ELECTRON PRO-TIPS™

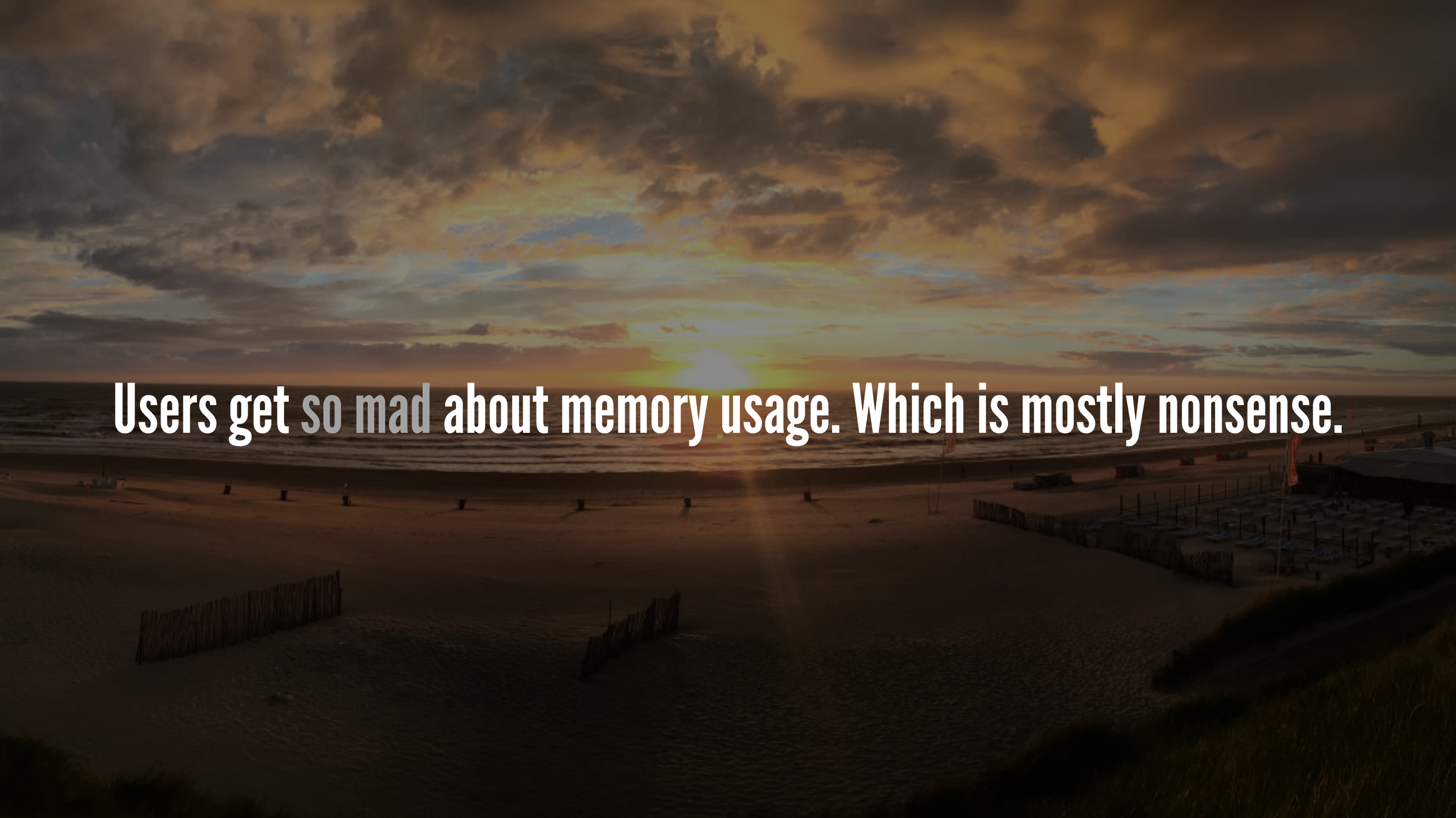
@PAULCBETTS (GITHUB, TWITTER)

HI.

HERE'S A FEW THINGS I NOTICE PEOPLE DOING IN ELECTRON APPS

(THAT MAKE USERS MAD)


MEMORY USAGE
MATTERS


Users get so mad about memory usage. Which is mostly nonsense.

