

# **shiny to EXE**

**Zarathu 김진환, 2023-04-11**

# 개요

## shiny 공홈 설명



Shiny is an R package that makes it easy to  
build interactive web apps straight from R.

# 개요

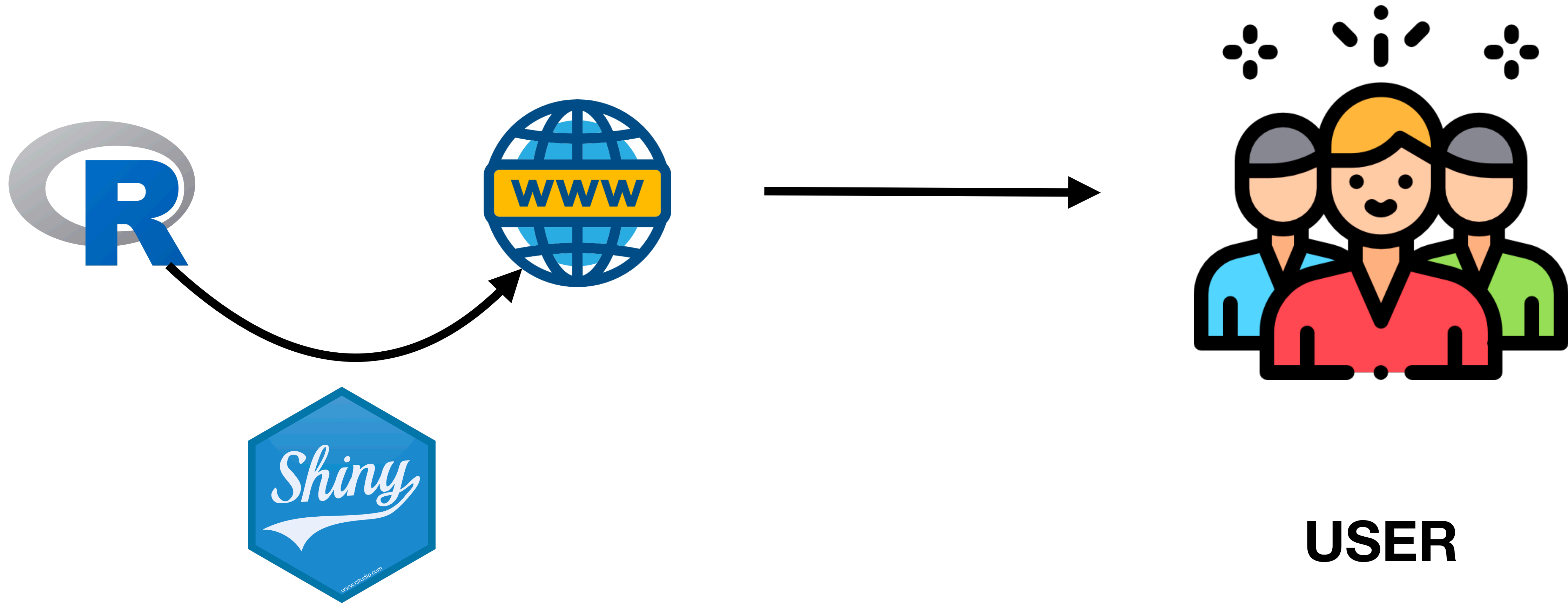
## shiny의 목적



R의 기능을 웹에서 이용할 수 있게 하는 Shiny

# 개요

## shiny의 주요 사용자



R의 기능은 이용하고 싶지만, R을 알고 싶진 않은 사용자들이 대상

# 1가지 전제조건

interactive web app을 사용하기 위한

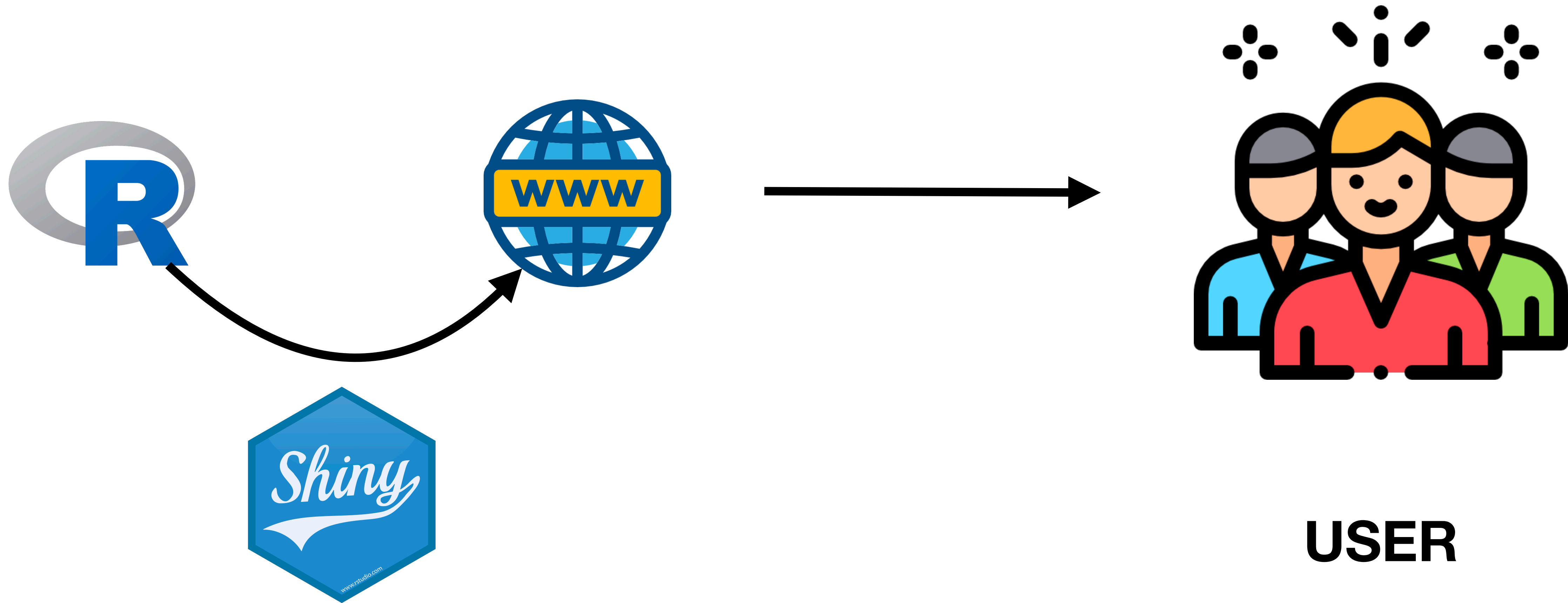


네트워크가 (인터넷) 연결 되어 있음

로컬의 app.R 을 chrome으로 실행 할 수도 있음

# 개요

## shiny의 주요 사용자



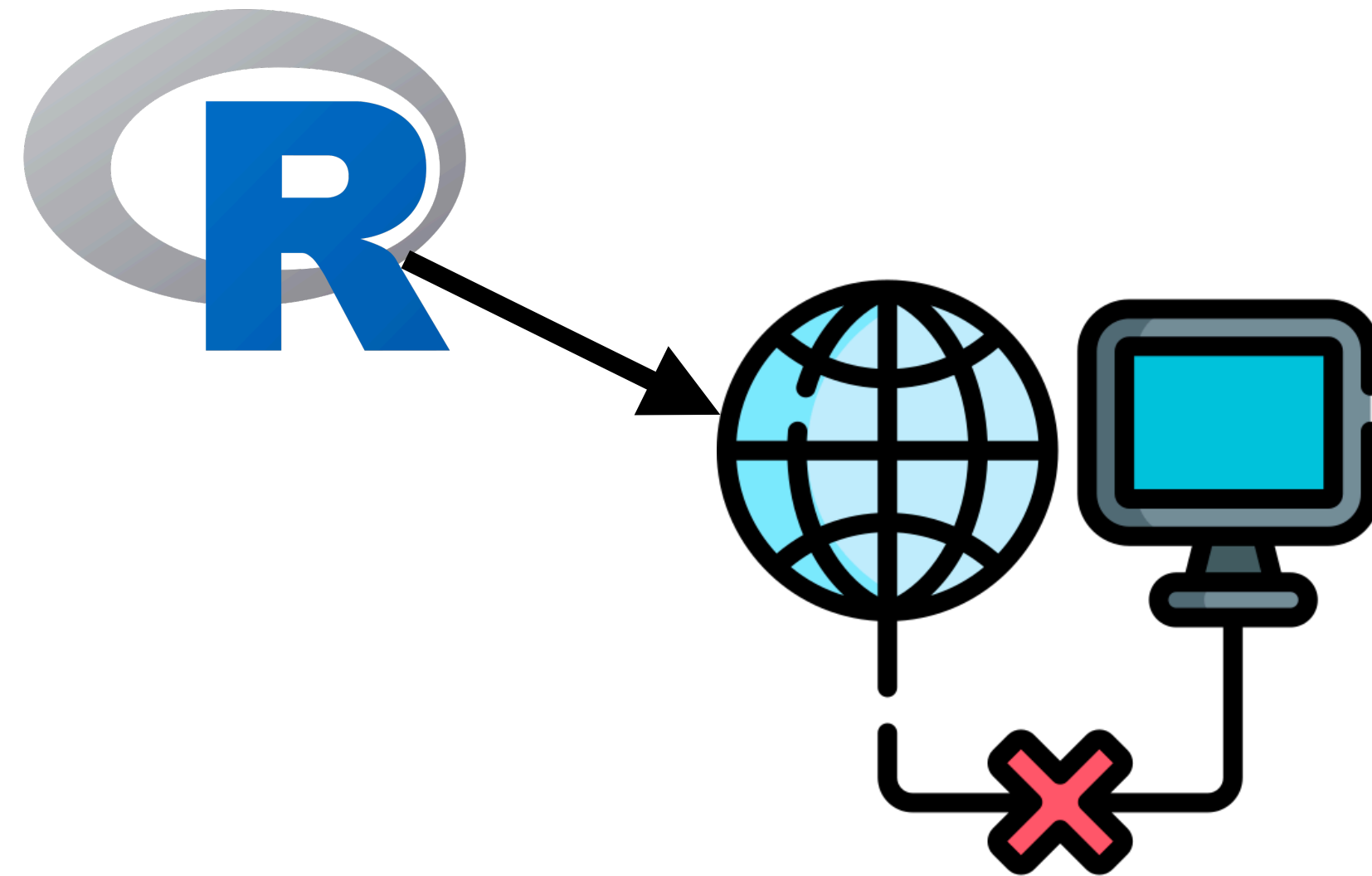
R의 기능은 이용하고 싶지만, R을 알고 싶진 않은 사용자들이 대상

# 1가지 전제조건



네트워크가 (인터넷) 연결 되어 있음

로컬의 app.R 을 chrome으로 실행 할 수도 있음





(보안상) 망분리가 필요한 경우

- 방산, 금융, 보안, 의료, 공공기관 등...
- 밖에서 빌드 후 반입
- shiny를 위해 R을 설치 !?



# shiny to exe (with electron)

 shiny application을 executable file로 만드는 방법은 어떻게 있어

 Shiny 애플리케이션을 실행 가능한 파일(exe)로 만드는 방법은 크게 두 가지가 있습니다.

1. Electron-forge를 사용한 패키징

Electron-forge는 Electron.js를 사용하여 데스크톱 애플리케이션을 만들기 위한 도구이며, Shiny 애플리케이션도 포함될 수 있습니다. 이를 사용하면 Shiny 애플리케이션을 패키징하여 실행 가능한 파일로 만들 수 있습니다.

다음은 Electron-forge를 사용하여 Shiny 애플리케이션을 패키징하는 방법입니다.

