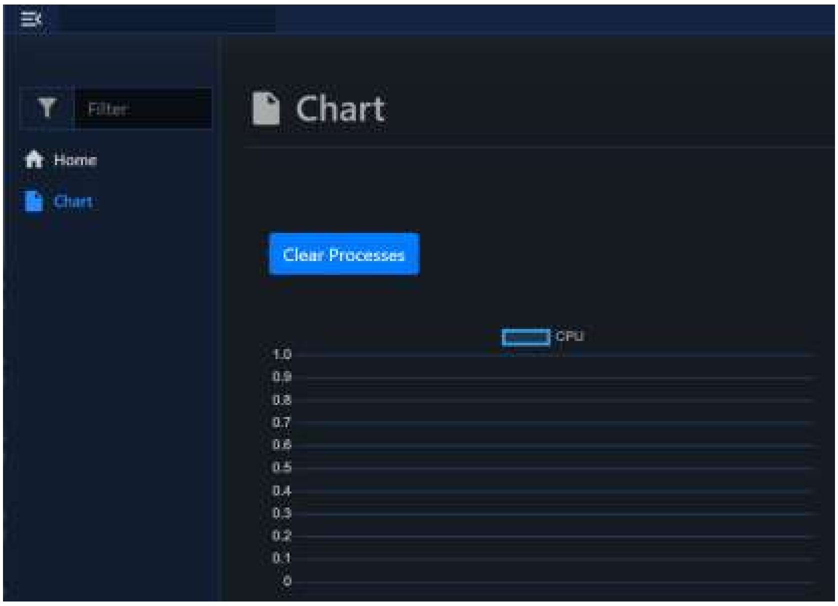


Clear

To clear a chart's data you can use Clear-PodeWebChart:

Example141.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Chart' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Clear Processes' -ScriptBlock {
        Clear-PodeWebChart -Name 'Processes'
    }

    New-PodeWebChart -Name 'Processes' -Type Line -NoRefresh -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 |
            ConvertTo-PodeWebChartData -LabelProperty ProcessName -DatasetProperty CPU
} ) } }
```


Filter
Home
Chart
Chart
Clear Processes
CPU
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

Checkbox

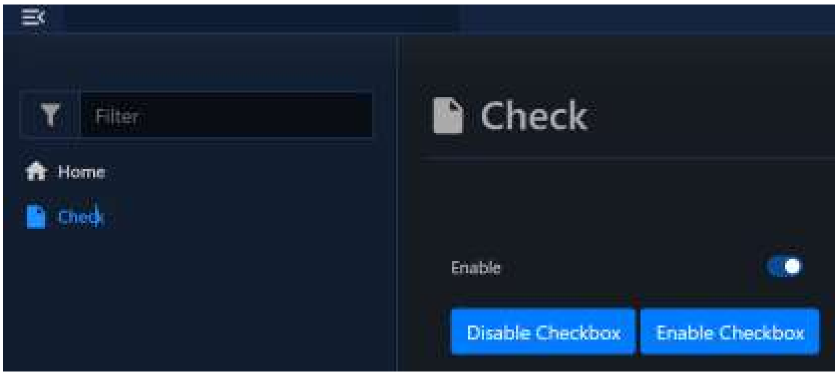
This page details the output actions available to Checkboxes.

Disable

To enable/disable a checkbox you can use Enable-PodeWebCheckbox or Disable-PodeWebCheckbox:

Example142.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Check' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebCheckbox -Name 'Enable' -AsSwitch -Checked

    New-PodeWebButton -Name 'Disable Checkbox' -ScriptBlock {
        Disable-PodeWebCheckbox -Name 'Enable'
    }
    New-PodeWebButton -Name 'Enable Checkbox' -ScriptBlock {
        Enable-PodeWebCheckbox -Name 'Enable'
    }
    )
} }
```

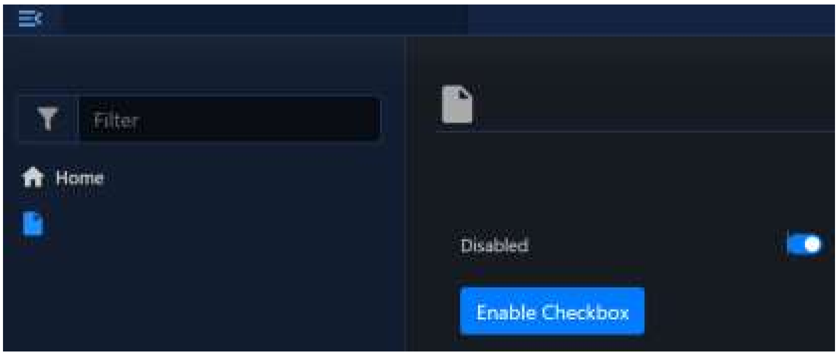


Enable

To enable a checkbox you can use Enable-PodeWebCheckbox:

Example143.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebCheckbox -Name 'Disabled' -AsSwitch -Disabled

    New-PodeWebButton -Name 'Enable Checkbox' -ScriptBlock {
        Enable-PodeWebCheckbox -Name 'Disabled'
    }
    )
} }
```



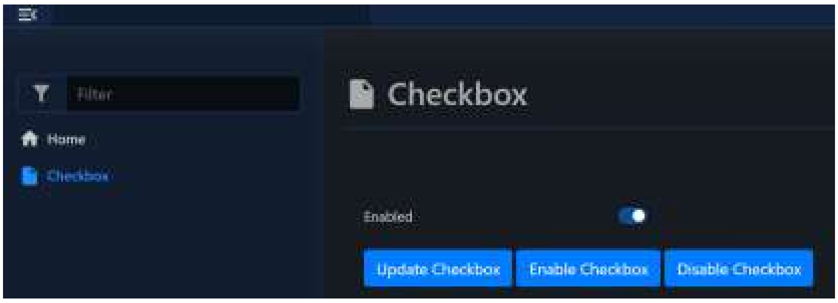
Update

To update a checkbox to be checked/unchecked, or to enable/disable, you can use Update-PodeWebCheckbox:

Example144.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Checkbox' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebCheckbox -Name 'Enabled' -AsSwitch

    New-PodeWebButton -Name 'Update Checkbox' -ScriptBlock {
        $checked = [bool](Get-Random -Minimum 0 -Maximum 2)
        Update-PodeWebCheckbox -Name 'Enabled' -Checked:$checked
    }
    New-PodeWebButton -Name 'Enable Checkbox' -ScriptBlock {
        Update-PodeWebCheckbox -Name 'Enabled' -Checked:$true -State Enabled
    }
    New-PodeWebButton -Name 'Disable Checkbox' -ScriptBlock {
        Update-PodeWebCheckbox -Name 'Enabled' -Checked:$false -State Disabled
    }
    )
} }
```

In the example above you can toggle checked/unchecked using Enable/Disable buttons. But if you perform the same using the Update button, you need to click it several times.



Code Editor

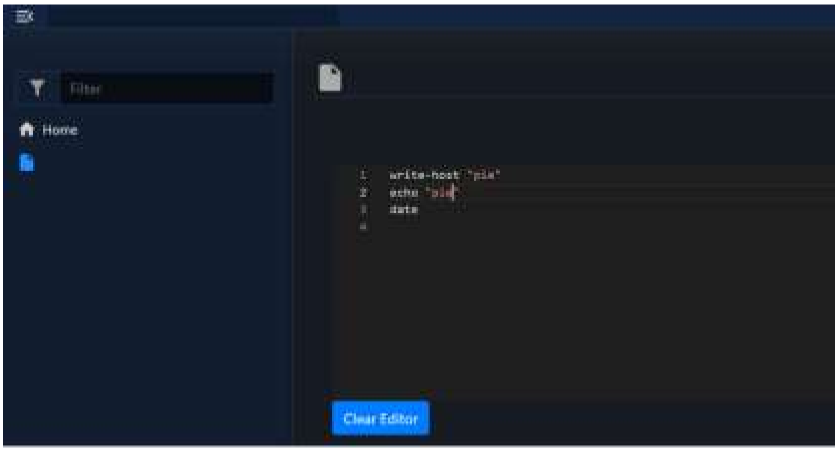
This page details the output actions available to Code Editors.

Clear

To clear the value of a Code Editor, you can use Clear-PodeWebCodeEditor:

Example145.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebCodeEditor -Language PowerShell -Name 'Code Editor' -Value "Write-Host 'hi!'"

    New-PodeWebButton -Name 'Clear Editor' -ScriptBlock {
        Clear-PodeWebCodeEditor -Name 'Code Editor'
    }
    )
} }
```



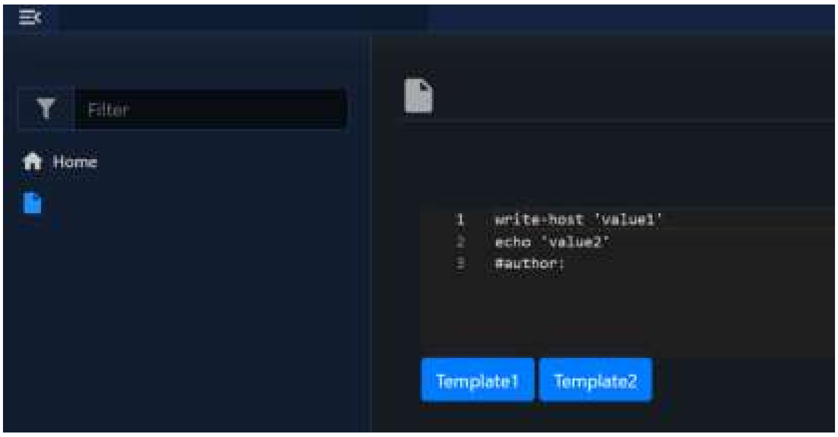
```
1   write-host 'ple'
2   echo 'ple
3   date
4
```

Update

To update the value/language of a Code Editor, you can use Update-PodeWebCodeEditor:

Example146.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebCodeEditor -Language PowerShell -Name 'Code Editor'

    New-PodeWebButton -Name 'Template1' -ScriptBlock {
        Update-PodeWebCodeEditor -Name 'Code Editor' `
                                 -Value '<optional-value>' -Language '<optional-language>'
    }
    New-PodeWebButton -Name 'Template2' -ScriptBlock {
        Update-PodeWebCodeEditor `
           -Name 'Code Editor' -Value "write-host 'value1'`necho 'value2'`n#author: "
} ) } }
```



```
write-host 'value1'
echo 'value2'
#author:
```

Pode.Web Components

This page details the output actions available to all components of Pode.Web.

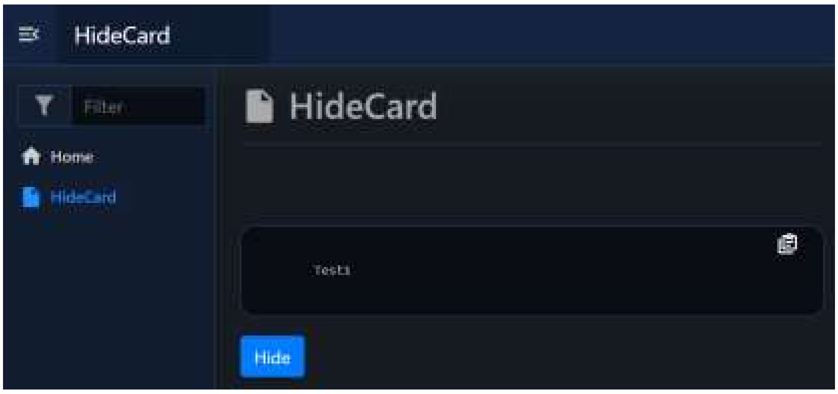
General

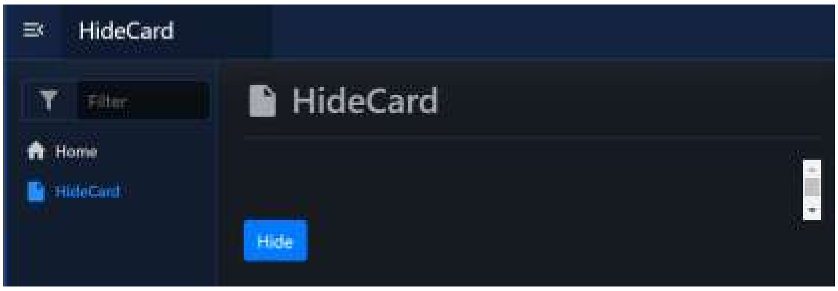
Hide

To hide a component on the frontend, you use Hide-PodeWebComponent. You can hide a component either by -Id, or by the component's -Name and -Type:

Example147.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title 'HideCard' -Theme Dark
    Add-PodeWebPage -name 'HideCard' -ScriptBlock {

    New-PodeWebCodeBlock -Value "Test1" -NoHighlight -Id "Test1"

