FLUTTER TUTORIALS

BY: **TECMAN**

TECMAN Lesson 1

What is Flutter

- Flutter is an open-source mobile application development framework created by Google. It is used to develop applications for both Android and iOS.
- In simple terms, Flutter is a framework for building mobile apps for cross-platform which is Android and iOS with one code base.
- Flutter promises fast development cycles, fast UI rendering and unique UI design, and native app performance on both platforms

TECMAN LIMITED 2019

Flutter Architecture

Flutter is made up of four major components, these include:

- Dart platform
- Flutter engine
- Foundation library
- Design-specific widgets

TECMAN LIMITED 2019

WHY FLUTTER

- Flutter is open-source, its native cross-platform mobile app development platform allows developers to build applications from one codebase based on the Dart programming language.
- Besides that, Flutter is fast, and because it is fast, it is faster to develop, faster to tweak, play, and prototype with, and it’s fast on the device.
- In less time than it used to take developers to write one app for one platform, developers can now write apps for both platforms that look and feel even better than if we had written them independently.

TECMAN LIMITED 2019

Introduction To Dart

What is Dart?

- Dart is a general-purpose programming language originally developed by Google and later approved as a standard by ECMA (ECMA-408). It is used to build web, server, desktop, and mobile applications.
- Dart is an object-oriented, class defined, garbage-collected language using a C-style syntax that transcompiles optionally into JavaScript. It supports interfaces, mixins, abstract classes, reified generics, static typing, and a sound type system.

WHY DART?

- Dart makes it easier to create smooth animations and transitions that run at 60fps. Dart can do object allocation and garbage collection without locks.
- And like JavaScript, Dart avoids preemptive scheduling and shared memory (and thus locks).
- Because Flutter apps are compiled to native code, they do not require a slow bridge between realms (e.g., JavaScript to native).
- They also start up much faster.

TECMAN LIMITED 2019

Prerequisite

You need to have done the necessary installations and done setting up your platforms.

Technologies you need are;

- Flutter SDK
- Dart SDK
- Visual Studio Code
- Android Studio

TECMAN LIMITED 2019

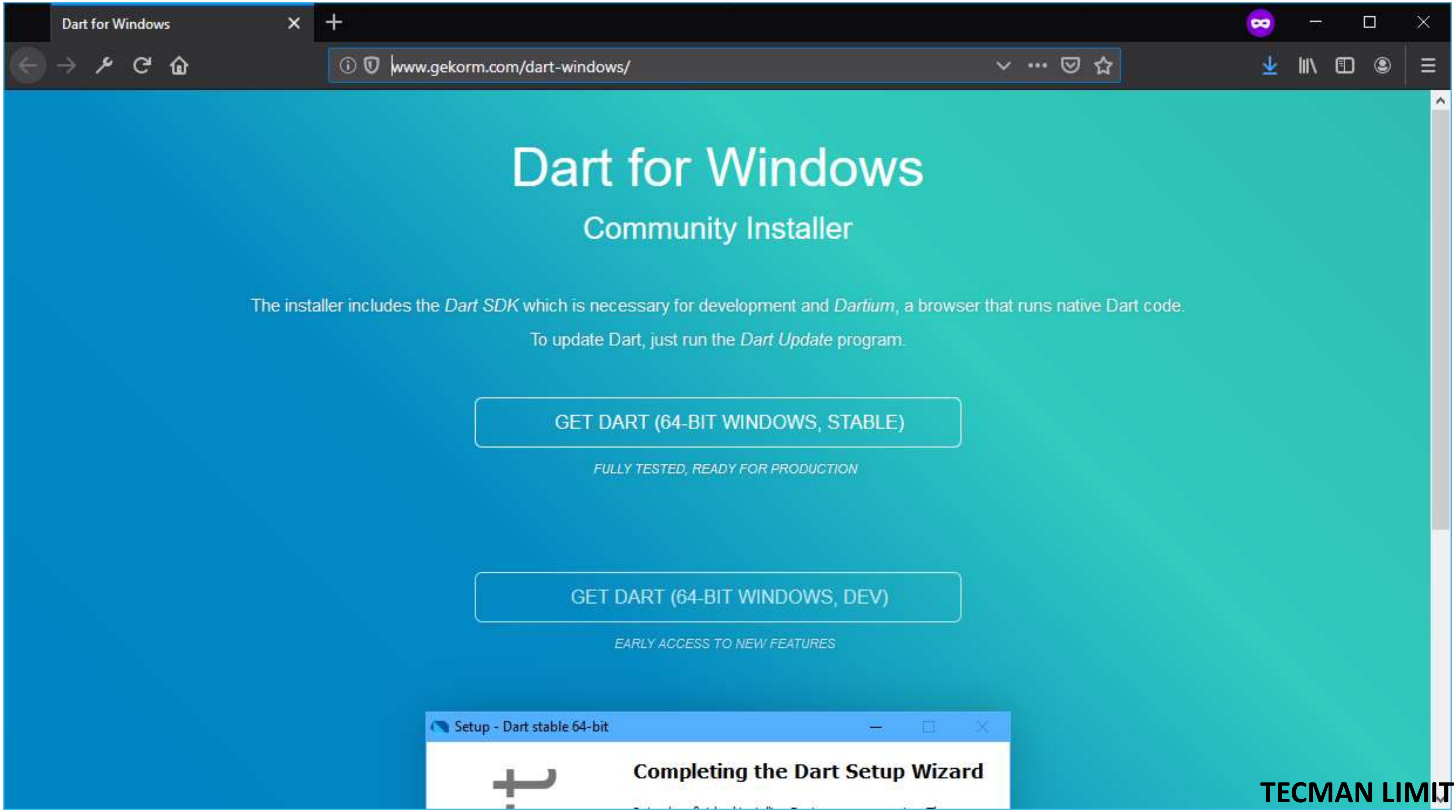
Setting Up Dart

Installing the Dart SDK.

Visit http://www.gekorm.com/dart-windows/ to download the latest version of the Dart Software Development Kit (SDK) for Windows.

Download the preferred version, as the time of writing, the version of Dark SDK was 2.2.0.

TECMAN LIMITED 2019


Dart for Windows
www.gekorm.com/dart-windows/
GET DART (64-BIT WINDOWS, STABLE)
GET DART (64-BIT WINDOWS, DEV)
Setup - Dart stable 64-bit

Dart for Windows

Community Installer

The installer includes the *Dart SDK* which is necessary for development and *Dartium*, a browser that runs native Dart code.

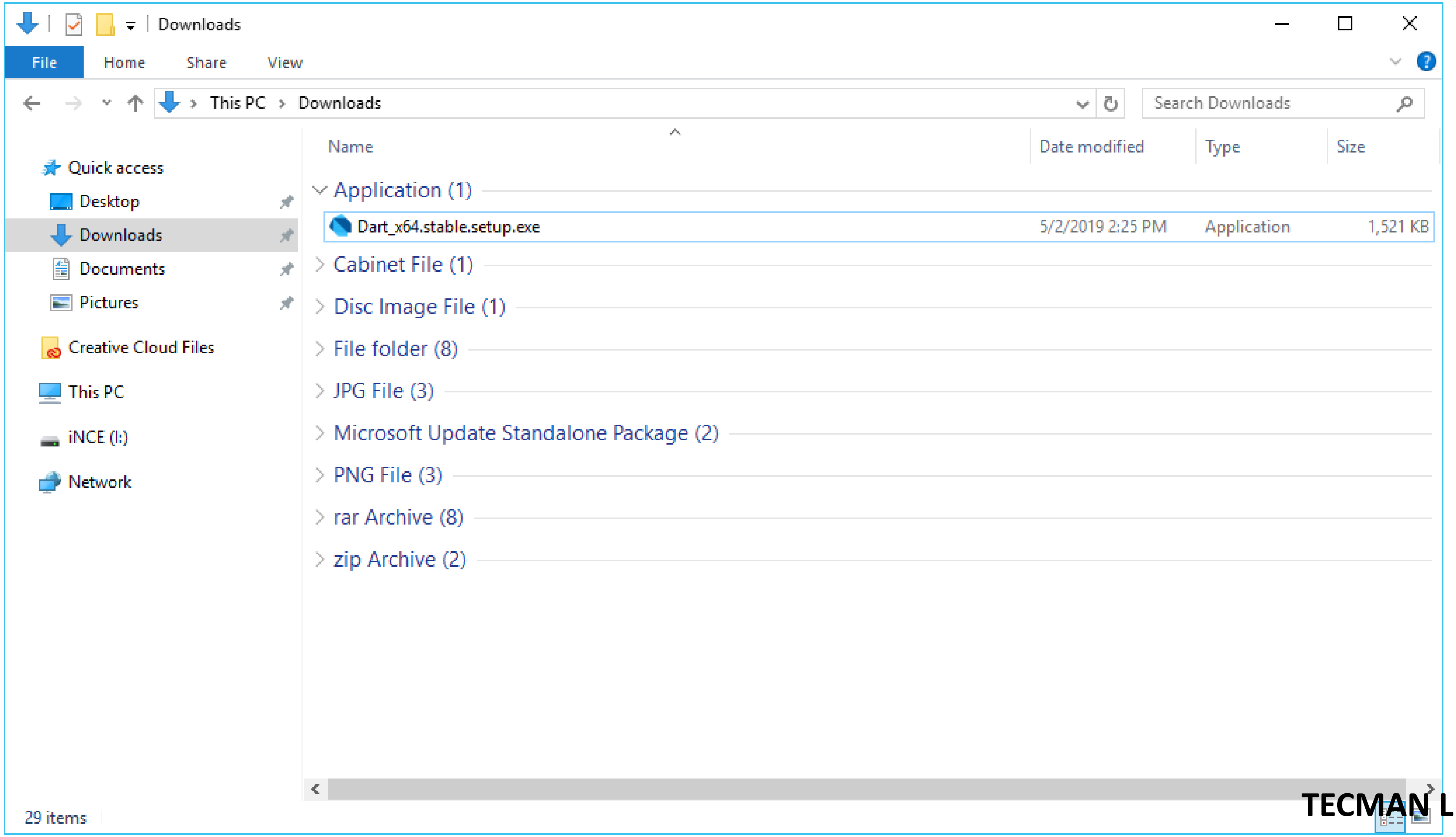
To update Dart, just run the *Dart Update* program.

FULLY TESTED, READY FOR PRODUCTION

EARLY ACCESS TO NEW FEATURES

Completing the Dart Setup Wizard

- Open Dart_x64.stable.setup.exe to start the installation process. Accept the default install settings and complete the installation process.

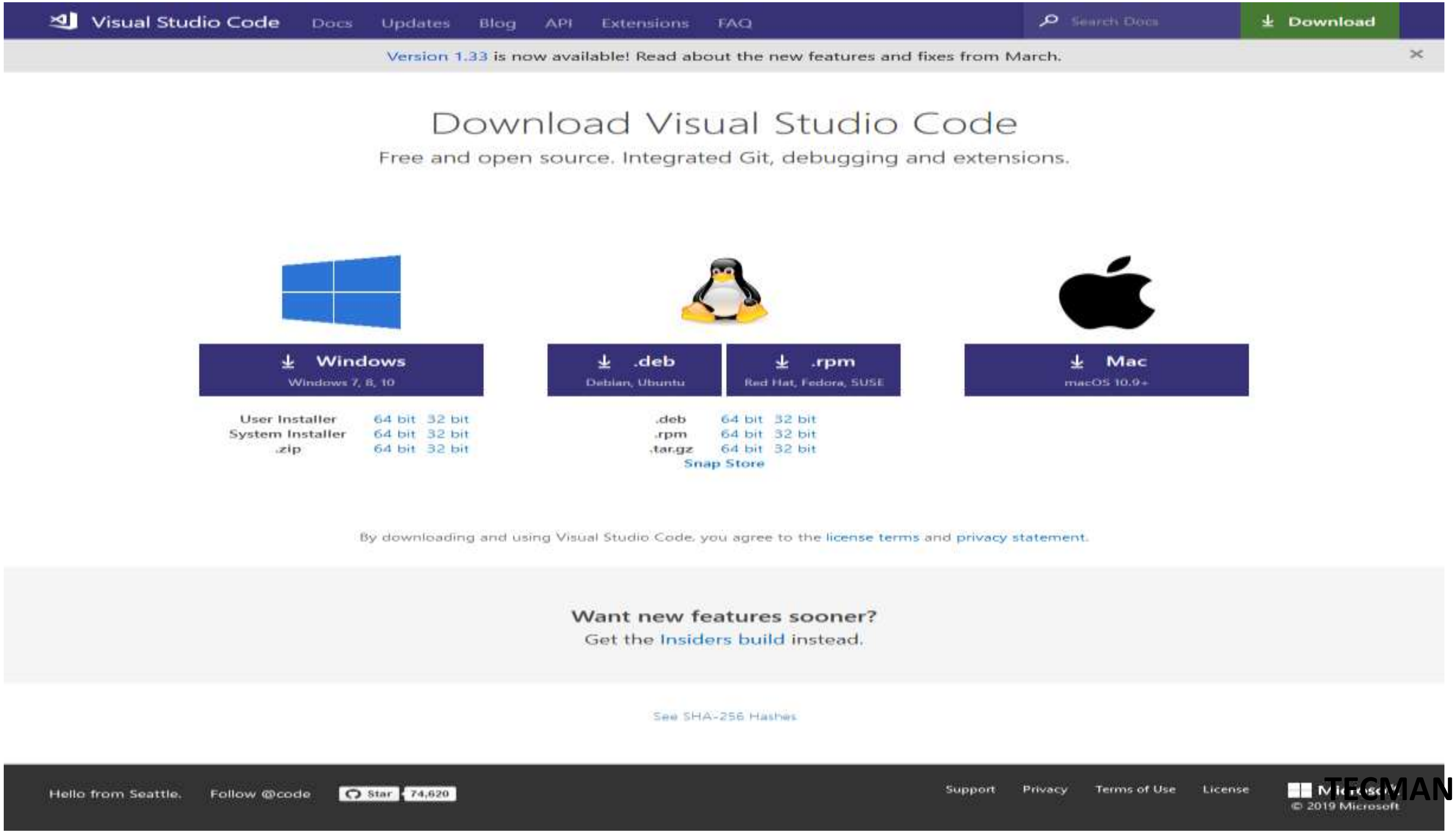


Name	Date modified	Type	Size
Application (1)			
Dart_x64.stable.setup.exe	5/2/2019 2:25 PM	Application	1,521 KB
Cabinet File (1)			
Disc Image File (1)			
File folder (8)			
JPG File (3)			
Microsoft Update Standalone Package (2)			
PNG File (3)			
rar Archive (8)			
zip Archive (2)			

TECMAN LIMITED 2019

Installing Visual Studio Code

- Visit https://code.visualstudio.com/Download to download the latest version of Visual Studio Code.



Version 1.33 is now available! Read about the new features and fixes from March.

Download Visual Studio Code

Free and open source. Integrated Git, debugging and extensions.

User Installer	64 bit	32 bit
System Installer	64 bit	32 bit
.zip	64 bit	32 bit

.deb	64 bit	32 bit
.rpm	64 bit	32 bit
.tar.gz	64 bit	32 bit

Snap Store

By downloading and using Visual Studio Code, you agree to the license terms and privacy statement.

Want new features sooner?

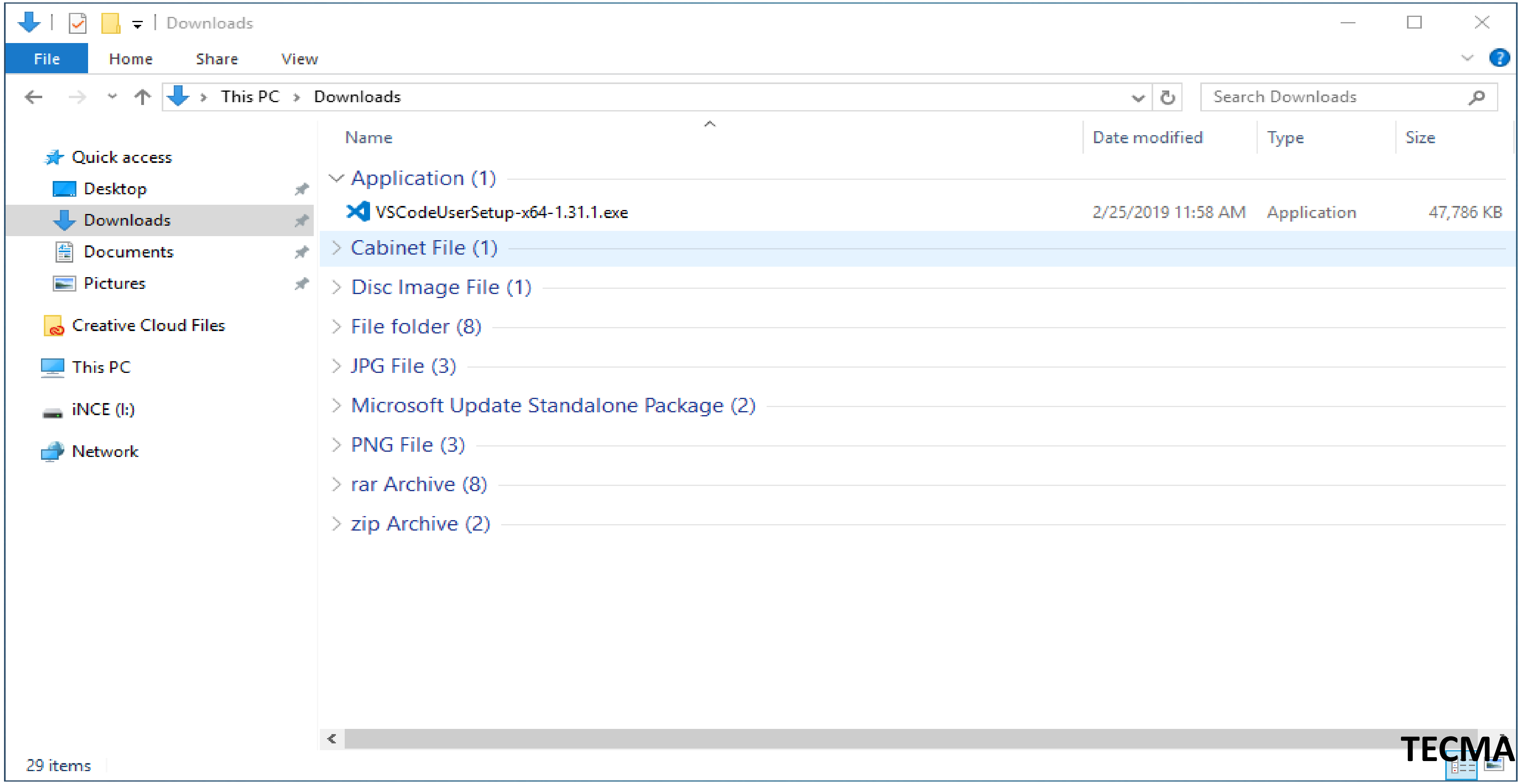
Get the Insiders build instead.

See SHA-256 Hashes

TECMAN LIMITED 2019

As the time of writing, Visual Studio Code version 1.33 was available.

