

Hew Playground

IPHONE, IPAD AND ANDROID USER GUIDE

The Hew Project

HEW PLAYGROUND

iPhone and iPad

- Open and run your first example
- Edit code and understand diagnostics
- Recent builds: choose where the program runs
- Save, reopen, import and export
- Share code or a link
- Learn and use reference material
- Use the iPad layout
- When something does not work

Android

- Open an example and run it
- Edit and navigate code
- Read output and errors
- Keep local files and export a backup
- Share a program
- Learn with tutorials and Quick Ref
- Troubleshoot a beta session

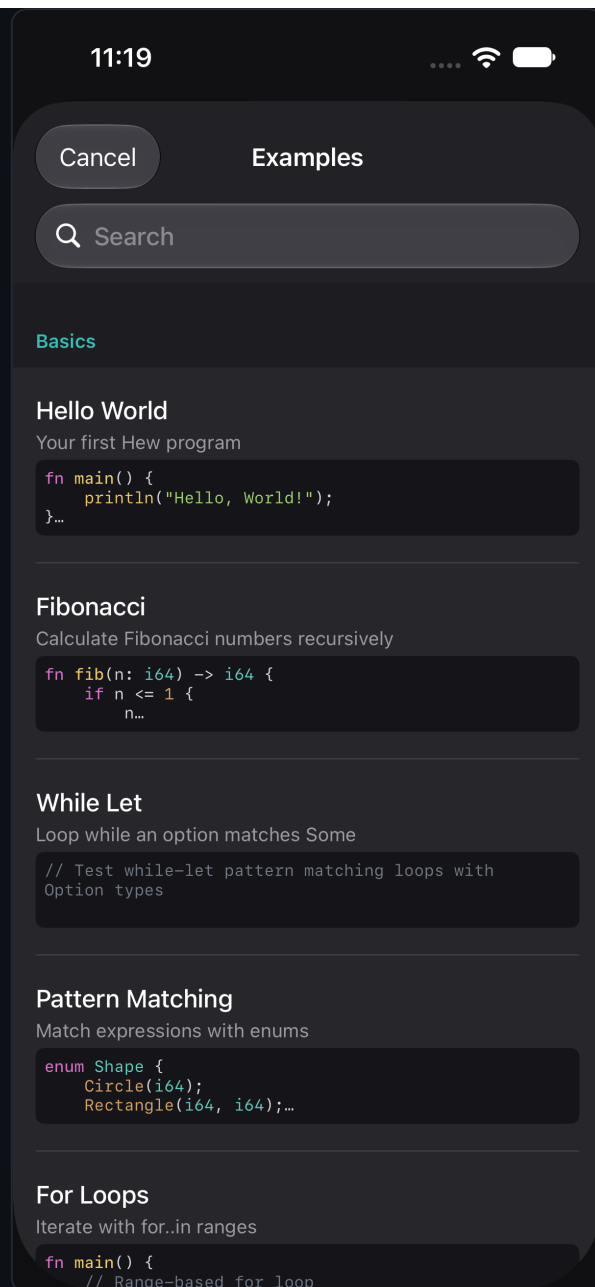
iPhone and iPad

Get [Hew Playground from the App Store](#) on your iPhone or iPad. You can start with a bundled example; no desktop compiler or account is needed for the editor.

The public App Store release is currently **0.5.0**. In that release, Run uses the online service and requires internet. This guide covers its example, editor and file workflows and labels the execution controls found in newer source builds, which now bundle Hew **0.9.2**. The real screenshots show a **0.6.1** build, including its **On device/Remote** chooser; those controls may be absent from your store version. Use the bundled examples for the app you have installed.

Open and run your first example

1. Open the **Library** menu, shown as a stack of books in the toolbar.
2. Choose **Examples**, then **Hello World**. On an older toolbar, tap Examples directly.
3. Tap the green **Run** button.
4. Read the **Output** panel below the editor. On iPad, the panels can sit side by side.



The native iPhone Examples picker. Selecting an example loads its source into the editor.