        New-PodeWebButton -Name 'Hide' -ScriptBlock {
            Hide-PodeWebComponent -Id "Test1"
        }
} }
```



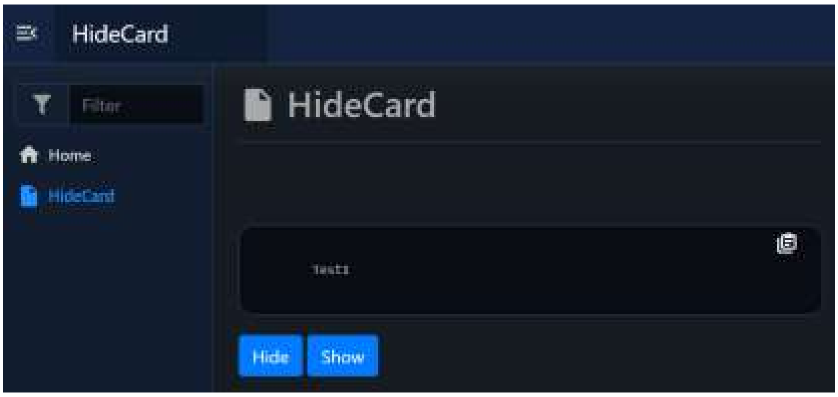


Show

To show a component on the frontend, you use Show-PodeWebComponent. You can show a component either by -Id, or by the component's -Name and -Type:

Example148.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title 'HideCard' -Theme Dark
    Add-PodeWebPage -name 'HideCard' -ScriptBlock {

    New-PodeWebCodeBlock -Value "Test1" -NoHighlight -Id "Test1"

        New-PodeWebButton -Name 'Hide' -ScriptBlock {
            Hide-PodeWebComponent -Id "Test1"
        }
        New-PodeWebButton -Name 'Show' -ScriptBlock {
            Show-PodeWebComponent -Id "Test1"
        }
} }
```



Classes

Add

You can add a class onto a component via Add-PodeWebComponentClass. You can update a component either by -Id, or by the component's -Name and -Type:

Example149.ps1

```
Add-PodeWebComponentClass -Type 'Textbox' -Name 'SomeTextboxName' -Class 'my-custom-class'

# or

Add-PodeWebComponentClass -Id 'textbox_somename' -Class 'my-custom-class'
```

Remove

You can remove a class from a component via Remove-PodeWebComponentClass. You can update a component either by -Id, or by the component's -Name and -Type:

Example150.ps1

```
Remove-PodeWebComponentClass -Type 'Textbox' -Name 'SomeTextboxName' -Class 'my-custom-class'

# or

Remove-PodeWebComponentClass -Id 'textbox_somename' -Class 'my-custom-class'
```

Styles

Set

You can set/update the value of a CSS property on a component via Set-PodeWebComponentStyle. You can update a component either by -Id, or by the component's -Name and -Type:

Example151.ps1

```
Set-PodeWebComponentStyle -Type 'Textbox' -Name 'SomeTextboxName' -Property 'color' -Value 'red'

# or

Set-PodeWebComponentStyle -Id 'textbox_somename' -Property 'color' -Value 'red'
```

Remove

You can remove a CSS property from a component via Remove-PodeWebComponentStyle. You can update a component either by -Id, or by the component's -Name and -Type:

Example152.ps1

```
Remove-PodeWebComponentStyle -Type 'Textbox' -Name 'SomeTextboxName' -Property 'color'

# or

Remove-PodeWebComponentStyle -Id 'textbox_somename' -Property 'color'
```

Desktop Notification

This page details the output actions available to Desktop Notifications.

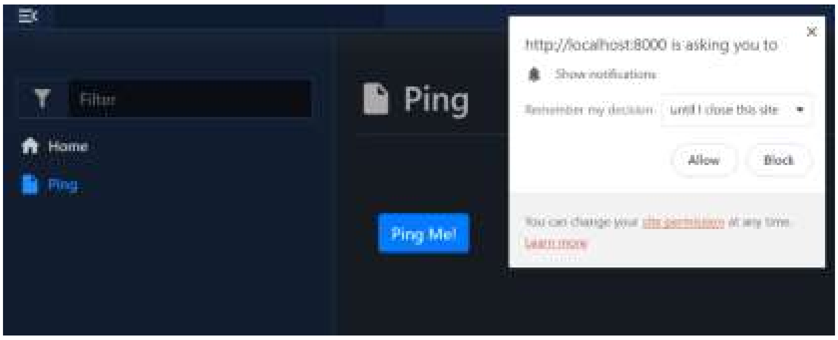
Show

You can show a desktop notification on the user's computer by using Show-PodeWebNotification. When called for the first time for a user, this will ack the user if they're OK for the page to show desktop notifications:

Example153.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8000 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Ping' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Ping Me!' -ScriptBlock {
        Show-PodeWebNotification -Title 'Hi!' -Body 'Hello, there!'
    }
    )

} }
```

If this will not give you a browser notification approval, try to change server port settings in the script and run it again. You should get something like this:



http://localhost:8000 is asking you to

Show notifications

Remember my decision: until I close this site

You can change your site permissions at any time.
Learn more

Error

This page details the output actions available to Errors.

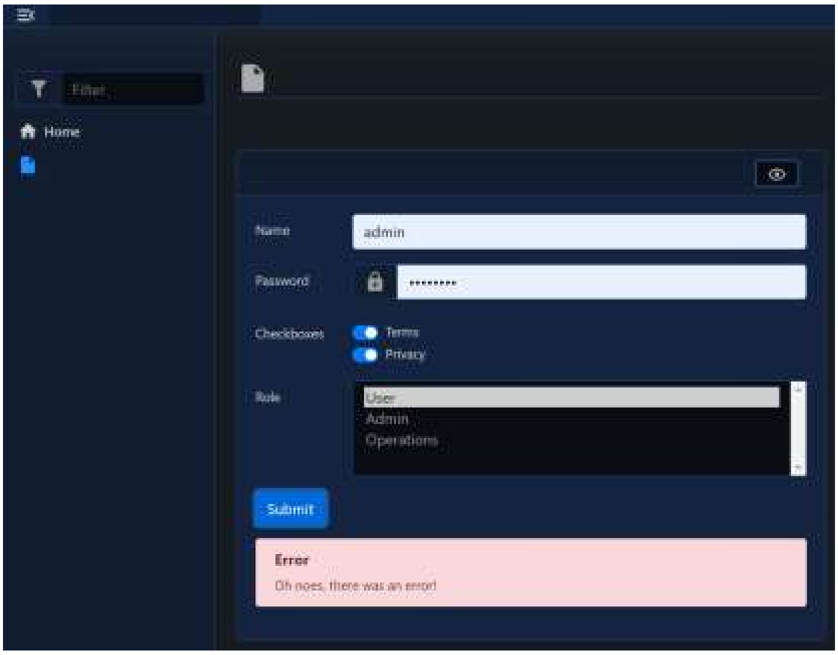
Out

To create/show an error alert, such as when a form throws an unexpected error, you can use Out-PodeWebError. This will render an error alert beneath the element/sender that triggered the action (such as a a form, or a button). These errors will automatically be removed if the form is resubmitted.

Example154.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        Out-PodeWebError -Message 'Oh noes, there was an error!'
    } -Content @(
        New-PodeWebTextbox -Name 'Name'
        New-PodeWebTextbox -Name 'Password' -Type Password -PrependIcon Lock
        New-PodeWebCheckbox -Name 'Checkboxes' -Options @('Terms', 'Privacy') -AsSwitch
        New-PodeWebSelect -Name 'Role' -Options @('User', 'Admin', 'Operations') -Multiple
    )
    )
} }
```

Which would look like below:



Name: admin

Password: ••••••••

Checkboxes: Terms, Privacy

Role: User, Admin, Operations

Error

Oh noes, there was an error!

FileStream

This page details the output actions available to FileStream elements.

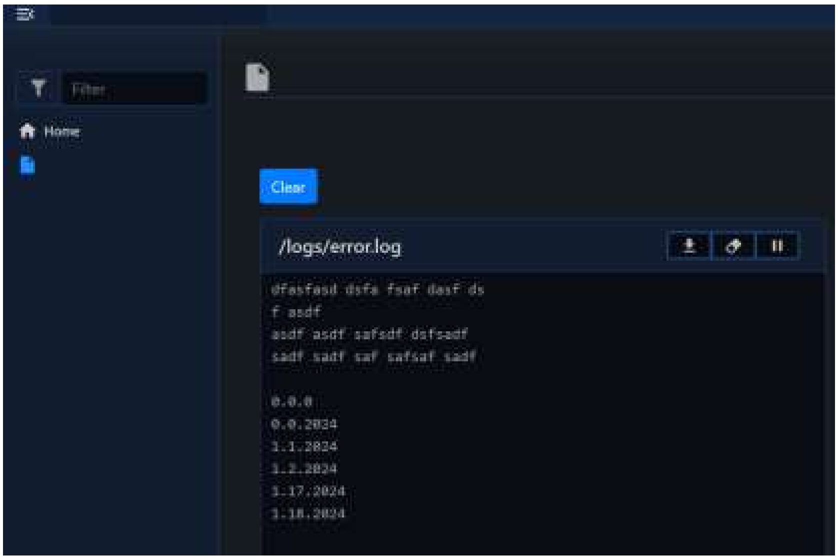
Clear

To clear the content of a FileStream, you can use Clear-PodeWebFileStream:

Example155.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Clear' -ScriptBlock {
        Clear-PodeWebFileStream -Name 'Example'
    }

    New-PodeWebFileStream -Name 'Example' -Url '/logs/error.log'
    )
} }
```

You can temporarily clean output but in 10 seconds contents will re-appear. To trace a filefrom a suggested path, put it into current sub-folder: **.\public\logs\**



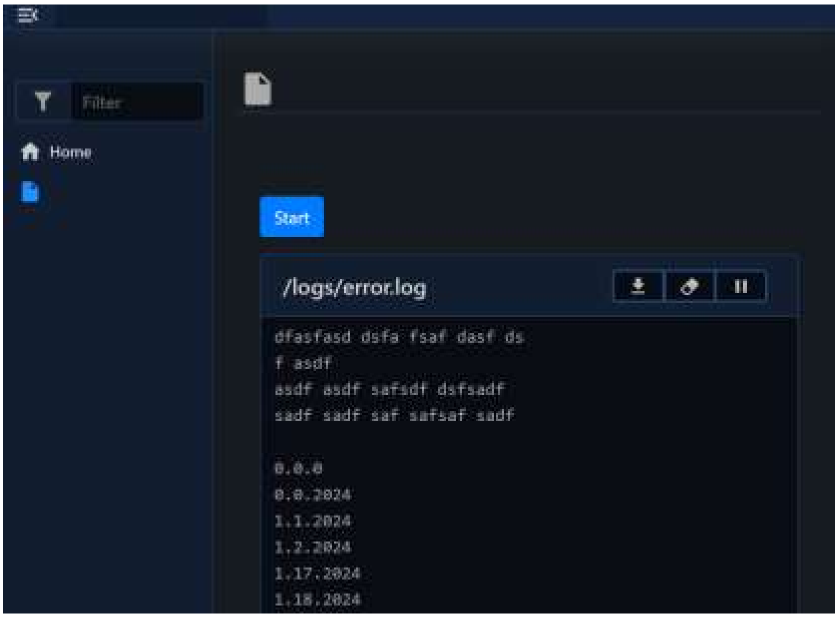
```
dfasfasd dsfa fsaf dasf ds
f asdf
asdf asdf safsdf dsfsadf
sadf sadf saf safsaf sadf

0.0.0
0.0.2024
1.1.2024
1.2.2024
1.17.2024
1.18.2024
```

Start

To start a FileStream that's paused, you can use Start-PodeWebFileSteam:

Example156.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Start' -ScriptBlock {
        Start-PodeWebFileStream -Name 'Example'
    }

    New-PodeWebFileStream -Name 'Example' -Url '/logs/error.log'
    )
} }
```



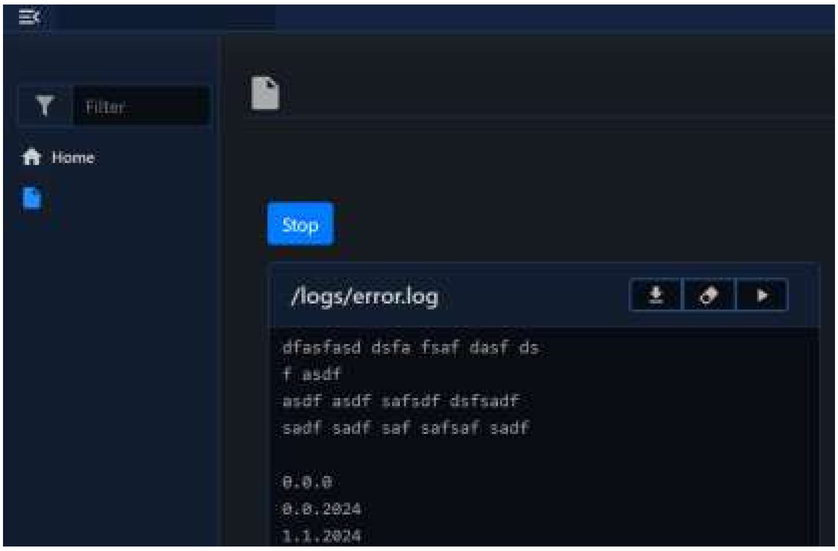
```
dfasfasd dsfa fsaf dasf ds
f asdf
asdf asdf safsdf dsfsadf
sadf sadf saf safsaf sadf

0.0.0
0.0.2024
1.1.2024
1.2.2024
1.17.2024
1.18.2024
```

Stop

To stop/pause a FileStream that's running, you can use Stop-PodeWebFileSteam:

Example157.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Stop' -ScriptBlock {
        Stop-PodeWebFileStream -Name 'Example'
    }

