GET STARTED

1 WHAT YOU'LL LEARN

- How to install Flutter SDK
- How to set up an editor for Flutter
- How to test your drive

2 INSTALL FLUTTER SDK

2.1 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.

2.2 Get the Flutter SDK

1. Download the latest stable release of the Flutter SDK: https://flutter.dev/docs/development/tools/sdk/releases
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.

2.3 Update Flutter 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** append the **full path to flutter\bin** using ; as a separator from existing values.
- Close and reopen any existing console windows for these changes to take effect.

- Use command: **where flutter dart** to check whether the flutter and dart commands originate from the same bin directory.

2.4 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 for other software you might need to install or further tasks to perform (shown in **bold** text).

3 SET UP AN EDITOR

You can build apps with Flutter using any text editor combined with our command-line tools. However, we recommend using one of our editor plugins for an even better experience. These plugins provide you with code completion, syntax highlighting, widget editing assists, run & debug support, and more.

Follow the steps below to add an editor plugin for Android Studio, IntelliJ, **VS Code**, or Emacs.

3.1 Install VS Code

VS Code is a lightweight editor with Flutter app execution and debug support.

- VS Code, latest stable version

  https://code.visualstudio.com/

3.2 Install the Flutter and Dart plugins

1. Start VS Code.
2. Invoke **View** > **Command Palette**….
3. Type "**install**", and select **Extensions**: **Install Extensions**.
4. Type "**flutter**" in the extensions search field, select **Flutter** in the list, and click **Install**. This also installs the required **Dart** plugin.

3.3 Validate your setup with the Flutter Doctor

1. Invoke **View** > **Command Palette**….
2. Type "**doctor**", and select the **Flutter**: **Run Flutter Doctor**.
3. Review the output in the **OUTPUT** pane for any issues. Make sure to select Flutter from the dropdown in the different Output Options.

4 TEST DRIVE

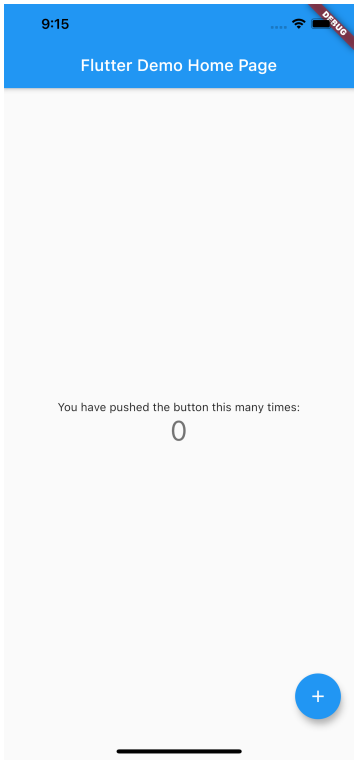
This section describes how to create a new Flutter app from templates, run it, and experience "hot reload" after you make changes to the app.

4.1 Create the app

1. Invoke **View** > **Command Palette**.
2. Type "flutter", and select the **Flutter**: **New Application Project**.
3. Create or select the parent directory for the new project folder.
4. Enter a project name, such as **myapp**, and press **Enter**.
5. Wait for project creation to complete and the **main.dart** file to appear.

4.2 Run the app

1. Locate the VS Code **status bar** (the blue bar at the bottom of the window)
2. Select a device from the **Device Selector** area.
3. **Invoke Run** > **Start Debugging** or press **F5**.
4. Wait for the app to launch — progress is printed in the **Debug Console** view.
5. After the app build completes, you'll see the starter app on your device



4.3 Try hot reload

Flutter offers a fast development cycle with Stateful Hot Reload, the ability to reload the code of a live running app without restarting or losing app state. Make a

change to app source, tell your IDE or command-line tool that you want to hot reload, and see the change in your simulator, emulator, or device.

1. Open lib/main.dart.
2. Change the string

```
Text(
  'You have pushed the button this many times:',
), // Text
```

to

```
Text(
  'You have clicked the button this many times:',
), // Text
```

3. Save your changes: invoke **Save All**, or click Hot Reload