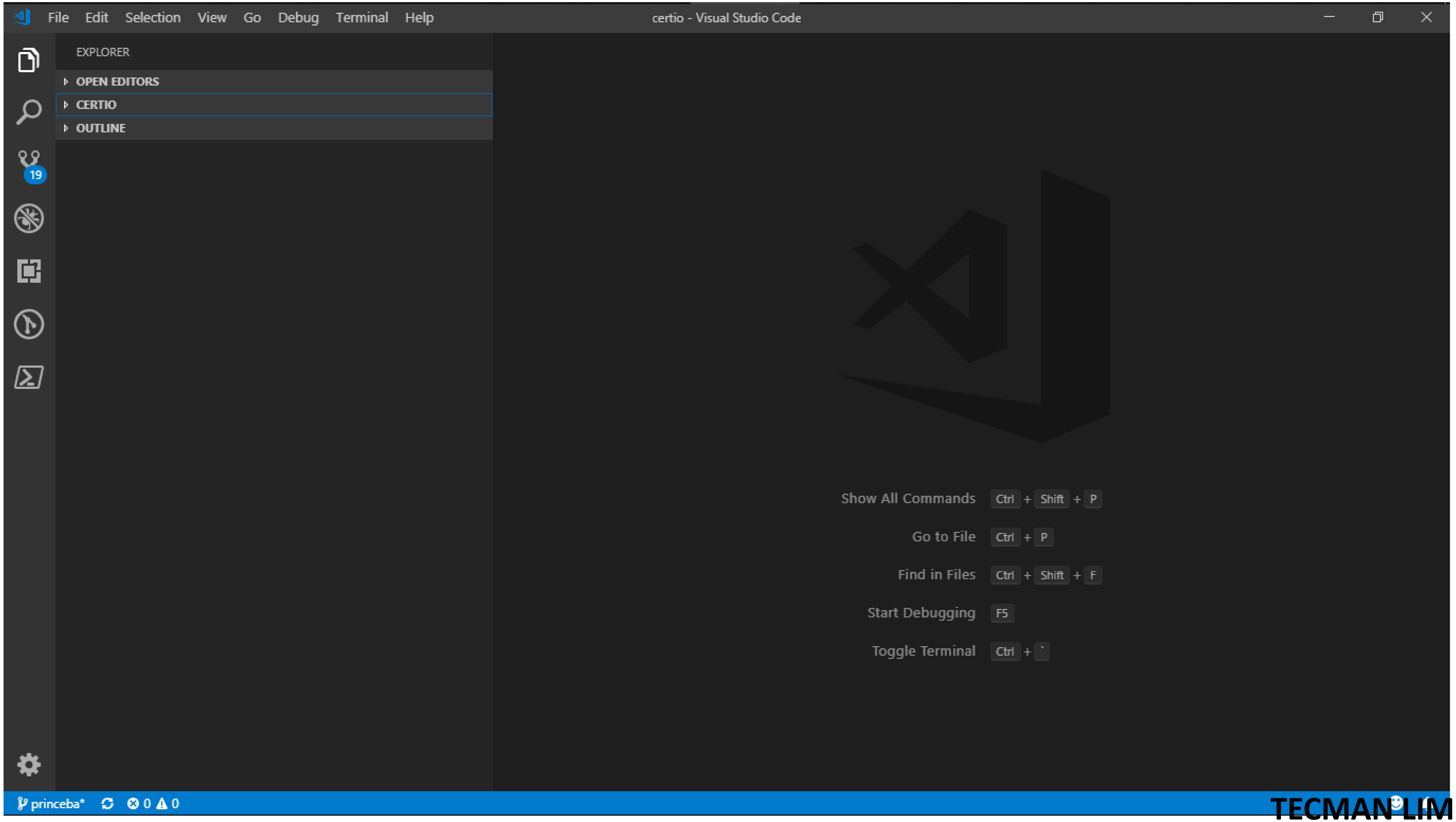
- Download your to your preferred operating system environment.
- Open the VSCodeUserSetup-x64-1.31.1.exe file to start the installation process, accept the defaults and complete the installation.



Name	Date modified	Type	Size
Application (1)			
VSCodeUserSetup-x64-1.31.1.exe	2/25/2019 11:58 AM	Application	47,786 KB
Cabinet File (1)			
Disc Image File (1)			
File folder (8)			
JPG File (3)			
Microsoft Update Standalone Package (2)			
PNG File (3)			
rar Archive (8)			
zip Archive (2)			

TECMAN LIMITED 2019

- Launch Visual Studio Code



File Edit Selection View Go Debug Terminal Help
certio - Visual Studio Code
EXPLORER
OPEN EDITORS
CERTIO
OUTLINE
19
princeba*
0 0

Show All Commands Ctrl + Shift + P

Go to File Ctrl + P

Find in Files Ctrl + Shift + F

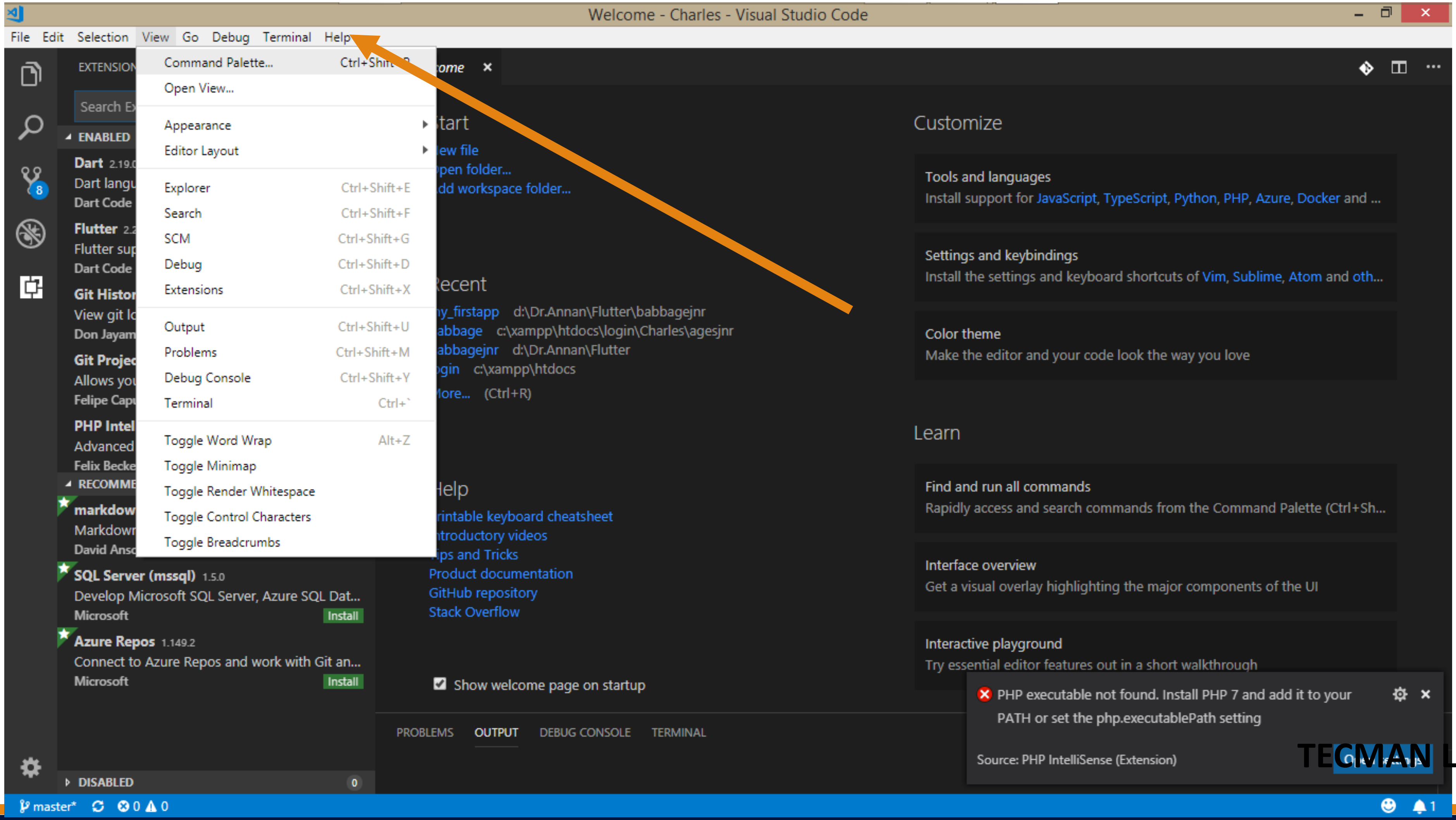
Start Debugging F5

Toggle Terminal Ctrl + `

TECMAN LIMITED 2019

Installing The Dart Plug-in in VS Code.

- Open Visual Studio Code (VSCODE)
- Locate the View tab and select command palette from the option. *(You can use the short cut: **Ctrl+Shift+P**)*

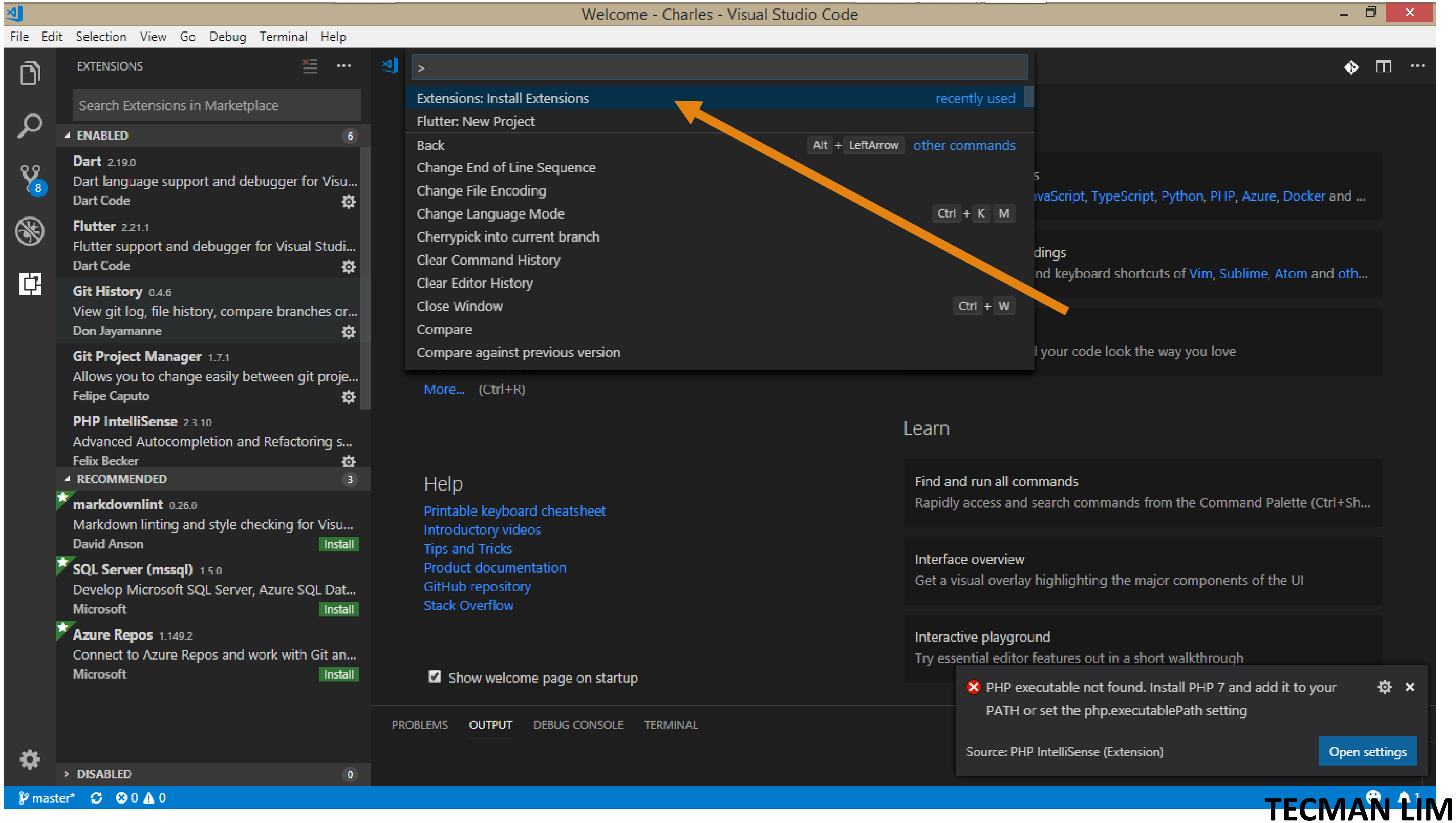


PHP executable not found. Install PHP 7 and add it to your PATH or set the php.executablePath setting

Source: PHP IntelliSense (Extension)

TECMAN LIMITED 2019

- Next, select Extension: Install Extensions


Welcome - Charles - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXTENSIONS
Search Extensions in Marketplace
ENABLED 6
RECOMMENDED 3
DISABLED 0
>
Extensions: Install Extensions recently used
Flutter: New Project
Back Alt + LeftArrow other commands
Change End of Line Sequence
Change File Encoding
Change Language Mode Ctrl + K M
Cherrypick into current branch
Clear Command History
Clear Editor History
Close Window Ctrl + W
Compare
Compare against previous version
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Open settings
master*
0 0
Install

Dart 2.19.0
Dart language support and debugger for Visu...
Dart Code

Flutter 2.21.1
Flutter support and debugger for Visual Studi...
Dart Code

Git History 0.4.6
View git log, file history, compare branches or...
Don Jayamanne

Git Project Manager 1.7.1
Allows you to change easily between git proje...
Felipe Caputo

PHP IntelliSense 2.3.10
Advanced Autocompletion and Refactoring s...
Felix Becker

markdownlint 0.26.0
Markdown linting and style checking for Visu...
David Anson

SQL Server (mssql) 1.5.0
Develop Microsoft SQL Server, Azure SQL Dat...
Microsoft

Azure Repos 1.149.2
Connect to Azure Repos and work with Git an...
Microsoft

...vaScript, TypeScript, Python, PHP, Azure, Docker and ...

...dings
...nd keyboard shortcuts of Vim, Sublime, Atom and oth...

...l your code look the way you love

More... (Ctrl+R)

Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow

☑ Show welcome page on startup

Learn

Find and run all commands
Rapidly access and search commands from the Command Palette (Ctrl+Sh...

Interface overview
Get a visual overlay highlighting the major components of the UI

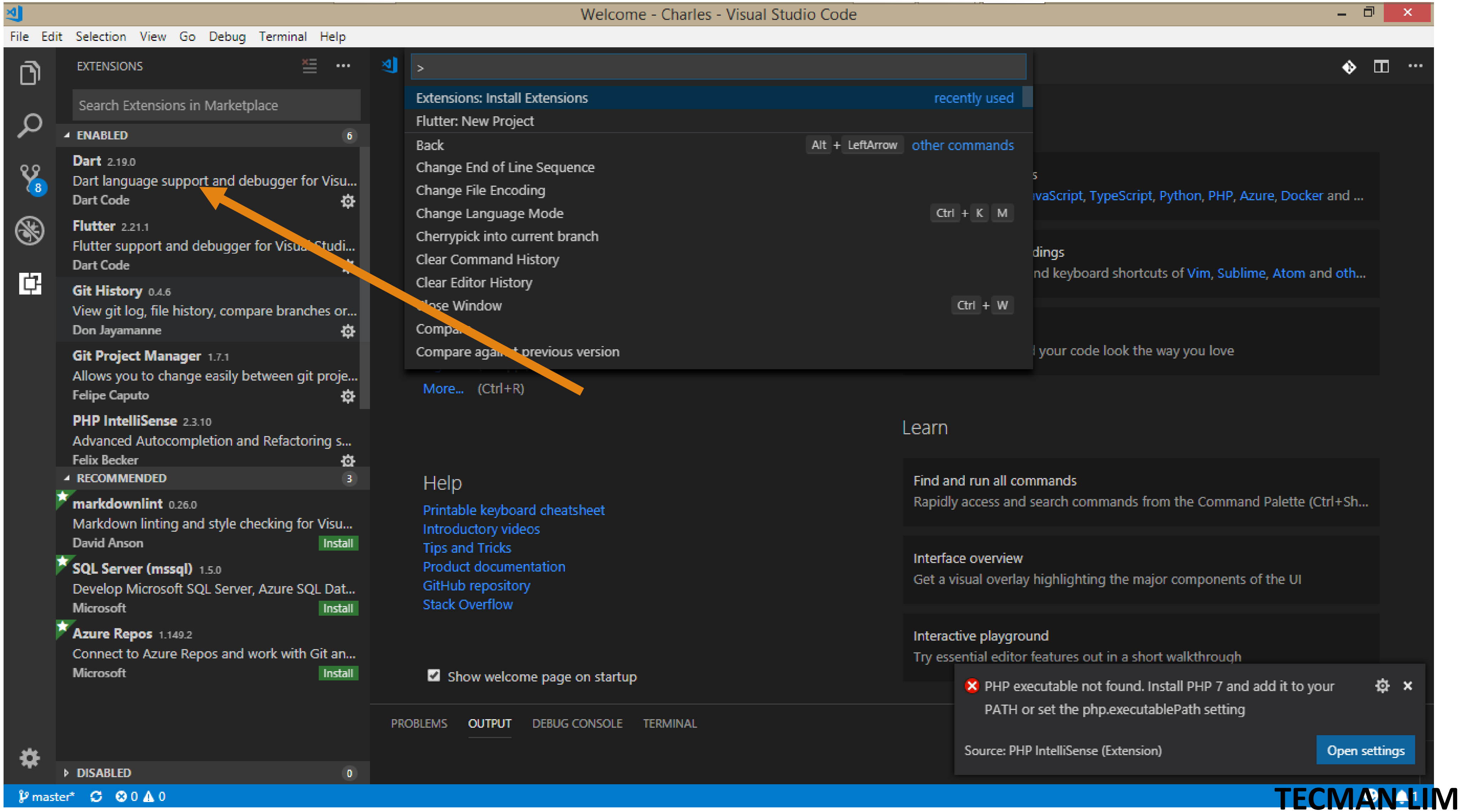
Interactive playground
Try essential editor features out in a short walkthrough

PHP executable not found. Install PHP 7 and add it to your PATH or set the php.executablePath setting

Source: PHP IntelliSense (Extension)

TECMAN LIMITED 2019

- Select the Dart plugin on the left pane of the VSCODE.



Welcome - Charles - Visual Studio Code

File Edit Selection View Go Debug Terminal Help

EXTENSIONS

Search Extensions in Marketplace

ENABLED 6

Dart 2.19.0
Dart language support and debugger for Visu...
Dart Code

Flutter 2.21.1
Flutter support and debugger for Visual Studi...
Dart Code

Git History 0.4.6
View git log, file history, compare branches or...
Don Jayamanne

Git Project Manager 1.7.1
Allows you to change easily between git proje...
Felipe Caputo

PHP IntelliSense 2.3.10
Advanced Autocompletion and Refactoring s...
Felix Becker

RECOMMENDED 3

markdownlint 0.26.0
Markdown linting and style checking for Visu...
David Anson Install

SQL Server (mssql) 1.5.0
Develop Microsoft SQL Server, Azure SQL Dat...
Microsoft Install

Azure Repos 1.149.2
Connect to Azure Repos and work with Git an...
Microsoft Install

DISABLED 0

>

Extensions: Install Extensions — recently used
Flutter: New Project
Back — Alt + LeftArrow — other commands
Change End of Line Sequence
Change File Encoding
Change Language Mode — Ctrl + K M
Cherrypick into current branch
Clear Command History
Clear Editor History
Close Window — Ctrl + W
Compa...
Compare against previous version

More... (Ctrl+R)

...vaScript, TypeScript, Python, PHP, Azure, Docker and ...

...dings
...nd keyboard shortcuts of Vim, Sublime, Atom and oth...

...your code look the way you love

Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow

☑ Show welcome page on startup

Learn

Find and run all commands
Rapidly access and search commands from the Command Palette (Ctrl+Sh...

Interface overview
Get a visual overlay highlighting the major components of the UI

Interactive playground
Try essential editor features out in a short walkthrough

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PHP executable not found. Install PHP 7 and add it to your PATH or set the php.executablePath setting

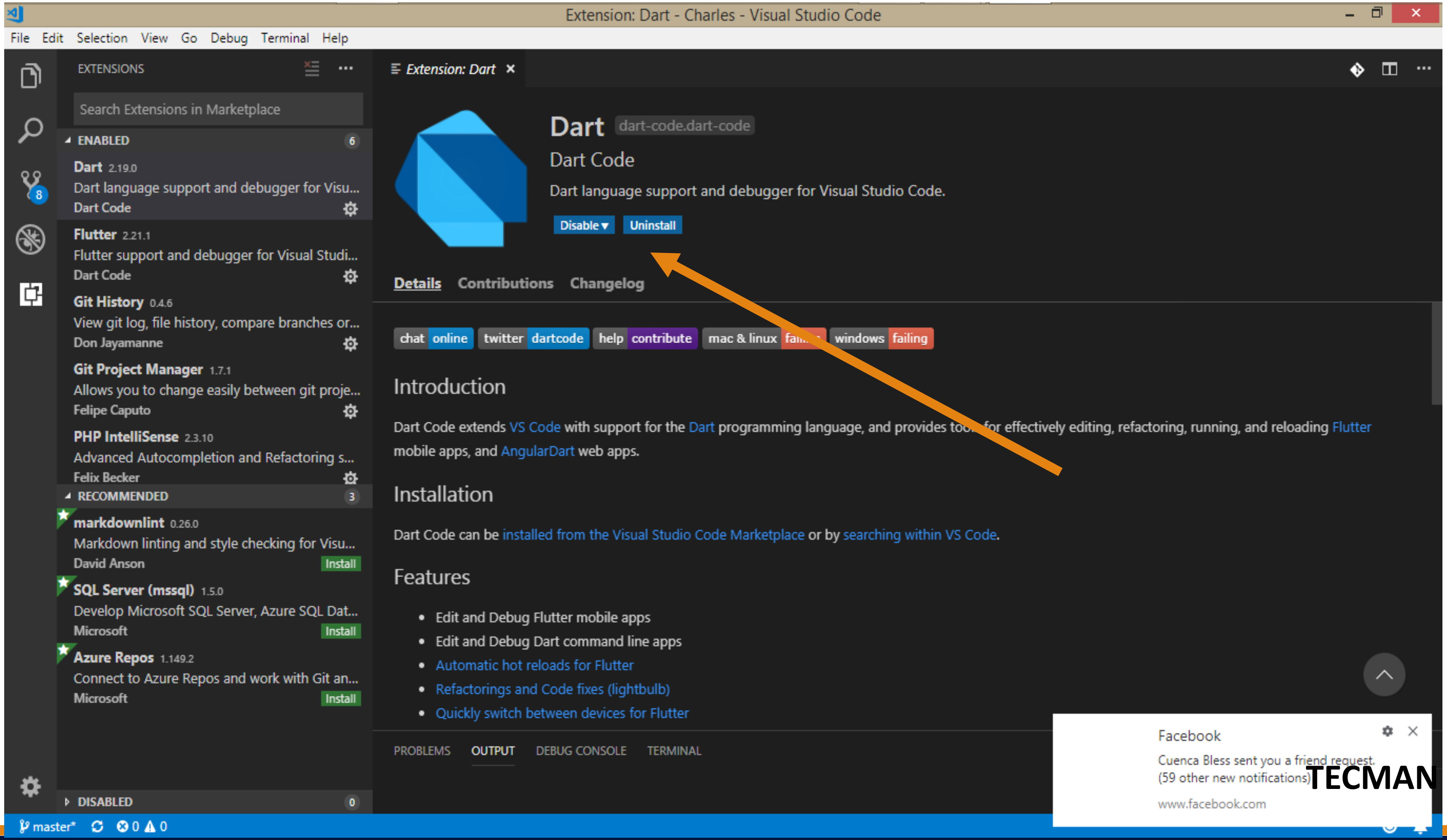
Source: PHP IntelliSense (Extension) — Open settings

master* ⊗ 0 ⚠ 0

TECMAN LIMITED 2019

- Next, click on the install button to install the Dart plugin.