    New-PodeWebFileStream -Name 'Example' -Url '/logs/error.log'
    )
} }
```



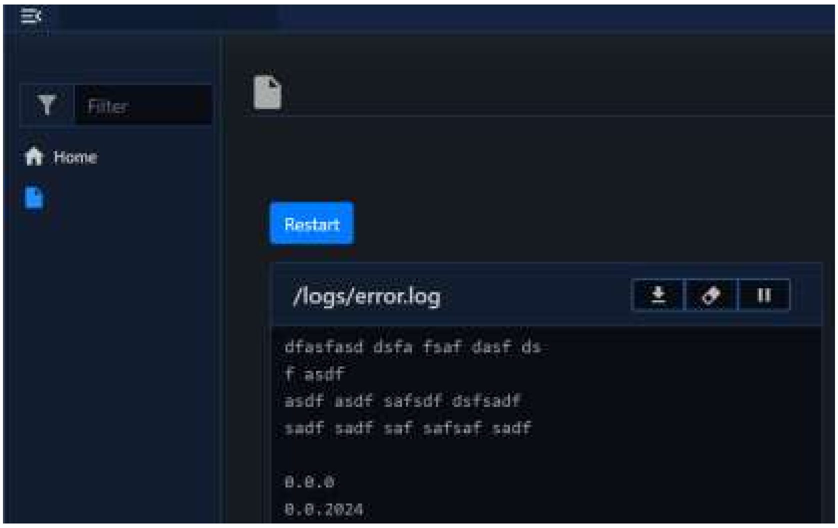
```
dfasfasd dsfa fsaf dasf ds
f asdf
asdf asdf safsdf dsfsadf
sadf sadf saf safsaf sadf

0.0.0
0.0.2024
1.1.2024
```

Restart

To restart a FileStream, you can use Restart-PodeWebFileSteam and this will stop, clear, and the start the FileStream element:

Example158.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Restart' -ScriptBlock {
        Restart-PodeWebFileStream -Name 'Example'
    }
    New-PodeWebFileStream -Name 'Example' -Url '/logs/error.log'
    )
} }
```



```
dfasfasd dsfa fsaf dasf ds
f asdf
asdf asdf safsdf dsfsadf
sadf sadf saf safsaf sadf

0.0.0
0.0.2024
```

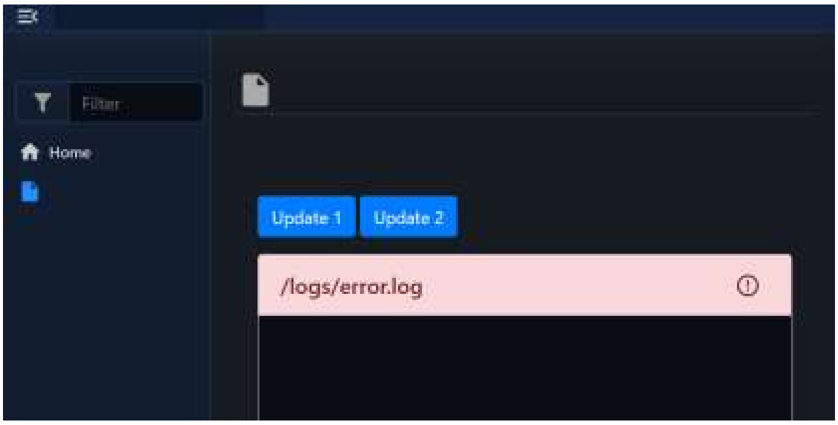
Update

To update the Url that a FileStream is currently streaming data from, you can use Update-PodeWebFileStream. This will stop, clear, update the Url, and the start the FileStream element:

Example159.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Update 1' -ScriptBlock {
        Update-PodeWebFileStream -Name 'Example' -Url '/logs/error.log'
    }
    New-PodeWebButton -Name 'Update 2' -ScriptBlock {
        Update-PodeWebFileStream -Name 'Example' -Url '/logs/error2.log'
    }

    New-PodeWebFileStream -Name 'Example' -Url '/logs/error.log'
    )
} }
```

Try to click Update 2 button. If there is no error2.log file, the border will switch color to notify you, that there is no file to stream it.



Form

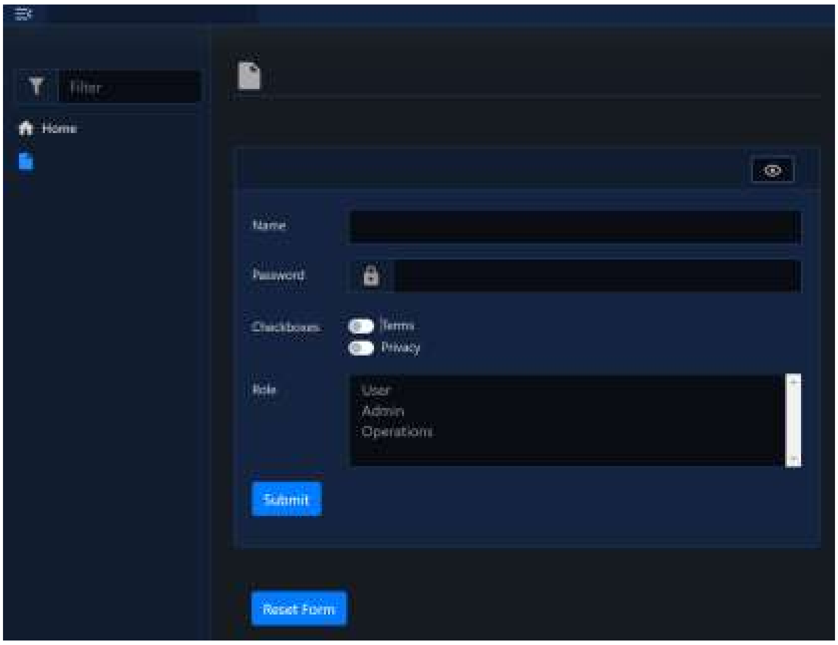
This page details the output actions available to Forms.

Reset

If at any point you need to reset a form, you can use Reset-PodeWebForm which will clear all elements of the specified form:

Example160.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {} -Content @(
        New-PodeWebTextbox -Name 'Name'
        New-PodeWebTextbox -Name 'Password' -Type Password -PrependIcon Lock
        New-PodeWebCheckbox -Name 'Checkboxes' -Options @('Terms', 'Privacy') -AsSwitch
        New-PodeWebSelect -Name 'Role' -Options @('User', 'Admin', 'Operations') -Multiple
    ) )
    New-PodeWebContainer -NoBackground -Content @(
        New-PodeWebButton -Name 'Reset Form' -ScriptBlock {
            Reset-PodeWebForm -Name 'Example'
} ) } }
```

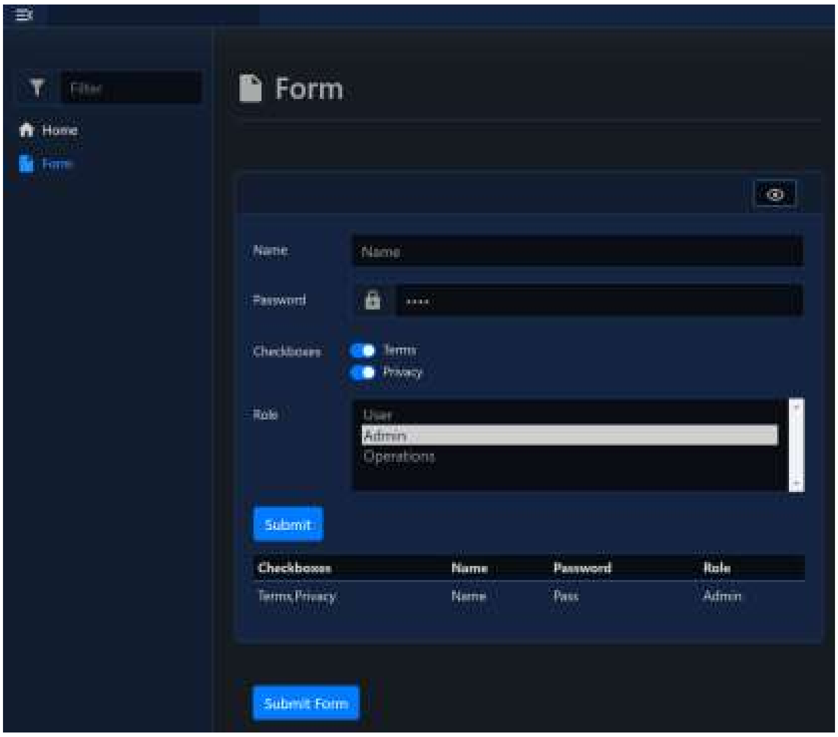


Submit

You can adhoc submit a form by using Submit-PodeWebForm:

Example161.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Add-PodeWebPage -name 'Form' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
                                    $WebEvent.data | Out-PodeWebTable
                                    } -Content @(
        New-PodeWebTextbox -Name 'Name'
        New-PodeWebTextbox -Name 'Password' -Type Password -PrependIcon Lock
        New-PodeWebCheckbox -Name 'Checkboxes' -Options @('Terms', 'Privacy') -AsSwitch
        New-PodeWebSelect -Name 'Role' -Options @('User', 'Admin', 'Operations') -Multiple

    ) )
    New-PodeWebContainer -NoBackground -Content @(
        New-PodeWebButton -Name 'Submit Form' -ScriptBlock {
            Submit-PodeWebForm -Name 'Example'
    }
    )
} }
```



Checkboxes	Name	Password	Role
Terms,Privacy	Name	Pass	Admin

iFrame

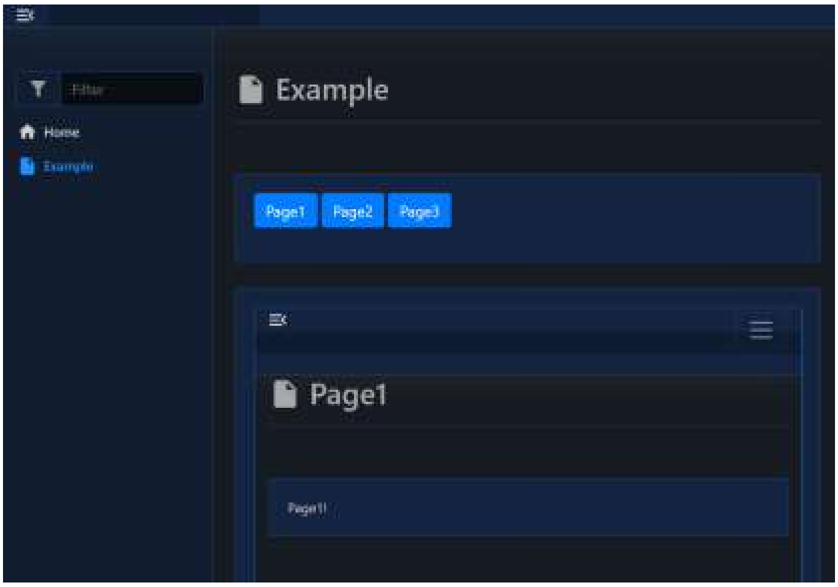
This page details the output actions available to IFrames.

Update

To update an iframe's URL you can use Update-PodeWebIFrame:

Example162.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
#   Add-PodeWebPage -name 'iFrame' -ScriptBlock {
    # set toggle buttons for different pages
    $con1 = New-PodeWebContainer -Content @(
    1..3 |  ForEach-Object {
        New-PodeWebButton -Name "Page$_" -ScriptBlock {
            Update-PodeWebIFrame -Name 'IFrame' -Url "/pages/$($ElementData.Name)"
        }
    }
    )
    # create iframe
    $con2 = New-PodeWebContainer -Content @(
            New-PodeWebIFrame -Name 'IFrame' -Url '/pages/Page1'
    )
    # add page with buttons/iframe
    Add-PodeWebPage -Name 'Example' -Layouts $con1, $con2
    # add 3 hidden pages for the iframe to toggle between
    1..3 |  ForEach-Object {
    Add-PodeWebPage -Name "Page$_" -Hide -Layouts @(
        New-PodeWebContainer -Content @(
            New-PodeWebText -Value "Page$_!"
        )
    )
    }
#}
}
```



Modal

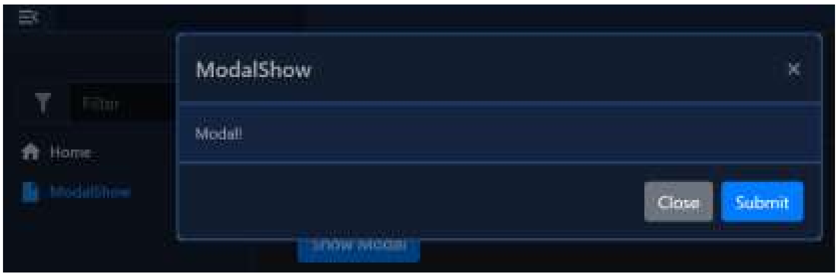
This page details the output actions available to Modals.

Show

You can show a modal by using Show-PodeWebModal, for this to work you must have created a modal via New-PodeWebModal first. When showing a modal, you can supply some further output -Actions to populate elements within the modal:

Example163.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'ModalShow' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Show Modal' -ScriptBlock {
        Show-PodeWebModal -Name 'ModalShow'
    }
    )
    New-PodeWebModal -Name 'ModalShow' -Content @(
        New-PodeWebText -Value "Modal!"
    ) -ScriptBlock {
                    Hide-PodeWebModal
} } }
```

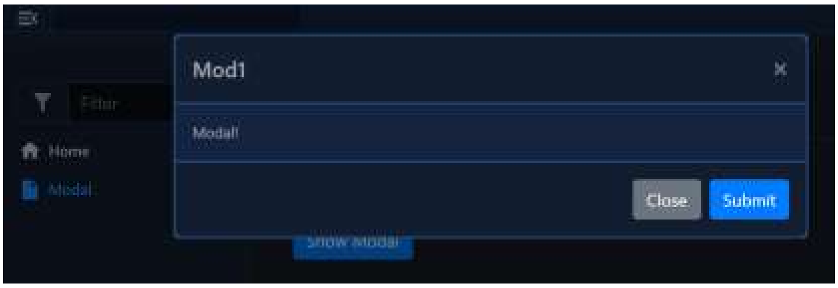


Hide

You can hide a shown modal via Hide-PodeWebModal. If call on its own with no parameters, this will hide an modal that is currently visible:

Example164.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Modal' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Show Modal' -ScriptBlock {
        Show-PodeWebModal -Name 'Mod1'
    } )

    New-PodeWebModal -Name 'Mod1' -Content @(
        New-PodeWebText -Value "Modal!"
    ) -ScriptBlock {
                    Hide-PodeWebModal
} } }
```



Mod1

Modal!

