Powershell 101

How to Code in Powershell

01

Coding Hygiene

Things to know when coding in Powershell

02

Essentials

The basics you will find in any language

03

Debugging

Finding and fixing errors in your code

04

Running Scripts

How to run your scripts after you make them

05

File Manipulation

Creating, writing, removing, and editing files

01

Coding Hygiene

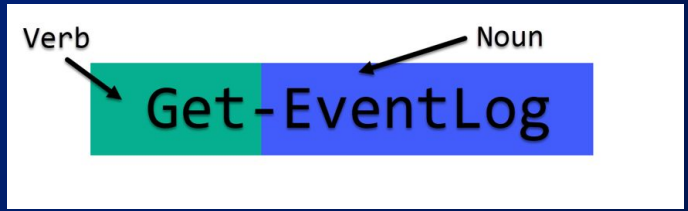
Things to know when coding in Powershell

Dos

How to write effective code:

- Use the complete cmdlet/function name (eg. Set-Location vs chdir)
- Utilize proper capitalization (eg. Write-Host vs write-host)
- Use consistent indentation and proper form
- Keep it simple (less is more)
- Use clear variable names (eg. $AppleTrees vs $appletrees

```
1  $UserTest = "John.Doe@company.com"
2  If (Get-Mailbox $UserTest) {
3      Write-Host "The mailbox for $UserTest was found".
4  } Else {
5      Write-Host "No mailbox was found with the email address of $UserTest."
6      If ($_.ServerName -match "DAGSERVER1") {
7          Write-Host "The mailbox for $UserTest was on DAGSERVER1."
8      } Else {
9          Write-Host "The mailbox for $UserTest was not on DAGSERVER1."
10     }
11 }
```



Don’ts

```
cd C:\Program Files\Google\Chrome
chdir C:\Users\Public

If($env:USERNAME == "user") { echo("hello")}
Else {
write-host "your bad" | grep "you" > file.txt}
```

```
1   $a = "I am the most important person in the world"
2   $b = $a + 2
3   $A = "I am the worst person in the world"
4   $aa = 17
5   $AAA = 923131
6   $aAaA = "124192301"
```

- Use aliases in your code (makes it hard to read)
- Use nondescript variable names (eg. $x vs $UserCount)
- Have inconsistent indentation and braces/brackets
- Be the only one who can read your code
    - (You are on a TEAM!)

02

Essentials

Programming Basics

IF Statements

If statements are used to determine conditional values on whether or not code should run.

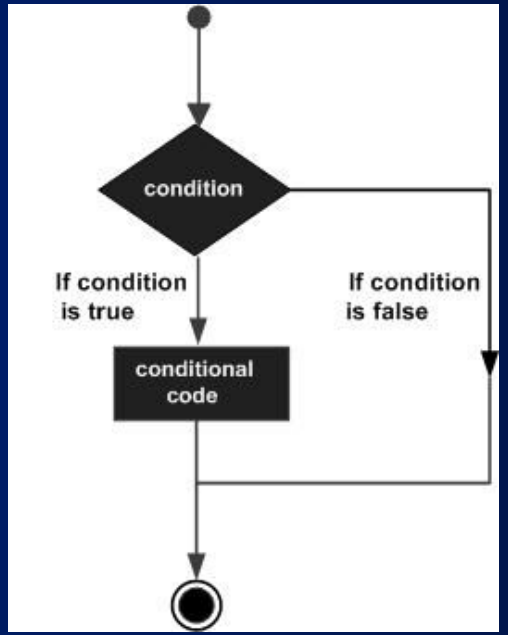
If's have 3 states: If, Elseif, and Else.