1. Electron-forge를 설치합니다.

```
npm install -g electron-forge
```

2. Shiny 애플리케이션을 Electron.js로 변환합니다. 이를 위해 electron-builder를 사용하고, Shiny 애플리케이션을 로드하는 코드를 추가합니다.

3. 애플리케이션을 패키징합니다.

```
go
```

```
electron-forge package
```

2. shinyjs 패키지를 사용한 패키징

shinyjs 패키지를 사용하여 Shiny 애플리케이션을 실행 가능한 파일로 만들 수 있습니다. Shiny 애플리케이션에서 JavaScript 코드를 실행할 수 있는 shinyjs 패키지를 사용합니다.

다음은 shinyjs를 사용하여 Shiny 애플리케이션을 패키징하는 방법입니다.

1. shinyjs 패키지를 설치합니다.

```
arduino
```

```
install.packages("shinyjs")
```

```
# Start the `shiny` "thread"
$shiny = Start-Job -Name "shiny" -ArgumentList($pwd) -ScriptBlock{
    param($workingdir);
    cd $workingdir;
    # Start the shiny app and print the URL
    & "rscript.exe" -e "shiny::runApp('.')" 2>&1 | % {
        if($_ -like '*Listening on*'){
            ("$_" -replace ".*Listening on ","').Trim()
        }
    }
}
```

아직 잘 모르겠음

At this point Shiny is started as a `job`, and jobs maintain a reference which can then be used to stop the job at a later time. This does add the problem that we need to read from the output stream slightly differently.

 rstudio.com  
<https://community.rstudio.com/convert-r-shiny-app-to-exe>

## Convert R-shiny app to ".exe" file or hide source code

2021. 12. 17. — Convert R-shiny app to ".exe" file Are there any new possibilities with R 4.x Why can't we create ".exe" for R-shiny apps, unlike Python app?



[https://community.rstudio.com/how-to-convert-r-shi...](https://community.rstudio.com/how-to-convert-r-shiny-app-to-exe)

## how to convert r shiny application to exe file

2019. 7. 17. — Hi All, Please any one help me how to convert r shiny application to exe file.

 stackoverflow.com  
[https://stackoverflow.com/questions/how-to-run-a-...](https://stackoverflow.com/questions/how-to-run-a-shiny-app-as-a-standalone-application)

## How to run a shiny app as a standalone application?

2021. 10. 10. — I've some shiny app and I want to execute and to make it standalone application (it will be awesome if it will open via chrome).

답변 1개 · 인기 답변: I have done some research on this issue. The commenters are basically correct: ...