(At the time of this tutorial, the dart plugin was already installed. You could see there is no install button on the image below. However, you should be able to see it on your screen if you are now about to install.)



Extension: Dart - Charles - Visual Studio Code

File Edit Selection View Go Debug Terminal Help

EXTENSIONS

Search Extensions in Marketplace

ENABLED 6

Dart 2.19.0
Dart language support and debugger for Visu...
Dart Code

Flutter 2.21.1
Flutter support and debugger for Visual Studi...
Dart Code

Git History 0.4.6
View git log, file history, compare branches or...
Don Jayamanne

Git Project Manager 1.7.1
Allows you to change easily between git proje...
Felipe Caputo

PHP IntelliSense 2.3.10
Advanced Autocompletion and Refactoring s...
Felix Becker

RECOMMENDED 3

markdownlint 0.26.0
Markdown linting and style checking for Visu...
David Anson Install

SQL Server (mssql) 1.5.0
Develop Microsoft SQL Server, Azure SQL Dat...
Microsoft Install

Azure Repos 1.149.2
Connect to Azure Repos and work with Git an...
Microsoft Install

DISABLED 0

Extension: Dart ×

Dart dart-code.dart-code
Dart Code
Dart language support and debugger for Visual Studio Code.
Disable ▾ Uninstall

Details Contributions Changelog

chat online twitter dartcode help contribute mac & linux failing windows failing

Introduction

Dart Code extends VS Code with support for the Dart programming language, and provides tools for effectively editing, refactoring, running, and reloading Flutter mobile apps, and AngularDart web apps.

Installation

Dart Code can be installed from the Visual Studio Code Marketplace or by searching within VS Code.

Features

- Edit and Debug Flutter mobile apps
- Edit and Debug Dart command line apps
- Automatic hot reloads for Flutter
- Refactorings and Code fixes (lightbulb)
- Quickly switch between devices for Flutter

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

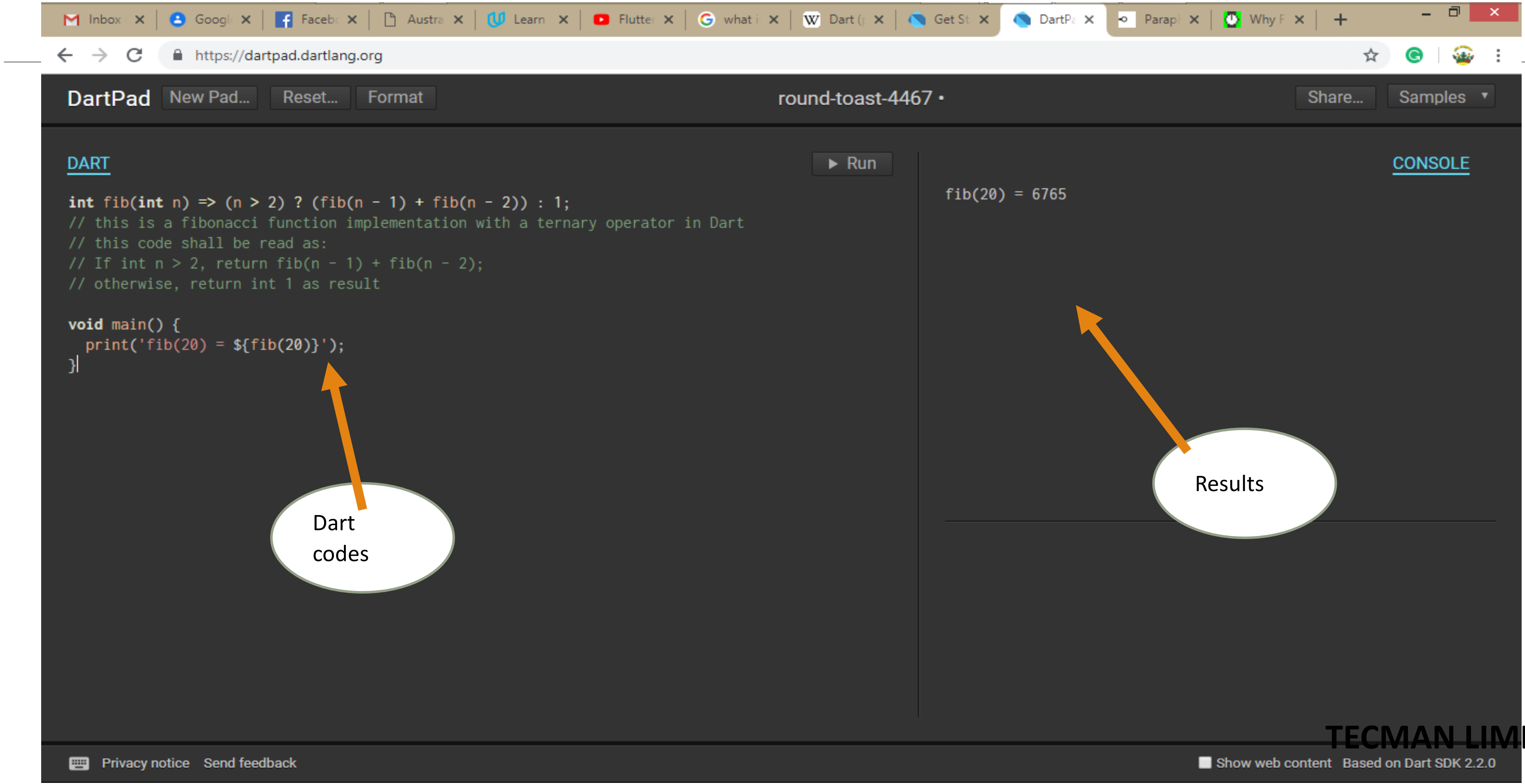
Facebook
Cuenca Bless sent you a friend request.
(59 other new notifications)
www.facebook.com

master* ⊗ 0 ⚠ 0

TECMAN LIMITED 2019

Writing your first Dart program

This is a Fibonacci function implementation with a ternary operator in Dart



```
int fib(int n) => (n > 2) ? (fib(n - 1) + fib(n - 2)) : 1;
// this is a fibonacci function implementation with a ternary operator in Dart
// this code shall be read as:
// If int n > 2, return fib(n - 1) + fib(n - 2);
// otherwise, return int 1 as result

void main() {
  print('fib(20) = ${fib(20)}');
}
```

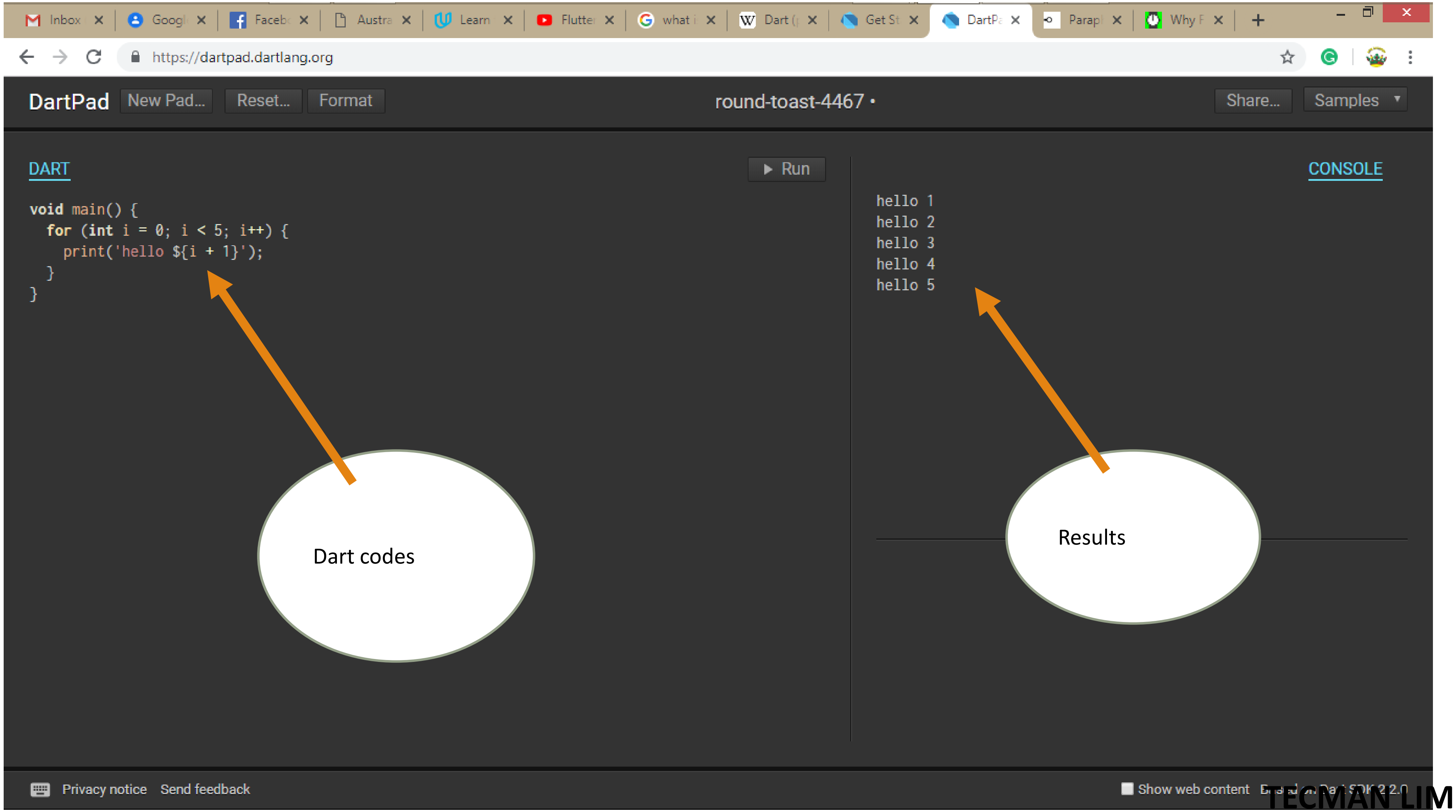
```
fib(20) = 6765
```

Dart codes

Results

TECMAN LIMITED 2019

A simple for loop function



```
void main() {
  for (int i = 0; i < 5; i++) {
    print('hello ${i + 1}');
  }
}
```

```
hello 1
hello 2
hello 3
hello 4
hello 5
```

TECMAN LIMITED 2019

Setting up Flutter.

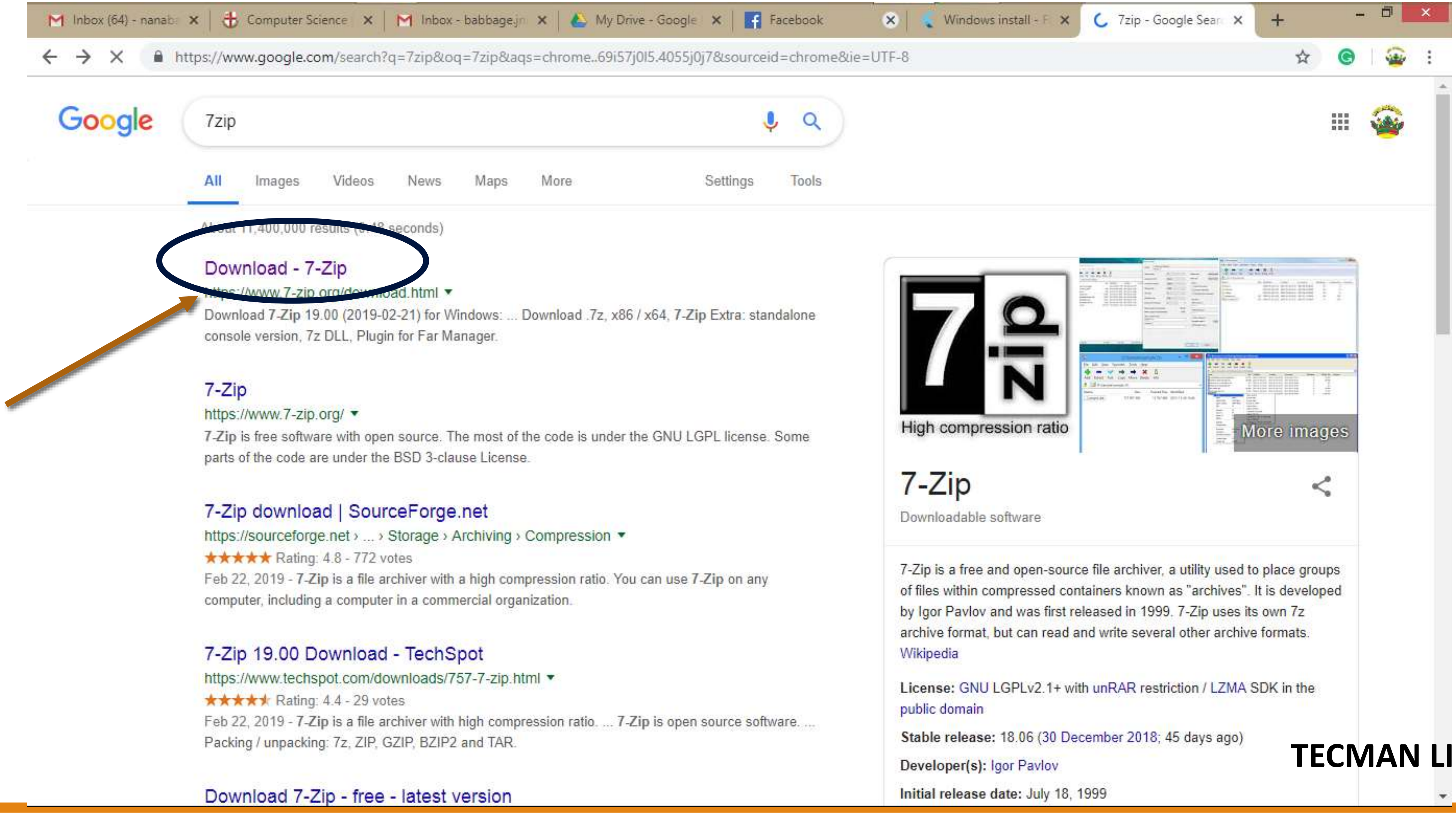
Things you need,

- Windows 7 and above.
- At least 400 MB of free disk space
- 7zip
- Visual Studio Code
- Dart SDK
- Flutter SDK
- Git for Windows

TECMAN LIMITED 2019

Installing 7zip

- Visit https://www.7-zip.org/download.html
- Download and install 7zip



https://www.google.com/search?q=7zip&oq=7zip&aqs=chrome..69i57j0l5.4055j0j7&sourceid=chrome&ie=UTF-8

About 11,400,000 results (0.48 seconds)

Download - 7-Zip
https://www.7-zip.org/download.html ▾
Download 7-Zip 19.00 (2019-02-21) for Windows: ... Download .7z, x86 / x64, 7-Zip Extra: standalone console version, 7z DLL, Plugin for Far Manager.

7-Zip
https://www.7-zip.org/ ▾
7-Zip is free software with open source. The most of the code is under the GNU LGPL license. Some parts of the code are under the BSD 3-clause License.

7-Zip download | SourceForge.net
https://sourceforge.net › ... › Storage › Archiving › Compression ▾
★★★★★ Rating: 4.8 - 772 votes
Feb 22, 2019 - 7-Zip is a file archiver with a high compression ratio. You can use 7-Zip on any computer, including a computer in a commercial organization.

7-Zip 19.00 Download - TechSpot
https://www.techspot.com/downloads/757-7-zip.html ▾
★★★★★ Rating: 4.4 - 29 votes
Feb 22, 2019 - 7-Zip is a file archiver with high compression ratio. ... 7-Zip is open source software. ... Packing / unpacking: 7z, ZIP, GZIP, BZIP2 and TAR.

Download 7-Zip - free - latest version

7-Zip
Downloadable software

7-Zip is a free and open-source file archiver, a utility used to place groups of files within compressed containers known as "archives". It is developed by Igor Pavlov and was first released in 1999. 7-Zip uses its own 7z archive format, but can read and write several other archive formats. Wikipedia

License: GNU LGPLv2.1+ with unRAR restriction / LZMA SDK in the public domain

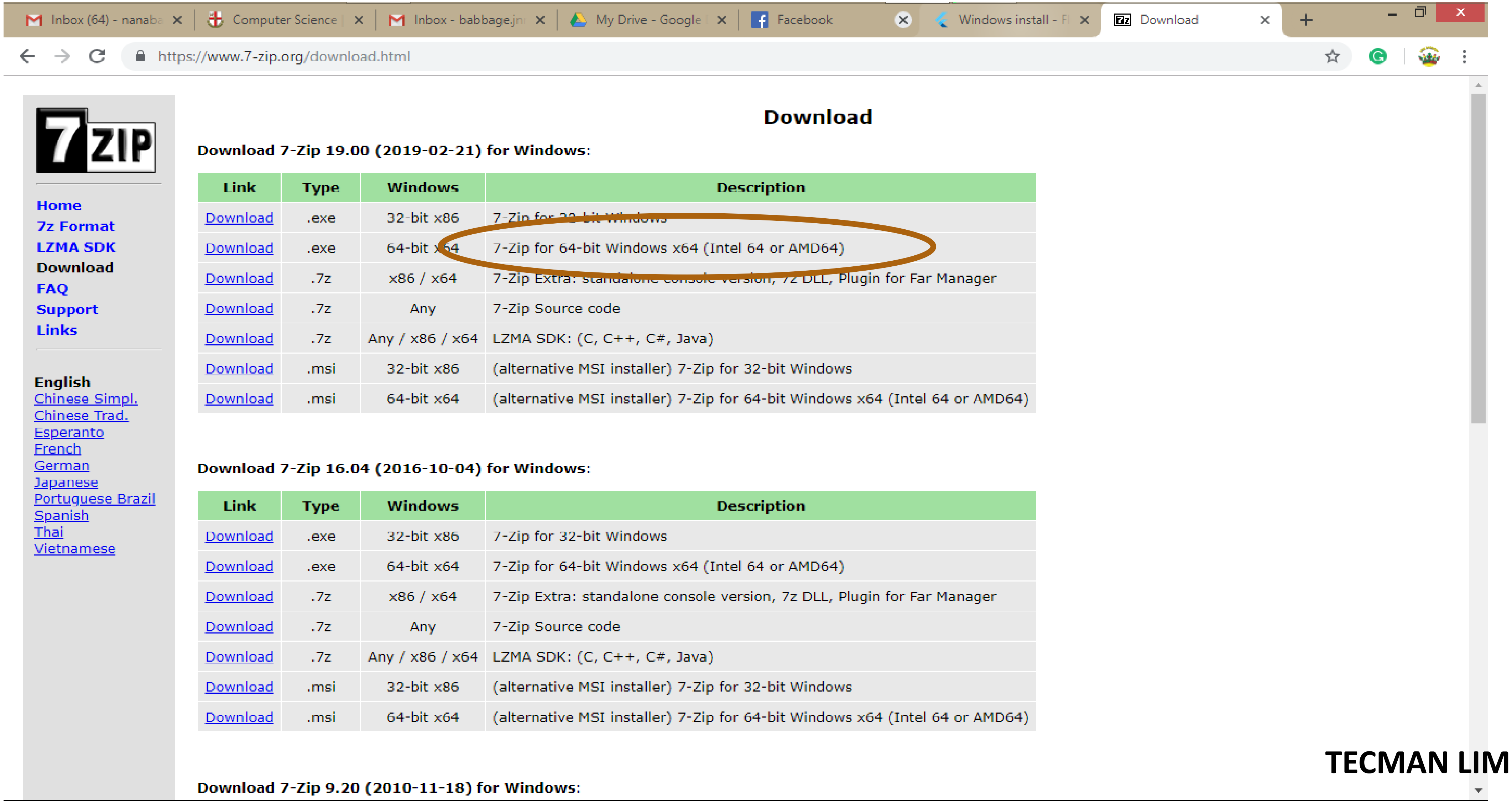
Stable release: 18.06 (30 December 2018; 45 days ago)

Developer(s): Igor Pavlov

Initial release date: July 18, 1999

TECMAN LIMITED 2019

- Select the one compatible with your system.