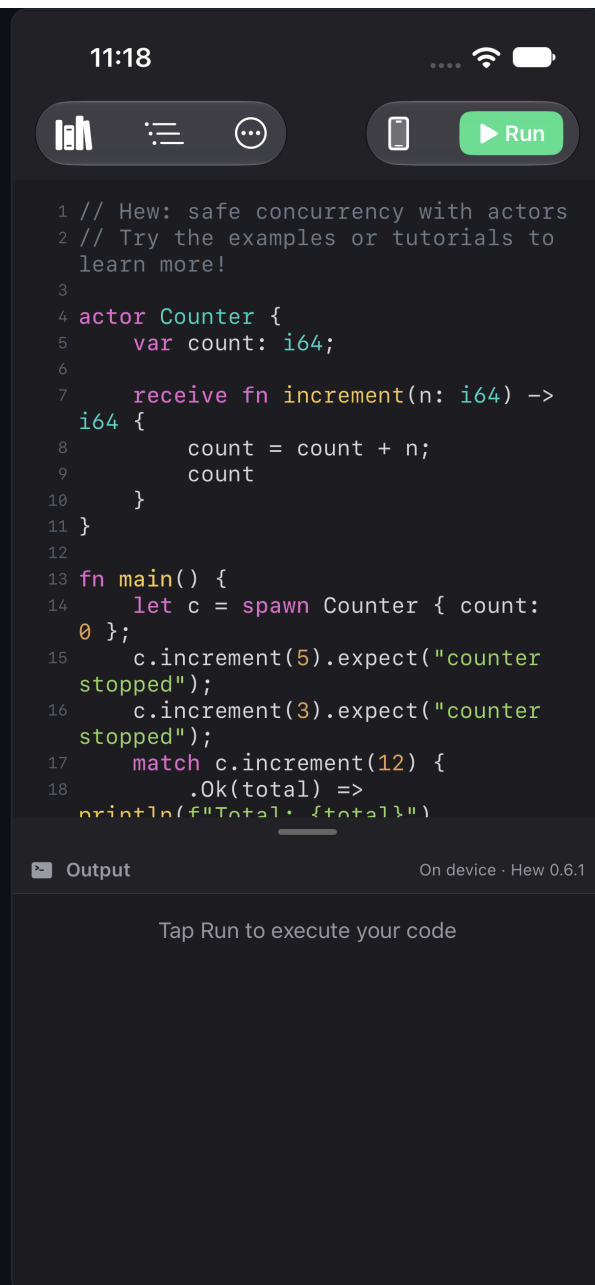
Change the text inside `println`, then run again. For a small program you can type yourself:

```
fn main() {  
  
    println("Hello from my phone!");  
  
}
```

Opening another example replaces the editor contents. Keep a working copy in **My Files**, or export it, before moving on; respond to any unsaved-changes prompt.

Edit code and understand diagnostics

Tap the editor to type. Line numbers, syntax colors, auto-indent, bracket pairing and the symbol accessory bar help with punctuation. Recent builds offer compiler-backed completion, type information, a document outline and diagnostics. In **More (...)** → **Settings**, adjust editor and analysis preferences.



The iPhone editor in a recent build. The phone icon beside Run selects the execution engine.

When an underline appears, read the diagnostic before changing more code. A compile error means the program could not build; a runtime error means execution failed after compilation. The Output panel distinguishes these from a connection error. Correct the first relevant error, then run again. Compiler quick fixes, when offered, apply to the selected diagnostic.

Recent builds: choose where the program runs

In builds with the engine selector, **On device** runs supported programs locally.

Remote sends the source to Hew's online execution service and needs internet.

The selected engine and compiler version appear with output. A local failure stays visible; choose **Run remotely** yourself when that action is offered and you want to send the program to the server.



A successful on-device run. This capture's output identifies its bundled 0.6.1 compiler.

Stop replaces Run while a program is running. Use it to cancel a long run before changing engines or starting another. On-device execution supports the bundled runtime's capabilities; operating-system resources and some native-only features may need a remote run. Keep a small bundled example handy when checking whether a problem is in your code or the chosen engine.

Save, reopen, import and export

Open **More (...)** → **My Files** for local working copies. Use **+** to create a document and select a document to reopen it. Edits to a working document are saved locally. Rename through the document's context menu; delete only when you no longer need the local copy.

- **Open File...** uses the native picker to import a `.hew` file into the editor.
- **Save As...** exports the current code as a `.hew` file to a destination you choose in Files.
- An imported or exported file and the app's local working copy are separate copies. Export again after later edits when you want to update your backup.

Use export for a portable copy before removing the app or changing devices. If a Files provider is unavailable or an export fails, keep the local document and retry with an available destination.