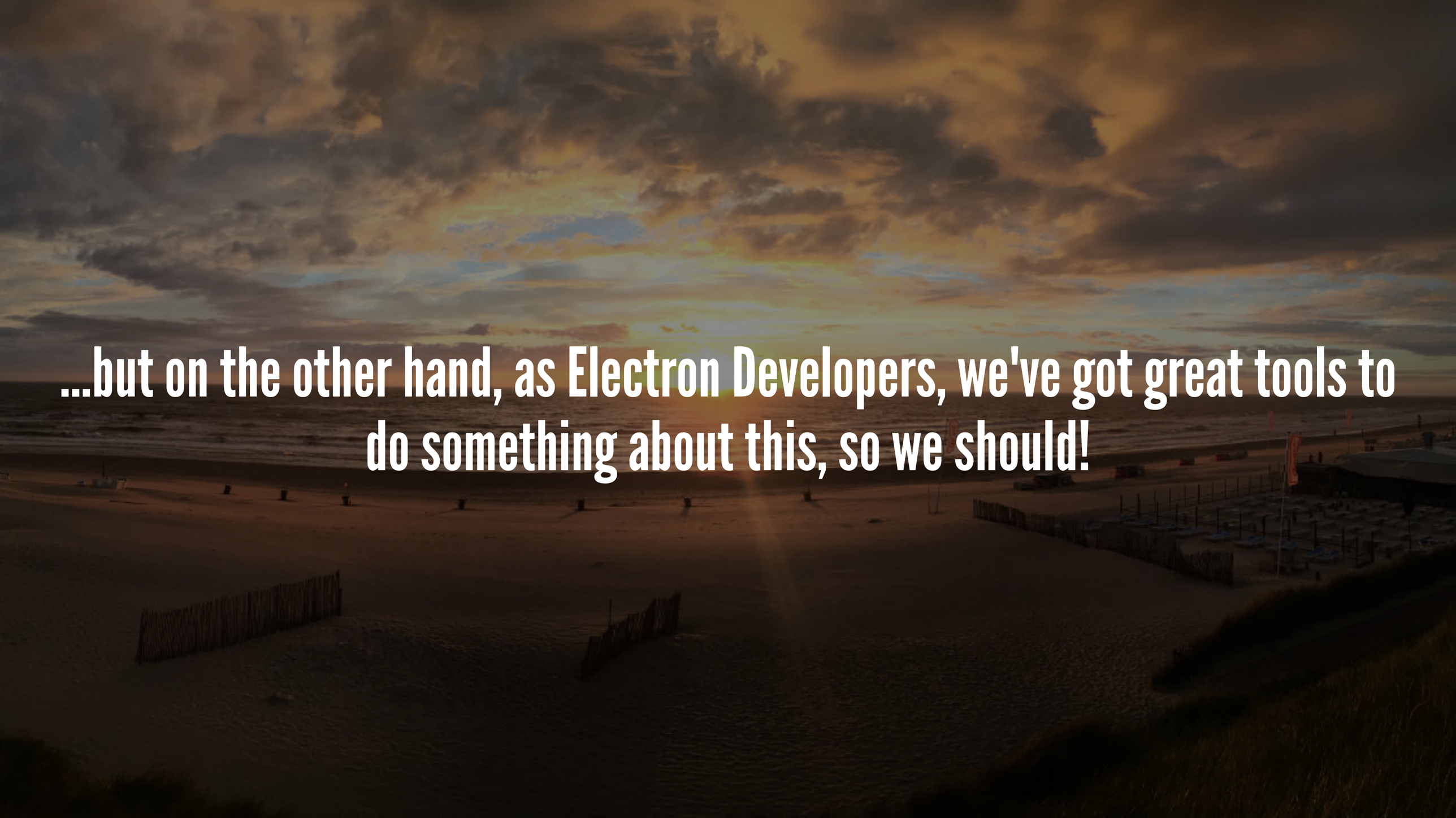
Every conversation I've ever had about Electron memory usage:

Them: IM SO MAD ABOUT MEMORY USAGE

Me: I understand! So, what's the "Commit Charge" say in Task Manager? That's the percentage of RAM that is actually in-use.

Them: Oh, it's 40%.