Download

Download 7-Zip 19.00 (2019-02-21) for Windows:

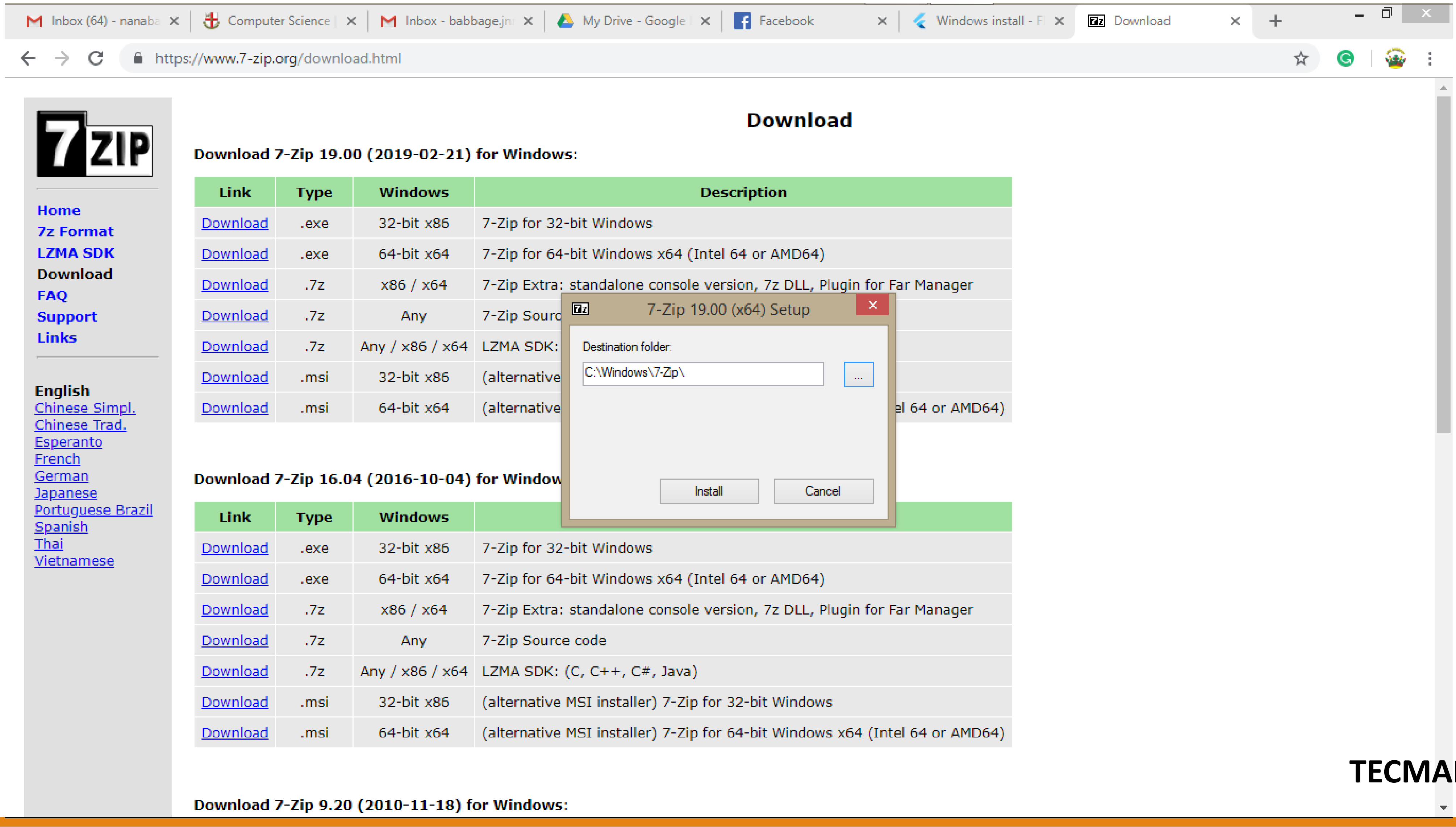
Link	Type	Windows	Description
Download	.exe	32-bit x86	7-Zip for 32-bit Windows
Download	.exe	64-bit x64	7-Zip for 64-bit Windows x64 (Intel 64 or AMD64)
Download	.7z	x86 / x64	7-Zip Extra: standalone console version, 7z DLL, Plugin for Far Manager
Download	.7z	Any	7-Zip Source code
Download	.7z	Any / x86 / x64	LZMA SDK: (C, C++, C#, Java)
Download	.msi	32-bit x86	(alternative MSI installer) 7-Zip for 32-bit Windows
Download	.msi	64-bit x64	(alternative MSI installer) 7-Zip for 64-bit Windows x64 (Intel 64 or AMD64)

Download 7-Zip 16.04 (2016-10-04) for Windows:

Link	Type	Windows	Description
Download	.exe	32-bit x86	7-Zip for 32-bit Windows
Download	.exe	64-bit x64	7-Zip for 64-bit Windows x64 (Intel 64 or AMD64)
Download	.7z	x86 / x64	7-Zip Extra: standalone console version, 7z DLL, Plugin for Far Manager
Download	.7z	Any	7-Zip Source code
Download	.7z	Any / x86 / x64	LZMA SDK: (C, C++, C#, Java)
Download	.msi	32-bit x86	(alternative MSI installer) 7-Zip for 32-bit Windows
Download	.msi	64-bit x64	(alternative MSI installer) 7-Zip for 64-bit Windows x64 (Intel 64 or AMD64)

Download 7-Zip 9.20 (2010-11-18) for Windows:

- Start the installation. (In downloads folder, double click the file setup to start installation.)



https://www.7-zip.org/download.html

Download

Download 7-Zip 19.00 (2019-02-21) for Windows:

Link	Type	Windows	Description
Download	.exe	32-bit x86	7-Zip for 32-bit Windows
Download	.exe	64-bit x64	7-Zip for 64-bit Windows x64 (Intel 64 or AMD64)
Download	.7z	x86 / x64	7-Zip Extra: standalone console version, 7z DLL, Plugin for Far Manager
Download	.7z	Any	7-Zip Sourc
Download	.7z	Any / x86 / x64	LZMA SDK:
Download	.msi	32-bit x86	(alternative
Download	.msi	64-bit x64	(alternative el 64 or AMD64)

7-Zip 19.00 (x64) Setup

Destination folder:

C:\Windows\7-Zip\

...

Install Cancel

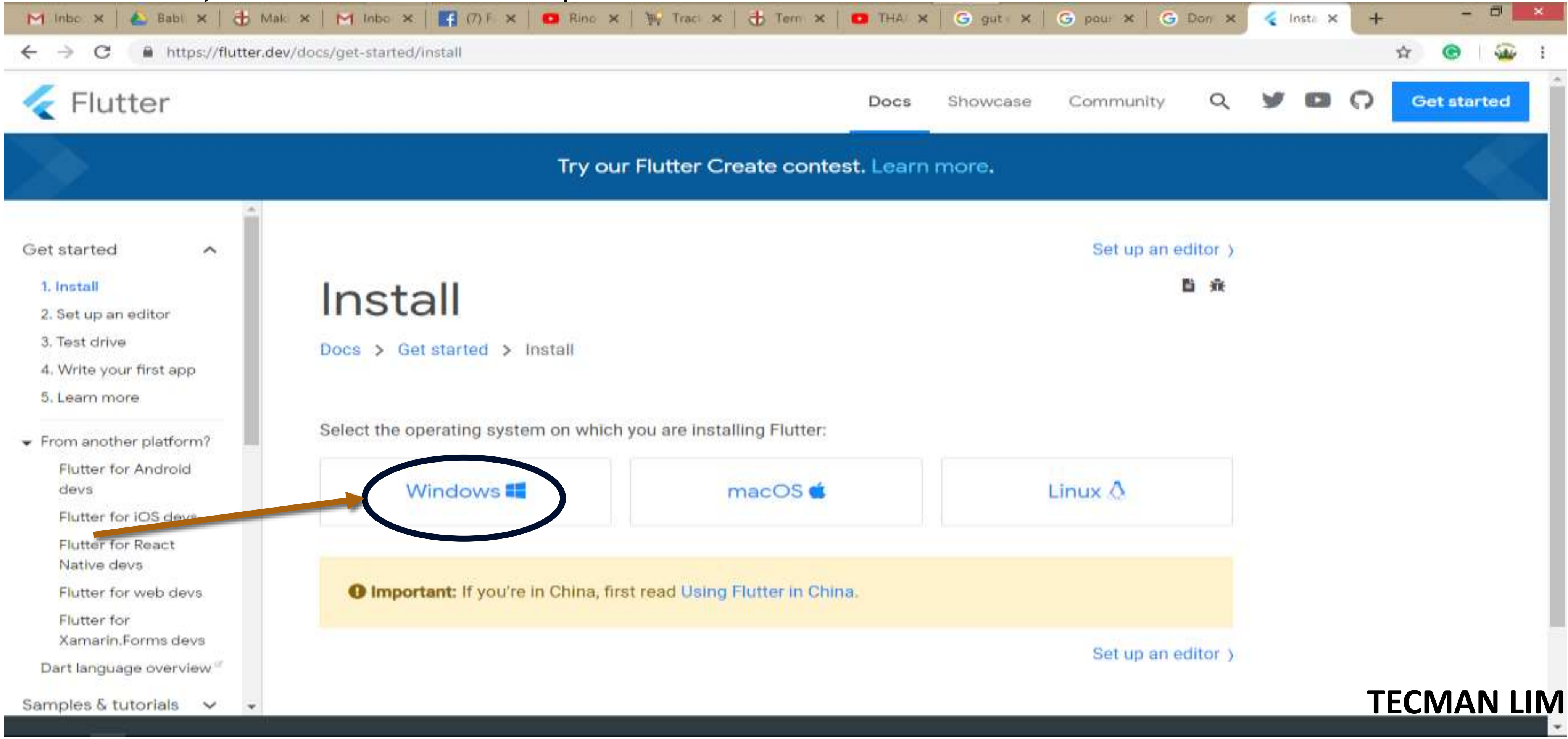
Download 7-Zip 16.04 (2016-10-04) for Window

Link	Type	Windows	
Download	.exe	32-bit x86	7-Zip for 32-bit Windows
Download	.exe	64-bit x64	7-Zip for 64-bit Windows x64 (Intel 64 or AMD64)
Download	.7z	x86 / x64	7-Zip Extra: standalone console version, 7z DLL, Plugin for Far Manager
Download	.7z	Any	7-Zip Source code
Download	.7z	Any / x86 / x64	LZMA SDK: (C, C++, C#, Java)
Download	.msi	32-bit x86	(alternative MSI installer) 7-Zip for 32-bit Windows
Download	.msi	64-bit x64	(alternative MSI installer) 7-Zip for 64-bit Windows x64 (Intel 64 or AMD64)

Download 7-Zip 9.20 (2010-11-18) for Windows:

Installing the Flutter SDK

- Visit https://flutter.dev/docs/get-started/install/windows
- Download the flutter windows SDK. (You can get the flutter SDK for windows from flutter's official website.)
- From the website, click on the windows option.



Try our Flutter Create contest. Learn more.

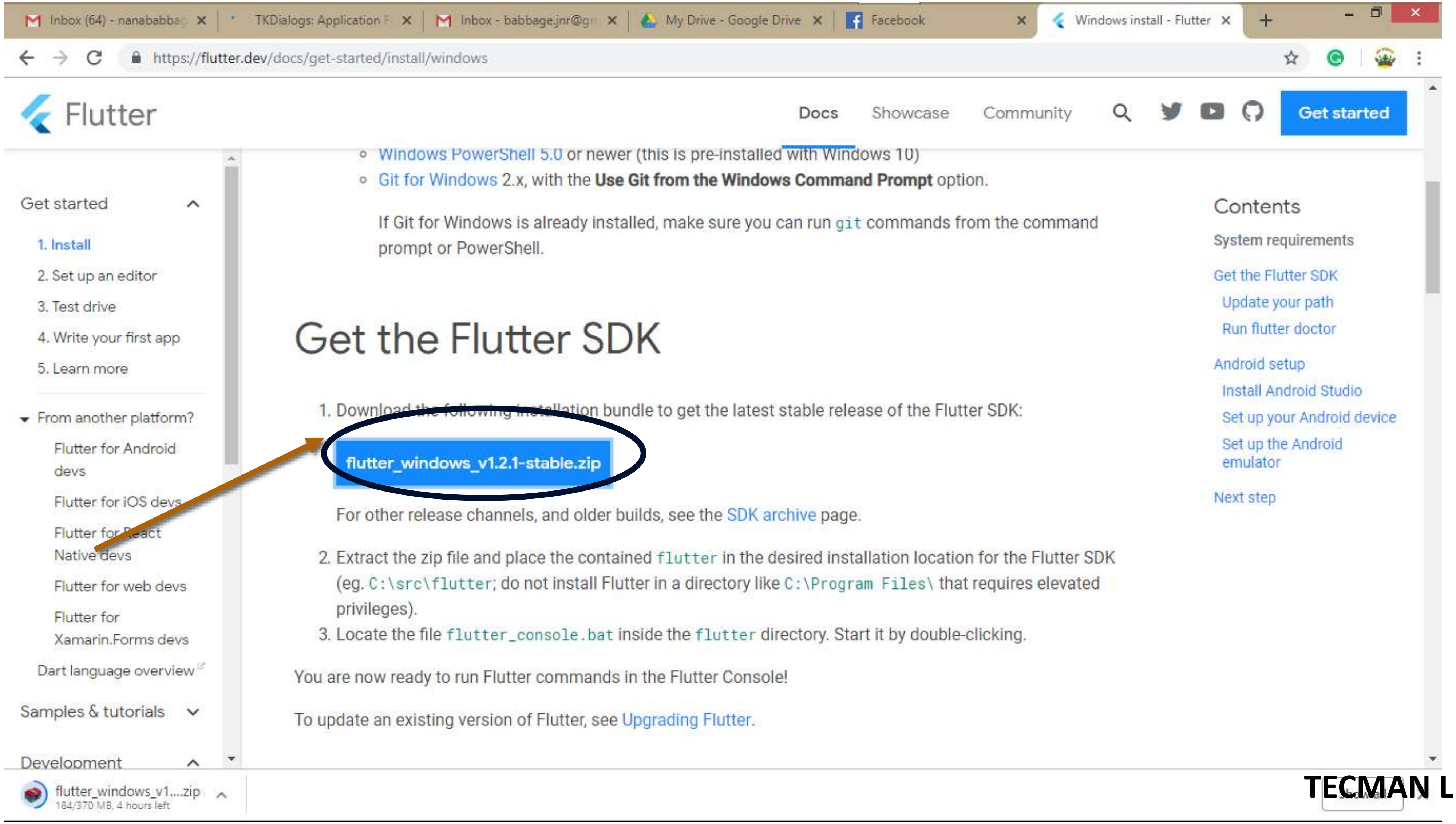
Install

Select the operating system on which you are installing Flutter:

Important: If you're in China, first read Using Flutter in China.

TECMAN LIMITED 2019

- Next, click on the button showing **“flutter_windows_v1.2.1-stable.zip”**



- Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)
- Git for Windows 2.x, with the **Use Git from the Windows Command Prompt** option.

  If Git for Windows is already installed, make sure you can run `git` commands from the command prompt or PowerShell.

Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:

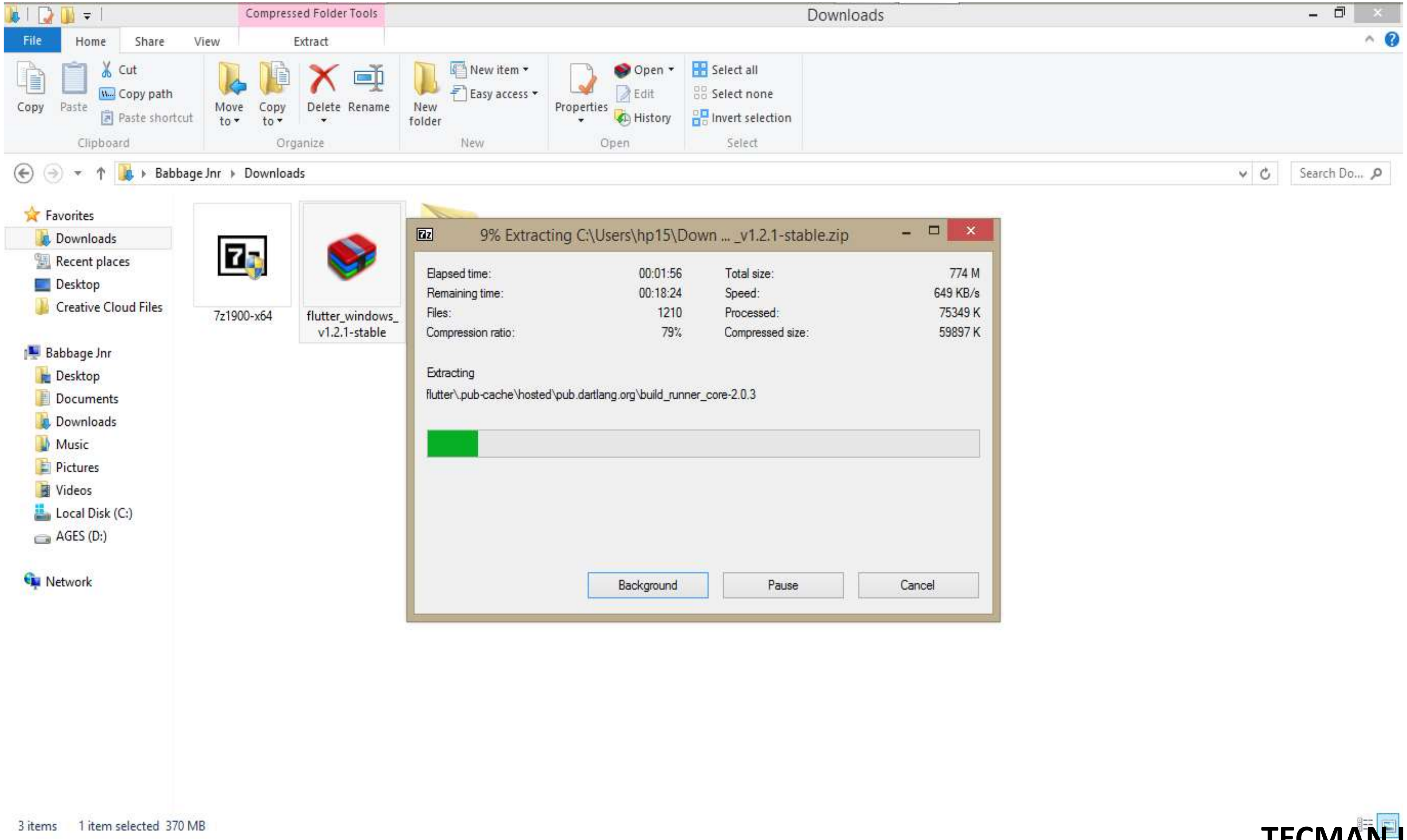
   flutter_windows_v1.2.1-stable.zip

   For other release channels, and older builds, see the SDK archive page.
2. Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (eg. `C:\src\flutter`; do not install Flutter in a directory like `C:\Program Files\` that requires elevated privileges).
3. Locate the file `flutter_console.bat` inside the `flutter` directory. Start it by double-clicking.

You are now ready to run Flutter commands in the Flutter Console!

To update an existing version of Flutter, see Upgrading Flutter.