Share code or a link

Use **More (...)** → **Share**:

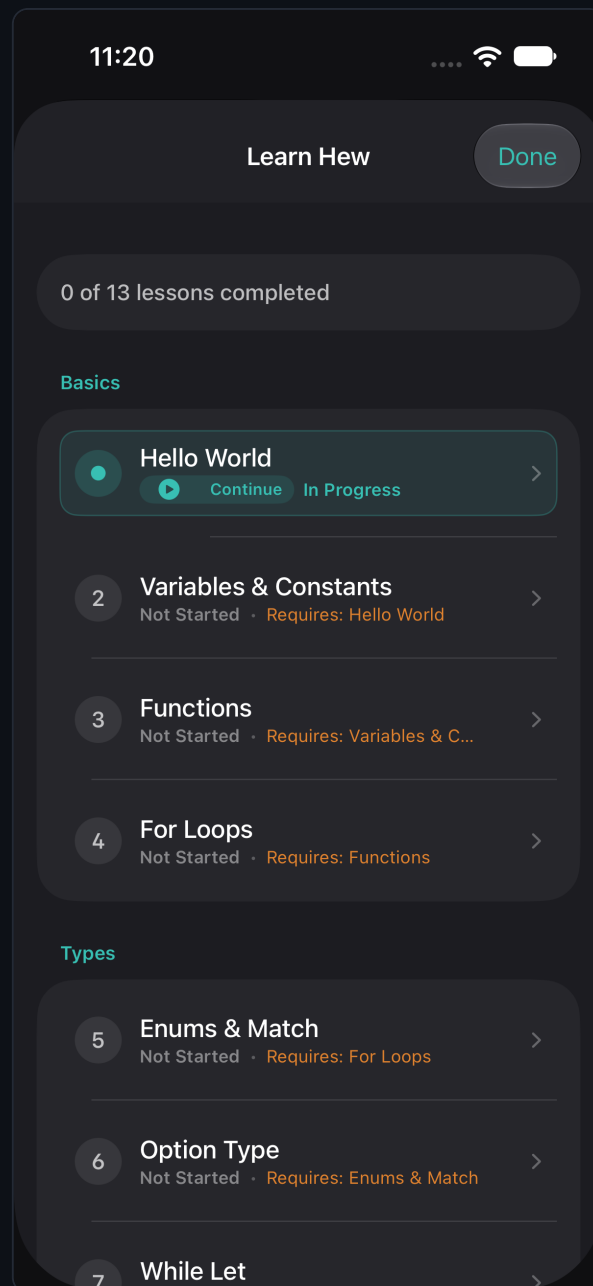
- **Copy Code** copies source text.
- **Copy Link** creates and copies a `hew.run` link.
- **Share Link...** opens the system share sheet for that link.

In Settings, the share format can be **Auto**, **Always Inline** or **Always Server**. Inline links carry compressed source in the URL fragment and can be created offline when the program fits. Auto can upload a larger program to create a server-backed link; Always Server uses the server while connected. Recent iOS builds try an inline link when offline, even if Always Server is selected. That offline link still contains the code in its fragment. If the program does not fit, the app reports the problem; copy the code or export a file instead.

Remote execution and server-backed sharing send source text to the service. Anyone receiving a code link can read its contents; remove passwords, tokens and private application data before sharing.

Learn and use reference material

Open **Library** → **Tutorials** for guided lessons and **Try it in the Editor** to work on one.



Choose a tutorial, read it and load the exercise into the editor.

Documentation → **Quick Ref** contains bundled syntax cards with **Try It** actions. **Docs** and **Learn** open live website pages and need internet; Quick Ref remains available offline.

