Set up an edit

Windows install

Contents

System requirements

To install and run Flutter, your development environment must meet these minimum requirements:

- **Operating Systems**: Windows 7 SP1 or later (64-bit), x86-64 based.
- **Disk Space**: 1.64 GB (does not include disk space for IDE/tools).
- **Tools**: Flutter depends on these tools being available in your environment.
  - 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 PowerSh

Get the Flutter SDK

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

   flutter_windows_2.2.3-stable.zip

   For other release channels, and older builds, see the SDK releases page.

2. Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, `C:\Users\<your-user-name>\Documents`).

Warning: Do not install Flutter in a directory like `C:\Program Files\` that requires elevated privileges.

If you don't want to install a fixed version of the installation bundle, you can skip steps 1 and 2. Instead, get the source code from Flutter repo on GitHub, and change branches or tags as needed. For example:

```
C:\src>git clone https://github.com/flutter/flutter.git -b stable
```

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

Update your path

If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the `PATH` environment variable:

- From the Start search bar, enter 'env' and select **Edit environment variables for your account**.
- Under **User variables** check if there is an entry called **Path**:
  - If the entry exists, append the full path to `flutter\bin` using `;` as a separator from existing values.
  - If the entry doesn't exist, create a new user variable named `Path` with the full path to `flutter\bin` as its value.

You have to close and reopen any existing console windows for these changes to take effect.

Note: As of Flutter's 1.19.0 dev release, the Flutter SDK contains the `dart` command alongside the `flutter` command so that you can more easily run Dart command-line programs. Downloading the Flutter SDK also downloads the compatible version of Dart, but if you've downloaded the Dart SDK separately, make sure that the Flutter version of `dart` is first in your path, as the two versions might not be compatible. The following command tells you whether the `flutter` and `dart` commands originate from the same `bin` directory and are therefore compatible.

```
  C:\>where flutter dart
  C:\path-to-flutter-sdk\bin\flutter
  C:\path-to-flutter-sdk\bin\flutter.bat
  C:\path-to-dart-sdk\bin\dart.exe        :: this should go after `C:\path-to-flutter-sdk\bin\`
commands
  C:\path-to-flutter-sdk\bin\dart
  C:\path-to-flutter-sdk\bin\dart.bat
```

As shown above, the command `dart` from the Flutter SDK doesn't come first. Update your path to use commands from `C:\path-to-flutter-sdk\bin\` before commands from `C:\path-to-dart-sdk\bin\` (in this case). After restarting your shel for the change to take effect, running the `where` command again should show that the `flutter` and `dart` commands from the same directory now come first.

```
  C:\>where flutter dart
  C:\dev\src\flutter\bin\flutter
  C:\dev\src\flutter\bin\flutter.bat
  C:\dev\src\flutter\bin\dart
  C:\dev\src\flutter\bin\dart.bat
  C:\dev\src\dart-sdk\bin\dart.exe
```

However, if you are using `PowerShell`, in it `where` is an alias of `Where-Object` command, so you need to use `where.exe` instead.

```
  PS C:\> where.exe flutter dart
```

To learn more about the `dart` command, run `dart -h` from the command line, or see the dart tool page.

Run `flutter doctor`

From a console window that has the Flutter directory in the path (see above), run the following command to see if there are any platform dependencies you need to complete the setup:

```
C:\src\flutter>flutter doctor
```

This command checks your environment and displays a report of the status of your Flutter installation. Check the output carefully other software you might need to install or further tasks to perform (shown in **bold** text).

For example:

```
[-] Android toolchain - develop for Android devices
    • Android SDK at D:\Android\sdk
    ✗ Android SDK is missing command line tools; download from https://goo.gl/XxQghQ
    • Try re-installing or updating your Android SDK,
      visit https://flutter.dev/setup/#android-setup for detailed instructions.
```

The following sections describe how to perform these tasks and finish the setup process. Once you have installed any missing dependencies, you can run the `flutter doctor` command again to verify that you've set everything up correctly.

Note: If `flutter doctor` returns that either the Flutter plugin or Dart plugin of Android Studio are not installed, move on to Set up an editor to resolve this issue.

Warning: The `flutter` tool uses Google Analytics to anonymously report feature usage statistics and basic crash reports. This data is used to help improve Flutter tools over time.

Flutter tool analytics are not sent on the very first run. To disable reporting, type `flutter config --no-analytics`. To display the current setting, type `flutter config`. If you opt out of analytics, an opt-out event is sent, and then no further information is sent by the Flutter tool.