- **If:** Necessary, determines to run code based on whether or not the input is True or False
- **Elseif:** Optional, runs another if statement if the previous one evaluated to False
- **Else:** Optional, if all statements prior evaluated to false, this statement is run.

```
$number=17

if($number -le 20)
{
    $number+=5
}
elseif($number -gt 20)
{
    $number-=12
}
else
{
    $number=21
}

Write-Host($number)
```



Nested Ifs:

If statements can be nested inside of each other to check multiple conditions.

```
$x = 30
$y = 10

if($x -eq 30){
    if($y -eq 10) {
        write-host("X = 30 and Y = 10")
    }
}
```

Multiple Conditions:

If statements can have multiple conditions using “-and” and “-or”

```
$x = 30
$y = 10

if(($x -eq 30) -and ($y -eq 10))
{
    Write-Host("X = 30 and Y = 10")
}
```

Comparison Operators

-eq

Equals; =
Evaluates to TRUE if both sides are equivalent

-ge

Greater or Equal to; >=
Evaluates to TRUE if the left side is greater than or equal to the right

-gt

Greater Than; >
Evaluates to TRUE if the left side is greater than the right

-ne

Not Equals; !=
Evaluates to TRUE if both sides are different

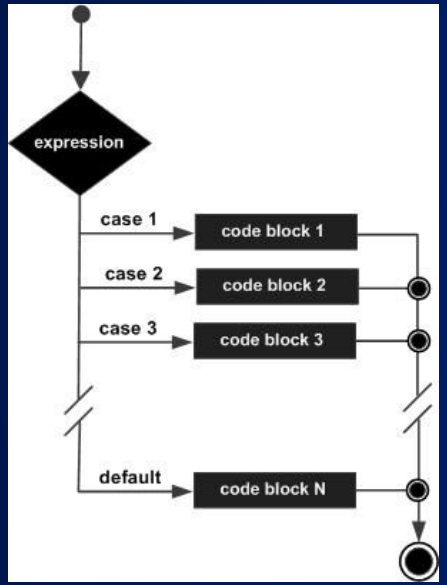
-le

Less or Equal to; <=
Evaluates to TRUE if the left side is less than or equal to the right

-lt

Less Than; <
Evaluates to TRUE if the left side is less than the right

Switch Statements




```
$x = 3

switch($x){
    1{"One"; break;}
    2{"Two"; break;}
    3{"Three"; break;}
    4{"Four"; break;}
    5{"Five"; break;}
    default{"Zero"; break;}
}
```

Switch statements are useful to avoid using long If-Else chains.

A switch statement takes a singular input and selects an output based on the available choices.

Switch statements run top to bottom, so having a “break;” after each condition is proper form.

Loops

Loops are useful for doing repetitive tasks or for executing a statement multiple times.

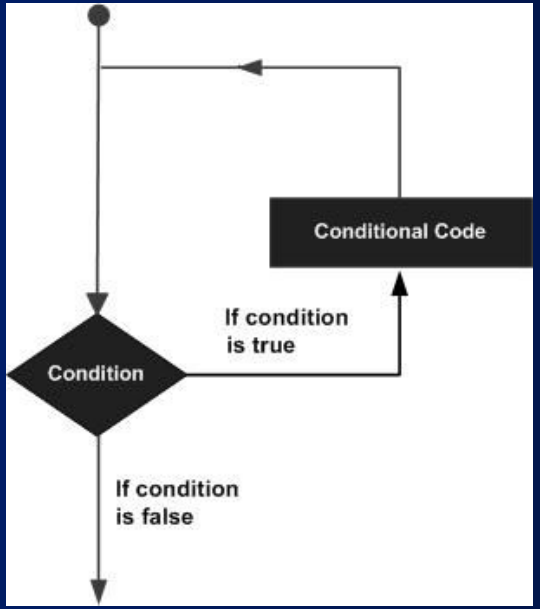
There are four types of loops: For, ForEach, While, Do...While:

