Build Instructions (Windows)

Follow the guidelines below for building **Electron itself** on Windows, for the purposes of creating custom Electron binaries. For bundling and distributing your app code with the prebuilt Electron binaries, see the application distribution guide.

Prerequisites

- Windows 10 / Server 2012 R2 or higher
- Visual Studio 2017 15.7.2 or higher - download VS 2019 Community Edition for free
  - See the Chromium build documentation for more details on which Visual Studio components are required.
  - If your Visual Studio is installed in a directory other than the default, you'll need to set a few environment variables to point the toolchains to your installation path.
    - `vs2019_install = DRIVE:\path\to\Microsoft Visual Studio\2019\Community`, replacing `2019` and `Community` with your installed versions and replacing `DRIVE:` with the drive that Visual Studio is on. Often, this will be `C:`.
    - `WINDOWSSDKDIR = DRIVE:\path\to\Windows Kits\10`, replacing `DRIVE:` with the drive that Windows Kits is on. Often, this will be `C:`.

- Node.js
- Git
- Debugging Tools for Windows of Windows SDK 10.0.15063.468 if you plan on creating a full distribution since `symstore.exe` is used for creating a symbol store from `.pdb` files.
  - Different versions of the SDK can be installed side by side. To install the SDK, open Visual Studio Installer, select `Modify` → `Individual Components`, scroll down and select the appropriate Windows SDK to install. Another option would be to look at the Windows SDK and emulator archive and download the standalone version of the SDK respectively.
  - The SDK Debugging Tools must also be installed. If the Windows 10 SDK was installed via the Visual Studio installer, then they can be installed by going to: `Control Panel` → `Programs` → `Programs and Features` → Select the "Windows Software Development Kit" → `Change` → `Change` → Check "Debugging Tools For Windows" → `Change`. Or, you can download the standalone SDK installer and use it to install the Debugging Tools.

If you don't currently have a Windows installation, dev.microsoftedge.com has timebombed versions of Windows that you can use to build Electron.

Building Electron is done entirely with command-line scripts and cannot be done with Visual Studio. You can develop Electron with any editor but support for building with Visual Studio will come in the future.

Note: Even though Visual Studio is not used for building, it's still **required** because we need the build toolchains it provides.

Exclude source tree from Windows Security

Windows Security doesn't like one of the files in the Chromium source code (see https://crbug.com/441184), so it will constantly delete it, causing `gclient sync` issues. You can exclude the source tree from being monitored by Windows Security by following these instructions.

Building

See Build Instructions: GN

32bit Build

To build for the 32bit target, you need to pass `target_cpu = "x86"` as a GN arg. You can build the 32bit target alongside the 64bit target by using a different output directory for GN, e.g. `out/Release-x86`, with different arguments.

```
$ gn gen out/Release-x86 --args="import(\"//electron/build/args/release.gn\")
target_cpu=\"x86\""
```

The other building steps are exactly the same.

Visual Studio project

To generate a Visual Studio project, you can pass the `--ide=vs2017` parameter to `gn gen`:

```
$ gn gen out/Testing --ide=vs2017
```

Troubleshooting

Command xxxx not found

If you encountered an error like `Command xxxx not found`, you may try to use the `VS2015 Command Prompt` console to execute the build scripts.

Fatal internal compiler error: C1001

Make sure you have the latest Visual Studio update installed.

LNK1181: cannot open input file 'kernel32.lib'

Try reinstalling 32bit Node.js.

Error: ENOENT, stat 'C:\Users\USERNAME\AppData\Roaming\npm'

Creating that directory should fix the problem:

```
$ mkdir ~\AppData\Roaming\npm
```

node-gyp is not recognized as an internal or external command

You may get this error if you are using Git Bash for building, you should use PowerShell or VS2015 Command Prompt instead.

cannot create directory at '...': Filename too long

node.js has some extremely long pathnames, and by default git on windows doesn't handle long pathnames correctly (even though windows supports them). This should fix it:

```
$ git config --system core.longpaths true
```

error: use of undeclared identifier 'DefaultDelegateCheckMode'

This can happen during build, when Debugging Tools for Windows has been installed with Windows Driver Kit. Uninstall Windows Driver Kit and install Debugging Tools with steps described above.

ImportError: No module named win32file

Make sure you have installed `pywin32` with `pip install pywin32`.

Build Scripts Hang Until Keypress

This bug is a "feature" of Windows' command prompt. It happens when clicking inside the prompt window with `QuickEdit` enabled and is intended to allow selecting and copying output text easily. Since each accidental click will pause the build process, you might want to disable this feature in the command prompt properties.