Page

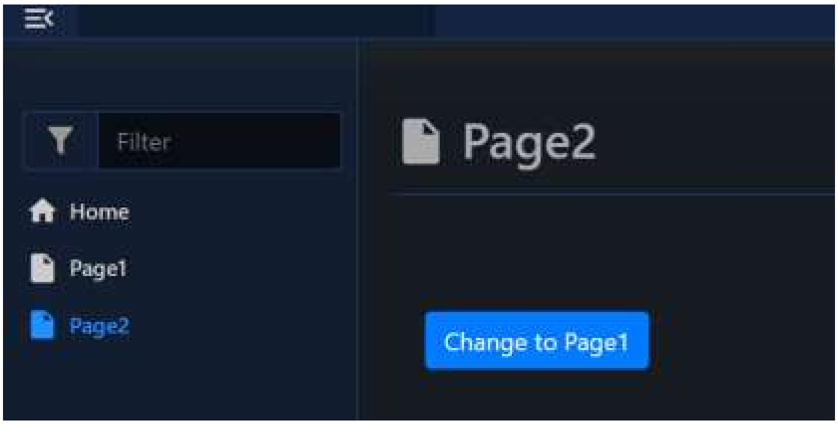
This page details the output actions available to Pages.

Move

You can redirect a user to another Page by using Move-PodeWebPage: Add-PodeWebPage -Name Page1 -Layouts @()

Example165.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark

    Add-PodeWebPage -Name Page1 -Layouts @(
    New-PodeWebContainer -NoBackground -Content @(
        New-PodeWebButton -Name 'Change to Page2' -ScriptBlock {
            Move-PodeWebPage -Name 'Page2'
    } ) )

    Add-PodeWebPage -Name Page2 -Layouts @(
    New-PodeWebContainer -NoBackground -Content @(
        New-PodeWebButton -Name 'Change to Page1' -ScriptBlock {
            Move-PodeWebPage -Name 'Page1'
} ) ) }
```



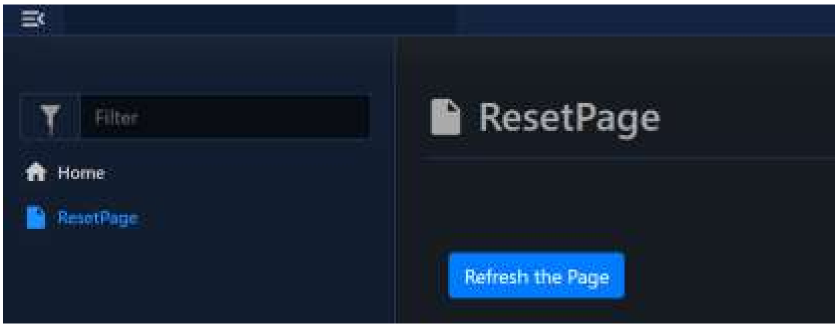
The Page can be opened in a new tab via the -NewTab switch.

Reset

You can refresh the current page by using Reset-PodeWebPage:

Example166.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'ResetPage' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Refresh the Page' -ScriptBlock {
        Reset-PodeWebPage
} ) } }
```


Filter
Home
ResetPage
ResetPage
Refresh the Page

Progress Bar

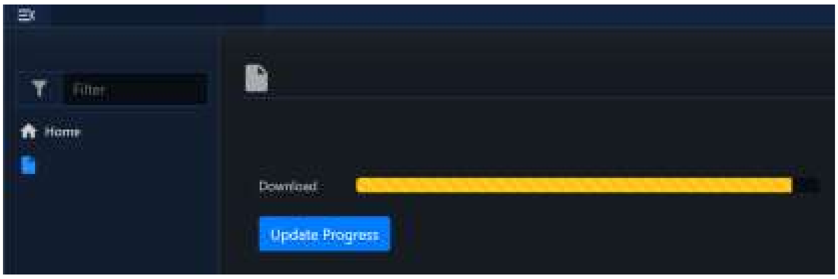
This page details the output actions available to Progress Bars.

Update

To update the value or the colour of a progress bar on the page, you can use Update-PodeWebProgress:

Example167.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebProgress -Name 'Download' -Value 0 -Colour Green -Striped -Animated

    New-PodeWebButton -Name 'Update Progress' -ScriptBlock {
        $rand = Get-Random -Minimum 0 -Maximum 3
        $colour = (@('Green', 'Yellow', 'Cyan'))[$rand]
        $value = Get-Random -Minimum 0 -Maximum 99
        Update-PodeWebProgress -Name 'Download' -Colour $colour -Value $value
} ) } }
```



Select

This page details the output actions available to Select elements.

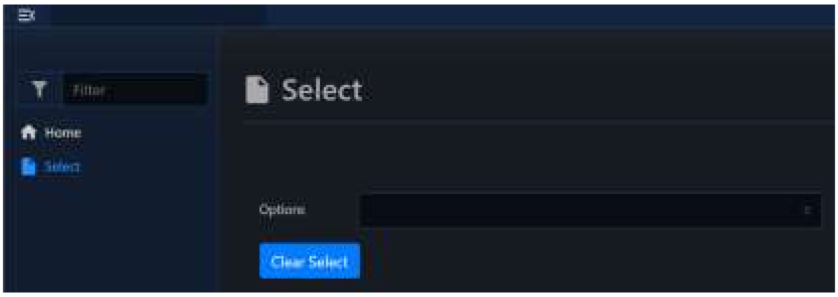
Clear

To clear the options of a Select element, you can use Clear-PodeWebSelect:

Example168.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Select' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebSelect -Name Options -Options Option1, Option2, Option3

    New-PodeWebButton -Name 'Clear Select' -ScriptBlock {
        Clear-PodeWebSelect -Name Options
} ) } }
```

This example will not just remove selection, it will also delete all options possible to choose from.

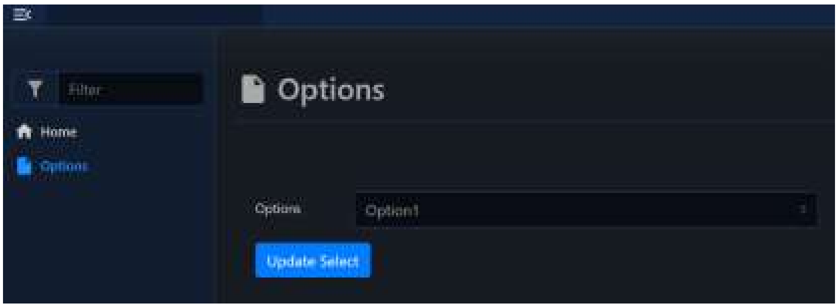


Set

To set the currently selected option/value of a select element, you can use Set-PodeWebSelect:

Example169.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Options' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebSelect -Name Options -Options Option1, Option2, Option3

    New-PodeWebButton -Name 'Update Select' -ScriptBlock {
        $rand = Get-Random -Minimum 0 -Maximum 3
        $opt = (@('Option1', 'Option2', 'Option3'))[$rand]
        Set-PodeWebSelect -Name Options -Value $opt
} ) } }
```



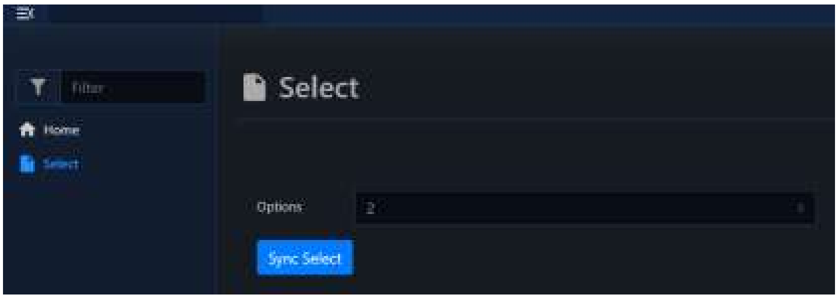
Options: Option1

Sync

If you built a Select element with the -ScriptBlock parameter, then you can re-invoke the scriptblock to update the element by using Sync-PodeWebSelect:

Example170.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Select' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebSelect -Name Options -ScriptBlock {
        return @(foreach ($i in (1..10)) {
            Get-Random -Minimum 1 -Maximum 10
        })
    }

    New-PodeWebButton -Name 'Sync Select' -ScriptBlock {
        Sync-PodeWebSelect -Name Options

} ) } }
```



Options: 2

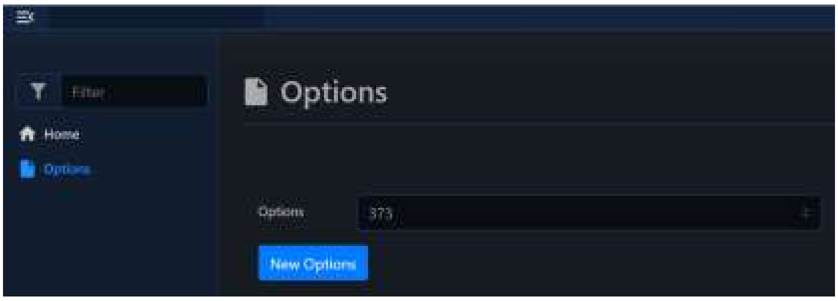
Update

You can update a Select element's options by using Update-PodeWebSelect:

Example171.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Options' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebSelect -Name Options -ScriptBlock {
        return @(foreach ($i in (1..10)) {
            Get-Random -Minimum 1 -Maximum 1000
        })
    }
    New-PodeWebButton -Name 'New Options' -ScriptBlock {
        $options = @(foreach ($i in (1..10)) {
            Get-Random -Minimum 1 -Maximum 1000
        })
```

```
          $options | Update-PodeWebSelect -Name Options
} ) } }
```



You can also optionally supply -DisplayOptions to alter the values displayed in the Select element, as well as also supply as -SelectedValue.

Table

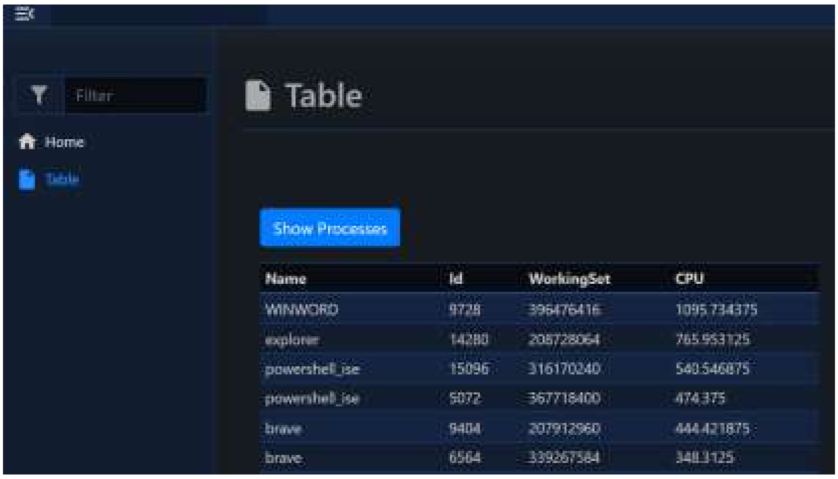
This page details the output actions available to Tables.

Out

To create a new table, usually appended beneath the sending element, you can use Out-PodeWebTable:

Example172.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Table' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Show Processes' -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 -Property Name, ID, WorkingSet, CPU |
            Out-PodeWebTable
} ) } }
```



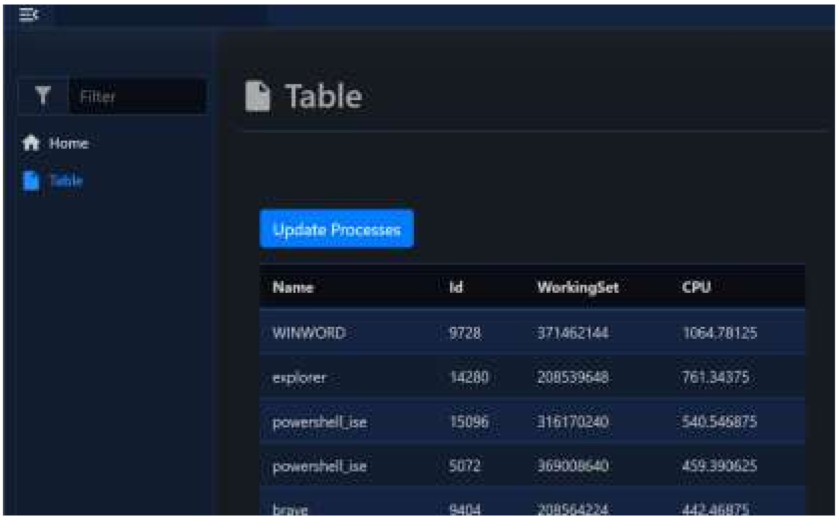
Name	Id	WorkingSet	CPU
WINWORD	9728	396476416	1095.734375
explorer	14280	208728064	765.953125
powershell_ise	15096	316170240	540.546875
powershell_ise	5072	367718400	474.375
brave	9404	207912960	444.421875
brave	6564	339267584	348.3125