<a href="#">possible to run RShiny app without opening an R environment?</a>	2013년 4월 8일
<a href="#">Deploying R shiny app as a standalone application [closed]</a>	2015년 11월 4일
<a href="#">Call exe in R shiny - Stack Overflow</a>	2018년 4월 18일
<a href="#">How to package "shiny" in R program? - Stack Overflow</a>	2022년 10월 31일

[stackoverflow.com](#) 검색결과 더보기

 medium.com  
[https://jefferey-cave.medium.com/turn-a-shiny-dash...](https://jefferey-cave.medium.com/turn-a-shiny-dashboard-into-a-desktop-app)


## Turn a Shiny Dashboard into a Desktop App | by Jefferey Cave

2021. 11. 6. — rscript.exe -e "shiny::runApp('.')" Your app should be started, but without having to have the customer load the IDE:

 pconlife.com  
<https://www.pconlife.com/viewfileinfo/shiny-exe>

## shiny.exe File Download & Fix For All Windows OS

shiny.exe. Comments. -. InternalName. Adobe Flash Player 10.1. ProductName. Shockwave Flash. CompanyName. Adobe Systems, Inc. LegalCopyright. Adobe? Flash?

 foretodata.com  
[https://foretodata.com/how-to-make-a-standalone-de...](https://foretodata.com/how-to-make-a-standalone-desktop-app-with-shiny)

## How To Make A Standalone Desktop Application With Shiny ...

2020. 6. 30. — How To Make A Standalone Desktop App Using Shiny and Electron. ... Double click the .exe file and it will open your app.

 github.com  
<https://github.com/lawalter/r-shiny-electron-app>

## How to Make an R Shiny Electron App - GitHub

2022. 4. 27. — memo: Guide to desktop app creation using R Shiny and Electron - GitHub ... Package and create the .exe on the command line with ...

이것 참고



# shiny to exe

## 방법들

- shinyShortcut -> 6년 전
- **electron-forge** -> 3년 전
- Rinno -> 5 ~ 7년 전
- shinyproxy -> Docker (Cloud)
- shinyjs -> ?
- command -> 사용자가 cmd에 익숙 +. R 설치가 필요



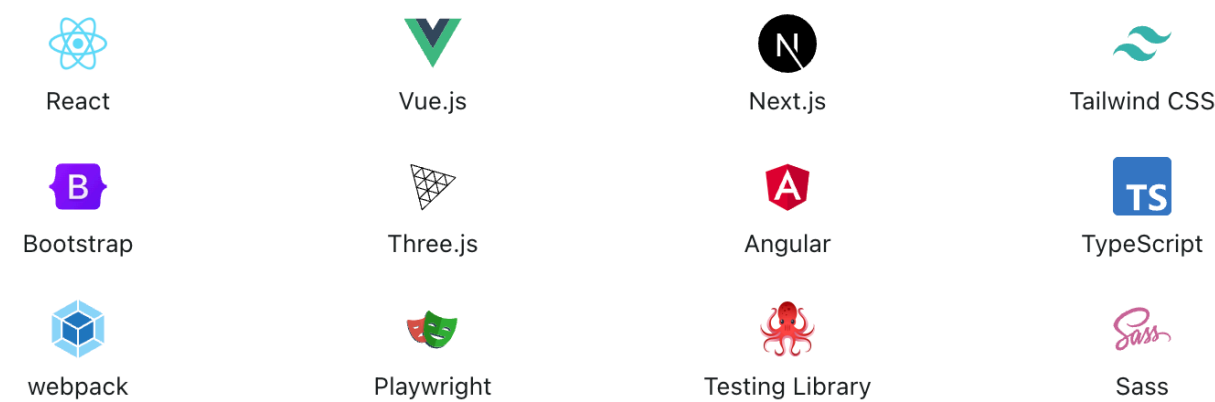
# electron.js

## web app to Desktop App




### Use the tools you love


With the power of modern Chromium, Electron gives you an unopinionated blank slate to build your app. Choose to integrate your favourite libraries and frameworks from the front-end ecosystem, or carve your own path with bespoke HTML code.



openjs foundation

MIT License

electron.js가 작동하는 원리를 설명해줘

Electron.js는 Chromium 브라우저와 Node.js 런타임을 기반으로한 오픈소스 프레임워크로서, 데스크톱 애플리케이션을 개발하기 위한 도구입니다.

Electron.js의 작동 원리는 크게 3단계로 나눌 수 있습니다.

1. 애플리케이션 초기화  
Electron.js는 먼저 Chromium 브라우저를 실행하고, Node.js를 로드합니다. 그리고 애플리케이션의 메인 프로세스(Main Process)를 시작합니다. 메인 프로세스는 사용자 인터페이스(UI)를 생성하고, 애플리케이션의 동작을 제어합니다.
2. 렌더러 프로세스 생성  
Electron.js는 Chromium 브라우저에서의 웹 페이지와 같이, 렌더러 프로세스(Renderer Process)를 생성합니다. 이는 Chromium의 멀티 프로세스 아키텍처를 따르며, 각각의 렌더러 프로세스는 독립된 JavaScript 실행 환경을 갖습니다.
3. 메인 프로세스와 렌더러 프로세스 간 통신  
메인 프로세스와 렌더러 프로세스 간의 통신은 Electron.js에서 중요한 역할을 합니다. 렌더러 프로세스에서는 HTML, CSS, JavaScript를 이용하여 UI를 구성하며, 이를 통해 메인 프로세스에 요청을 보내고, 응답을 받습니다. 이를 통해 렌더러 프로세스는 메인 프로세스에서 제공하는 다양한 기능을 사용할 수 있습니다.