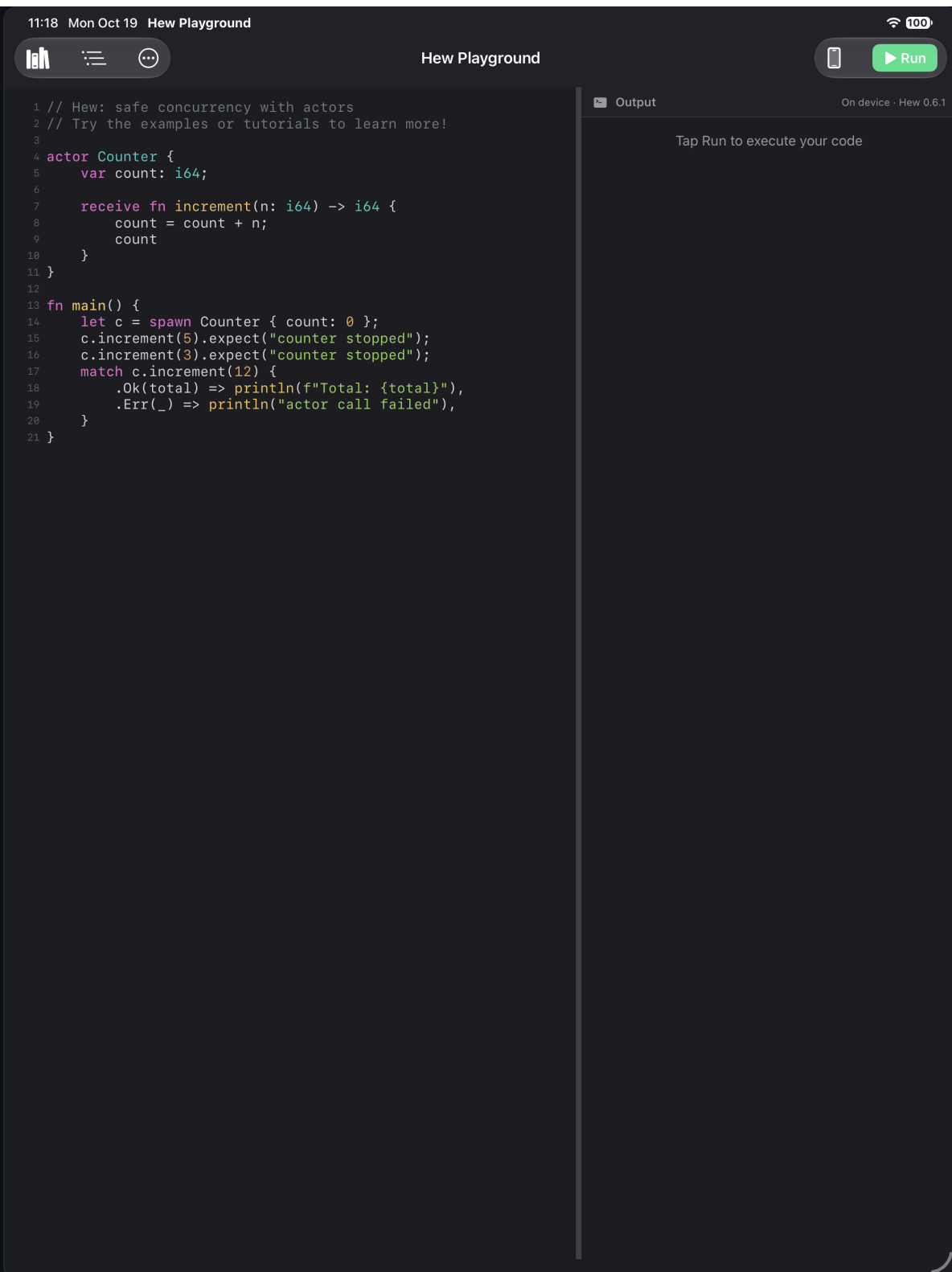
When you want the ideas behind the syntax, start with the [core concepts map](#). Use the [language reference](#) and [standard-library guide](#) when you need exact rules or API details. These website pages track the current Hw release and can be newer than the compiler bundled with your installed app.



Quick Ref supplies offline syntax examples;
Try It loads a snippet into the editor.

Use the iPad layout

Drag the divider between editor and output to give the panel you are using more space. Rotate or use Split View as needed; the toolbar and panels adapt to the available width. A hardware keyboard is optional.



The native iPad layout from the same app capture set.

When something does not work

- **Run needs a connection:** reconnect for the online engine, or select On device if your build offers it and the local engine is ready.
- **Analysis is loading:** wait for the language service to finish starting. Check Settings for its status; unavailable analysis does not mean the source is valid.
- **An example uses unfamiliar syntax:** use examples bundled with your installed build and check the reported compiler version. A website example for a newer compiler may need a newer app.
- **No output:** expand the Output panel and check for compile, runtime or connection errors. A successful program can also finish without printing anything.
- **A shared link fails:** check the connection for server-backed links and ask for the full link, source text or a `.hew` file if decoding fails.

Continue with the [Android guide](#) or [mobile guide overview](#). The [printable manual](#) contains both platform guides.