- **For:** Execute a sequence of statements multiple times
- **ForEach:** Enhanced For Loop. Executes statements on each element in a collection
- **While:** Repeats a statement while the given condition is TRUE
- **Do...While:** A while loop, except the test condition exists at the end of the loop

```
$number=17

if($number -le 20)
{
    $number+=5
}
elseif($number -gt 20)
{
    $number-=12
}
else
{
    $number=21
}

Write-Host($number)
```



For Loop:

Given a variable, repeats the loop until a condition is met. Similar in syntax to Java

```
$array = @("item1","item2","item3")

for($i = 0; $i -lt $array.length; $i++)
{
    $array[$i]
}
```

ForEach Loop:

Given a collection of items, such as an array, repeats a statement for each item

```
$array = @("item1","item2","item3")

foreach($element in $array)
{
    $element
}
```

```
$array = @("item1","item2","item3")
$counter = 0;

while($counter -lt $array.length){
    $array[$counter]
    $counter += 1
}
```

While Loops:

As long as the conditional statement is TRUE, it will continue to execute the code

```
$array = @("item1","item2","item3")
$counter = 0;

do {
    $array[$counter]
    $counter += 1
} while($counter -lt $array.length)
```

Do...While Loops:

Same thing as a while loop, except it checks for the conditional statement at the end. This means that the code is guaranteed to run at least once.

03

File Manipulation

Writing, Reading, Editing, Executing, and Deleting

Folders

New-Item -ItemType Directory

A directory is equivalent to a Folder. By running this command it will spawn a new folder at PATH.

Copy-Item PATH_A PATH_B

Copies a folder from PATH_A to PATH_B. This also maintains all contents of the folder.

Move-Item PATH_A PATH_B

This removes the folder from the current path, PATH_A, and moves it to a new location, PATH_B.

Test-Path

If there exists a folder path at this location, then it will return TRUE, otherwise it'll return FALSE.

Remove-Item

Removes the folder at the given location. Also deletes the contents recursively.

Rename-Item

Simply renames the folder. The folders should be exactly the same, the only difference being the name.

Files

New-Item -ItemType File

File is data written to a location. By running this command it will spawn a new file at PATH.

Copy-Item PATH_A PATH_B

Copies a file from PATH_A to PATH_B. This also maintains all contents of the file.

Move-Item PATH_A PATH_B

This removes the file from the current path, PATH_A, and moves it to a new location, PATH_B.

Test-Path

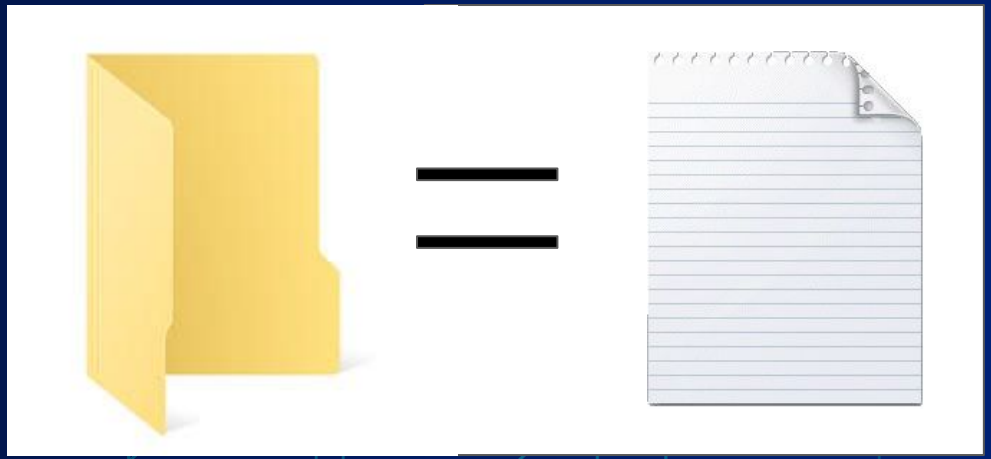
If there exists a filepath at this location, then it will return TRUE, otherwise it'll return FALSE.