Electron.js의 작동 원리는 이와 같이 간단합니다. Chromium 브라우저와 Node.js를 이용하여 데스크톱 애플리케이션을 개발하면서, 메인 프로세스와 렌더러 프로세스 간의 통신을 통해 유연하게 개발할 수 있습니다.

이해 못해도 빌드에 큰 지장 없었음

# r-shiny-electron-app

<https://github.com/lawalter/r-shiny-electron-app>

This repository has been archived by the owner on Apr 28, 2022. It is now read-only.

lawalter / r-shiny-electron-app Public archive Watch 2

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

master 2 branches 0 tags [Go to file](#) [Code](#)

lawalter Update README.md 4766a45 on Mar 19, 2020 92 commits

shiny	Update app.R	3 years ago
src	added linux option to launch shiny	3 years ago
.gitignore	removed folder	3 years ago
LICENSE	Update LICENSE	3 years ago
README.md	Update README.md	3 years ago
add-cran-binary-pkgs.R	Update add-cran-binary-pkgs.R	3 years ago
get-r-linux.sh	Add files via upload	3 years ago
get-r-mac.sh	Update get-r-mac.sh	3 years ago
get-r-win.sh	Update get-r-win.sh	3 years ago
r-shiny-electron-app.Rproj	removed folder	3 years ago
start-shiny.R	Update start-shiny.R	3 years ago

2020년 3월에 최종수정

업데이트가 필요

# [TL;DR] shiny electron template

<https://github.com/zarathucorp/shiny-electron-template-m1-2023>



Repo에 가이드 써놓음

## Creating Standalone Apps from Shiny with Electron [2023, macOS M1]

Posted on March 21, 2023 by jhk0530 in R bloggers | 0 Comments

[This article was first published on [R-posts.com](#), and kindly contributed to [R-bloggers](#). (You can report issue about the content on this page [here](#))

Want to share your content on R-bloggers? [click here](#) if you have a blog, or [here](#) if you don't.

f Share

Tweet

- 정리된 글 (영어):



- <https://www.r-bloggers.com/2023/03/creating-standalone-apps-from-shiny-with-electron-2023-macos-m1/>

# make APP

## 0. 개발 환경 설정

- R, Rstudio 최신으로 업데이트 (*Terminal* 기능 활용)
- **Node, NPM** 설치 (Node Package Manager : CRAN과 비슷)
  - Terminal에서 node -v, npm -v로 버전 확인
  - 18.12.0 & 9.5.1 -> NPM 버전이 그새 오름...
- Repo Fork & Clone
  - fork 안해도 될 듯..?

# make APP

## 1. 템플릿 빌드

- Rproj를 열고
- > npx create-electron-app helloworld
  - 시간 엄청 걸림
- helloworld의 파일을 repo의 파일로 수정
- start\_shiny.R
- add\_cran-binary\_pkgs.R
- get-r-mac.sh
- /shiny
- /src



# make APP

## 2. EXE R 설치

- > sh ./get-r-mac.sh
  - /r-mac 디렉토리 확인
- automagic R 패키지 설치

template...의 코드를 기반으로 R app을 만들기

-> R 을 설치

get-r-mac.sh

```
# Copyright (c) 2018 Dirk Schumacher, Noam Ross, Rich FitzJohn

#!/usr/bin/env bash
set -e

# Download and extract the main Mac Resources directory
mkdir -p r-mac
curl -o r-mac/latest_r.pkg \
https://cloud.r-project.org/bin/macosx/big-sur-arm64/base/R-4.2.2-arm64.pkg

cd r-mac
xar -xf latest_r.pkg
rm -r Resources tcltk.pkg texinfo.pkg Distribution latest_r.pkg
# cat R-app.pkg/Payload | gunzip -dc | cpio -i
cat R-fw.pkg/Payload | gunzip -dc | cpio -i
mv R.framework/Versions/Current/Resources/* .
rm -r R-fw.pkg R.framework

# Patch the main R script
sed -i.bak '/^R_HOME_DIR=/d' bin/R
sed -i.bak 's;/Library/Frameworks/R.framework/Resources;${R_HOME};g' \
bin/R
chmod +x bin/R
rm -f bin/R.bak

# Remove unnecessary files
rm -r doc tests
rm -r lib/*.dSYM
```



# make APP

## 3. JSON 수정

- electron node 관련 작업 (R package의 description)
- **dependencies, devDependencies 업데이트**
  - **template repo에 있음.**
- license를 수정하지 않으면 빌드할때 warning
  - 지장은 없음
- 이후 > sudo npm install
  - 시간 조금 걸림