- Extract the zip with 7zip and copy the flutter folder to C:\Windows



9% Extracting C:\Users\hp15\Down ... _v1.2.1-stable.zip

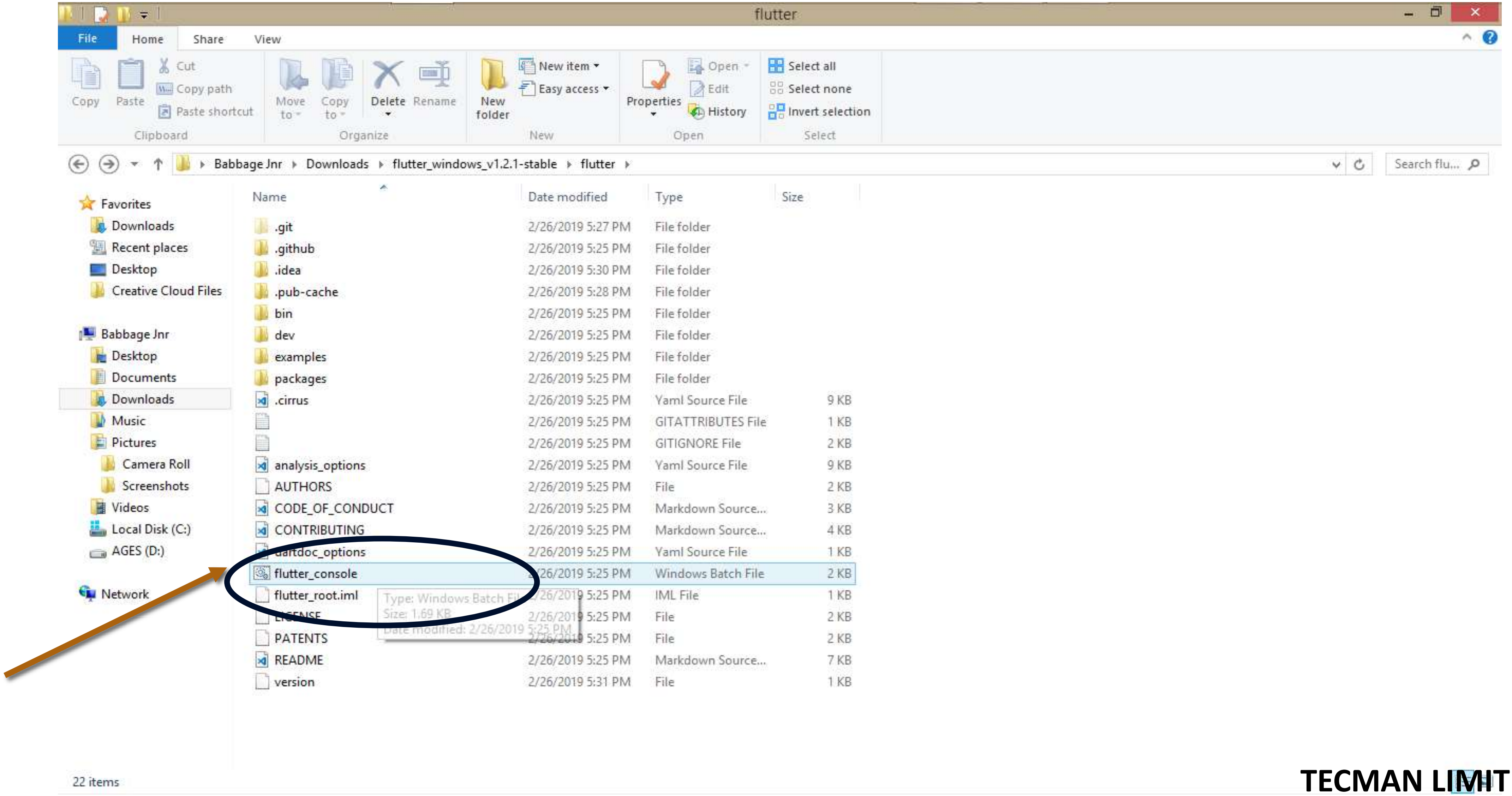
```
Elapsed time:        00:01:56     Total size:           774 M
Remaining time:      00:18:24     Speed:             649 KB/s
Files:                   1210     Processed:          75349 K
Compression ratio:        79%     Compressed size:    59897 K
```

Extracting

flutter\.pub-cache\hosted\pub.dartlang.org\build_runner_core-2.0.3

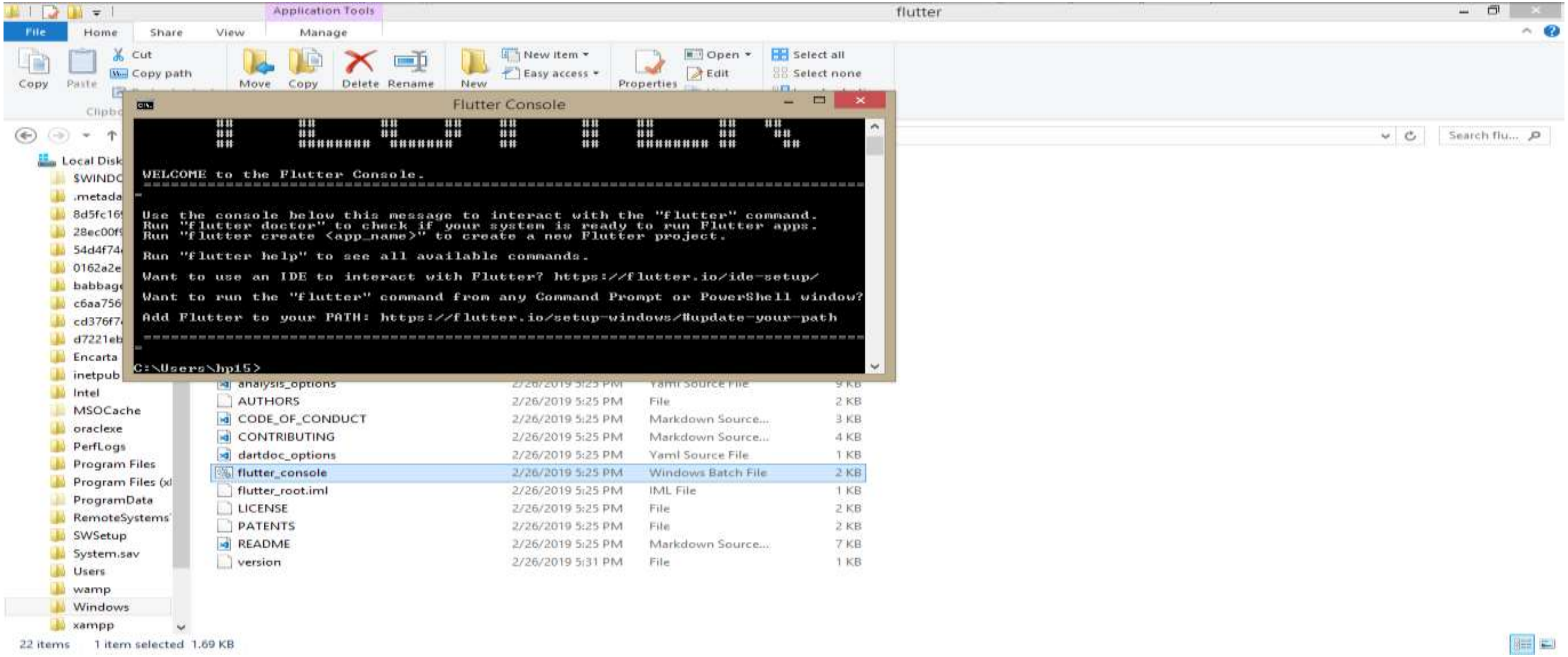
3 items   1 item selected 370 MB

- Open the flutter folder and locate the file flutter_console.bat
- Double click on flutter_console.bat



Name	Date modified	Type	Size
.git	2/26/2019 5:27 PM	File folder	
.github	2/26/2019 5:25 PM	File folder	
.idea	2/26/2019 5:30 PM	File folder	
.pub-cache	2/26/2019 5:28 PM	File folder	
bin	2/26/2019 5:25 PM	File folder	
dev	2/26/2019 5:25 PM	File folder	
examples	2/26/2019 5:25 PM	File folder	
packages	2/26/2019 5:25 PM	File folder	
.cirrus	2/26/2019 5:25 PM	Yaml Source File	9 KB
	2/26/2019 5:25 PM	GITATTRIBUTES File	1 KB
	2/26/2019 5:25 PM	GITIGNORE File	2 KB
analysis_options	2/26/2019 5:25 PM	Yaml Source File	9 KB
AUTHORS	2/26/2019 5:25 PM	File	2 KB
CODE_OF_CONDUCT	2/26/2019 5:25 PM	Markdown Source...	3 KB
CONTRIBUTING	2/26/2019 5:25 PM	Markdown Source...	4 KB
dartdoc_options	2/26/2019 5:25 PM	Yaml Source File	1 KB
flutter_console	2/26/2019 5:25 PM	Windows Batch File	2 KB
flutter_root.iml	2/26/2019 5:25 PM	IML File	1 KB
LICENSE	2/26/2019 5:25 PM	File	2 KB
PATENTS	2/26/2019 5:25 PM	File	2 KB
README	2/26/2019 5:25 PM	Markdown Source...	7 KB
version	2/26/2019 5:31 PM	File	1 KB

- Congrats, you are now ready to run flutter from the command console



```
WELCOME to the Flutter Console.
================================================================================

Use the console below this message to interact with the "flutter" command.
Run "flutter doctor" to check if your system is ready to run Flutter apps.
Run "flutter create <app_name>" to create a new Flutter project.

Run "flutter help" to see all available commands.

Want to use an IDE to interact with Flutter? https://flutter.io/ide-setup/

Want to run the "flutter" command from any Command Prompt or PowerShell window?
Add Flutter to your PATH: https://flutter.io/setup-windows/#update-your-path

================================================================================

C:\Users\hp15>
```

analysis_options	2/26/2019 5:25 PM	Yaml Source File	9 KB
AUTHORS	2/26/2019 5:25 PM	File	2 KB
CODE_OF_CONDUCT	2/26/2019 5:25 PM	Markdown Source...	3 KB
CONTRIBUTING	2/26/2019 5:25 PM	Markdown Source...	4 KB
dartdoc_options	2/26/2019 5:25 PM	Yaml Source File	1 KB
flutter_console	2/26/2019 5:25 PM	Windows Batch File	2 KB
flutter_root.iml	2/26/2019 5:25 PM	IML File	1 KB
LICENSE	2/26/2019 5:25 PM	File	2 KB
PATENTS	2/26/2019 5:25 PM	File	2 KB
README	2/26/2019 5:25 PM	Markdown Source...	7 KB
version	2/26/2019 5:31 PM	File	1 KB

- Verify your flutter SDK by typing flutter doctor

(This means that at any point in time if want to run flutter you need to run it from the C:\Windows\flutter what if you don’t want it that way? And you want to run it from the command console any time you want? Well there’s a way.)

TECMAN LIMITED 2019

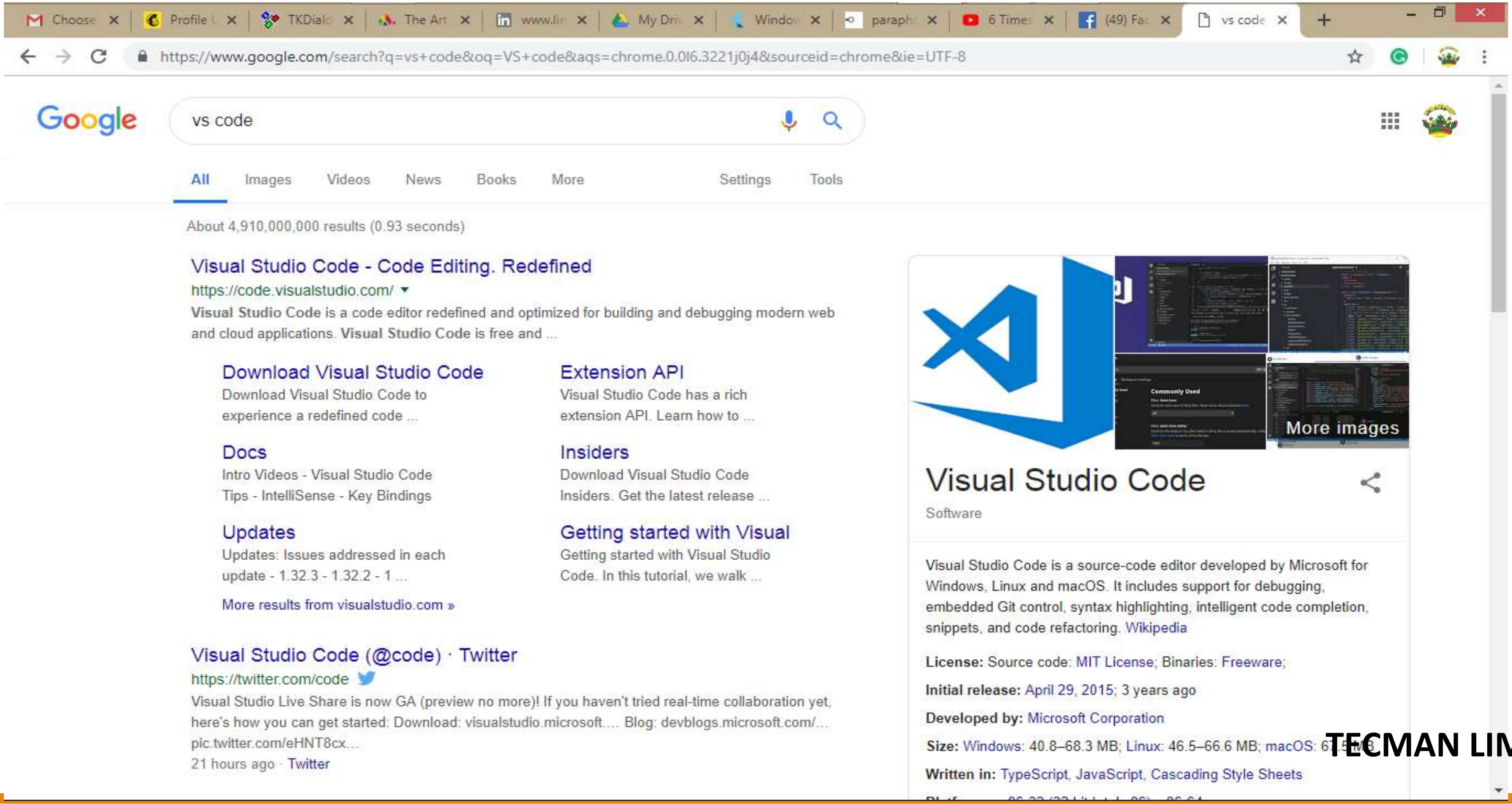
ADDING FLUTTER TO PATH

- Go to the folder where flutter SDK was copied and copy the path *C:\Windows\flutter\bin*
- Click on Start and type Control Panel
- Open Control Panel and click on System and Security
- Click on System, from the left pane click on Advance System Settings
- Locate Environment Variables…
- Under User Variable locate the Path under Variable click on Edit
- Now scroll to the last empty path and double click
- Paste the path which was copied to clipboard from step one
- Click OK to save the changes
- Voila, now flutter is ready to run anywhere from the command console
- Verify again by typing command prompt from Start
- Now run ‘flutter doctor’

SETTING UP VS CODE

Flutter can be built using any IDE of choice but for the purpose of this training, VS Code will be used as a choice of IDE.

- Download and Install VS Code
- VS Code 1.31



https://www.google.com/search?q=vs+code&oq=VS+code&aqs=chrome.0.0l6.3221j0j4&sourceid=chrome&ie=UTF-8

Google vs code

About 4,910,000,000 results (0.93 seconds)

Visual Studio Code - Code Editing. Redefined
https://code.visualstudio.com/
Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications. Visual Studio Code is free and ...

Download Visual Studio Code
Download Visual Studio Code to experience a redefined code ...

Extension API
Visual Studio Code has a rich extension API. Learn how to ...

Docs
Intro Videos - Visual Studio Code Tips - IntelliSense - Key Bindings

Insiders
Download Visual Studio Code Insiders. Get the latest release ...

Updates
Updates: Issues addressed in each update - 1.32.3 - 1.32.2 - 1 ...

Getting started with Visual
Getting started with Visual Studio Code. In this tutorial, we walk ...

More results from visualstudio.com »

Visual Studio Code (@code) · Twitter
https://twitter.com/code
Visual Studio Live Share is now GA (preview no more)! If you haven't tried real-time collaboration yet, here's how you can get started: Download: visualstudio.microsoft.... Blog: devblogs.microsoft.com/... pic.twitter.com/eHNT8cx...
21 hours ago · Twitter

Visual Studio Code
Software

Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. Wikipedia

License: Source code: MIT License; Binaries: Freeware;

Initial release: April 29, 2015; 3 years ago

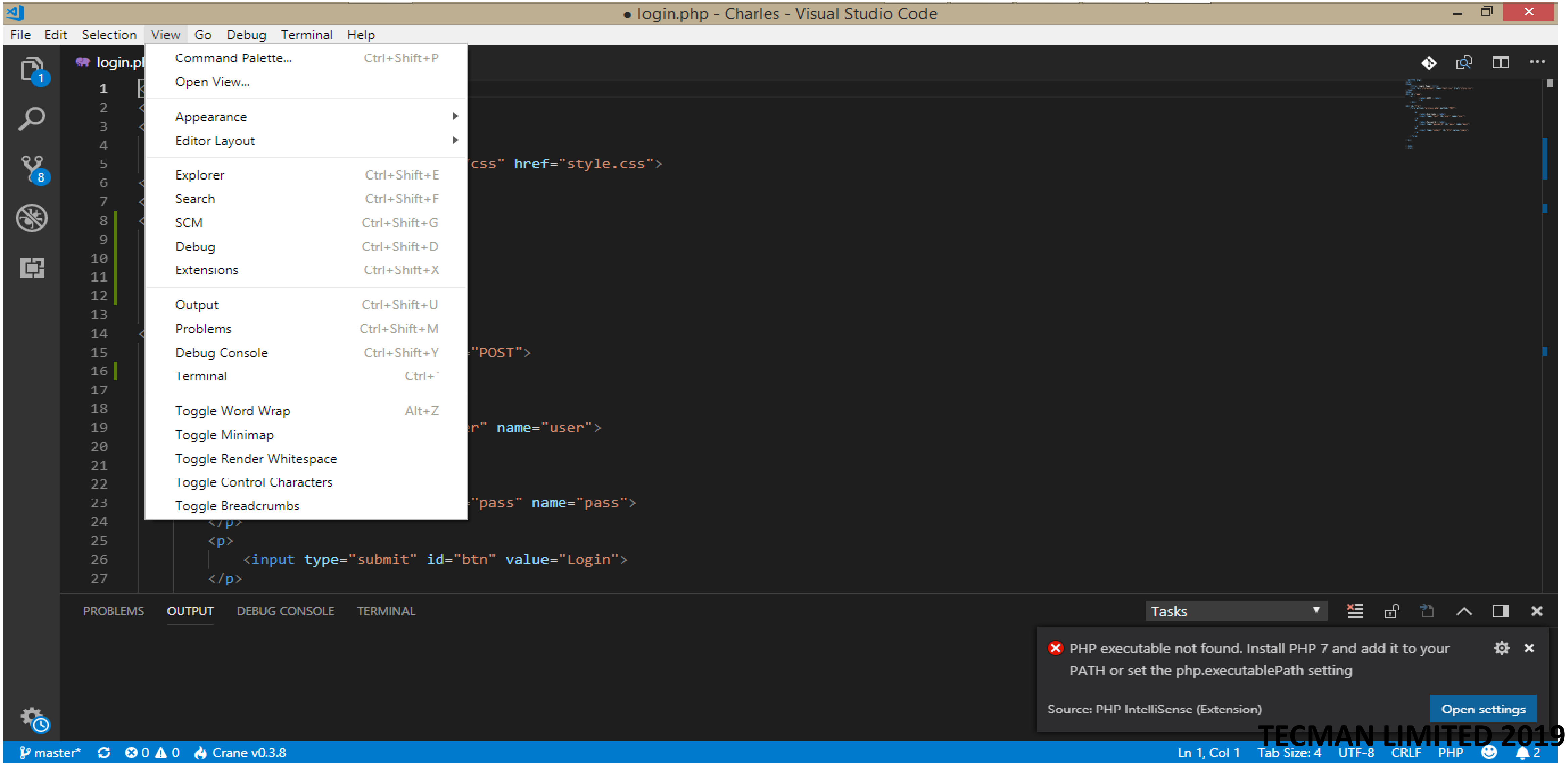
Developed by: Microsoft Corporation

Size: Windows: 40.8–68.3 MB; Linux: 46.5–66.6 MB; macOS: 67.5 MB

Written in: TypeScript, JavaScript, Cascading Style Sheets

TECMAN LIMITED 2019

- Open VS Code and Install flutter plugins
- Go to View -> Command Palette….