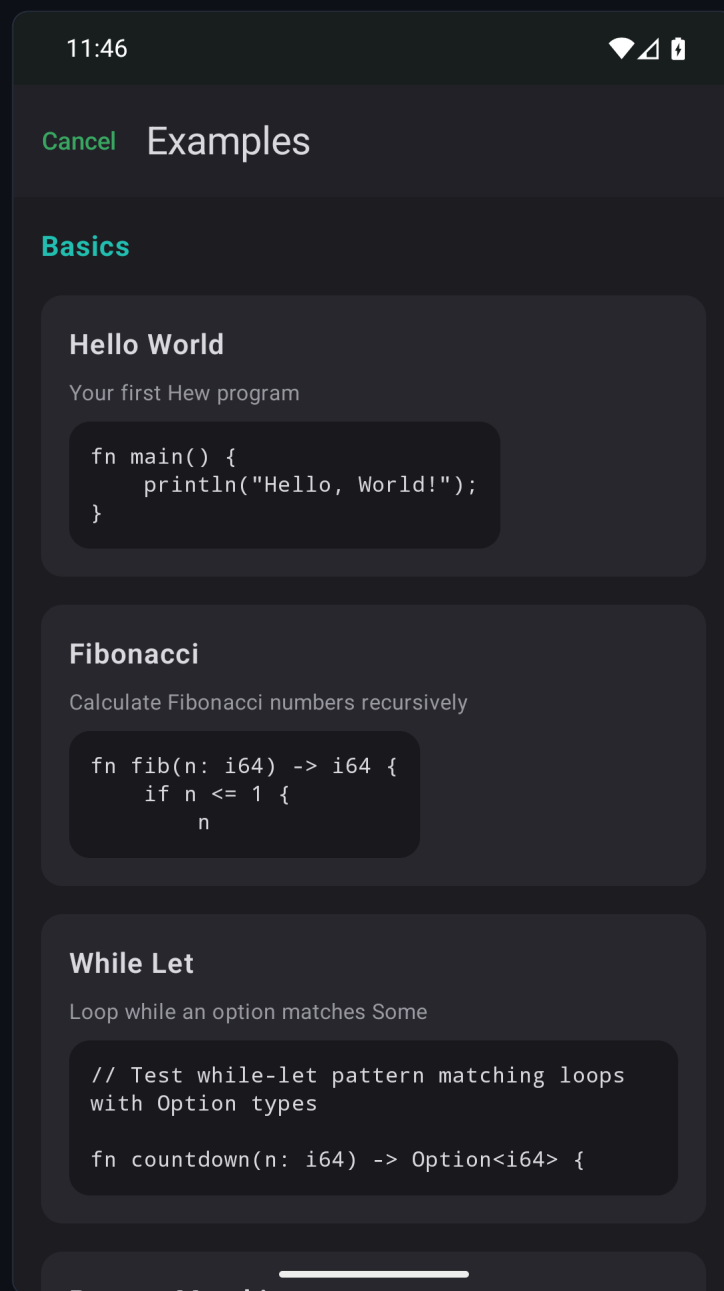
Android

Join the [Hew Playground Android beta](#), follow Google Play's opt-in steps and install the app. Android is in development; this public pre-release can change as work continues. Keep exported copies of code you want to retain.

This guide follows the current Android app source, which bundles the official Hew 0.9.2 compiler and sandbox engine. Its native screenshots show the learner and document screens captured from a signed 0.6.1 build. The beta installed through Play can be an earlier version; names and available features may vary.

Open an example and run it

1. Open **Library** → **Examples**. Older builds show Examples directly in the toolbar.
2. Choose **Hello World** to load it into the editor.
3. Tap **Run** and read the **Output** panel below the code.
4. Change the string printed by `println`, then tap Run again.



Choose a bundled example to start from code that matches the app.

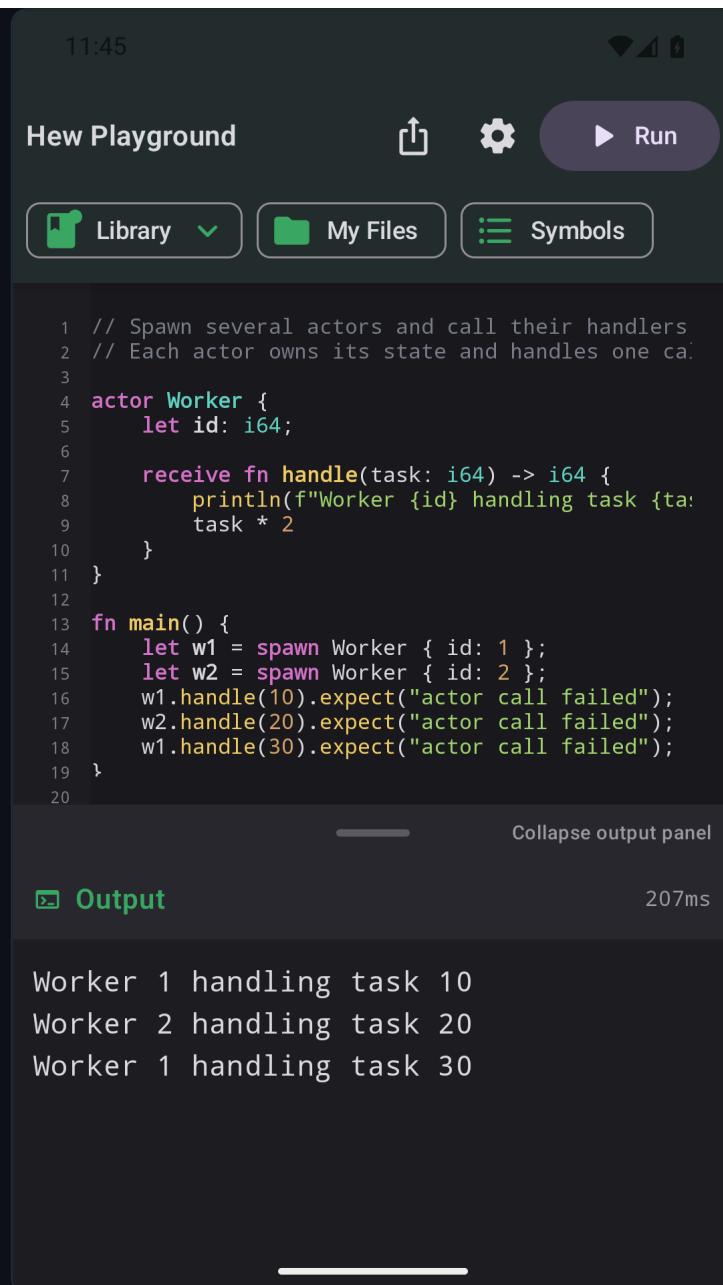
You can also type this small program:

```
fn main() {  
  
    println("Hello from Android!");  
  
}
```