Remove-Item

Removes the file at the given location. Also deletes the contents.

Rename-Item

Simply renames the file. The files should be exactly the same, the only difference being the name.



Why are folders and files the same?

In Powershell, both treated as objects. The reason they can both be manipulated the exact same is because they are Items. Items generally have basic properties such as name and location. In this way we can manipulate those properties of the Folder and File Items.

Reading Content:

Get-Content reads the information from inside any file.

```
PS C:\Users\Aidan Masinsin\Desktop\Windows Scripts> Get-Content text.txt
Hello World!
```

Writing Content:

Set-Content changes the contents of the specified file.

```
Set-Location "C:\Users\Aidan Masinsin\Desktop\Windows Scripts"
New-Item text.txt
Set-Content text.txt "Hello World!"
Get-Content text.txt
Set-Content text.txt "Change da world. My final message. Goodbye."
Get-Content text.txt
```

Adding Content:

Append-Content adds input content onto the end of the file instead of replacing it

```
Set-Location "C:\Users\Aidan Masinsin\Desktop\Windows Scripts"
New-Item text.txt
Set-Content text.txt "One Fish, Two Fish."
Get-Content text.txt
Append-Content text.txt "Red Fish, Blue Fish."
Get-Content text.txt
```

Erasing Content:

Clear-Content clears all of the content inside of the file

```
Set-Location "C:\Users\Aidan Masinsin\Desktop\Windows Scripts"
New-Item text.txt
Set-Content text.txt "One Fish, Two Fish."
Get-Content text.txt
Clear-Content text.txt
Get-Content text.txt
```

Reading File Line-by-Line

Using Foreach in Script:

```
$regex = "(8000)[0-9]{5}"

foreach($line in Get-Content file.txt) {
    if($line -match $regex){
        $line
    }
}
```

Using Foreach-Object with Pipe:

```
$regex = "(8000)[0-9]{5}"

Get-Content file.txt | ForEach-Object {
    if($_ -match $regex){
        $_
    }
}
```

04

Running Scripts

Getting the thing to run

Why is my script not running?

Set-ExecutionPolicy

The execution policy is the level in which scripts in powershell are allowed to run. This is to prevent any harmful or malicious scripts from being run without the user's knowledge. There are 5 levels of execution policy:

- Restricted
  - Will not let ANY Powershell scripts to run, only individual commands
- AllSigned
  - Only SIGNED Powershell scripts can run
- RemoteSigned
  - Allows unsigned local scripts and SIGNED REMOTE scripts to run
- Unrestricted
  - Allows unsigned scripts to run, but will warn when running downloaded scripts
- Bypass
  - ALL scripts are allowed to run, no warnings or prompts

Powershell File Types

.ps1

Powershell Scripts; most common type of PS file and similar to formats such as .bat or.sh

.psm1

Powershell Modules; these contain multiple function scripts that interact as a package. These are imported with Import-Module or Install-Module

.psd1

Powershell Data File; provide information and instructions for .psm1 files. Can serve as a help for Powershell Modules

.pssc

Powershell Session Configuration File; Describes how the session environment is built. (We won't be using this one)

.psrc

Powershell Role Capability File; Define a set of capabilities used in session configuration and .pssc files

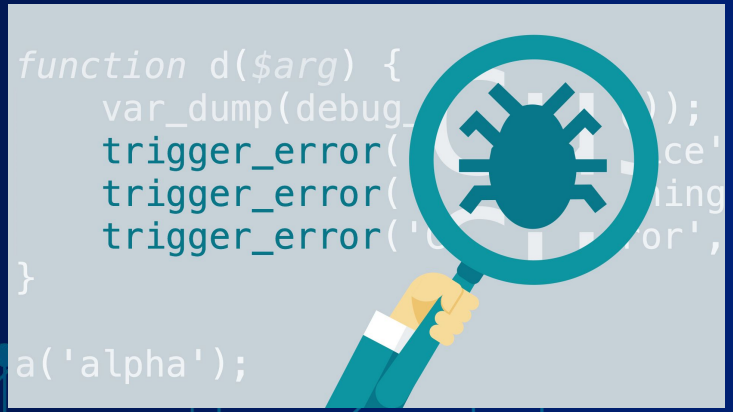
.ps1xml or .cdxml

Powershell XML; XML is a common file format that marks up text. This is pretty much a .psd1 in an XML format.