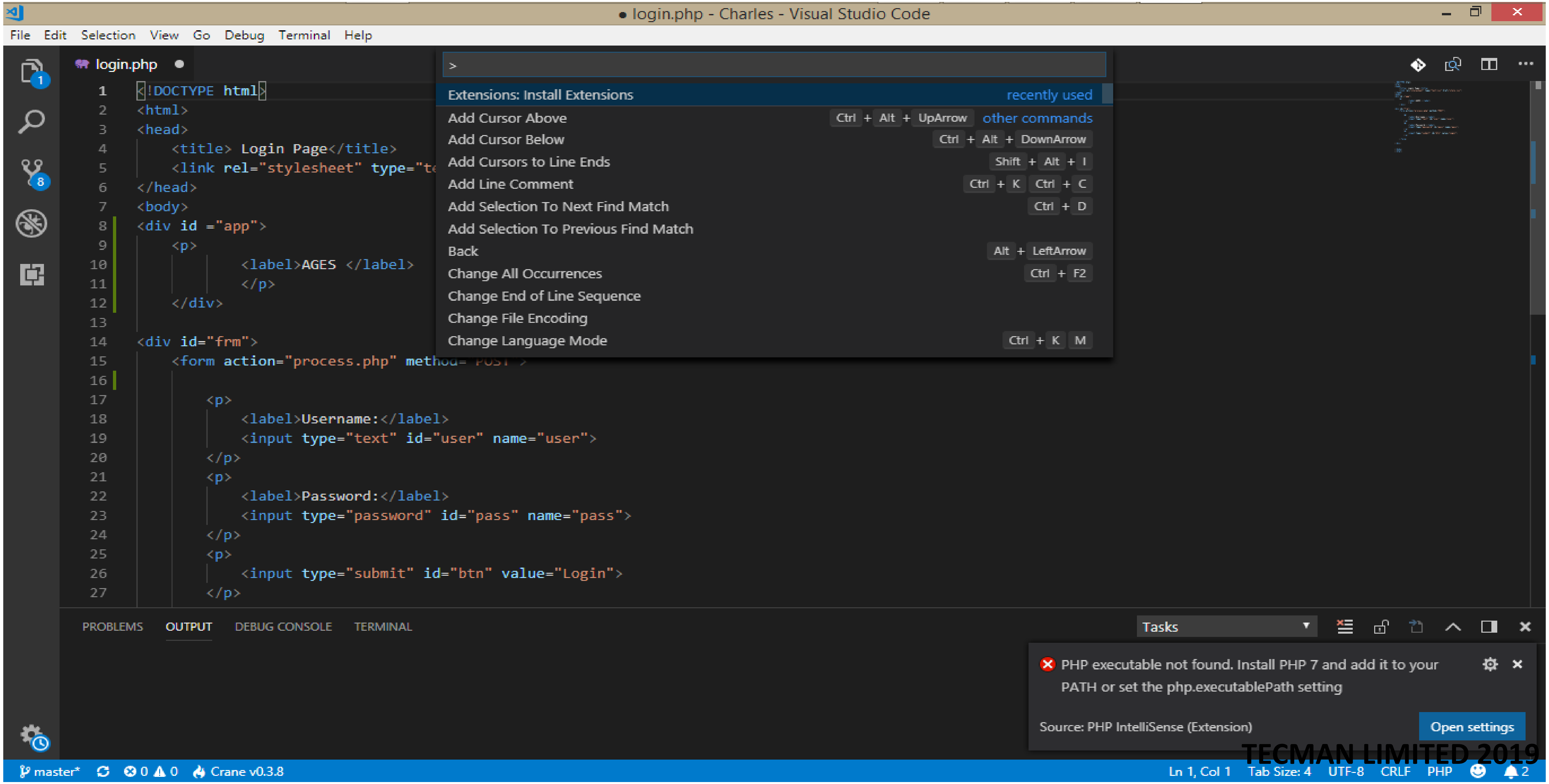


```
css" href="style.css">
"POST">
r" name="user">
"pass" name="pass">
</p>
<p>
    <input type="submit" id="btn" value="Login">
</p>
```

PHP executable not found. Install PHP 7 and add it to your PATH or set the php.executablePath setting

Source: PHP IntelliSense (Extension)

- Start typing install and select Extension: Install Extensions



```
>
```

Extensions: Install Extensions — recently used
Add Cursor Above — Ctrl + Alt + UpArrow — other commands
Add Cursor Below — Ctrl + Alt + DownArrow
Add Cursors to Line Ends — Shift + Alt + I
Add Line Comment — Ctrl + K Ctrl + C
Add Selection To Next Find Match — Ctrl + D
Add Selection To Previous Find Match
Back — Alt + LeftArrow
Change All Occurrences — Ctrl + F2
Change End of Line Sequence
Change File Encoding
Change Language Mode — Ctrl + K M

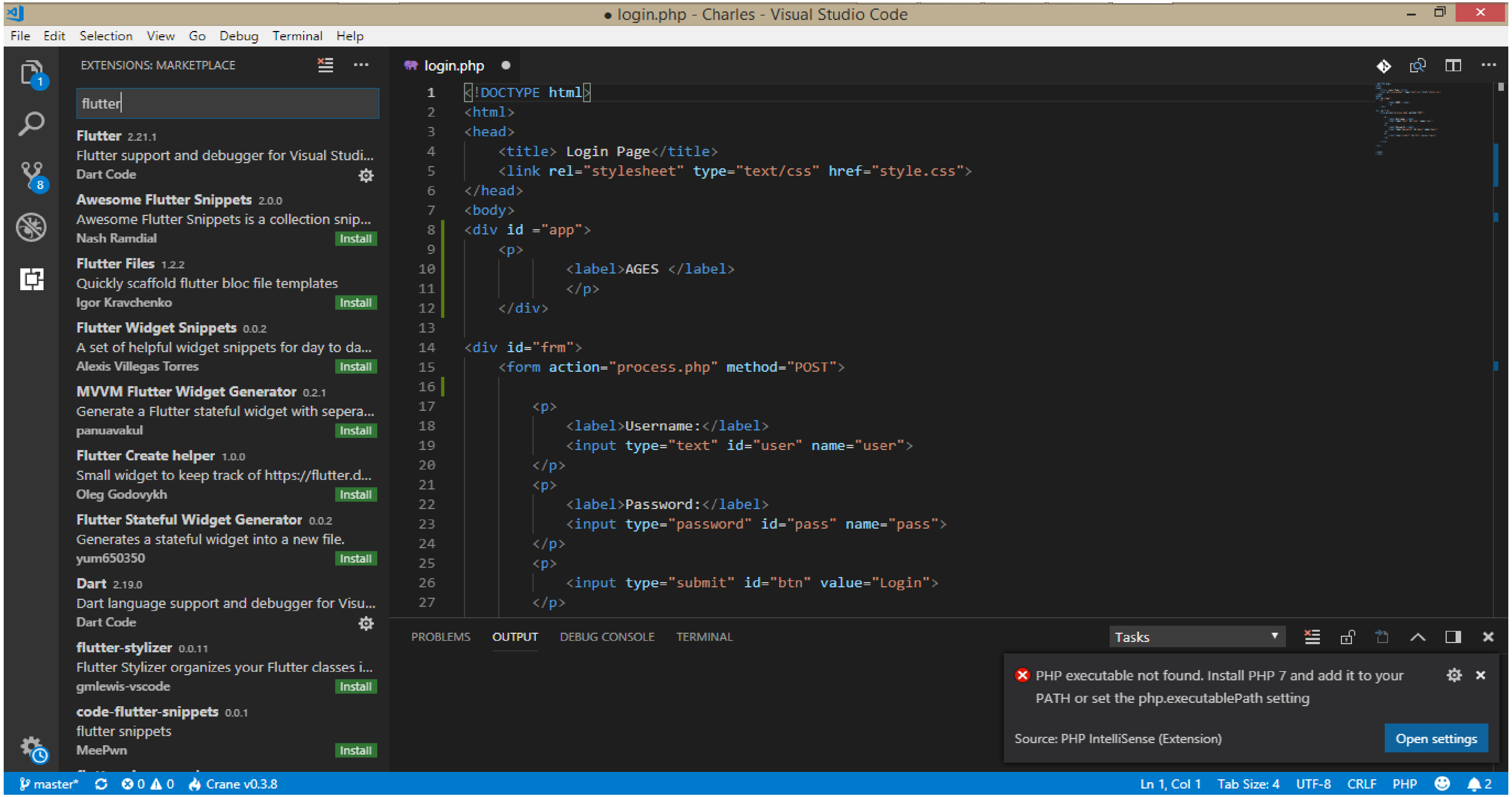
```
<!DOCTYPE html>
<html>
<head>
    <title> Login Page</title>
    <link rel="stylesheet" type="te
</head>
<body>
<div id ="app">
    <p>
            <label>AGES </label>
            </p>
    </div>

<div id="frm">
    <form action="process.php" method="POST">

        <p>
            <label>Username:</label>
            <input type="text" id="user" name="user">
        </p>
        <p>
            <label>Password:</label>
            <input type="password" id="pass" name="pass">
        </p>
        <p>
            <input type="submit" id="btn" value="Login">
        </p>
```

PHP executable not found. Install PHP 7 and add it to your PATH or set the php.executablePath setting

Source: PHP IntelliSense (Extension)

Open settings

- Now type flutter in the extension search bar on the left and click install



flutter

Flutter 2.21.1
Flutter support and debugger for Visual Studi...
Dart Code

Awesome Flutter Snippets 2.0.0
Awesome Flutter Snippets is a collection snip...
Nash Ramdial — Install

Flutter Files 1.2.2
Quickly scaffold flutter bloc file templates
Igor Kravchenko — Install

Flutter Widget Snippets 0.0.2
A set of helpful widget snippets for day to da...
Alexis Villegas Torres — Install

MVVM Flutter Widget Generator 0.2.1
Generate a Flutter stateful widget with sepera...
panuavakul — Install

Flutter Create helper 1.0.0
Small widget to keep track of https://flutter.d...
Oleg Godovykh — Install

Flutter Stateful Widget Generator 0.0.2
Generates a stateful widget into a new file.
yum650350 — Install

Dart 2.19.0
Dart language support and debugger for Visu...
Dart Code

flutter-stylizer 0.0.11
Flutter Stylizer organizes your Flutter classes i...
gmlewis-vscode — Install

code-flutter-snippets 0.0.1
flutter snippets
MeePwn — Install

```
<!DOCTYPE html>
<html>
<head>
    <title> Login Page</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<div id ="app">
    <p>
            <label>AGES </label>
            </p>
    </div>

<div id="frm">
    <form action="process.php" method="POST">

        <p>
            <label>Username:</label>
            <input type="text" id="user" name="user">
        </p>
        <p>
            <label>Password:</label>
            <input type="password" id="pass" name="pass">
        </p>
        <p>
            <input type="submit" id="btn" value="Login">
        </p>
```

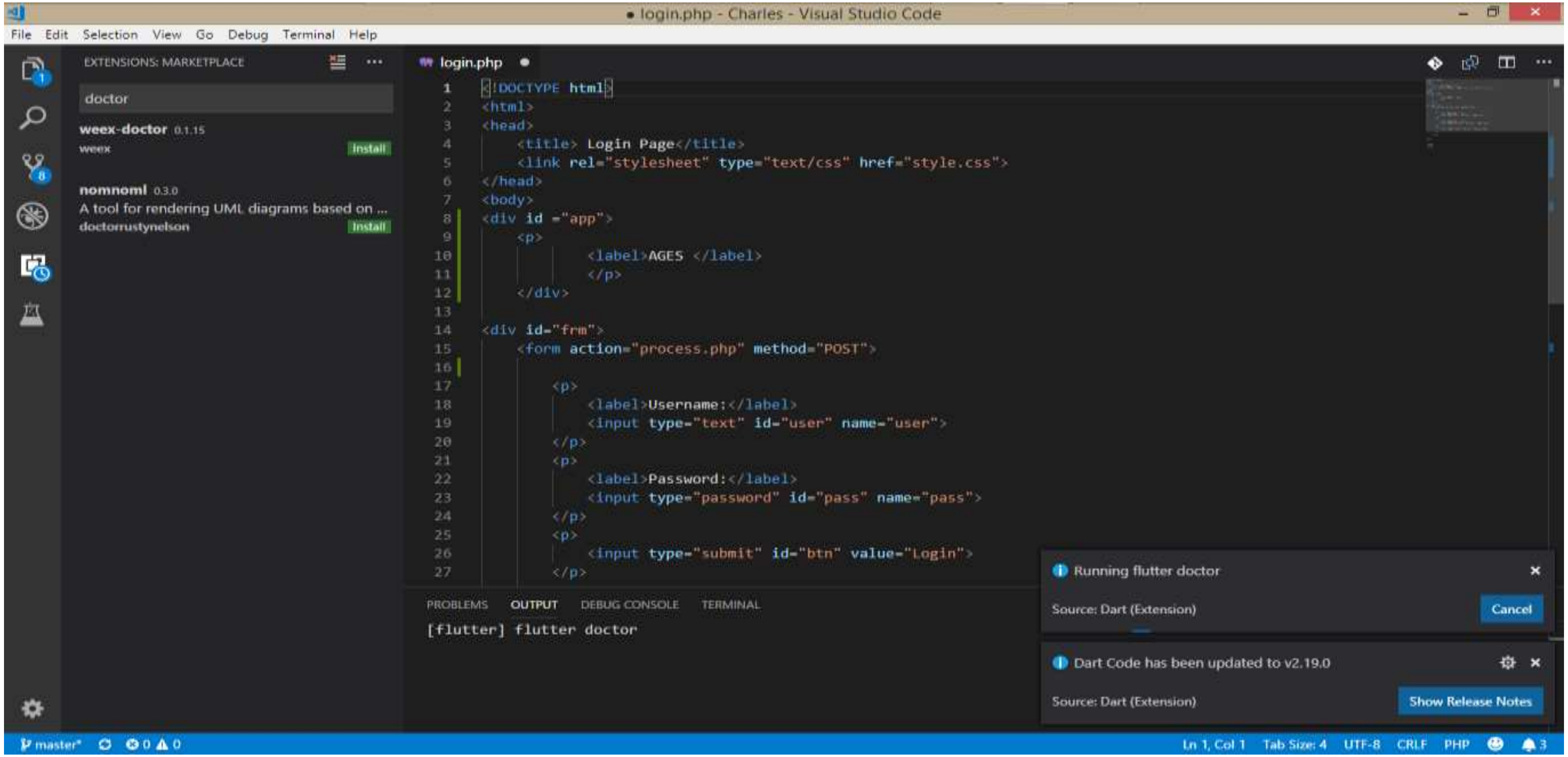
PHP executable not found. Install PHP 7 and add it to your PATH or set the php.executablePath setting

Source: PHP IntelliSense (Extension) — Open settings

- Wait for the installation to complete and restart

Verify your installation

- Open VS Code
- Go to View -> Command Palette…
- Start typing ‘doctor’ and select Flutter: Run Flutter Doctor



• login.php - Charles - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXTENSIONS: MARKETPLACE
doctor
weex-doctor 0.1.15
weex
Install
nomnoml 0.3.0
A tool for rendering UML diagrams based on ...
doctorrustynelson
Install
login.php
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
master* ⊗ 0 ⚠ 0
Ln 1, Col 1 Tab Size: 4 UTF-8 CRLF PHP 3

```
<!DOCTYPE html>
<html>
<head>
    <title> Login Page</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<div id ="app">
    <p>
        <label>AGES </label>
        </p>
    </div>

<div id="frm">
    <form action="process.php" method="POST">

        <p>
            <label>Username:</label>
            <input type="text" id="user" name="user">
        </p>
        <p>
            <label>Password:</label>
            <input type="password" id="pass" name="pass">
        </p>
        <p>
            <input type="submit" id="btn" value="Login">
        </p>
```

```
[flutter] flutter doctor
```

Running flutter doctor

Source: Dart (Extension)

Cancel

Dart Code has been updated to v2.19.0

Source: Dart (Extension)

Show Release Notes

- Review the install through the output pane

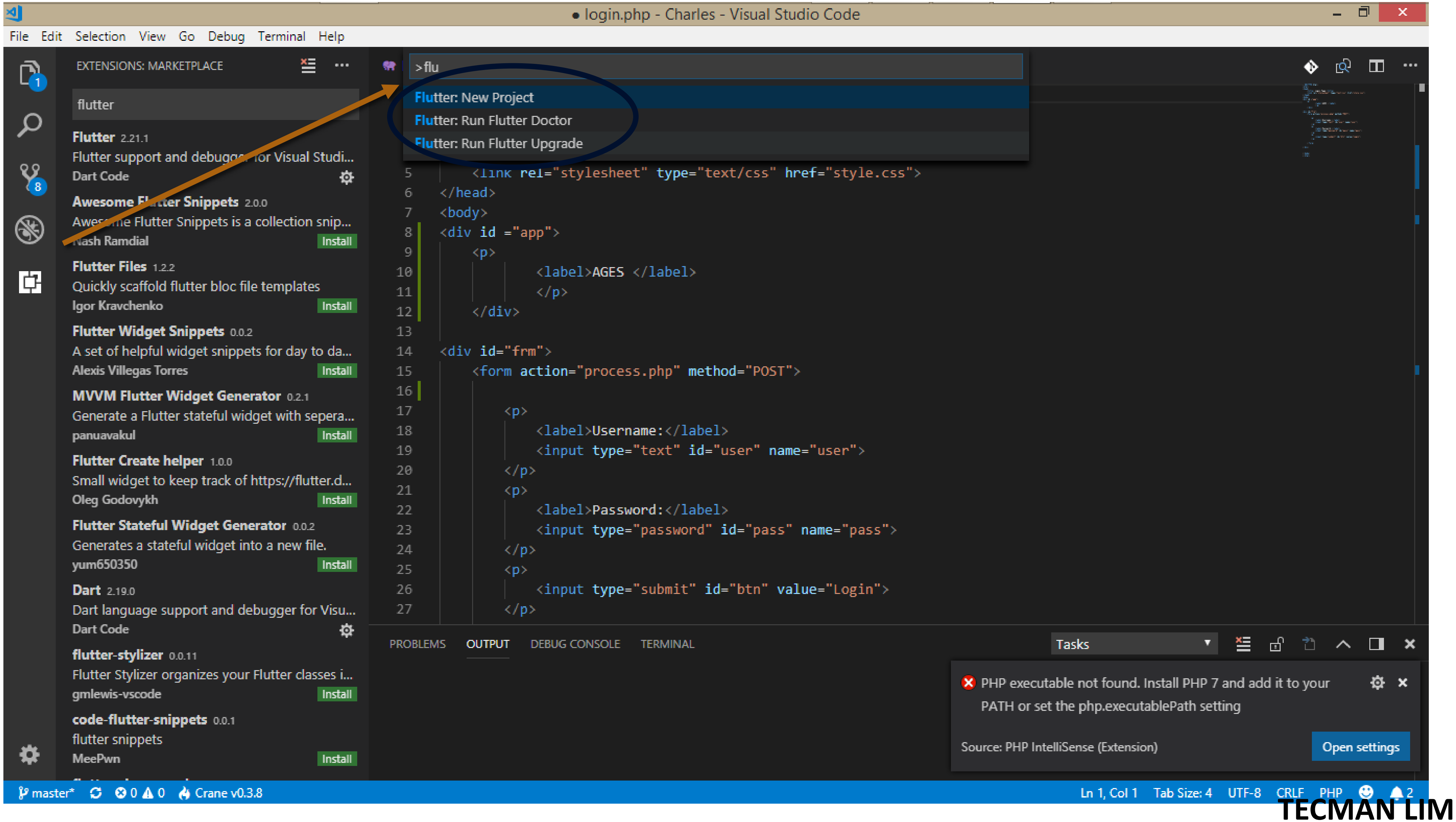
TECMAN LIMITED 2019

CREATING YOUR FIRST APP

Flutter SDK comes with command line codes to install and create an app on the go. Be sure to verify the setting up processes before continuing to this step.

1. Open VS Code
2. Go to View -> Command Palette…
3. Start typing ‘flutter’ and select the Flutter: New Project.
4. Enter your preferred project name and press Enter
5. Wait for the project to build.