By downloading the Flutter SDK, you agree to the Google Terms of Service. Note: The Google Privacy Policy describes how data is handled in this service.

Moreover, Flutter includes the Dart SDK, which may send usage metrics and crash reports to Google.

Android setup

Note: Flutter relies on a full installation of Android Studio to supply its Android platform dependencies. However, you can write your Flutter apps in a number of editors; a later step discusses that.

Install Android Studio

1. Download and install Android Studio.
2. Start Android Studio, and go through the 'Android Studio Setup Wizard'. This installs the latest Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, which are required by Flutter when developing for Android.
3. Run `flutter doctor` to confirm that Flutter has located your installation of Android Studio. If Flutter cannot locate it, run `flutter config --android-studio-dir <directory>` to set the directory that Android Studio is installed to.

Set up your Android device

To prepare to run and test your Flutter app on an Android device, you need an Android device running Android 4.1 (API level 16) o higher.

1. Enable **Developer options** and **USB debugging** on your device. Detailed instructions are available in the Android documenta
2. Windows-only: Install the Google USB Driver.
3. Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access you device.
4. In the terminal, run the `flutter devices` command to verify that Flutter recognizes your connected Android device. By defa Flutter uses the version of the Android SDK where your `adb` tool is based. If you want Flutter to use a different installation o Android SDK, you must set the `ANDROID_SDK_ROOT` environment variable to that installation directory.

Set up the Android emulator

To prepare to run and test your Flutter app on the Android emulator, follow these steps:

1. Enable VM acceleration on your machine.
2. Launch **Android Studio**, click the **AVD Manager** icon, and select **Create Virtual Device…**
   - In older versions of Android Studio, you should instead launch **Android Studio > Tools > Android > AVD Manager** and select **Create Virtual Device…**. (The **Android** submenu is only present when inside an Android project.)
   - If you do not have a project open, you can choose **Configure > AVD Manager** and select **Create Virtual Device…**
3. Choose a device definition and select **Next**.
4. Select one or more system images for the Android versions you want to emulate, and select **Next**. An *x86* or *x86_64* image i recommended.
5. Under Emulated Performance, select **Hardware - GLES 2.0** to enable hardware acceleration.
6. Verify the AVD configuration is correct, and select **Finish**.

   For details on the above steps, see Managing AVDs.

7. In Android Virtual Device Manager, click **Run** in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

Agree to Android Licenses

Before you can use Flutter, you must agree to the licenses of the Android SDK platform. This step should be done after you have installed the tools listed above.

1. Make sure that you have a version of Java 8 installed and that your `JAVA_HOME` environment variable is set to the JDK's folde

   Android Studio versions 2.2 and higher come with a JDK, so this should already be done.

2. Open an elevated console window and run the following command to begin signing licenses.

```
$ flutter doctor --android-licenses
```

3. Review the terms of each license carefully before agreeing to them.
4. Once you are done agreeing with licenses, run `flutter doctor` again to confirm that you are ready to use Flutter.

Windows setup

Warning: Beta (Win32) and Dev (UWP)! This area covers Windows desktop support, which is available in beta release (Win32) and alpha release (UWP).

The Win32 variant still has notable feature gaps, including accessibility support, while the UWP variant is still in very active development.

You can try a beta snapshot of Win32 desktop support on the stable channel, or you can keep up with the latest changes to desktop on the `beta` channel. For Windows UWP you need to be on the `dev` channel.

For more information, see the **Desktop** section in [What's new in Flutter 2.2](), a free article on Medium.

Additional Windows requirements

For Windows desktop development, you need the following in addition to the Flutter SDK:

- [Visual Studio 2019]() (not to be confused with Visual Studio *Code*). For Win32 you need the "Desktop development with C++" workload installed, including all of its default components. For UWP you need the "Universal Windows Platform developmer workload installed, with the optional UWP C++ tools.

Enable desktop support

At the command line, perform the following command to enable Win32 desktop support:

```
$ flutter config --enable-windows-desktop
```

For Windows UWP desktop support perform the following commands to switch to the `dev` channel, upgrade Flutter, and enable U

```
$ flutter channel dev
$ flutter upgrade
$ flutter config --enable-windows-uwp-desktop
```

For more information, see [Desktop support for Flutter]()

Web setup

Flutter has support for building web applications in the `stable` channel. Any app created in Flutter 2 automatically builds for the v
To add web support to an existing app, follow the instructions on [Building a web application with Flutter]() when you've completed t
setup above.

Next step

Set up your preferred editor.

[Set up an edito]()