Update

To update a table on the page, you can use Update-PodeWebTable:

Example173.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Table' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Update Processes' -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 -Property Name, ID, WorkingSet, CPU |
            Update-PodeWebTable -Name 'Processes'
    }

    New-PodeWebTable -Name 'Processes' -NoRefresh -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 -Property Name, ID, WorkingSet, CPU
} ) } }
```



Table

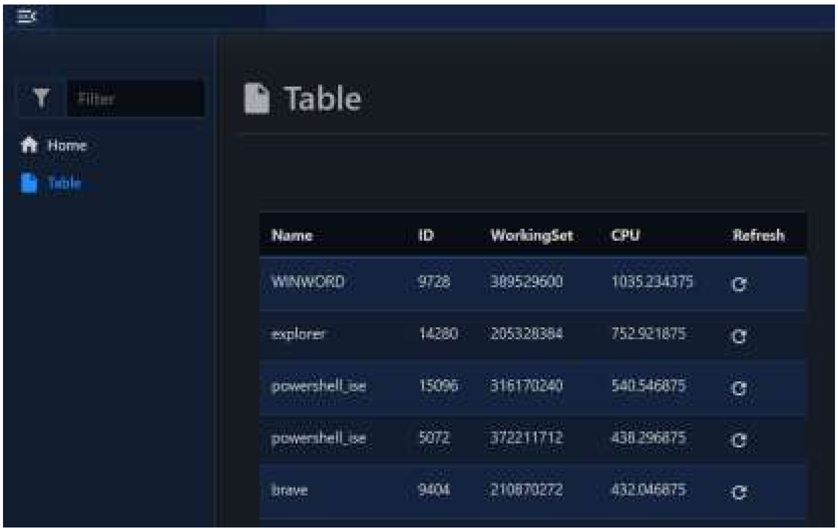
Name	Id	WorkingSet	CPU
WINWORD	9728	371462144	1064.78125
explorer	14280	208539648	761.34375
powershell_ise	15096	316170240	540.546875
powershell_ise	5072	369008640	459.390625
brave	9404	208564224	442.46875

Update Row

To update a single row in the table you can use Update-PodeWebTableRow. You need to supply the table's ID/Name, and then either the index of the row, or the value of that row's -DataColumn. The -Data is a HashTable/PSCustomObject containing the properties/columns that you want to update:

Example174.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Table' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebTable -Name 'Processes' -DataColumn ID -NoRefresh -ScriptBlock {
        $refreshBtn = New-PodeWebButton -Name 'Refresh' -Icon 'refresh' -IconOnly -
ScriptBlock {
            $processId = $WebEvent.Data.Value

            Get-Process -Id $processId |
                Select-Object -Property WorkingSet, CPU |
                Update-PodeWebTableRow -Name 'Processes' -DataValue $processId
        }

        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 |
            ForEach-Object {
                [ordered]@{
                    Name        = $_.Name
                    ID          = $_.ID
                    WorkingSet  = $_.WorkingSet
                    CPU         = $_.CPU
                    Refresh     = @($refreshBtn)
} } } ) } }
```



Table

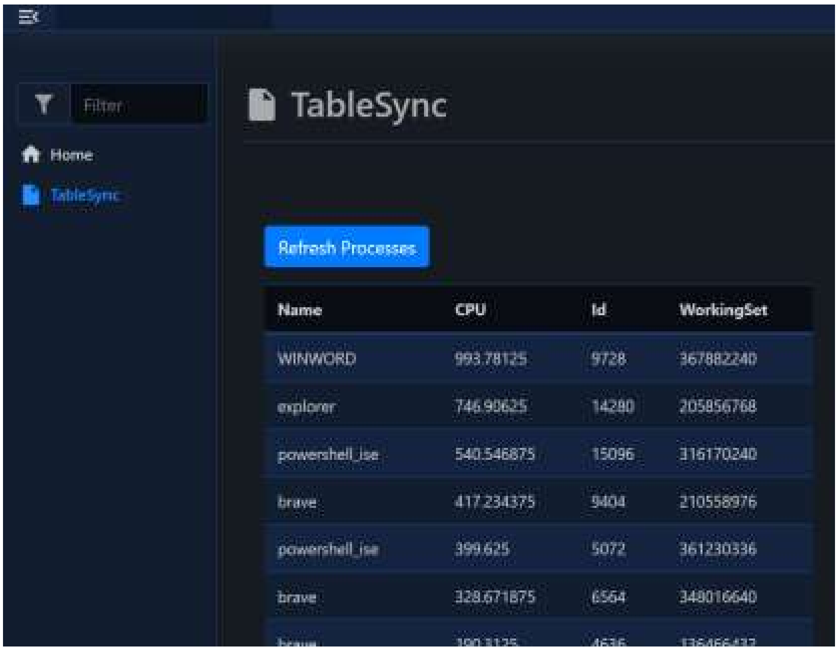
Name	ID	WorkingSet	CPU	Refresh
WINWORD	9728	389529600	1035.234375	↻
explorer	14280	205328384	752.921875	↻
powershell_ise	15096	316170240	540.546875	↻
powershell_ise	5072	372211712	438.296875	↻
brave	9404	210870272	432.046875	↻

Sync

To force a table to refresh its data you can use Sync-PodeWebTable:

Example175.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'TableSync' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Refresh Processes' -ScriptBlock {
        Sync-PodeWebTable -Name 'Processes'
    }

    New-PodeWebTable -Name 'Processes' -NoRefresh -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 -Property Name, CPU, ID, WorkingSet
} ) } }
```



TableSync

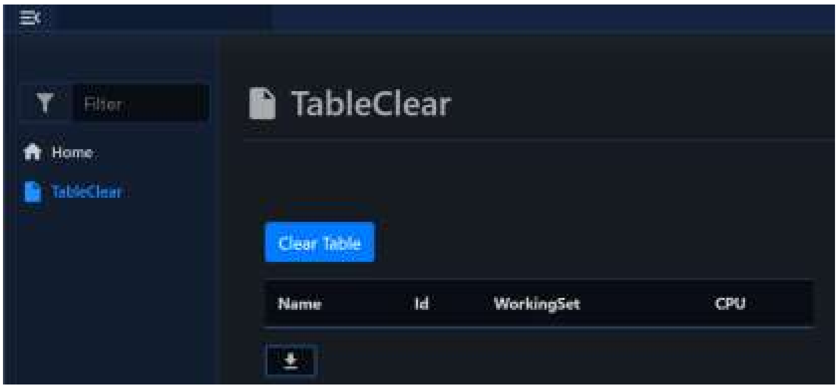
Name	CPU	Id	WorkingSet
WINWORD	993.78125	9728	367882240
explorer	746.90625	14280	205856768
powershell_ise	540.546875	15096	316170240
brave	417.234375	9404	210558976
powershell_ise	399.625	5072	361230336
brave	328.671875	6564	348016640
brave	190.3125	4636	136466432

Clear

To clear a table's data you can use Clear-PodeWebTable:

Example176.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'TableClear' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Clear Table' -ScriptBlock {
        Clear-PodeWebTable -Name 'Processes'
    }

    New-PodeWebTable -Name 'Processes' -NoRefresh -ScriptBlock {
        Get-Process |
            Sort-Object -Property CPU -Descending |
            Select-Object -First 15 -Property Name, ID, WorkingSet, CPU
} ) } }
```

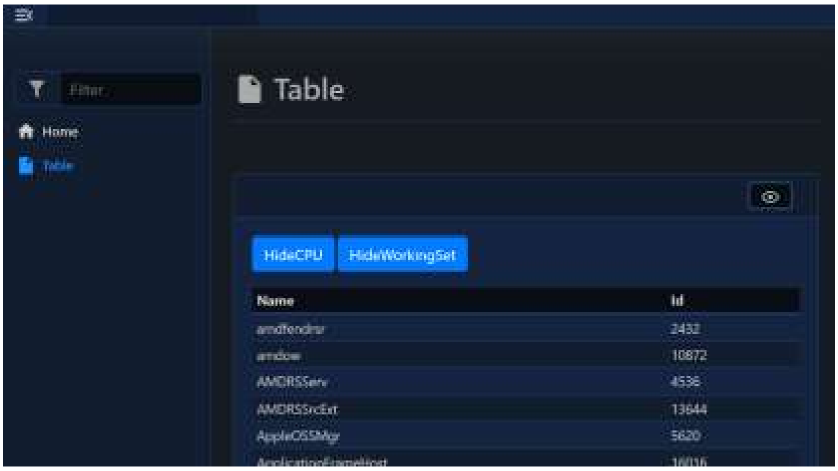


Hide Column

To hide a column within a table, you can use Hide-PodeWebTableColumn. You'll need to supply the table's ID/Name and then the Key of column, specified via Initialize-PodeWebTableColumn (or the Key used in a PSCustomObject or Hashtable used to build the table):

Example177.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Table' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebButton -Name 'HideCPU' -ScriptBlock {
        Hide-PodeWebTableColumn -Name 'Processes' -Key 'CPU'
    }
    New-PodeWebButton -Name 'HideWorkingSet' -ScriptBlock {
        Hide-PodeWebTableColumn -Name 'Processes' -Key 'WorkingSet'
    }

    New-PodeWebTable `
        -Name 'Processes' `
        -Paginate `
        -Compact `
        -ScriptBlock {
            $processes = Get-Process | Select-Object -Property Name, ID, WorkingSet, CPU

            $totalCount = $processes.Length
            $pageIndex = [int]$WebEvent.Data.PageIndex
            $pageSize = [int]$WebEvent.Data.PageSize
            $processes = $processes[(($pageIndex - 1) * `
                          $pageSize) .. (($pageIndex * $pageSize) - 1)]

            $processes | Update-PodeWebTable -Name $ElementData.Name `
                                             -PageIndex $pageIndex `
                                             -TotalItemCount $totalCount
        } `
        -Columns @(
            Initialize-PodeWebTableColumn -Key 'Name'
            Initialize-PodeWebTableColumn -Key 'ID'
            Initialize-PodeWebTableColumn -Key 'WorkingSet' `
                                                           -Name 'Memory' `
                                                           -Alignment Center `
                                                           -Width 10
            Initialize-PodeWebTableColumn -Key 'CPU'
) ) } }
```



Name	Id
amdfendrsr	2432
amdow	10872
AMDRSServ	4536
AMDRSSrcExt	13644
AppleOSSMgr	5620
ApplicationFrameHost	16016

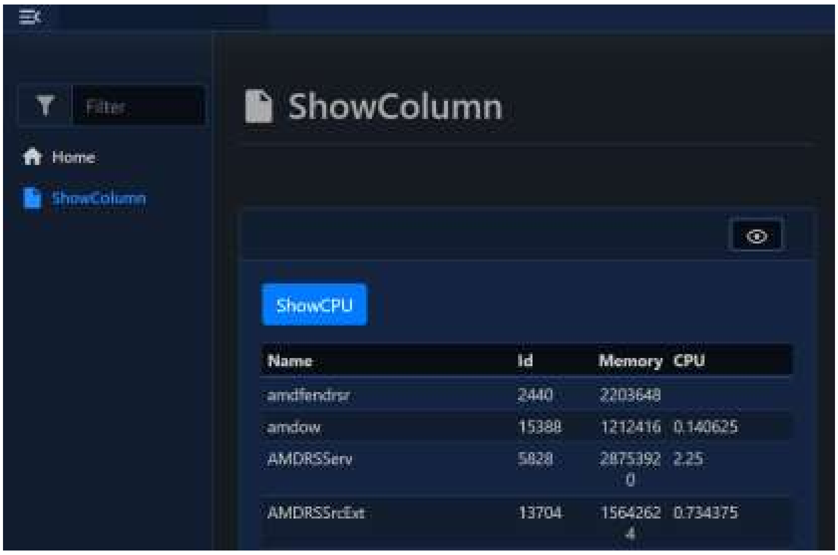
Show Column

To show a column within a table, you can use Show-PodeWebTableColumn. You'll need to supply the table's ID/Name and then the Key of column, specified via Initialize-PodeWebTableColumn (or the Key used in a PSCustomObject or Hashtable used to build the table):

Example178.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'ShowColumn' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebButton -Name 'ShowCPU' -ScriptBlock {
        Show-PodeWebTableColumn -Name 'Processes' -Key 'CPU'
    }

    New-PodeWebTable `
        -Name 'Processes' `
        -Paginate `
        -Compact `
        -ScriptBlock {
            $processes = Get-Process |
                         Select-Object -Property Name, ID, WorkingSet, CPU

            $totalCount = $processes.Length
            $pageIndex = [int]$WebEvent.Data.PageIndex
            $pageSize = [int]$WebEvent.Data.PageSize
            $processes = $processes[(($pageIndex - 1) * `
                         $pageSize) .. (($pageIndex * $pageSize) - 1)]

            $processes | Update-PodeWebTable -Name $ElementData.Name `
                                             -PageIndex $pageIndex `
                                             -TotalItemCount $totalCount
        } `
        -Columns @(
            Initialize-PodeWebTableColumn -Key 'Name'
            Initialize-PodeWebTableColumn -Key 'ID'
            Initialize-PodeWebTableColumn -Key 'WorkingSet' -Name 'Memory' `
                                                 -Alignment Center -Width 10
            Initialize-PodeWebTableColumn -Key 'CPU' -Hide
) ) } }
```

Show hidden column



ShowColumn

Name	Id	Memory	CPU
amdfendrsr	2440	2203648	
amdow	15388	1212416	0.140625
AMDRSServ	5828	28753920	2.25
AMDRSSrcExt	13704	15642624	0.734375

Tabs

This page details the output actions available to control a Tab layout.