TECMAN LIMITED 2019


● login.php - Charles - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXTENSIONS: MARKETPLACE

flutter

Flutter 2.21.1
Flutter support and debugger for Visual Studi...
Dart Code

Awesome Flutter Snippets 2.0.0
Awesome Flutter Snippets is a collection snip...
Nash Ramdial Install

Flutter Files 1.2.2
Quickly scaffold flutter bloc file templates
Igor Kravchenko Install

Flutter Widget Snippets 0.0.2
A set of helpful widget snippets for day to da...
Alexis Villegas Torres Install

MVVM Flutter Widget Generator 0.2.1
Generate a Flutter stateful widget with sepera...
panuavakul Install

Flutter Create helper 1.0.0
Small widget to keep track of https://flutter.d...
Oleg Godovykh Install

Flutter Stateful Widget Generator 0.0.2
Generates a stateful widget into a new file.
yum650350 Install

Dart 2.19.0
Dart language support and debugger for Visu...
Dart Code

flutter-stylizer 0.0.11
Flutter Stylizer organizes your Flutter classes i...
gmlewis-vscode Install

code-flutter-snippets 0.0.1
flutter snippets
MeePwn Install

>flu

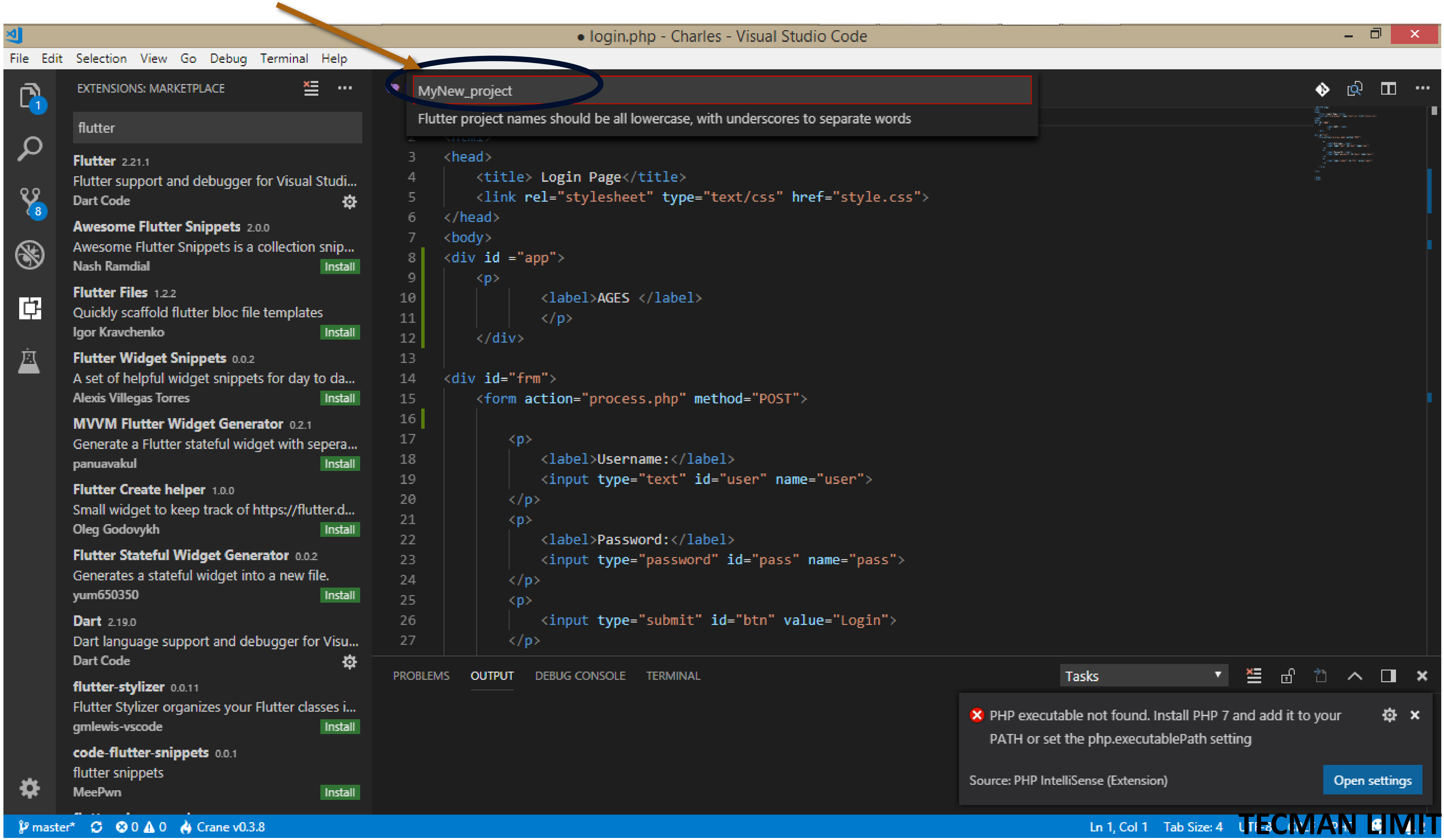
Flutter: New Project
Flutter: Run Flutter Doctor
Flutter: Run Flutter Upgrade

```
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<div id ="app">
    <p>
            <label>AGES </label>
            </p>
    </div>

<div id="frm">
    <form action="process.php" method="POST">

        <p>
            <label>Username:</label>
            <input type="text" id="user" name="user">
        </p>
        <p>
            <label>Password:</label>
            <input type="password" id="pass" name="pass">
        </p>
        <p>
            <input type="submit" id="btn" value="Login">
        </p>
```

PHP executable not found. Install PHP 7 and add it to your PATH or set the php.executablePath setting

Source: PHP IntelliSense (Extension) Open settings

TECMAN LIMITED 2019



flutter

MyNew_project

Flutter project names should be all lowercase, with underscores to separate words

Flutter 2.21.1
Flutter support and debugger for Visual Studi...
Dart Code

Awesome Flutter Snippets 2.0.0
Awesome Flutter Snippets is a collection snip...
Nash Ramdial — Install

Flutter Files 1.2.2
Quickly scaffold flutter bloc file templates
Igor Kravchenko — Install

Flutter Widget Snippets 0.0.2
A set of helpful widget snippets for day to da...
Alexis Villegas Torres — Install

MVVM Flutter Widget Generator 0.2.1
Generate a Flutter stateful widget with sepera...
panuavakul — Install

Flutter Create helper 1.0.0
Small widget to keep track of https://flutter.d...
Oleg Godovykh — Install

Flutter Stateful Widget Generator 0.0.2
Generates a stateful widget into a new file.
yum650350 — Install

Dart 2.19.0
Dart language support and debugger for Visu...
Dart Code

flutter-stylizer 0.0.11
Flutter Stylizer organizes your Flutter classes i...
gmlewis-vscode — Install

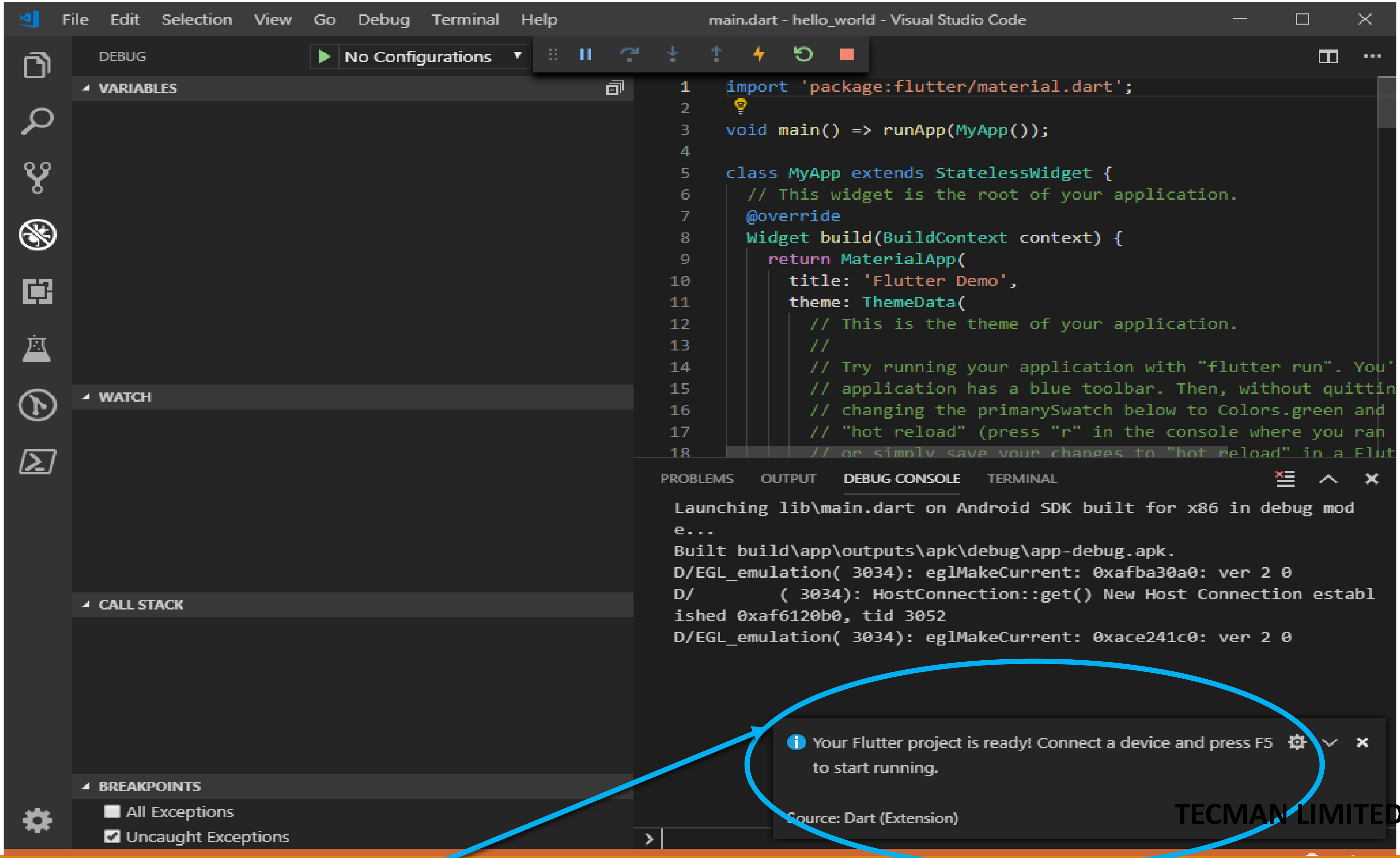
code-flutter-snippets 0.0.1
flutter snippets
MeePwn — Install

```
<head>
    <title> Login Page</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<div id ="app">
    <p>
            <label>AGES </label>
            </p>
    </div>

<div id="frm">
    <form action="process.php" method="POST">

        <p>
            <label>Username:</label>
            <input type="text" id="user" name="user">
        </p>
        <p>
            <label>Password:</label>
            <input type="password" id="pass" name="pass">
        </p>
        <p>
            <input type="submit" id="btn" value="Login">
        </p>
```

PHP executable not found. Install PHP 7 and add it to your PATH or set the php.executablePath setting

Source: PHP IntelliSense (Extension)

TECMAN LIMITED 2019



```dart
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'
        // application has a blue toolbar. Then, without quittin
        // changing the primarySwatch below to Colors.green and
        // "hot reload" (press "r" in the console where you ran
        // or simply save your changes to "hot reload" in a Flut
```

```
Launching lib\main.dart on Android SDK built for x86 in debug mod
e...
Built build\app\outputs\apk\debug\app-debug.apk.
D/EGL_emulation( 3034): eglMakeCurrent: 0xafba30a0: ver 2 0
D/        ( 3034): HostConnection::get() New Host Connection establ
ished 0xaf6120b0, tid 3052
D/EGL_emulation( 3034): eglMakeCurrent: 0xace241c0: ver 2 0
```

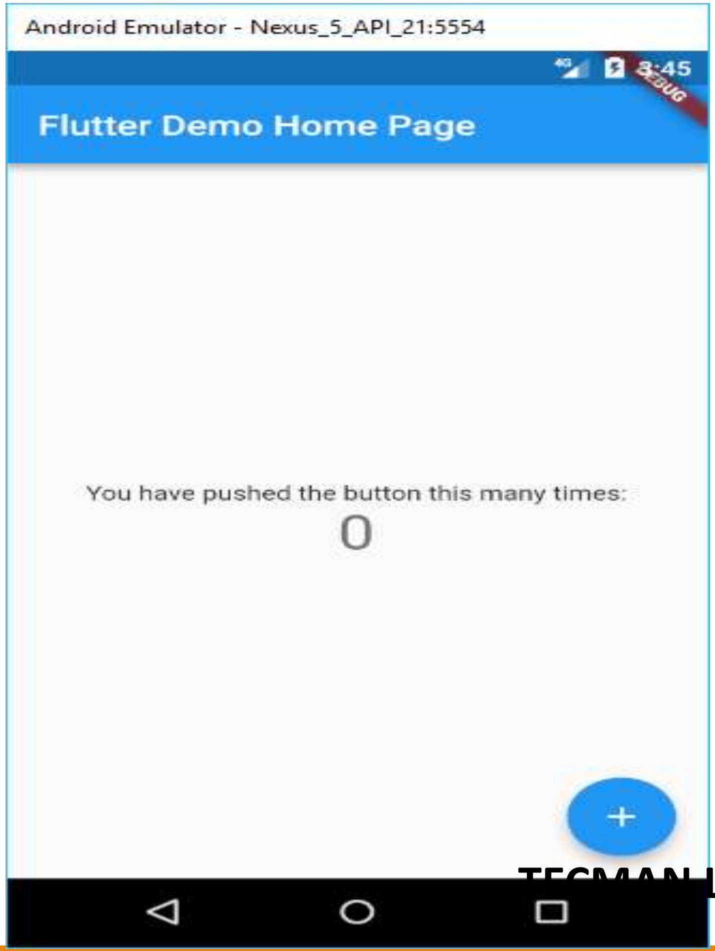
Your Flutter project is ready! Connect a device and press F5 to start running.

Source: Dart (Extension)

TECMAN LIMITED 2019

RUNNING YOUR FIRST FLUTTER APP

- Locate the VS Code status bar at the bottom of the window
- Select a device and wait for the device to load
- Now Invoke Debug > Start Debugging
- Wait for the app to launch
- Monitor the progress printed in the Debug Console view



Android Emulator - Nexus_5_API_21:5554

Flutter Demo Home Page

You have pushed the button this many times:

0

TECMAN LIMITED 2019

Introduction To Widget

WHAT IS A WIDGET?

- Flutter widgets are built using a modern framework that takes inspiration from React. The central idea is that you build your UI out of widgets.
- Widgets describe what their view should look like given their current configuration and state.
- When a widget's state changes, the widget rebuilds its description, which the framework diffs against the previous description in order to determine the minimal changes needed in the underlying render tree to transition from one state to the next.

TECMAN LIMITED 2019

WHY WIDGET?

- A widget's main job is to implement a build function, which describes the widget in terms of other, lower-level widgets.
- The framework builds those widgets in turn until the process bottoms out in widgets that represent the underlying Render Object, which computes and describes the geometry of the widget.

TECMAN LIMITED 2019

MaterialApp and Scalfold Widget

- In flutter, Scaffold implements the basic material design visual layout structure.
- The Scaffold is good enough to create a general purpose mobile application and contains almost everything you need to create a functional and responsive app.

TECMAN LIMITED 2019

APP BAR

- An app bar consists of a toolbar and potentially other widgets, such as a TabBar and a FlexibleSpaceBar.
- App bars typically expose one or more common actions with IconButtons which are optionally followed by a PopupMenuButton for less common operations.
- This is sometimes known as the "overflow menu.“
- App bars are typically used in the Scaffold.appBar property, which places the app bar as a fixed-height widget at the top of the screen.

TECMAN LIMITED 2019

FloatingActionButton

- A floating action button is a circular icon button that hovers over content to promote a primary action in the application.
- Floating action buttons are most commonly used in the Scaffold.floatingActionButton field.
- Use at most a single floating action button per screen. Floating action buttons should be used for positive actions such as "create", "share", or "navigate".
- If more than one floating action button is used within aRoute, then make sure that each button has a unique heroTag, otherwise an exception will be thrown.

GRIDVIEW

- The most commonly used grid layouts are GridView.count, which creates a layout with a fixed number of tiles in the cross axis, and GridView.extent, which creates a layout with tiles that have a maximum cross-axis extent.
- A custom SliverGridDelegate can produce an arbitrary 2D arrangement of children, including arrangements that are unaligned or overlapping.
- To create a grid with a large (or infinite) number of children,use the GridView.builder constructor with either a SliverGridDelegateWithFixedCrossAxisCount or a SliverGridDelegateWithMaxCrossAxisExtent for the gridDelegate.

AlertDialog

- An alert dialog informs the user about situations that require acknowledgement.
- An alert dialog has an optional title and an optional list of actions.
- The title is displayed above the content and the actions are displayed below the content.
- If the content is too large to fit on the screen vertically, the dialog will display the title and the actions and let the content overflow, which is rarely desired.
- Consider using a scrolling widget for content, such as SingleChildScrollView, to avoid overflow. (However, be aware that since AlertDialog tries to size itself using the intrinsic dimensions of its children, widgets such as ListView, GridView, and CustomScrollView, which use lazy viewports, will not work. If this is a problem, consider using Dialog directly.)

More Widgets

➢Text

- The Text widget displays a string of text with single style.
- The string might break across multiple lines or might all be displayed on the same line depending on the layout constraints. The style argument is optional.
- When omitted, the text will use the style from the closest enclosing DefaultTextStyle.
- Sample

```
new Text(
'Hello, World!',
textAlign: TextAlign.center,
style: new TextStyle(fontWeight: FontWeight.bold),
)
```

MORE WIDGETS CONT'D

- **Row, Column:** These flex widgets let you create flexible layouts in both the horizontal (Row) and vertical (Column) directions. Its design is based on the web's flexbox layout model.

- **Stack**: Instead of being linearly oriented (either horizontally or vertically), a Stack widget lets you stack widgets on top of each other in paint order.

- You can then use the Positioned widget on children of a Stack to position them relative to the top, right, bottom, or left edge of the stack. Stacks are based on the web's absolute positioning layout model.

TECMAN LIMITED 2019

More Widgets Cont'd

- **Container:** The Container widget lets you create a rectangular visual element. A container can be decorated with a BoxDecoration, such as a background, a border, or a shadow. A Container can also have margins, padding, and constraints applied to its size. In addition, a Container can be transformed in three dimensional space using a matrix.

- **The RichText:** widget displays text that uses multiple different styles. The text to display is described using a tree of TextSpan objects, each of which has an associated style that is used for that subtree. The text might break across multiple lines or might all be displayed on the same line depending on the layout constraints.

TECMAN LIMITED 2019

MORE WIDGETS CONT'D

- **Center:** A widget that centers its child within itself. This widget will be as big as possible if its dimensions are constrained and widthFactor and heightFactorare null. If a dimension is unconstrained and the corresponding size factor is null then the widget will match its child's size in that dimension.

- If a size factor is non-null then the corresponding dimension of this widget will be the product of the child's dimension and the size factor.

- For example if widthFactor is 2.0 then the width of this widget will always be twice its child's width.

TECMAN LIMITED 2019

MORE WIDGETS CONT’D

- **Padding:** A widget that insets its child by the given padding. When passing layout constraints to its child, padding shrinks the constraints by the given padding, causing the child to layout at a smaller size.

- Padding then sizes itself to its child's size, inflated by the padding, effectively creating empty space around the child.

TECMAN LIMITED 2019

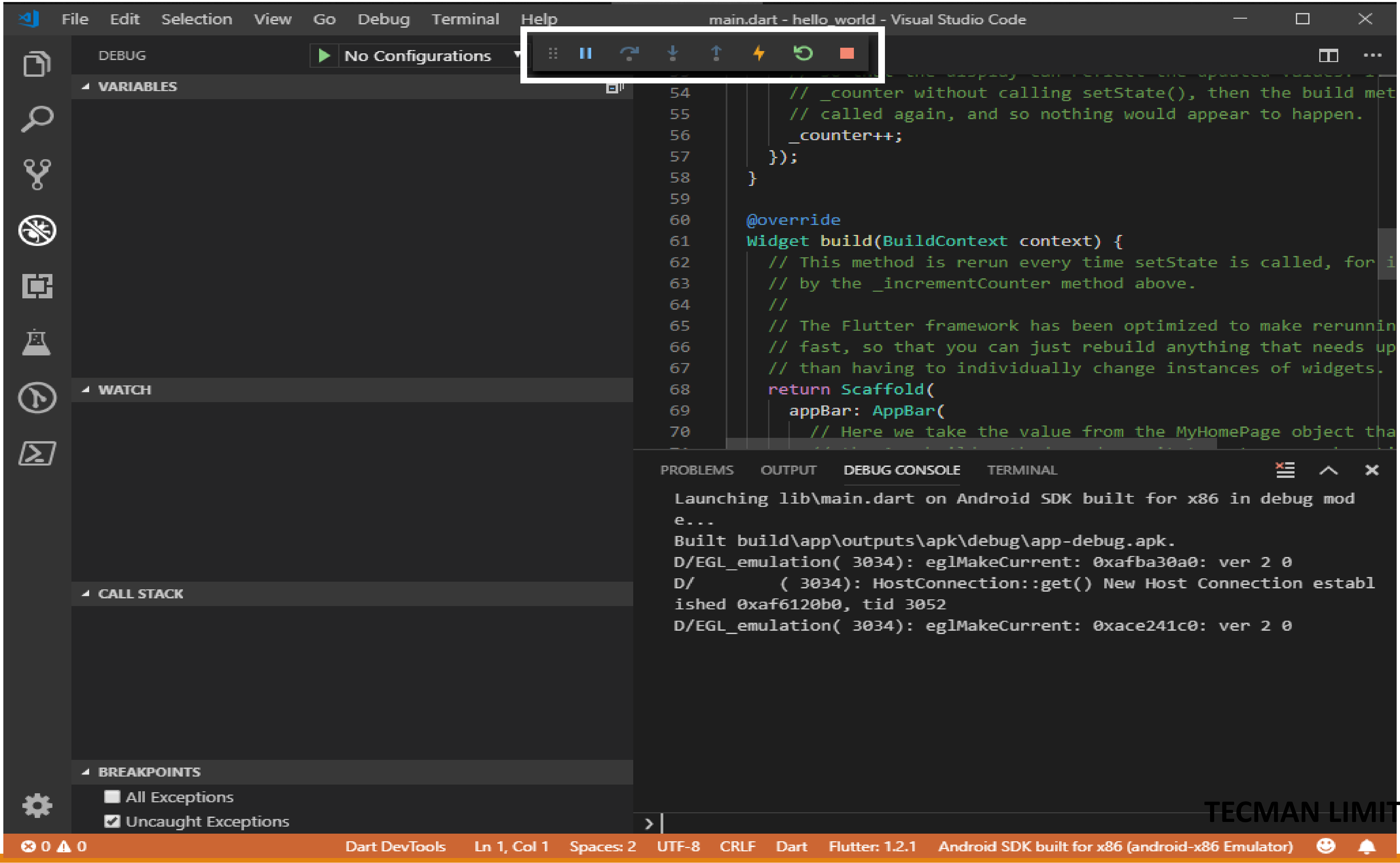
HOT RELOAD AND HOT RESTART

Flutter’s hot reload feature helps you quickly and easily experiment, build UIs, add features, and fix errors. Hot reload works by injecting updated source code files into the running Dart Virtual Machine (VM). After the VM updates classes with the new versions of fields and functions, the Flutter framework automatically rebuilds the widget tree, allowing you to quickly view the effects of your changes.

