To build for a target architecture different than the host (e.g. using x64 to build for ARM), you'll need to do the following, in addition to the setup under How to Contribute:

One-Time Setup

1. Install build toolchain and chroot/rootfs prerequisites:

```
sudo apt-get install qemu qemu-user-static debootstrap gcc-arm-linux-
gnueabihf g++-arm-linux-gnueabihf
```

2. Create a chroot/rootfs for the target architecture:

```
sudo qemu-debootstrap --arch=armhf --variant=minbase xenial rootfs
```

3. Install libx11-dev on the chroot/rootfs:

```
sudo chroot rootfs apt-get install -y libx11-dev
```

Build

Because cross-compiling isn't officially supported by the Visual Studio Code team, some workarounds are required to make the app build correctly:

1. Point to the target toolchain on the build host:

```
export CC=$(which arm-linux-gnueabihf-gcc)
export CXX="$(which arm-linux-gnueabihf-g++) -L$(pwd)/rootfs/usr/lib/arm-
linux-gnueabihf/"
```

   note the -L linker argument pointing to the absolute path of libx11 on the chroot/rootfs

2. Tell `yarn` you want to cross-compile native modules for ARM:

```
export npm_config_arch=arm
```

3. Build VS Code and create a .deb file (for easier installation on the target device) as usual:

```
yarn
yarn run gulp vscode-linux-arm-min
yarn run gulp vscode-linux-arm-build-deb
```