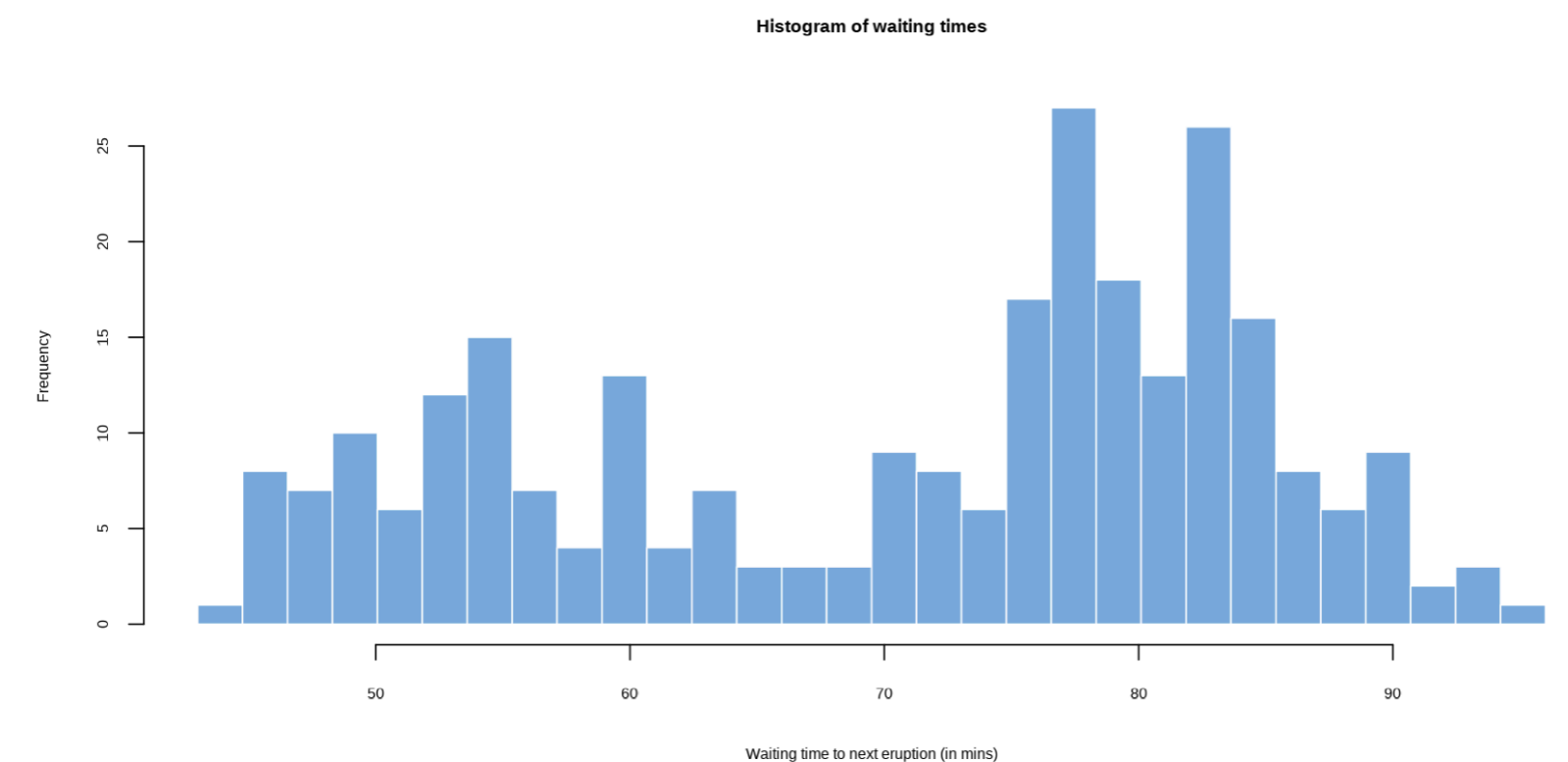
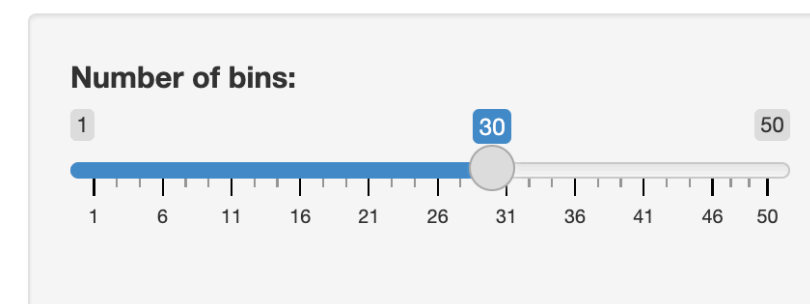
```
package.json
1 {
2   "name": "helloworld",
3   "productName": "helloworld",
4   "version": "1.0.0",
5   "description": "My Electron application description",
6   "main": "src/index.js",
7   "scripts": {
8     "start": "electron-forge start",
9     "package": "electron-forge package",
10    "make": "electron-forge make",
11    "publish": "electron-forge publish",
12    "lint": "echo \\\"No linting configured\\\""
13  },
14  "keywords": [],
15  "author": {
16    "name": "Jinhwan Kim",
17    "email": "hwanistic@gmail.com"
18  },
19  "license": "MIT",
20  "repository": {
21    "type": "git",
22    "url": "https://github.com/jhk0530/some_repo"
23  },
24  "dependencies": {
25    "axios": "0.27.2",
26    "esm": "^3.2.25",
27    "execa": "^5.1.1",
28    "electron-squirrel-startup": "^1.0.0"
29  },
30  "devDependencies": {
31    "@babel/core": "^7.21.0",
32    "@babel/plugin-transform-async-to-generator": "^7.20.7",
33    "@babel/preset-env": "^7.20.2",
34    "@babel/preset-react": "^7.18.6",
35    "eslint": "^8.35.0",
36    "eslint-config-airbnb": "^19.0.4",
37    "eslint-plugin-import": "^2.27.5",
38    "eslint-plugin-jsx-a11y": "^6.7.1",
39    "eslint-plugin-react": "^7.32.2",
40    "eslint-plugin-react-hooks": "^4.6.0",
41    "fs-extra": "^11.1.0",
42    "@electron-forge/cli": "^6.0.5",
43    "@electron-forge/maker-deb": "^6.0.5",
44    "@electron-forge/maker-rpm": "^6.0.5",
45    "@electron-forge/maker-squirrel": "^6.0.5",
46    "@electron-forge/maker-zip": "^6.0.5",
47    "electron": "23.1.3"
48  }
49 }
50
```