05

Debugging

Squish those bugs



debugging

/ˌdēˈbəgiNG/

Learn to pronounce

noun

the process of identifying and removing errors from computer hardware or software.

"software debugging"

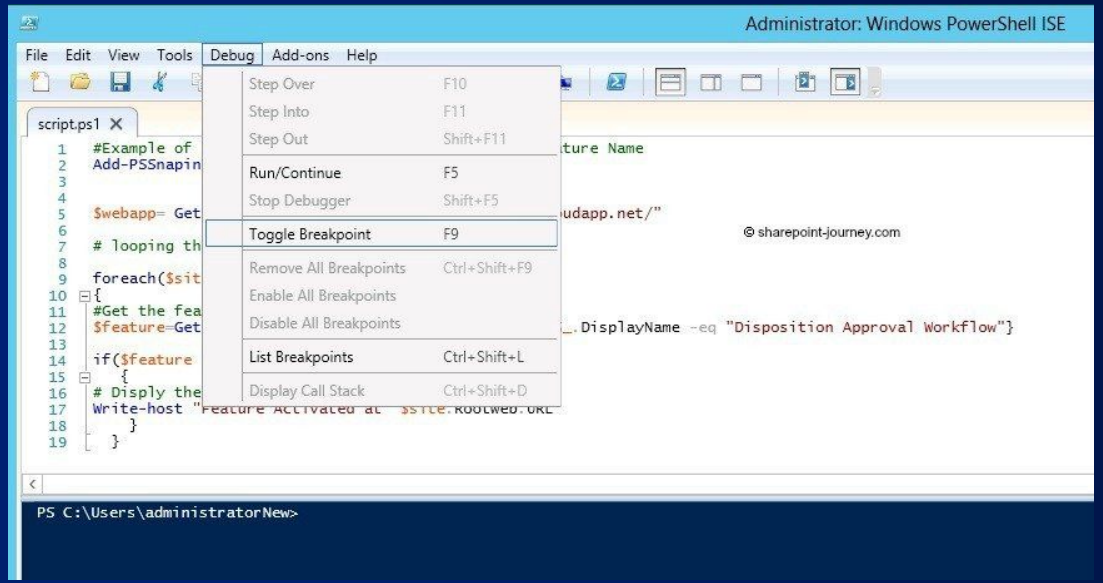
Breakpoints

Breakpoints are points in the code in which your code will stop in order to figure out what is wrong.

There are two ways to toggle breakpoints:

- Right Click on Line + Toggle Breakpoint
- F9 Key

During a breakpoint you can execute code in what is known as the debugger. This is a shell which is running the current instance of your script. From here you can choose to continue your script or write code in the debug shell.



Thanks!

Do you have any questions?
youremail@freepik.com
+91 620 421 838
yourcompany.com

Alternative Resources

Resources

Did you like the resources on this template? Get them for free at our other websites.

PHOTOS:

- Friendly colleagues with devices at desk
- Close-up happy guy wearing vr glasses

MOCK-UPS:

- Simple modern business stationery

VECTORS:

- Geometric logo collection
- Neon lights technology background
- Blue abstract hud technology background
- Technology background

ICONS:

- Technology Icon Pack

Instructions for use

In order to use this template, you must credit **Slidesgo** by keeping the **Thanks** slide.

You are allowed to:

- Modify this template.
- Use it for both personal and commercial projects.