The editor changes to the selected example or lesson. Keep a local file or export the code before replacing work you want to retain.

Edit and navigate code

Tap the editor to type. It provides line numbers, syntax highlighting, auto-indent and bracket/quote pairing. **Settings** controls font size, line numbers and analysis preferences. When the local compiler is ready, completion chips, diagnostics, hover information and the **Symbols** outline help you inspect code. Long-press in the editor for hover information; tap a completion chip to insert it.



The editor and Output panel after running the multiple-actor example.

Drag the divider or use the output-panel control to adjust how much code and output you see. A larger screen gives the adaptive layout more room.

Read output and errors

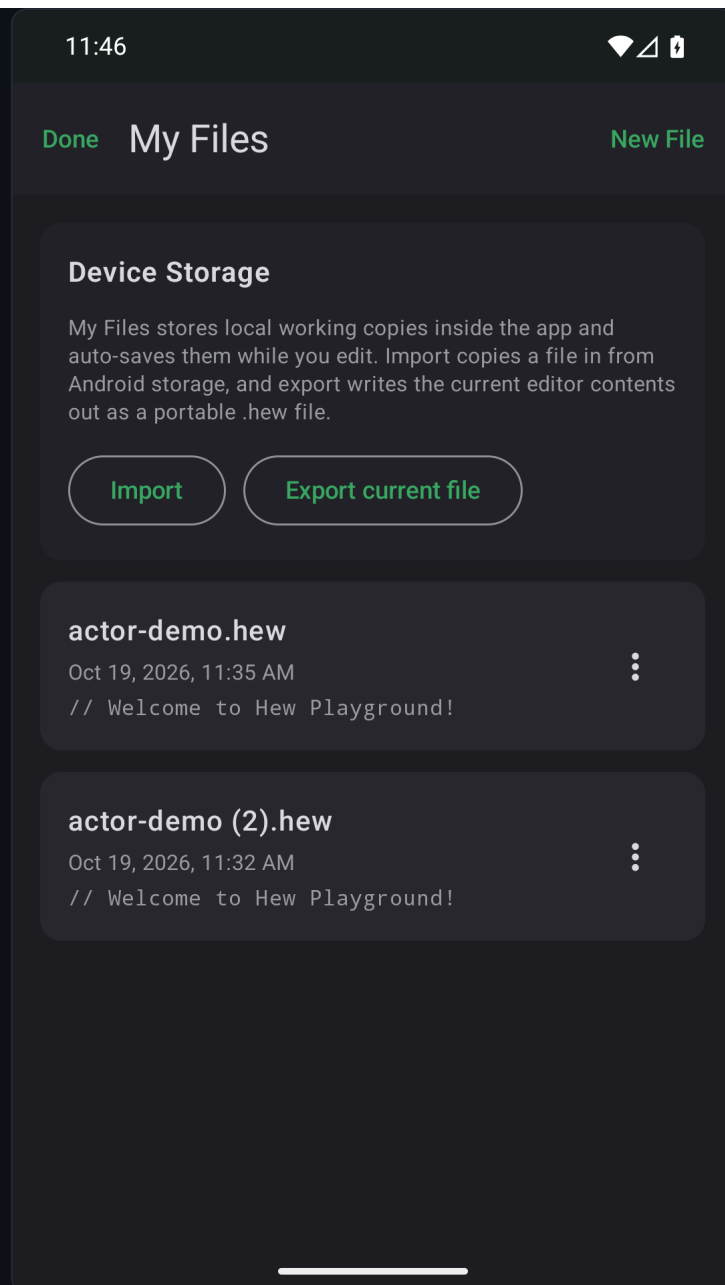
A successful run prints its standard output in **Output**. Compile errors, runtime failures and connection errors appear there instead of being treated as successful output. Fix the first relevant diagnostic, then run again. A program that does not print can finish successfully with an empty output panel.