# make APP

## 4. make shiny App

- `shiny::runExample('01_hello')`
- 코드를 바로 제공 하기 때문에 그대로 활용
- `hellowor Id/shiny/app.R`
  - template 엔 이미 적용되어 있음

Hello Shiny!



# make APP

## 5. EXE R shiny App 빌드

- > Rscript add-cran-binary-pkgs.R
- 앱을 위한 CRAN R패키지를 설치 (automagic사용)
- CRAN 패키지가 아닌 경우...? (테스트 안해봄)
- > electron-forge start 로 EXE 테스트
  - 실행 안되는 경우 Rstudio restart 하면 됨
- 실행 되면 > electron-forge make 로 EXE 빌드
  - /out에서 앱 확인

```
add-cran-binary-pkgs.R

# Copyright (c) 2018 Dirk Schumacher, Noam Ross, Rich FitzJohn

# !/usr/bin/env Rscript

# Script to find dependencies of a pkg list, download binaries and put them
# In the standalone R library.

library(automagic)

options(repos = "https://cloud.r-project.org")

cran_pkgs <- setdiff(unique(c("shiny", automagic::get_dependent_packages("shiny"))), "automagic")

install_bins <- function(cran_pkgs, library_path, type, decompress,
                          remove_dirs = c(
                            "help", "doc", "tests", "html",
                            "include", "unitTests",
                            file.path("libs", "*dSYM")
                          )) {
  installed <- list.files(library_path) # check installed packages
  cran_to_install <- sort(setdiff(
    unique(unlist(
      c(
        cran_pkgs,
        tools::package_dependencies(cran_pkgs,
          recursive = TRUE,
          which = c("Depends", "Imports", "LinkingTo")
        )
      )
    ),
    installed
  ))
  if (!length(cran_to_install)) {
    message("No packages to install")
  } else {
    td <- tempdir()
    downloaded <- download.packages(cran_to_install, destdir = td, type = type)
    apply(downloaded, 1, function(x) decompress(x[2], exdir = library_path))
    unlink(downloaded[, 2])
  }
  z <- lapply(
    list.dirs(library_path, full.names = TRUE, recursive = FALSE),
    function(x) {
      unlink(file.path(x, remove_dirs), force = TRUE, recursive = TRUE)
    }
  )
  invisible(NULL)
}

# https://cloud.r-project.org/bin/macosx/big-sur-arm64/contrib/

if (dir.exists("r-mac")) {
  install_bins(
    cran_pkgs = cran_pkgs, library_path = file.path("r-mac", "library"),
    type = "mac.binary.big-sur-arm64", decompress = untar
  )
}

if (dir.exists("r-win")) {
  install_bins(
    cran_pkgs = cran_pkgs, library_path = file.path("r-win", "library"),
    type = "win.binary", decompress = unzip
  )
}
```

# Standalone의 Advantage

## 1. Security

- 네트워크 연결이 없으므로 외부 유출이 없음.