You are not allowed to:

- Sublicense, sell or rent any of Slidesgo Content (or a modified version of Slidesgo Content).
- Distribute Slidesgo Content unless it has been expressly authorized by Slidesgo.
- Include Slidesgo Content in an online or offline database or file.
- Offer Slidesgo templates (or modified versions of Slidesgo templates) for download.
- Acquire the copyright of Slidesgo Content.

For more information about editing slides, please read our FAQs or visit Slidesgo School:

https://slidesgo.com/faqs and https://slidesgo.com/slidesgo-school

Fonts & colors used

This presentation has been made using the following fonts:

Montserrat
(https://fonts.google.com/specimen/Montserrat)

Montserrat Alternates
(https://fonts.google.com/specimen/Montserrat+Alternates)

#06bad6

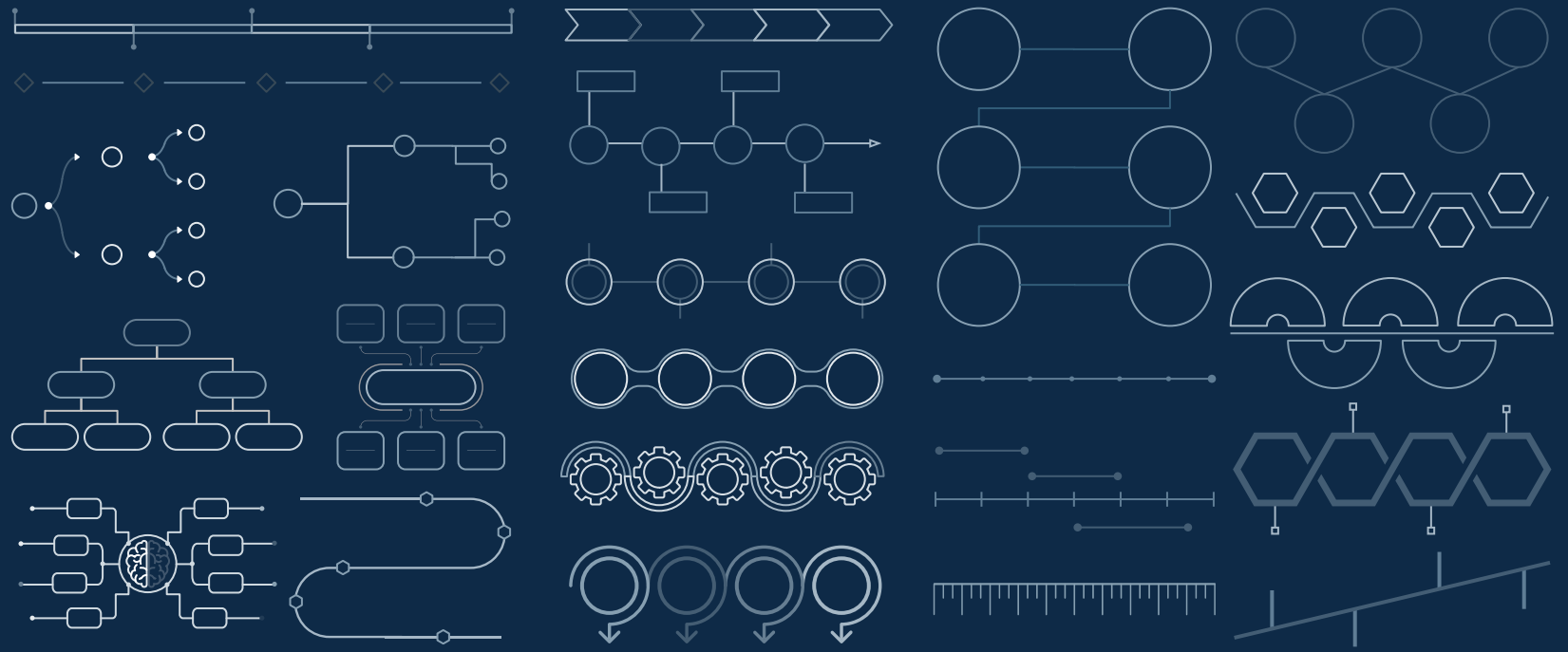
#f3f3f3

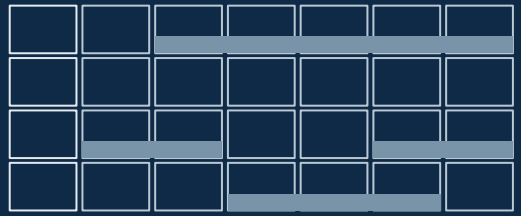
Use our editable graphic resources...

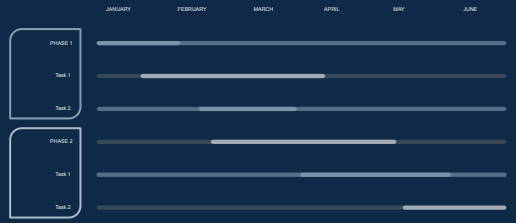
You can easily resize these resources without losing quality. To change the color, just ungroup the resource and click on the object you want to change. Then, click on the paint bucket and select the color you want. Group the resource again when you're done.

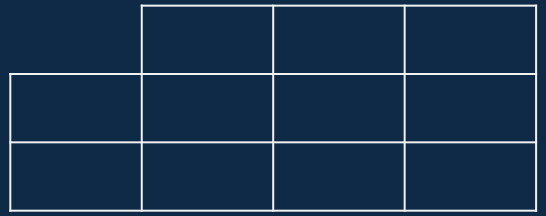


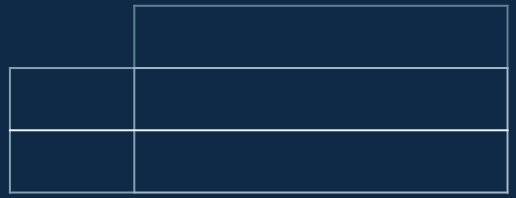


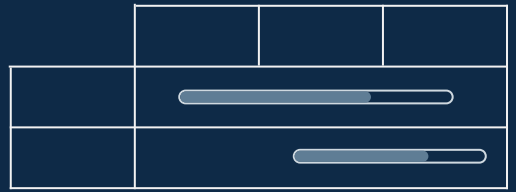



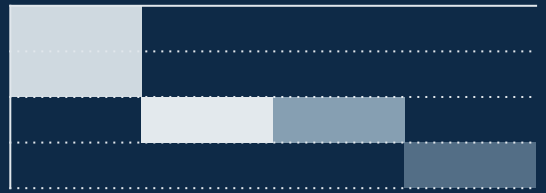


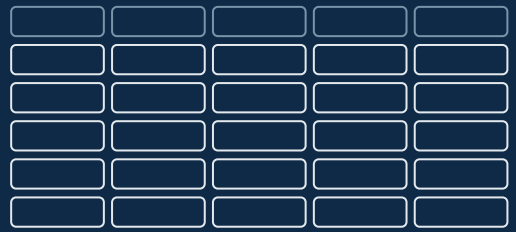

JANUARY
FEBRUARY
MARCH
APRIL
MAY
JUNE
PHASE 1
Task 1
Task 2
PHASE 2
Task 1
Task 2