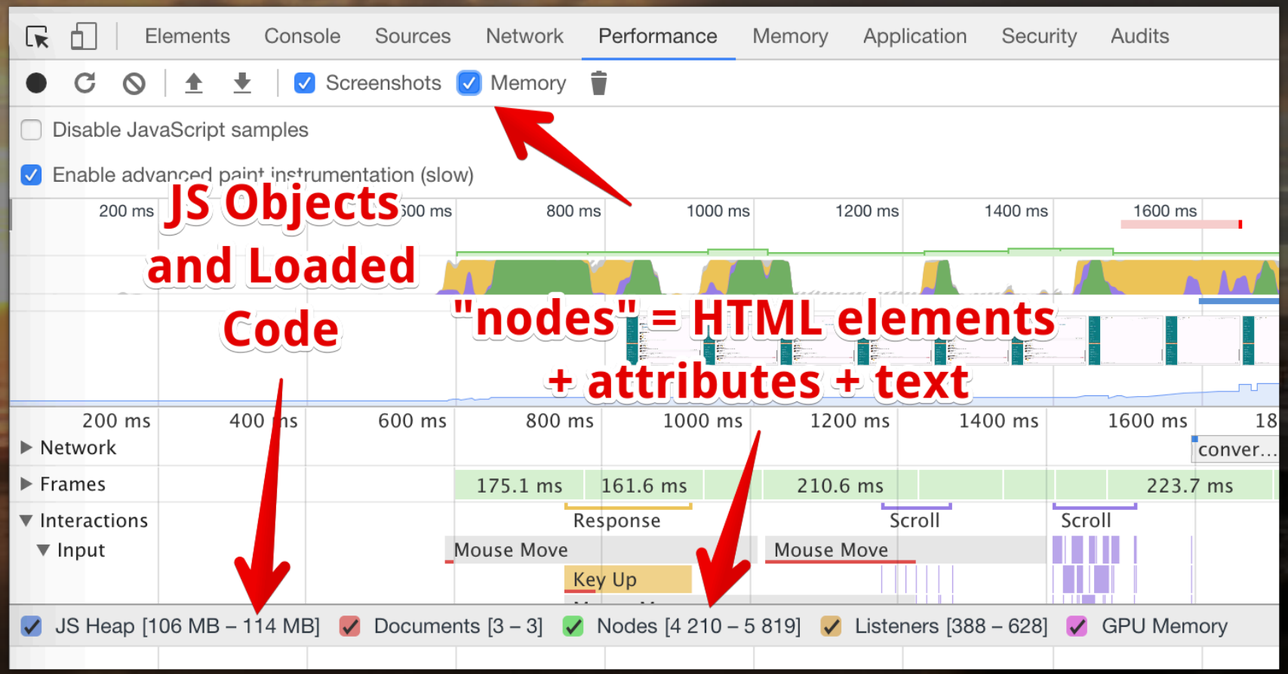

烈
BUZZ OOOOOOOOOOFF


...but on the other hand, as Electron Developers, we've got great tools to do something about this, so we should!