## 2. User Experience

- Shiny는 결국 R 코드 (그 실행 방법을) 를 공유 하는 방법
- Install {R, Rstudio, Package, Github, Dependency}, Run VS unzip, Run

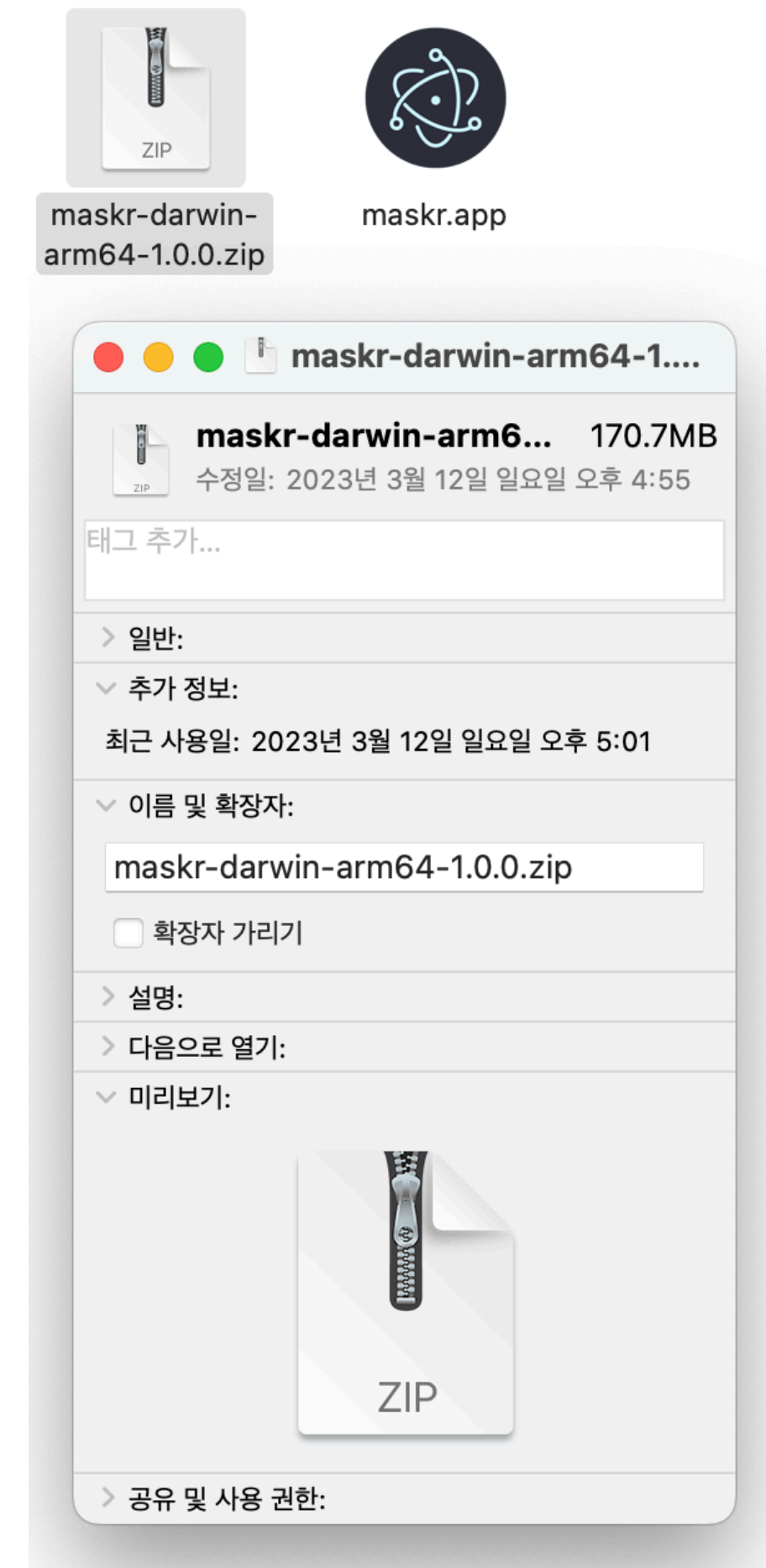
# Standalone의 Limit

## 1. Hard

- 터미널 기반 작업 필요
- 환경 설정 난해함
- R 만으로는 되지 않음. (Node, NPM)

## 2. Heavy

- 모든 필요한 파일을 "패키징" (170mb)
  - Electron 때문인지 R 때문인지 는 모르겠음
- github / 클라우드에 올리면 상관 없는 문제



# Not checked

그러나 곧 알게 될 것 같은

- not-CRAN Package in Shiny
- DBMS 연결 or 사용자 권한
- CI / CD (Update)
- OS Dependency (Dev / User)
- Customize Electron.app (제 경우 앱의 실행 사이즈만 바꿨음)
- Not Chromium browser



# Further...?

- shiny app to build app.R code to standalone app
- <https://github.com/Wandrysf-dev/shinybox>





# Q&A

- or **jinhwan@zarathu.com** 