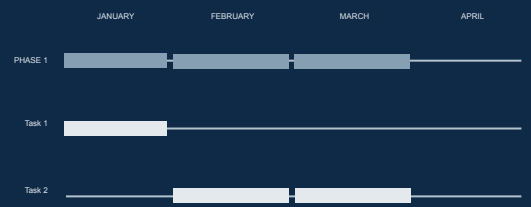


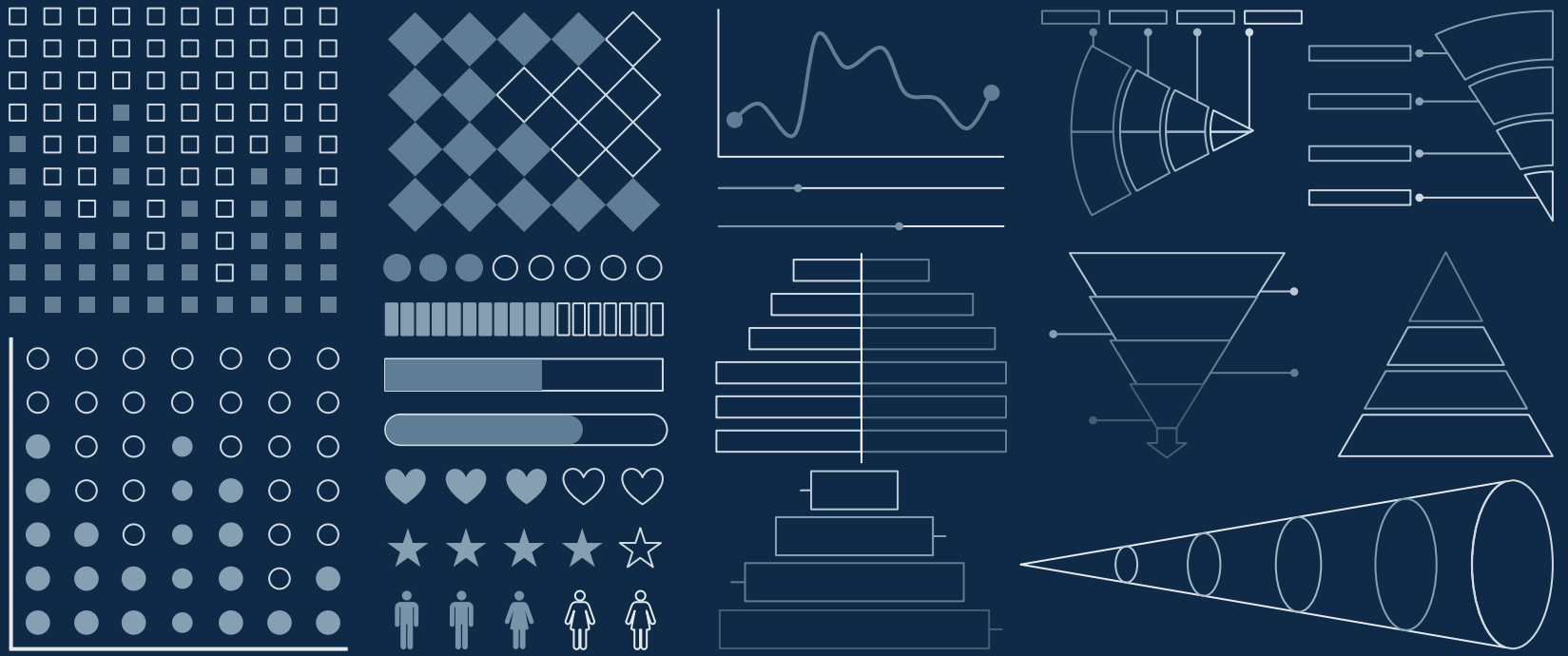









JANUARY
FEBRUARY
MARCH
APRIL
PHASE 1
Task 1
Task 2



...and our sets of editable icons

You can resize these icons without losing quality.
You can change the stroke and fill color; just select the icon and click on the paint bucket/pen.
In Google Slides, you can also use Flaticon's extension, allowing you to customize and add even more icons.

Educational Icons

Medical Icons

Business Icons

Teamwork Icons

Help & Support Icons

Avatar Icons

Creative Process Icons

Performing Arts Icons

Nature Icons

SEO & Marketing Icons


slidesgo