LOAD LESS STUFF

LOAD LESS STUFF

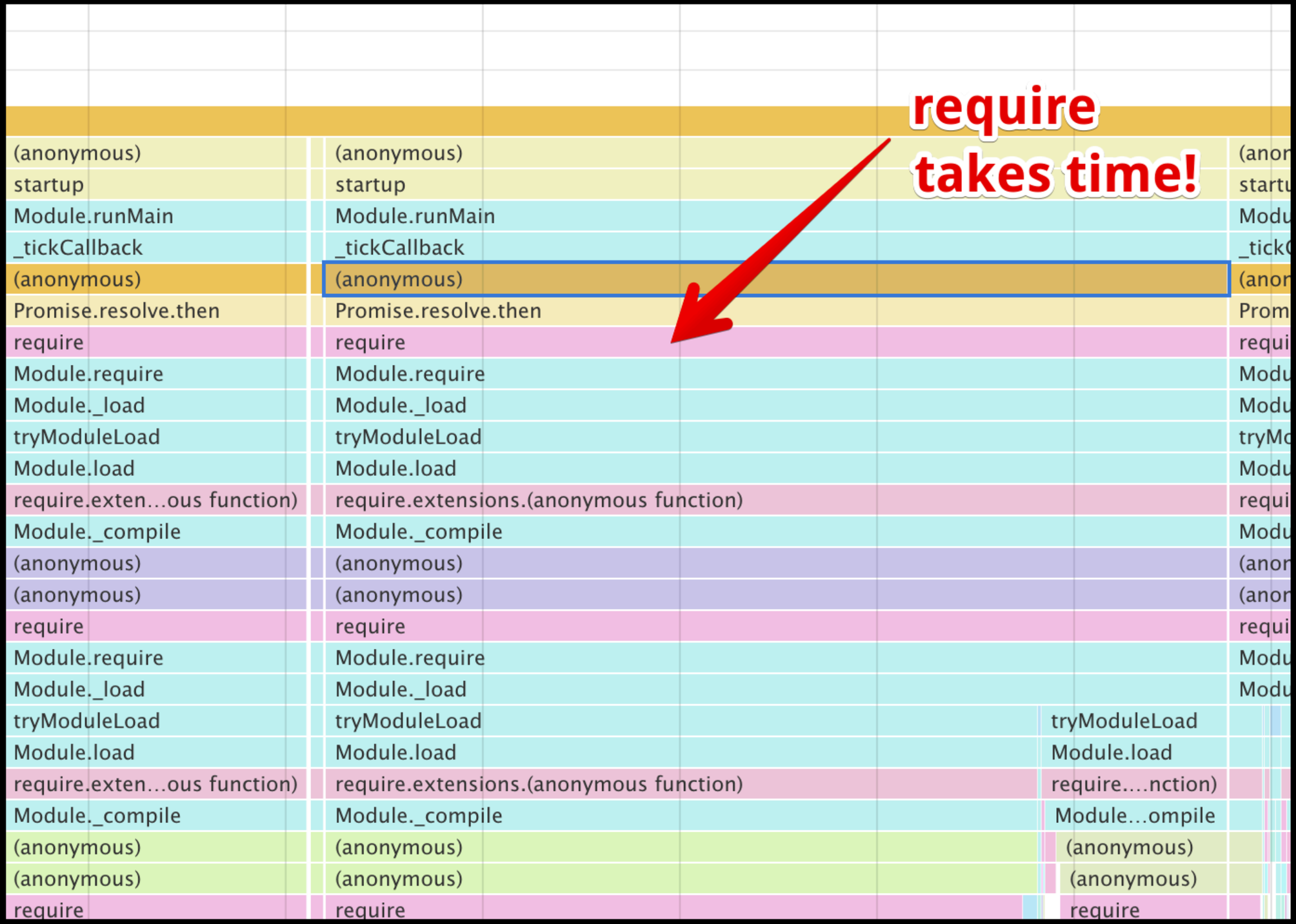
- Lots and Lots of DOM Elements
  - Especially Images
    - JS Heap



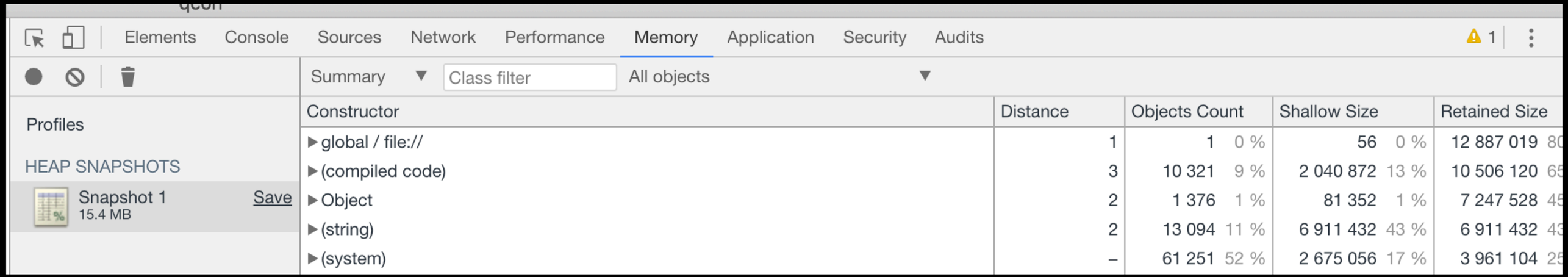

USE REACT OR
VUE
(AND VIRUALIZING LISTS)

LOAD LESS STUFF

- Libraries that you load in your app never get unloaded
- Bad for startup performance and for memory usage!


require
takes time!
(anonymous)
startup
Module.runMain
_tickCallback
(anonymous)
Promise.resolve.then
require
Module.require
Module._load
tryModuleLoad
Module.load
require.exten…ous function)
Module._compile
(anonymous)
(anonymous)
require
Module.require
Module._load
tryModuleLoad
Module.load
require.exten…ous function)
Module._compile
(anonymous)
(anonymous)
require
tryModuleLoad
Module.load
require.…nction)
Module…ompile
(anonymous)
(anonymous)
require