To hot reload a Flutter app:

1. Run the app from a supported Flutter editor or a terminal window. Either a physical or virtual device can be the target. Only Flutter apps in debug mode can be hot reloaded.
2. Modify one of the Dart files in your project. Most types of code changes can be hot reloaded
3. If you're working in an IDE/editor that supports Flutter's IDE tools, select **Save All** (cmd-s/ctrl-s), or click the Hot Reload button on the toolbar

TECMAN LIMITED 2019



```
      // _counter without calling setState(), then the build met
      // called again, and so nothing would appear to happen.
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    // This method is rerun every time setState is called, for i
    // by the _incrementCounter method above.
    //
    // The Flutter framework has been optimized to make rerunnin
    // fast, so that you can just rebuild anything that needs up
    // than having to individually change instances of widgets.
    return Scaffold(
      appBar: AppBar(
        // Here we take the value from the MyHomePage object tha
```

```
Launching lib\main.dart on Android SDK built for x86 in debug mod
e...
Built build\app\outputs\apk\debug\app-debug.apk.
D/EGL_emulation( 3034): eglMakeCurrent: 0xafba30a0: ver 2 0
D/        ( 3034): HostConnection::get() New Host Connection establ
ished 0xaf6120b0, tid 3052
D/EGL_emulation( 3034): eglMakeCurrent: 0xace241c0: ver 2 0
```

Stateless and Stateful Widgets

➢**STATELESS WIDGET:**

A stateless widget is a widget that describes part of the user interface by building a constellation of other widgets that describe the user interface more concretely. The building process continues recursively until the description of the user interface is fully concrete (e.g., consists entirely of RenderObjectWidgets, which describe concrete RenderObjects).

Note*: A Stateless widget is a widget that does not require mutable state.*

TECMAN LIMITED 2019

Sample Code

```
class MyWidget extends StatelessWidget {
  const MyWidget({ Key key }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(color: const Color(0xFF2DBD3A));
  }
}
```

STATEFUL WIDGET

A stateful widget is a widget that describes part of the user interface by building a constellation of other widgets that describe the user interface more concretely.

Note: *A Stateful Widget is a widget that has mutable state*

```
class YellowBird extends StatefulWidget {
  const YellowBird({ Key key }) : super(key: key);

  @override
  _YellowBirdState createState() => _YellowBirdState();
}

class _YellowBirdState extends State<YellowBird> {
  @override
  Widget build(BuildContext context) {
    return Container(color: const Color(0xFFFFE306));
  }
}
```

TECMAN LIMITED 2019

DEFINING A 'STATE' WIDGET

- State is information that (1) can be read synchronously when the widget is built and (2) might change during the lifetime of the widget. It is the responsibility of the widget implementer to ensure that the State is promptly notified when such state changes, using State.setState.

➢SETTING SETSTATE() METHOD

- The setState() method notifies the framework that the internal state of this object has changed

setState(() { _myState = newValue });

TECMAN LIMITED 2019

Navigation and Routing

➢NAVIGATOR

- A widget that manages a set of child widgets with a stack discipline. Many apps have a navigator near the top of their widget hierarchy in order to display their logical history using an Overlay with the most recently visited pages visually on top of the older pages.

- Using this pattern lets the navigator visually transition from one page to another by moving the widgets around in the overlay. Similarly, the navigator can be used to show a dialog by positioning the dialog widget above the current page.

TECMAN LIMITED 2019

Sample Code

```
Navigator.push(context, MaterialPageRoute<void>(
  builder: (BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('My Page')),
      body: Center(
        child: FlatButton(
          child: Text('POP'),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
    );
  },
));
```

TECMAN LIMITED 2019

ROUTES

- Mobile apps typically reveal their contents via full-screen elements called "screens" or "pages". In Flutter these elements are called routes and they're managed by a Navigator widget. The navigator manages a stack of Route objects and provides methods for managing the stack, like Navigator.push and Navigator.pop.

- An abstraction for an entry managed by a Navigator. This class defines an abstract interface between the navigator and the "routes" that are pushed on and popped off the navigator For example, '/home' will take you to HomeScreen or '/login' will take you to LoginScreen. '/' will be your initial route. Most routes have visual affordances, which they place in the navigators Overlayusing one or more OverlayEntry objects.

TECMAN LIMITED 2019

Sample code:

```
new MaterialApp(
  home: new Screen1(),
  routes: <String, WidgetBuilder> {
    '/screen1': (BuildContext context) => new Screen1(),
    '/screen2' : (BuildContext context) => new Screen2(),
    '/screen3' : (BuildContext context) => new Screen3(),
    '/screen4' : (BuildContext context) => new Screen4()
  },
)
```

push ( )

- In Flutter, when we navigate to another screen, we use the pushmethods and Navigator widget adds the new screen onto the top of the stack.
- Naturally, the pop methods would remove that screen from the stack.

Sample Code:

```
void _openMyPage() {
  Navigator.push(context,
MaterialPageRoute(builder:
(BuildContext context) => MyPage()));
}
```

POP ( )

When we want to get rid of the last visited screen, we would need to pop Routes from the Navigator's stack using the pop methods

Sample Code

```
void _close() {
  Navigator.pop(context);
}
```

TECMAN LIMITED 2019

PushNamed ( )

Push a named route onto the navigator that most tightly encloses the given context.

Sample Code

```
void _didPushButton() {
  Navigator.pushNamed(context, '/settings');
}
```

Handling User Input

Sometimes, it can be handy to run a callback function every time the text in a text field changes. For example, we might want to build a search screen with autocomplete functionality. In this case, we would want to update the results as the user types.

How can we run a callback function every time the text changes? With Flutter, we have two options:

1. Supply an onChanged callback to a TextField
2. Use a TextEditingController

TECMAN LIMITED 2019

Supply an onChanged callback to a TextField

The simplest approach is to supply an onChanged callback to a TextField. Whenever the text changes, the callback will be invoked. One downside to this approach is it does not work withTextFormField Widgets

The example will print the current value of the text field to the console every time the text changes.

Sample:

```
TextField(
  onChanged: (text) {
    print("First text field: 
$text");
  },
);
```

TECMAN LIMITED 2019

Use a TextEditingController

A more powerful, but more elaborate approach, is to supply a TextEditingController as the controllerproperty of the TextField or a TextFormField

To be notified when the text changes, we can listen to the controller using its addListener method

TECMAN LIMITED 2019

Create a TextEditingController

We need to create a TextEditingController before we can proceed to the following steps. In the subsequent steps, we will supply the TextEditingController to a TextField. Once we have wired these two classes together, we can listen for changes to the text field.

Sample

```
// Define a Custom Form Widget
class MyCustomForm extends StatefulWidget {
  @override
  _MyCustomFormState createState() => _MyCustomFormState();
}

// Define a corresponding State class. This class will hold the data
related to
// our Form.
class _MyCustomFormState extends State<MyCustomForm> {
  // Create a text controller. We will use it to retrieve the current value
  // of the TextField!
  final myController = TextEditingController();
```

Sample cont’d

```
@override
  void dispose() {
    // Clean up the controller when the Widget is removed from
the Widget tree
    myController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    // We will fill this out in the next step!
  }
}
```

Supply the TextEditingController to a TextField

In order to work, the TextEditingController must be supplied to either a TextField or a TextFormField. Once it’s wired up, we can begin listening for changes to the text field.

```
TextField(
  controller: myController,
);
```

Create a function to print the latest value

Now, we will need a function that should run every time t text changes. In this example, we will create a method tha prints out the current value of the text field.

Sample

```
_printLatestValue() {
  print("Second text field:
${myController.text}");
}
```

Listen to the controller for changes

Finally, we need to listen to the TextEditingController and run the _printLatestValue method whenever the text changes. We will use the addListener method to achieve this task.

In the ***example code listen***, we will begin listening for changes when the _MyCustomFormState class is initialized, and stop listening when the _MyCustomFormState is dispose

Example Code Listen

```
class _MyCustomFormState extends State<MyCustomForm> {
  @override
  void initState() {
    super.initState();

    // Start listening to changes
    myController.addListener(_printLatestValue);
  }
}
```

TECMAN LIMITED 2019

Complete code

```
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Retrieve Text Input',
      home: MyCustomForm(),
    );
  }
}
// Define a Custom Form Widget
class MyCustomForm extends StatefulWidget {
  @override
  _MyCustomFormState createState() => _MyCustomFormState();
}
```

Complete code cont’d

```
// Define a corresponding State class. This class will hold the
data related to

// our Form.

class _MyCustomFormState extends State<MyCustomForm> {

  // Create a text controller. We will use it to retrieve the
current value

  // of the TextField!

  final myController = TextEditingController();


  @override

  void initState() {

    super.initState();


    myController.addListener(_printLatestValue);

  }
```

```
@override
  void dispose() {
    // Clean up the controller when the Widget is removed from the
Widget tree
    // This also removes the _printLatestValue listener
    myController.dispose();
    super.dispose();
  }
  _printLatestValue() {
    print("Second text field: ${myController.text}");
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Retrieve Text Input'),
      ),
body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: <Widget>[
            TextField(
              onChanged: (text) {
                print("First text field: $text");
              },
            ),
            TextField(
              controller: myController,
            ),
          ],
        ),
      ),
    );
  }
}
```

User Interface

Theming

Flutter makes it possible to applying any kind of theme you want. A Theme basically describes the colors and typographic choices of an application. Descendant widgets obtain the current theme's ThemeData object using Theme.of.

When a widget uses Theme.of, it is automatically rebuilt if the theme later changes, so that the changes can be applied. The Theme widget implies an IconTheme widget, set to the value of the ThemeData.iconTheme of the data for the Theme.

Applying ThemeData

Theme is an inherited widget that you more or less use to set things like Color and Font, and it automatically applies these settings to all widgets below it in the Widget tree.

MaterialApp Widgets are the only widgets that accept a Theme. These Widgets will always be at the top of a Widget tree. For all intents and purposes, when you set ThemeData, it sets properties in your entire app.

TECMAN LIMITED 2019

Applying the Default ThemeData

```
return new MaterialApp(
    title: 'Flutter Demo',
    // This is actually redundant.
    // Flutter defaults to 'blue' for the
ThemeData
    theme: new ThemeData(
      primarySwatch: Colors.blue,
    ),
    home: new MyHomePage(title: 'Flutter
Demo Home Page'),
  );
```

Applying a Dark ThemeData

```
return new MaterialApp(
    title: 'Flutter Demo',
      theme: new ThemeData(
        brightness: Brightness.dark,
// new
      ),
    home: new MyHomePage(title: 'Flutter
Demo Home Page'),
  );
```

TECMAN LIMITED 2019

Applying a Custom ThemeData

```
Widget build(BuildContext context) {
  return new MaterialApp(
    title: 'Flutter Demo',
    theme: new ThemeData(
      primaryColor: Colors.amber,
      textTheme: new TextTheme(
        body1: new TextStyle(color: Colors.red),
      ),
    ),
    home: new MyHomePage(title: 'Flutter Demo Home Page'),
  );
}
```

THE BASIC SCREEN LAYOUT

Layouts in flutter consists a hierarchy of widgets with the outer widgets usually deal with alignment and structure whereas the inner elements are usually the visible elements on the page itself, like buttons and images, etc.

Sample code

```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
// This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

```
class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  // This widget is the home page of your application. It is stateful, meaning
  // that it has a State object (defined below) that contains fields that affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in this
  // case the title) provided by the parent (in this case the App widget) and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".
```

```
 final String title;
  @override
  _MyHomePageState createState() => _MyHomePageState();
}
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      // This call to setState tells the Flutter framework that something has
      // changed in this State, which causes it to rerun the build method below
      // so that the display can reflect the updated values. If we changed
      // _counter without calling setState(), then the build method would not
be
      // called again, and so nothing would appear to happen.
      _counter++;
```

TECMAN LIMITED 2019

A basic flutter screen layout can consist of:

- Visible Element Widgets

Visible Element Widgets

Visible Element Widgets are commonly used widgets we know of as descried in the previous chapters

.

Text

A text widget holds text.

```
new Text(
'Hello, World!',
textAlign: TextAlign.center,
style: new TextStyle(fontWeight: FontWeight.bold),
)
```

TECMAN LIMITED 2019

Button

A button allows you to perform an action based on a click

```
new RaisedButton(
  child: Text("Click here"),
  onPressed: () {
    // Do something in here
  },
),
```

Image

The image widget is a placeholder to hold an image. The image can be fetched from multiple sources like assets defined in a directory or from a direct URL

Icon

The icon widget is a container for an icon in Flutter.

sample

```
new Icon(
  Icons.add,
  size: 36.0,
)
```

Colum

A Column is a widget which arranges all its children widgets in a vertical stack. It can space the widgets according to the mainAxisAlignment and crossAxisAlignment property. Here the “main axis” is the vertical)

Sample

```
new Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    new Text(
      "Column 1",
    ),
    new Text(
      "Column 2"
    ),
    new Text(
      "Column 3"
    ),
  ],
),
```

Structuring and Alignment Widgets

Colum

A Column is a widget which arranges all its children widgets in a vertical stack. It can space the widgets according to the mainAxisAlignment and crossAxisAlignment property. Here the “main axis” is the vertical axis and the “cross axis” is the horizontal axis.

```
new Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    new Text(
      "Column 1",
    ),
    new Text(
      "Column 2"
    ),
    new Text(
      "Column 3"
    ),
  ],
),
```

Row

A Column is a widget which arranges all its children widgets in a vertical stack.

```
new Row(
  mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    new Text(
      "Row 1",
    ),
    new Text(
      "Row 2"
    ),
    new Text(
      "Row 3"
    ),
  ],
),
```

Center

A center widget takes a child and centers it in the available space.

```
Center(
  child: new Row(
    mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
    children: <Widget>[
      new Text(
        "Center 1",
      ),
      new Text(
        "Center 2"
      ),
      new Text(
        "Center 3"
      ),
    ],
  ),
),
```

Padding

A padding is a widget that wraps other widgets to give them padding in all or specified directions.

```
Padding(
  padding: const EdgeInsets.all(8.0),
  child: new Text(
    "Padding 1",
  ),
),
```

TECMAN LIMITED 2019

Scaffold

A Scaffold Widget provdies a framework for adding common material design elements such as AppBars, Drawers, Floating Action Buttons, Bottom Navigation, etc.

```
new Scaffold(
  appBar: new AppBar(
    title: new Text(widget.title),
  ),
  body: Center(
  ),
  floatingActionButton: FloatingActionButton(
      child:Icon(Icons.add),
      onPressed: () {
      }
  ),
)
```

FONTS

- Flutter makes it possible for changing the default fonts of your application. Fonts can be downloaded and loaded from the assets directory.
- You can use custom fonts for your flutter application by including them the pubspec.yaml file which can be located under the fonts leading in your flutter section.

TECMAN LIMITED 2019

```yaml
flutter:
  fonts:
    - family: Raleway
      fonts:
        - asset: assets/fonts/Raleway-Regular.ttf
        - asset: assets/fonts/Raleway-Medium.ttf
          weight: 500
        - asset: assets/fonts/Raleway-SemiBold.ttf
          weight: 600
    - family: AbrilFatface
      fonts:
        - asset: assets/fonts
```

Applying Custom Fonts

- Flutter works out of the box with custom fonts. You can apply fonts across an entire app or to individual Widgets.
- In order to use custom fonts, you need to import the font file into your application and declare the font files in your pubspec.yaml

```
flutter:
  fonts:
    - family: Raleway
      fonts:
        - asset: fonts/Raleway-Regular.ttf
        - asset: fonts/Raleway-Italic.ttf
          style: italic
    - family: RobotoMono
      fonts:
        - asset: fonts/RobotoMono-Regular.ttf
        - asset: fonts/RobotoMono-Bold.ttf
          weight: 700
```

Now set the custom font as the default to override the system default fonts. There are two ways to set the custom fonts, (1) setting as the default and (2) setting in a specific widget.

Setting fonts as default font

To use a font as the default, set the fontFamily property as part of the app’s theme. The value provided to fontFamily must match the family name declared in the pubspec.yaml.

```
MaterialApp(
  title: 'Custom Fonts',
  // Set Raleway as the default app font
  theme: ThemeData(fontFamily: 'Raleway'),
  home: MyHomePage(),
);
```

Setting fonts in a specific widget

If you want to apply the font to a specific Widget, such as a Text Widget, provide a TextStyle to the Widget.

In this example, you’ll apply the RobotoMono font to a single Text Widget. Once again, the fontFamily must match the family name declared in the pubspec.yaml.

```
Text(
  'Roboto Mono sample',
  style: TextStyle(fontFamily: 'RobotoMono'),
);
```

Asynchronous Functions

The ‘Future’ Function

In flutter it is common to have code that works in an asynchronous way. An example of code that works asynchronously is having an app that retrieves data from a remote server. In most cases you will need actions to handle the state which is either failed or succeed. There is the need for the “future” function.

The class Future is included in the dart:async package.

A Future object can be in two states:

- pending - In this state, the computation represented by this Future is still in progress, and no result is available.
- completed - In this state, the computation is completed either successfully or with failure and the result is available. We can further divide this state into two sub-states: completed with value and completed with error.

TECMAN LIMITED 2019

Future.then

- The method Future.then is used to add callbacks when the future completes. We can add callbacks for both states completed with value and completed with error. In the following code, we use the factory constructor
- Future.delayed to create a new Future object that will be completed after 3 seconds. Based on the random boolean value, the Future may complete with the value 100 or an error boom!. We use then to add callbacks for both cases and output different messages to the console.

```
import 'dart:async';
import 'dart:math';
void main() {
  var random = new Random();
  var future = new Future.delayed(new
Duration(seconds: 3), () {
    if (random.nextBool()) {
      return 100;
    } else {
      throw 'boom!';
    }
  });
  future.then((value) {
    print('completed with value $value');
  }, onError: (error) {
    print('completed with error $error');
  });
}
```

```
import 'dart:async';


void main() {
  var future = new Future.value('a').then((v1)
{
    return new Future.value('$v1 b').then((v2)
{
      return new Future.value('$v2
c').then((v3) {
        return new Future.value('$v3 d');
      });
    });
  });
  future.then(print, onError: print);
}
```

Future.catchError

- Future.catchError adds a callback to handle errors in the Future. By default the callback handles all errors. We can also pass a predicate function test to check if an error should be handled. In the code below, the test predicate specifies that only error objects with type String are handled.
- Future.error creates a new Future completed with the given error.

```
import 'dart:async';


void main() {
  new
Future.error('boom!').catchError(print,
test: (error) {
    return error is String;
  });
}
```

Future.whenComplete

The pattern of using then().catchError() may remind you of the common try-catch structure, and Future.whenComplete is the missing part - finally. The callback added with Future.whenComplete is always called after the Future is completed. whenComplete also returns a new Future object.

```
import 'dart:async';
import 'dart:math';

void main() {
  var random = new Random();
  new Future.delayed(new Duration(seconds: 3), () {
    if (random.nextBool()) {
      return 100;
    } else {
      throw 'boom!';
    }
  }).then(print).catchError(print).whenComplete(() {
    print('done!');
  });
}
```

Future.timeout

- The asynchronous computation of a Future may take quite a long time to complete. For example, a remote service call may be blocked due to heavy load on the server. Future.timeout allows to time-out a Future after specified time limit has passed. The time limit is specified as a Duration.
- In the code below, the original Future will be completed after 5 seconds delay, but the timeout value specified in timeout is 2 seconds, so the returned Future is completed with a TimeoutException error after 2 seconds.

```
import 'dart:async';


void main() {
  new Future.delayed(new Duration(seconds: 5),
() {
    return 1;
  }).timeout(new Duration(seconds:
2)).then(print).catchError(print);
}
```

Create Future

- Now we know how to work with Futures in Flutter, but how should we get these Future objects? Usually we can get Future objects by using libraries. For example, methods get, post and put in the package http return Future objects with the HTTP response. We can also create Future objects using constructors.
- The code below shows the very basic way to create Future objects by passing the computation to run. The created Future is completed with the result of the function, either the function's return value or the error thrown in the function. The passed-in function is invoked asynchronously with Timer.run.

```
import 'dart:async';


void main() {
  new Future(() {
    var sum = 0;
    for (var i = 0; i < 50000; i++) {
      sum += i;
    }
    return sum;
  }).then(print);
}
```