The current Android implementation tries the on-device sandbox when it is ready. If it is unavailable, or the program is rejected only for native-only capabilities, execution can use the remote service and send the source to the server. Genuine compile errors and other local failures stay visible. Android does not currently expose the iOS On device/Remote engine chooser.

On-device supported programs can run offline after the engine is ready. A remote run needs internet and the service's supported capabilities. If you need to keep code off the network, do not assume pressing Run is local in every situation. Check your installed version and the available engine status before running sensitive source.

Keep local files and export a backup

Choose **My Files**. It stores working copies inside the app and auto-saves them while you edit. Use **New File** to create one, tap a file to reopen it, and use its menu to rename or delete it.



My Files keeps app-local documents. Import and Export current file use Android storage providers.

- **Import** copies a `.hew` file from the Android document picker into the app.
- **Export current file** writes the current editor contents as a portable `.hew` file to the chosen destination.
- Export again after later changes to update an external copy. Cancelling the picker does not create an exported backup.

Local documents are kept in app-private storage and are excluded from Android backup/device transfer in the current implementation. Export important code before

uninstalling, clearing app data or changing devices. If a storage provider fails, retain the working copy and retry elsewhere.

Share a program

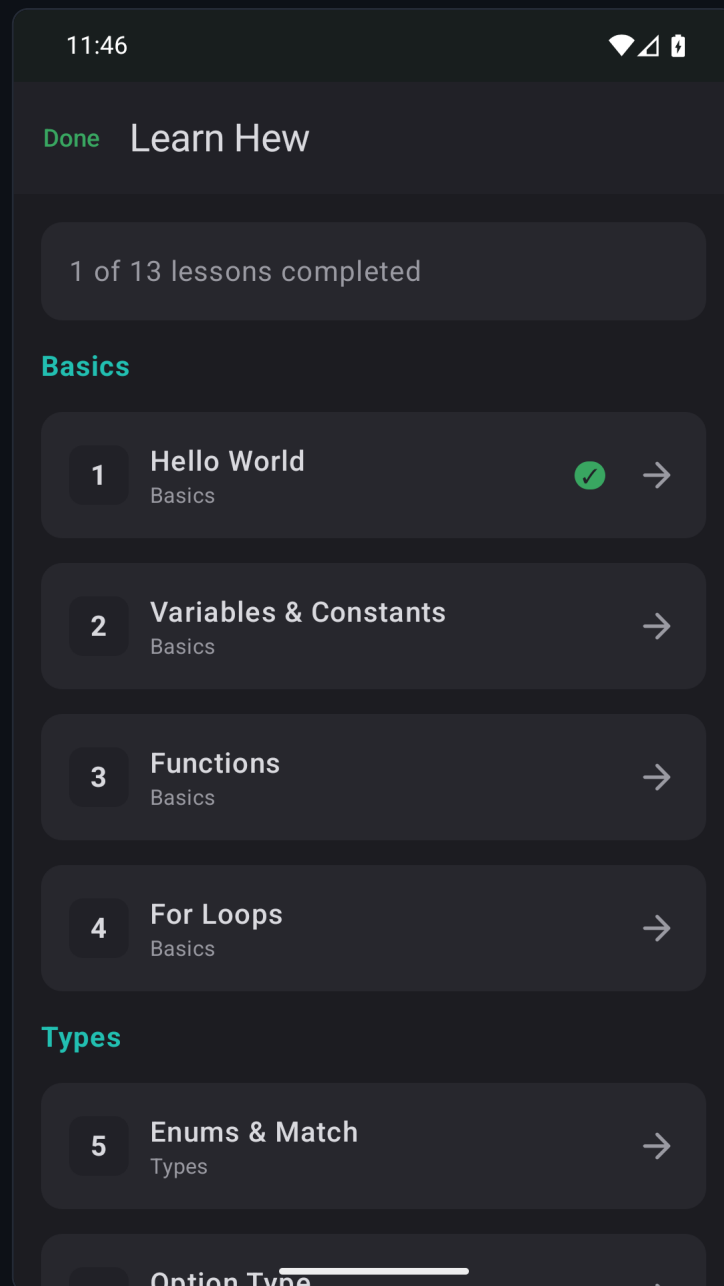
Tap the **Share** icon in the top bar to create a `hew.run` link and open Android's share chooser. In **Settings** → **Sharing & Links**, choose the default share format:

FORMAT	WHAT HAPPENS
Auto	Creates an inline link if it fits; otherwise uploads source for a server-backed link.
Always Inline	Keeps compressed source in the URL fragment; reports an error if the program is too large.
Always Server	Uploads source to create a short link; requires a connection.