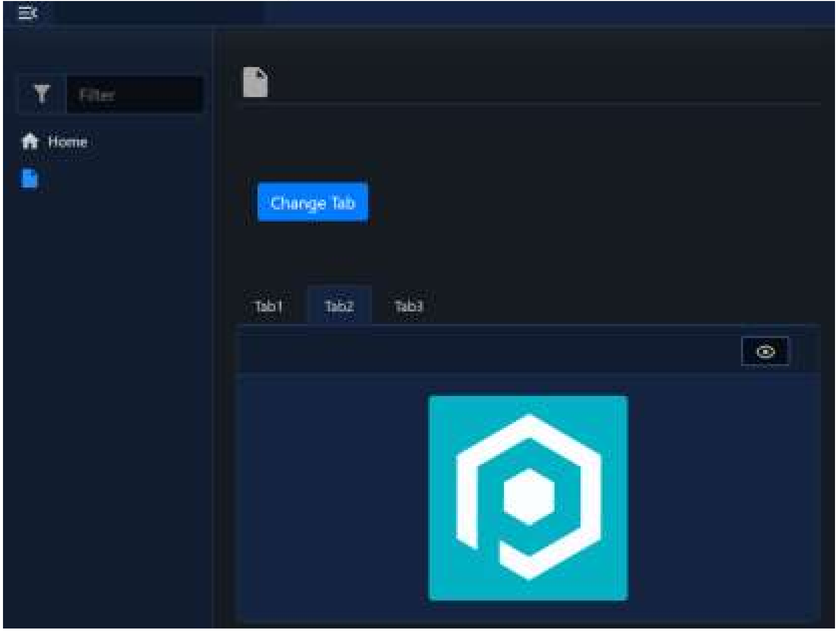
Move

You can change the current active tab by using Move-PodeWebTab. This will make the specified tab become the active one.

Example179.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Change Tab' -ScriptBlock {
        Move-PodeWebTab -Name "Tab$(Get-Random -Minimum 1 -Maximum 4)"
    }
    )
    New-PodeWebTabs -Tabs @(
    New-PodeWebTab -Name Tab1 -Layouts @(
        New-PodeWebCard -Content @(
            New-PodeWebImage -Source '/pode.web/images/icon.png' `
                             -height 200 `
                             -Alignment Center
        )
    )
    New-PodeWebTab -Name Tab2 -Layouts @(
        New-PodeWebCard -Content @(
            New-PodeWebImage -Source '/pode.web/images/icon.png' `
                             -height 200 `
                             -Alignment Center
        )
    )
    New-PodeWebTab -Name Tab3 -Layouts @(
        New-PodeWebCard -Content @(
            New-PodeWebImage -Source '/pode.web/images/icon.png' `
                             -height 200 `
                             -Alignment Center
) ) ) } }
```

Random TABs changer



Text

This page details the output actions available to Text elements, or elements available to have text updated within them.

This output action differs slightly, as it doesn't just update elements created by New-PodeWebText, but instead applies to the following list:

- Alert
- Badge
- Code
- CodeBlock
- Header
- Paragraph
- Quote
- Text

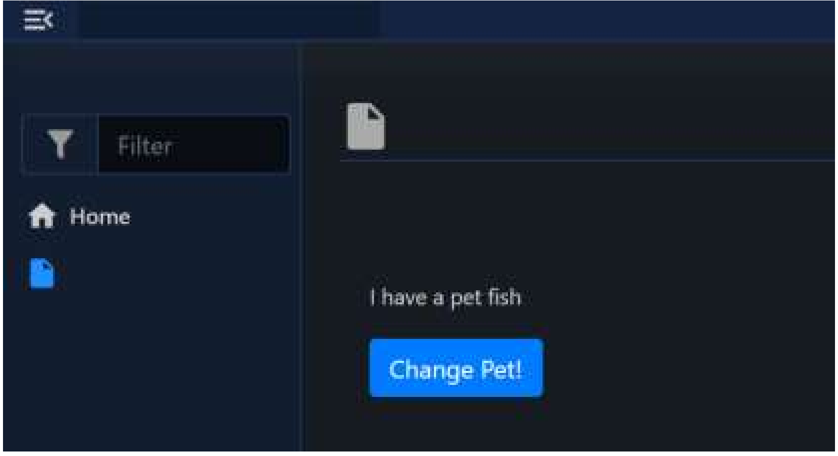
Update

To update the textual value of one of the above elements, you can use Update-PodeWebText:

Example180.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebText -Value 'I have a pet'
    New-PodeWebText -Value 'dog' -Id 'pet_type'

    New-PodeWebParagraph -value ' '

    New-PodeWebButton -Name 'Change Pet!' -ScriptBlock {
        $rand = Get-Random -Minimum 0 -Maximum 5
        $pet = (@('dog', 'cat', 'fish', 'bear'))[$rand]
        Update-PodeWebText -Id 'pet_type' -Value $pet
} ) } }
```

Random pet text changer



– 206 –

Textbox – not working as expected…

This page details the output actions available to Textboxes.

Out

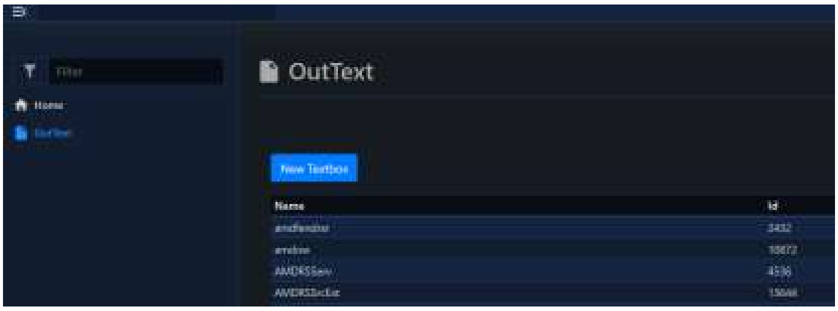
To create a new textbox, usually appended beneath the sending element, you can use Out-PodeWebTextbox. The -Value supplied can be a string/object array, any objects will be converted to a string:

This is not working at all. Instead you can use: Out-PodeWebTable

Example181.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'OutText' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'New Textbox' -ScriptBlock {
        Get-Process |
            Select-Object Name, ID, WorkingSet, CPU |
            Out-PodeWebTable

     #   Get-Process |
     #       Select-Object Name, ID, WorkingSet, CPU |
     #       Out-PodeWebTextBox -Multiline -Preformat -ReadOnly
} ) } }
```

By default it shows the first 10 lines of text, this can be altered using the -Size parameter.



Name	Id
amdfendrsr	3432
amdow	10872
AMDRSServ	4536
AMDRSSrcExt	13044

Update

To update the value of a textbox on the page, you can use Update-PodeWebTextbox. The -Value supplied can be a string/object array, any objects will be converted to a string:

Example182.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'TextBox' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebTextBox -Name 'Content'
    New-PodeWebButton -Name 'Update Textbox' -ScriptBlock {
        Update-PodeWebTexBox -Name 'Content' -Value $([datetime]::Now.ToString())
} ) } }
}
```

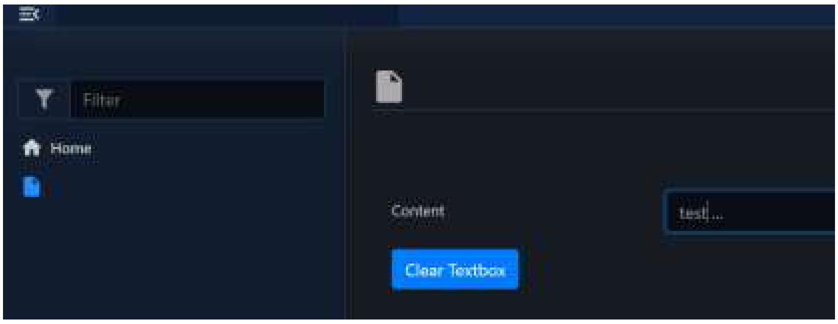
Clear

You can clear the content of a textbox by using Clear-PodeWebTextbox:

Example183.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebTextbox -Name 'Content'

    New-PodeWebButton -Name 'Clear Textbox' -ScriptBlock {
        Clear-PodeWebTextbox -Name 'Content'
} ) } }
```

This one is working fine.



Theme

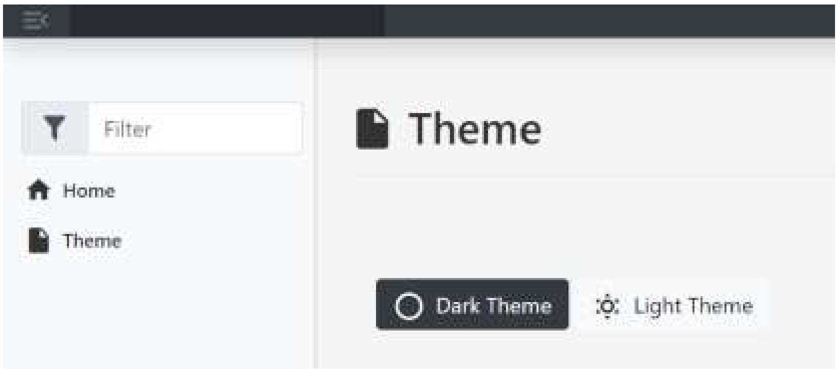
This page details the output actions available to the Theme of pages.

Update

To update the theme for a user you can use Update-PodeWebTheme. This will update the frontend cookie, and then reload the page to toggle the rendering theme:

Example184.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'Theme' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Dark Theme' -Icon 'moon-new' -Colour Dark -ScriptBlock {
        Update-PodeWebTheme -Name Dark
    }
    New-PodeWebButton -Name 'Light Theme' -Icon 'weather-sunny' -Colour Light -ScriptBlock {
        Update-PodeWebTheme -Name Light
} ) } }
```

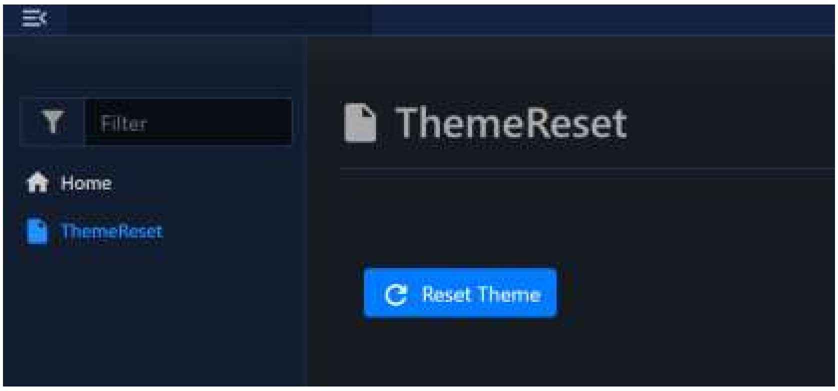


Reset

To reset a user's theme back to the default, you can use Reset-PodeWebTheme:

Example185.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'ThemeReset' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Reset Theme' -Icon 'refresh' -ScriptBlock {
        Reset-PodeWebTheme
} ) } }
```


Filter
Home
ThemeReset
ThemeReset
Reset Theme

Tiles

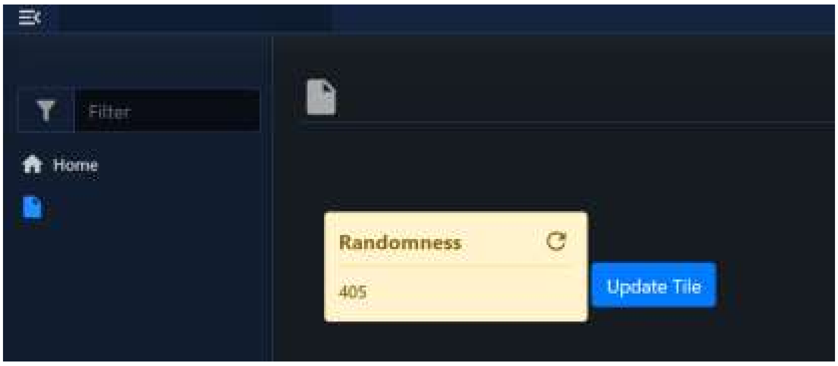
This page details the output actions available to Tiles.

Update

To update the value or the colour of a tile on the page, you can use Update-PodeWebTile:

Example186.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebTile -Name 'Randomness' -ScriptBlock {
        return (Get-Random -Minimum 0 -Maximum 1000)
    }

    New-PodeWebButton -Name 'Update Tile' -ScriptBlock {
        $rand = Get-Random -Minimum 0 -Maximum 3
        $colour = (@('Green', 'Yellow', 'Cyan'))[$rand]
        Update-PodeWebTile -Name 'Randomness' -Colour $colour
} ) } }
```

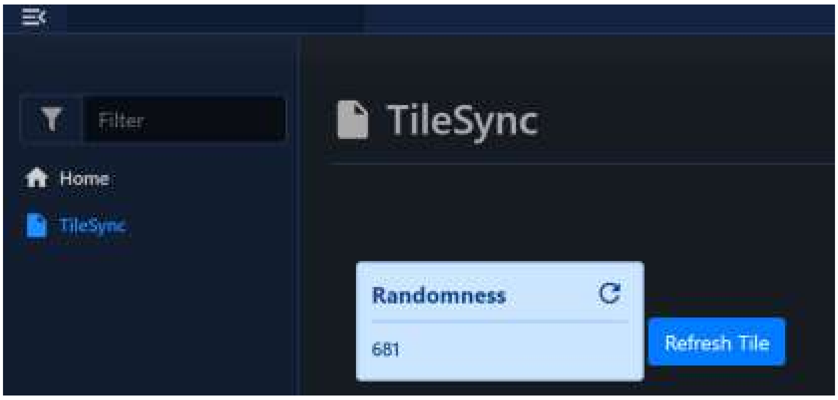


Sync

To force a tile to refresh its data you can use Sync-PodeWebTile:

Example187.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'TileSync' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebTile -Name 'Randomness' -ScriptBlock {
        return (Get-Random -Minimum 0 -Maximum 1000)
    }

    New-PodeWebButton -Name 'Refresh Tile' -ScriptBlock {
        Sync-PodeWebTile -Name 'Randomness'
} ) } }
```


Filter
Home
TileSync
TileSync
Randomness
681
Refresh Tile

Toast

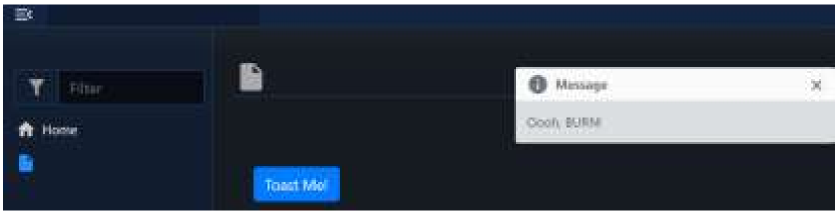
This page details the output actions available to Toast messages.

Show

You can show a quick toast message to a user, displayed in the top-right of the page, by using Show-PodeWebToast. By default a toast is displayed for 3secs, but can be customised via -Duration (in ms):

Example188.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'Toast Me!' -ScriptBlock {
        Show-PodeWebToast -Message 'Oooh, BURN!' -Duration 5000
} ) } }
```



Oooh, BURN!

