# Windows on ARM

If your app runs with Electron 6.0.8 or later, you can now build it for Windows 10 on Arm. This considerably improves performance, but requires recompilation of any native modules used in your app. It may also require small fixups to your build and packaging scripts.

## Running a basic app

If your app doesn't use any native modules, then it's really easy to create an Arm version of your app.

1. Make sure that your app's `node_modules` directory is empty.
2. Using a *Command Prompt*, run `set npm_config_arch=arm64` before running `npm install`/`yarn install` as usual.
3. If you have Electron installed as a development dependency, npm will download and unpack the arm64 version. You can then package and distribute your app as normal.

## General considerations

### Architecture-specific code

Lots of Windows-specific code contains if… else logic that selects between either the x64 or x86 architectures.

```
if (process.arch === 'x64') {
  // Do 64-bit thing...
} else {
  // Do 32-bit thing...
}
```

If you want to target arm64, logic like this will typically select the wrong architecture, so carefully check your application and build scripts for conditions like this. In custom build and packaging scripts, you should always check the value of `npm_config_arch` in the environment, rather than relying on the current process arch.

### Native modules

If you use native modules, you must make sure that they compile against v142 of the MSVC compiler (provided in Visual Studio 2017). You must also check that any pre-built `.dll` or `.lib` files provided or referenced by the native module are available for Windows on Arm.

### Testing your app

To test your app, use a Windows on Arm device running Windows 10 (version 1903 or later). Make sure that you copy your application over to the target device

- Chromium's sandbox will not work correctly when loading your application assets from a network location.

## Development prerequisites

### Node.js/node-gyp

Node.js v12.9.0 or later is recommended. If updating to a new version of Node is undesirable, you can instead update npm's copy of node-gyp manually to version 5.0.2 or later, which contains the required changes to compile native modules for Arm.

### Visual Studio 2017

Visual Studio 2017 (any edition) is required for cross-compiling native modules. You can download Visual Studio Community 2017 via Microsoft's Visual Studio Dev Essentials program. After installation, you can add the Arm-specific components by running the following from a *Command Prompt*:

```
vs_installer.exe ^
--add Microsoft.VisualStudio.Workload.NativeDesktop ^
--add Microsoft.VisualStudio.Component.VC.ATLMFC ^
--add Microsoft.VisualStudio.Component.VC.Tools.ARM64 ^
--add Microsoft.VisualStudio.Component.VC.MFC.ARM64 ^
--includeRecommended
```

**Creating a cross-compilation command prompt** Setting `npm_config_arch=arm64` in the environment creates the correct arm64 `.obj` files, but the standard *Developer Command Prompt for VS 2017* will use the x64 linker. To fix this:

1. Duplicate the *x64_x86 Cross Tools Command Prompt for VS 2017* shortcut (e.g. by locating it in the start menu, right clicking, selecting *Open File Location*, copying and pasting) to somewhere convenient.
2. Right click the new shortcut and choose *Properties*.
3. Change the *Target* field to read `vcvarsamd64_arm64.bat` at the end instead of `vcvarsamd64_x86.bat`.

If done successfully, the command prompt should print something similar to this on startup:

```
**********************************************************************
** Visual Studio 2017 Developer Command Prompt v15.9.15
** Copyright (c) 2017 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x64_arm64'
```

If you want to develop your application directly on a Windows on Arm device, substitute `vcvarsx86_arm64.bat` in *Target* so that cross-compilation can happen with the device's x86 emulation.

#### Linking against the correct `node.lib`

By default, `node-gyp` unpacks Electron's node headers and downloads the x86 and x64 versions of `node.lib` into `%APPDATA%\..\Local\node-gyp\Cache`, but it does not download the arm64 version (a fix for this is in development.) To fix this:

1. Download the arm64 `node.lib` from https://electronjs.org/headers/v6.0.9/win-arm64/node.lib
2. Move it to `%APPDATA%\..\Local\node-gyp\Cache\6.0.9\arm64\node.lib`

Substitute `6.0.9` for the version you're using.

## Cross-compiling native modules

After completing all of the above, open your cross-compilation command prompt and run `set npm_config_arch=arm64`. Then use `npm install` to build your project as normal. As with cross-compiling x86 modules, you may need to remove `node_modules` to force recompilation of native modules if they were previously compiled for another architecture.

## Debugging native modules

Debugging native modules can be done with Visual Studio 2017 (running on your development machine) and corresponding Visual Studio Remote Debugger running on the target device. To debug:

1. Launch your app `.exe` on the target device via the *Command Prompt* (passing `--inspect-brk` to pause it before any native modules are loaded).
2. Launch Visual Studio 2017 on your development machine.
3. Connect to the target device by selecting *Debug > Attach to Process...* and enter the device's IP address and the port number displayed by the Visual Studio Remote Debugger tool.
4. Click *Refresh* and select the appropriate Electron process to attach.
5. You may need to make sure that any symbols for native modules in your app are loaded correctly. To configure this, head to *Debug > Options...* in Visual Studio 2017, and add the folders containing your `.pdb` symbols under *Debugging > Symbols*.
6. Once attached, set any appropriate breakpoints and resume JavaScript execution using Chrome's remote tools for Node.

## Getting additional help

If you encounter a problem with this documentation, or if your app works when compiled for x86 but not for arm64, please file an issue with "Windows on Arm" in the title.