An inline link still contains the program. Anyone with the link can read it. Remove secrets and private data before sharing. If link creation fails, check the connection or export a `.hew` file instead. Supported share links can open in the app when **Open share links in-app** is enabled.

Learn with tutorials and Quick Ref

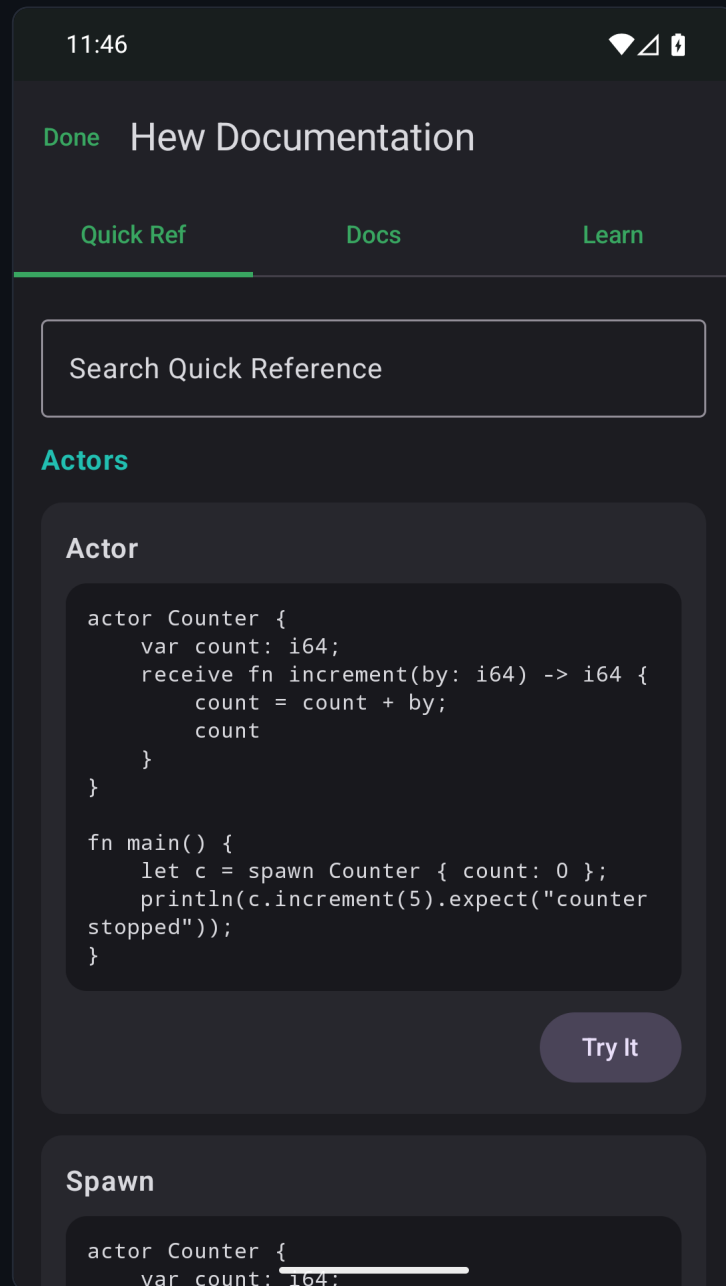
Choose **Library** → **Tutorials** for guided lessons. Read a lesson, then use **Try it in the Editor** to load its code. Tutorial progress and settings are stored locally.



The bundled tutorial list keeps progress on the device.

Library → **Documentation** opens **Quick Ref**, **Docs** and **Learn**. Quick Ref's syntax cards are bundled and available offline; **Try It** loads a snippet. Docs and Learn open live website content and require internet.

When you want the ideas behind the syntax, start with the [core concepts map](#). Use the [language reference](#) and [standard-library guide](#) when you need exact rules or API details. These website pages track the current Hew release and can be newer than the compiler bundled with your installed beta.



Quick Ref works offline. Docs and Learn use the live website.

Troubleshoot a beta session

- **Run is unavailable or fails offline:** wait for the local engine to start; reconnect if the run needs the remote service.
- **Language-service banner:** read its loading/error status. Missing diagnostics are not a successful compile result.
- **Syntax differs from a website example:** start from a bundled example or update the app; the installed beta and website compiler may differ.
- **No printed output:** expand Output and check errors, then try Hello World to distinguish a quiet program from a failed run.
- **Sharing or importing fails:** keep the current code, check the connection or provider, and retry. Use an exported file when a link is unsuitable.

Continue with the [iPhone/iPad guide](#) or [mobile guide overview](#). Both guides are also in the [printable manual](#).