URL

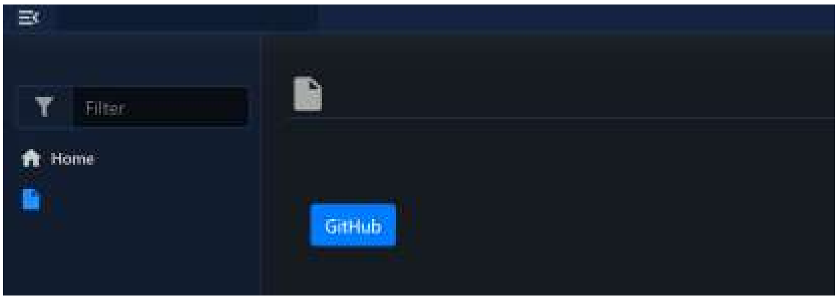
This page details the output actions available to redirect users to other URLs.

Move

You can redirect a user to another URL by using Move-PodeWebUrl:

Example189.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebContainer -NoBackground -Content @(
    New-PodeWebButton -Name 'GitHub' -ScriptBlock {
        Move-PodeWebUrl -Url 'https://github.com'
} ) } }
```



The URL can also be opened in a new tab via the -NewTab switch.

Validation

This page details the output actions available to input form element Validation messages.

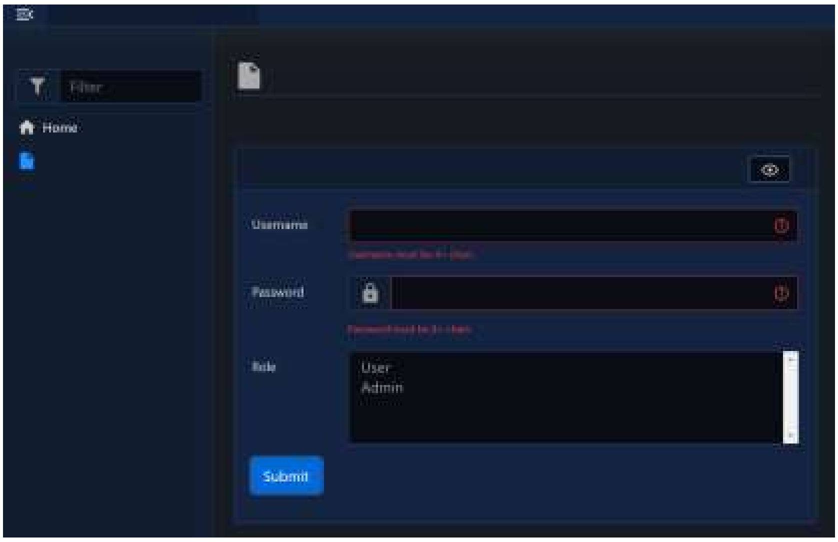
Out

To show an error validation message beneath an element - usually form input elements - you can use Out-PodeWebValidation. This will turn the element red, and show a red message beneath it - useful for username lengths, etc:

Example190.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebForm -Name 'Example' -ScriptBlock {
        $invalid = $false

        if ($WebEvent.Data['Username'].Length -le 3) {
            Out-PodeWebValidation -Name 'Username' -Message 'Username must be 4+ chars'
            $invalid = $true
        }

        if ($WebEvent.Data['Password'].Length -le 7) {
            Out-PodeWebValidation -Name 'Password' -Message 'Password must be 8+ chars'
            $invalid = $true
        }

        if ($invalid) {
            return
        }
    } -Content @(
        New-PodeWebTextbox -Name 'Username'
        New-PodeWebTextbox -Name 'Password' -Type Password -PrependIcon Lock
        New-PodeWebSelect -Name 'Role' -Options @('User', 'Admin') -Multiple
) ) } }
```

Which would look like below:



Username must be 4+ chars

Password must be 8+ chars

Video

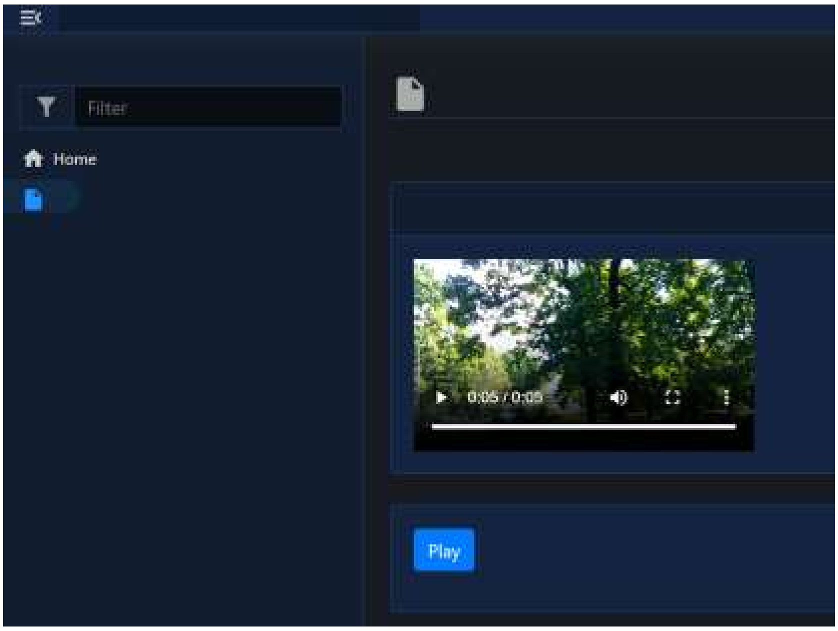
This page details the output actions available to Video.

Start

To play video that's currently stopped/paused, you can use Start-PodeWebVideo:

Example191.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebVideo -Name 'example' `
       -Thumbnail 'https://samplelib.com/lib/preview/mp4/sample-5s.jpg' -Source @(
        New-PodeWebVideoSource -Url 'https://samplelib.com/lib/preview/mp4/sample-5s.mp4'
    ) )
    New-PodeWebContainer -Content @(
    New-PodeWebButton -Name 'Play' -ScriptBlock {
        Start-PodeWebVideo -Name 'example'
} ) } }
```

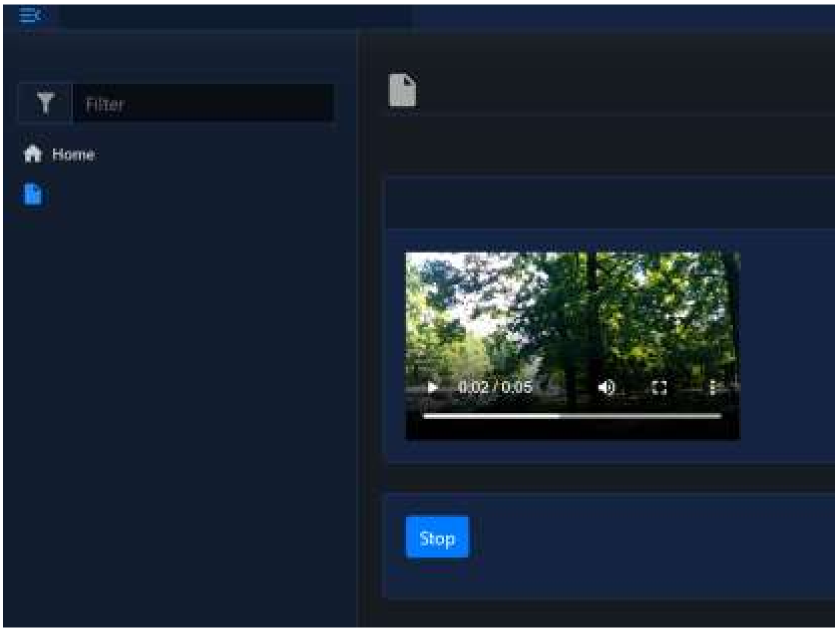


Stop

To pause video that's currently playing, you can use Stop-PodeWebVideo:

Example192.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebVideo -Name 'example' `
       -Thumbnail 'https://samplelib.com/lib/preview/mp4/sample-5s.jpg' `
       -Source @(
        New-PodeWebVideoSource `
           -Url 'https://samplelib.com/lib/preview/mp4/sample-5s.mp4'
    ) )

    New-PodeWebContainer -Content @(
        New-PodeWebButton -Name 'Stop' -ScriptBlock {
            Stop-PodeWebVideo -Name 'example'
} ) } }
```

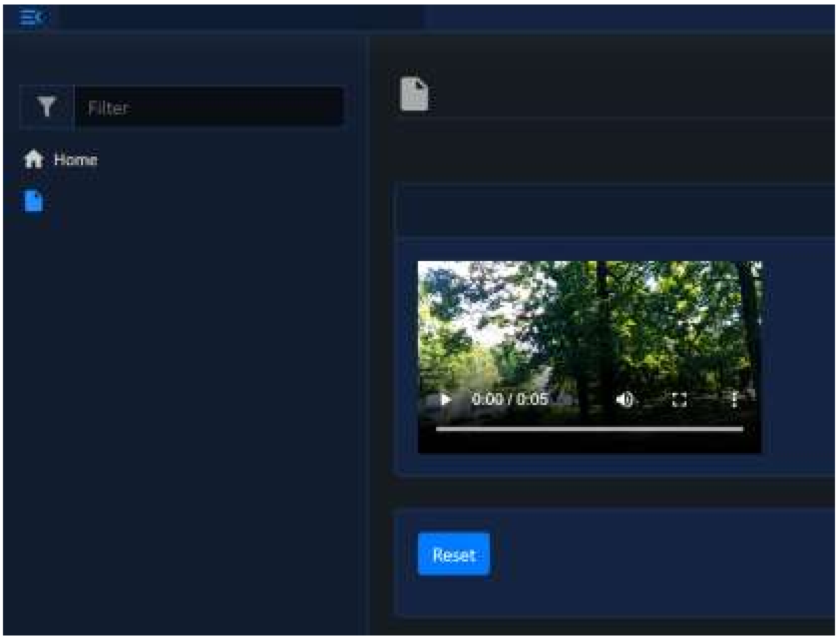


Reset

To reload an video element, and also reset the video back to the start, you can use Reset-PodeWebVideo:

Example193.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name ' ' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebVideo -Name 'example' `
       -Thumbnail 'https://samplelib.com/lib/preview/mp4/sample-5s.jpg' -Source @(
        New-PodeWebVideoSource -Url 'https://samplelib.com/lib/preview/mp4/sample-5s.mp4'
    ) )
    New-PodeWebContainer -Content @(
    New-PodeWebButton -Name 'Reset' -ScriptBlock {
        Reset-PodeWebVideo -Name 'example'
} ) } }
```

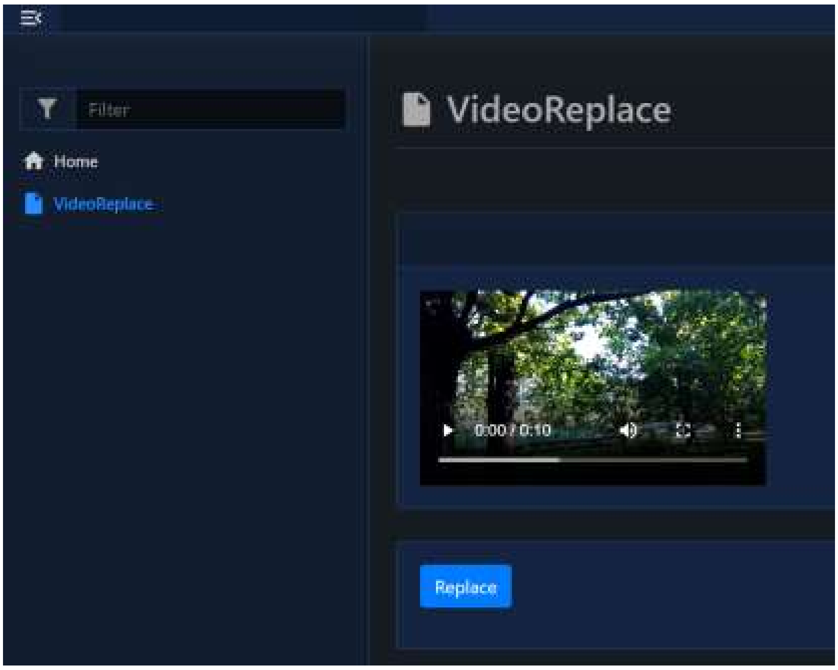


Update

To update the sources/tracks of an video element, you can use Update-PodeWebVideo. This will clear all current sources/tracks, add the new ones, and then reload the element:

Example194.ps1

```
Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
    New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
    Use-PodeWebTemplates -Title ' ' -Theme Dark
    Add-PodeWebPage -name 'VideoReplace' -ScriptBlock {

    New-PodeWebCard -Content @(
    New-PodeWebVideo -Name 'sample' `
       -Thumbnail 'https://samplelib.com/lib/preview/mp4/sample-5s.jpg' -Source @(
        New-PodeWebVideoSource `
           -Url 'https://samplelib.com/lib/preview/mp4/sample-5s.mp4'
    ) )
    New-PodeWebContainer -Content @(
    New-PodeWebButton -Name 'Replace' -ScriptBlock {
        Update-PodeWebVideo -Name 'sample' `
              -Thumbnail 'https://samplelib.com/lib/preview/mp4/sample-10s.jpg' `
              -Source @( New-PodeWebVideoSource `
              -Url 'https://samplelib.com/lib/preview/mp4/sample-10s.mp4'
) } ) } }
```


Filter
Home
VideoReplace
VideoReplace
0:00 / 0:10
Replace

Aditional Bugs discovered

To find information about most of the bugs please visit the github:
https://badgerati.github.io/Pode.Web/release-notes

Below are some bugs that I faced while working on my project:

- Function Out-PodeWebText not working as expected while passing an object to it
- if adding -SimpleFilter and -SimpleSort to the table, all (big-data) data might not get processed
- if u create a query form that will run longer than 30s, the system will cycle (3 times?), running icon on your button will continue to run for 120 seconds and you will not ne able to click the other links on your page for that time.
- Fast reader might come to some crazy idea on how to combine functions available from Pode.Web. And this is allowed, no extra warnings will show. But if you did it the wrong way, the result will not be as expoected. That’s very frustrating.

Timeouts

Long running example with unpredicted behaviour:

- I was trying to get processes with pre-defined columns, and then also all possible columns, which will run for more than 30s.

- Author however warns about this in pode (not in pode.web !)
- https://badgerati.github.io/Pode/Tutorials/RequestLimits/#timeout

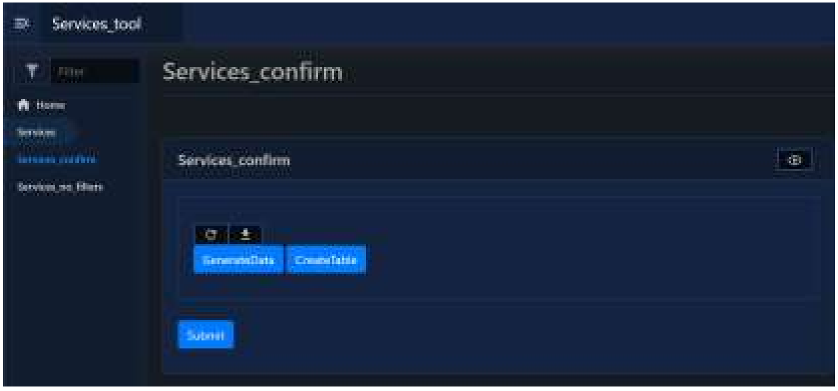
You can edit the timeout in the server.psd1 file which should accompany your server.ps1 file.

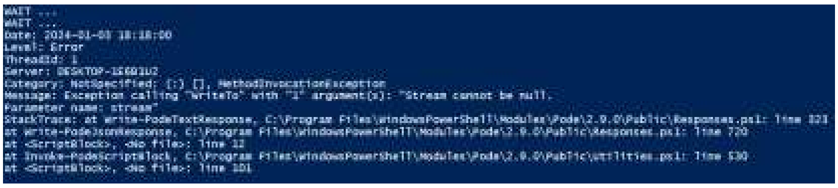
```
@{
    Server = @{
        Request = @{
            Timeout = 30
        }
    }
}
```

Example – timeout bugs.ps1

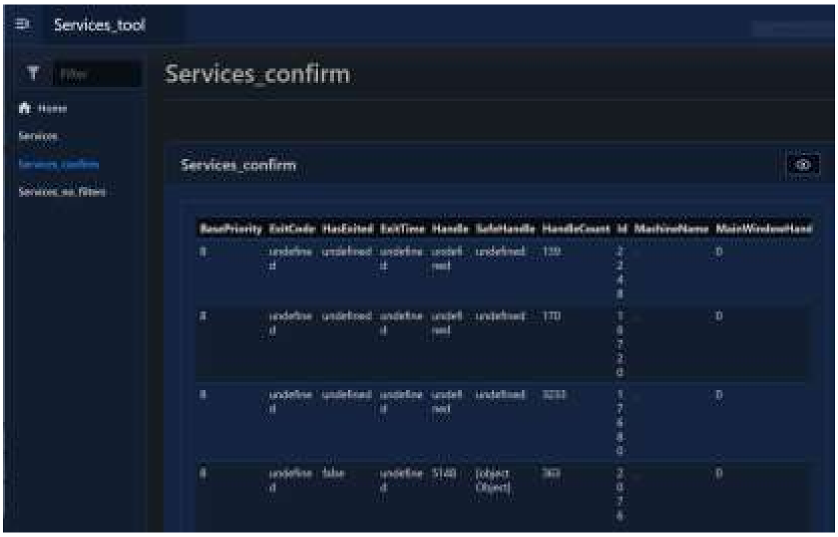
```
Start-PodeServer -EnablePool Tasks {
   Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
   New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
   Use-PodeWebTemplates -Title 'Services_tool' -Theme Dark


Add-PodeWebPage -Name Services -Icon Settings -ScriptBlock {
    New-PodeWebForm -Name " " -SubmitText "DISPLAY" -AsCard `
       -Content @(New-PodeWebContainer `
       -Content @( New-PodeWebTable -Name 'Services' `
       -Compact -SimpleFilter -SimpleSort ) ) `
       -ScriptBlock { $a=get-process|select     `
                      'ProcessName','Path','Company','Description'
                      update-PodeWebTable -name "Services" -Data @( $a ) }
}

Add-PodeWebPage -Name Services_confirm -Icon Settings `
                -ScriptBlock {
    New-PodeWebForm -Name "Services_confirm" -AsCard `
                    -ScriptBlock { echo "A" | out-default } `
                    -Content @(New-PodeWebContainer -Content @(
                                   New-PodeWebTable -Compact `
                                                    -Name "ServicesTable"
    New-PodeWebButton -Name 'GenerateData' `
                      -ScriptBlock {
                                    $global:a=get-process
                                    $a|out-default
                                    echo "B" | out-default
                                   }
    New-PodeWebButton -Name 'CreateTable' `
                      -ScriptBlock {
                             update-PodeWebTable -name 'ServicesTable' -Data @( $a )
                             Sync-PodeWebTable -Name 'ServicesTable'
                             echo "WAIT ..." | out-default
                                   }
)
)
}

Add-PodeWebPage -Name Services_no_filters -Icon Settings `
                -ScriptBlock {
                   New-PodeWebForm -Name " " -SubmitText "DISPLAY" -AsCard `
                                   -Content @(New-PodeWebContainer `
                                   -Content @( New-PodeWebTable `
                                   -Name "Services_no_filters" -Compact  ) ) `
                                   -ScriptBlock { $a=get-process
                                                  update-PodeWebTable -name `
                                                 "Services_no_filters" -Data `
                                                 @( $a ) }
}
}


<#
    if adding -Filter and -Sort to the table,
    all (big-data) might not get processed
 #>
```

In my case services are processed with preceding errors





```
WAIT ...
WAIT ...
Date: 2024-01-03 18:18:00
Level: Error
ThreadId: 1
Server: DESKTOP-1E6B1U2
Category: NotSpecified: (:) [], MethodInvocationException
Message: Exception calling "WriteTo" with "1" argument(s): "Stream cannot be null.
Parameter name: stream"
StackTrace: at Write-PodeTextResponse, C:\Program Files\WindowsPowerShell\Modules\Pode\2.9.0\Public\Responses.ps1: line 323
at Write-PodeJsonResponse, C:\Program Files\WindowsPowerShell\Modules\Pode\2.9.0\Public\Responses.ps1: line 720
at <ScriptBlock>, <No file>: line 12
at Invoke-PodeScriptBlock, C:\Program Files\WindowsPowerShell\Modules\Pode\2.9.0\Public\Utilities.ps1: line 530
at <ScriptBlock>, <No file>: line 101
```



Services_confirm

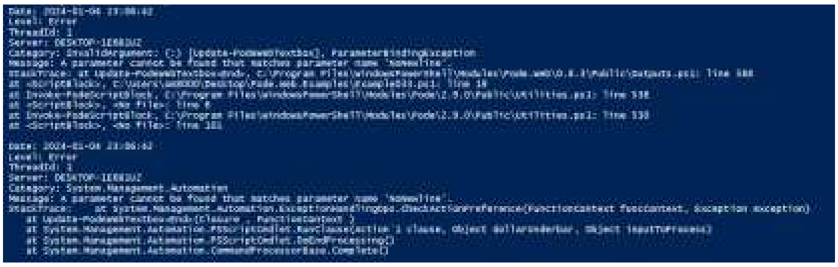
Services_confirm

BasePriority	ExitCode	HasExited	ExitTime	Handle	SafeHandle	HandleCount	Id	MachineName	MainWindowHand
8	undefined	undefined	undefined	undefined	undefined	139	2248	.	0
8	undefined	undefined	undefined	undefined	undefined	170	18720	.	0
8	undefined	undefined	undefined	undefined	undefined	3233	17480	.	0
8	undefined	false	undefined	5140	[object Object]	363	2076	.	0

Text Box not being updated

This one is also not working as expected:

```
Start-PodeServer {
 Add-PodeEndpoint -Address localhost -Port 8888 -Protocol Http
 New-PodeLoggingMethod -Terminal | Enable-PodeErrorLogging
 Use-PodeWebTemplates -Title ' ' -Theme Dark

 Add-PodeWebPage -Name Sample `
                -ScriptBlock {New-PodeWebCard -Content @(
                              New-PodeWebTextBox -name 'aa' -value $a
                              New-PodeWebTextBox -name 'bb' -value $b
                              New-PodeWebTextBox -name 'cc' -value $c
                              New-PodeWebButton `
                                 -Name 'Click Me' `
                                 -ArgumentList 'Value1', 2, 3 `
                                 -ScriptBlock {
                                              param($value1, $value2, $value3)
                                              $global:a=$value1
                                              $global:b=$value2
                                              $global:c=$value2
                                              Update-PodeWebTextBox -name 'aa' -value $a
                                              Update-PodeWebTextBox -name 'bb' -value $b
                                              Update-PodeWebTextBox -name 'cc' -value $c
                                              $a=0;$b=0;$c=0
                                             }
                              )
                             }
}
```



```
Date: 2024-01-04 23:06:42
Level: Error
ThreadId: 1
Server: DESKTOP-1E6B1UZ
Category: InvalidArgument: (:) [Update-PodeWebTextbox], ParameterBindingException
Message: A parameter cannot be found that matches parameter name 'NoNewLine'.
StackTrace: at Update-PodeWebTextbox<End>, C:\Program Files\WindowsPowerShell\Modules\Pode.Web\0.8.3\Public\Outputs.ps1: line 588
at <ScriptBlock>, C:\Users\[illegible]\Desktop\Pode.Web.Examples\Example511.ps1: line 18
at Invoke-PodeScriptBlock, C:\Program Files\WindowsPowerShell\Modules\Pode\2.9.0\Public\Utilities.ps1: line 538
at <ScriptBlock>, <No file>: line 8
at Invoke-PodeScriptBlock, C:\Program Files\WindowsPowerShell\Modules\Pode\2.9.0\Public\Utilities.ps1: line 538
at <ScriptBlock>, <No file>: line 101

Date: 2024-01-04 23:06:42
Level: Error
ThreadId: 1
Server: DESKTOP-1E6B1UZ
Category: System.Management.Automation
Message: A parameter cannot be found that matches parameter name 'NoNewLine'.
StackTrace:    at System.Management.Automation.ExceptionHandlingOps.CheckActionPreference(FunctionContext funcContext, Exception exception)
   at Update-PodeWebTextbox<End>(Closure , FunctionContext )
   at System.Management.Automation.PSScriptCmdlet.RunClause(Action`1 clause, Object dollarUnderbar, Object inputToProcess)
   at System.Management.Automation.PSScriptCmdlet.DoEndProcessing()
   at System.Management.Automation.CommandProcessorBase.Complete()
```

Get all Pode.Web functions:

```
import-module pode.web
(get-module pode.web).ExportedCommands.keys
```

O. Mike
2024

Pode.Web

a Powershell tool for rapid web applications development
by Matthew Kelly Badgeratty