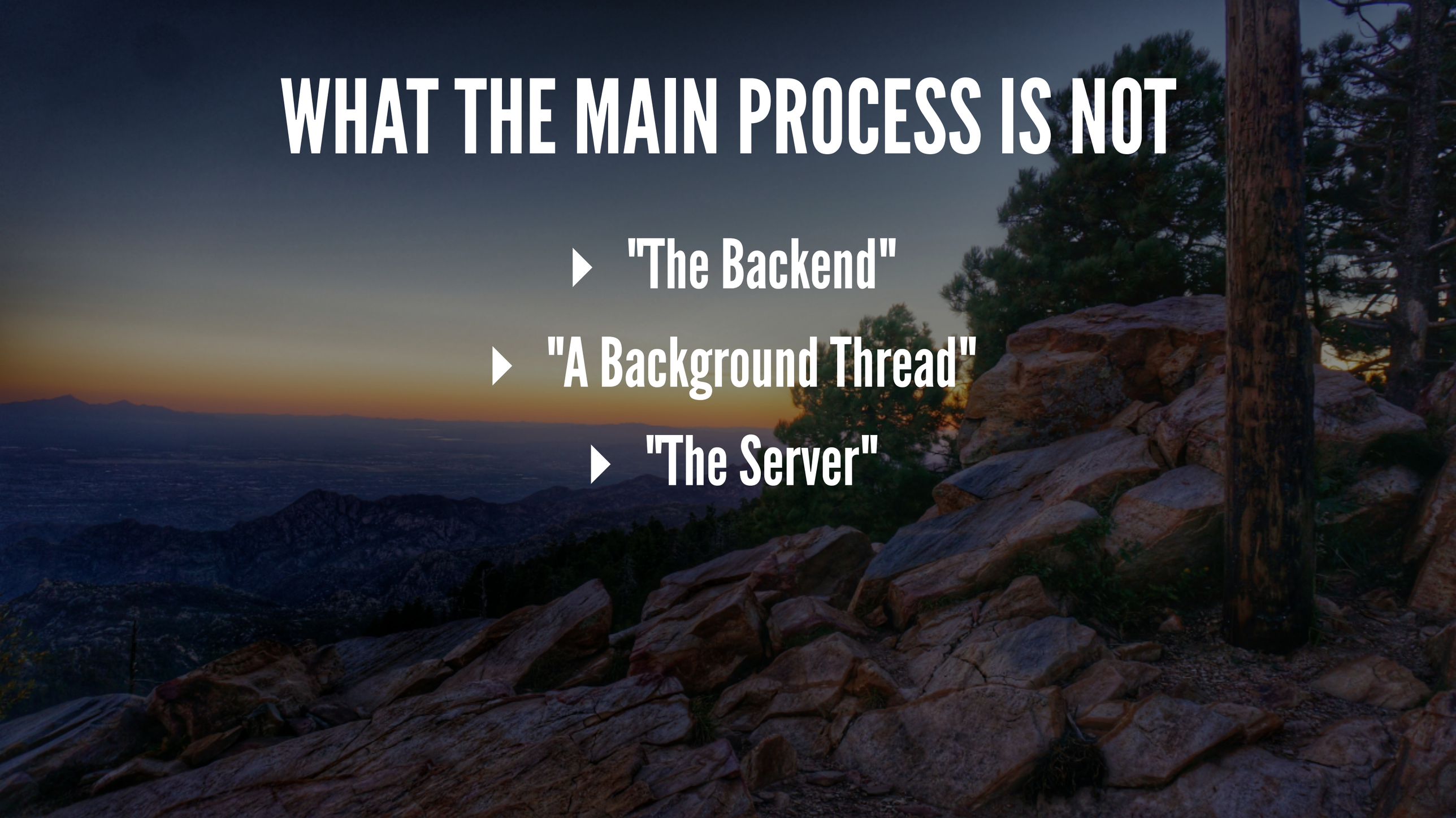

USE THE HEAP
PROFILER


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
1
Summary
Class filter
All objects
Profiles
HEAP SNAPSHOTS
Snapshot 1
15.4 MB
Save
Constructor
Distance
Objects Count
Shallow Size
Retained Size
▶ global / file://
1
1 0 %
56 0 %
12 887 019
▶ (compiled code)
3
10 321 9 %
2 040 872 13 %
10 506 120
▶ Object
2
1 376 1 %
81 352 1 %
7 247 528
▶ (string)
2
13 094 11 %
6 911 432 43 %
6 911 432
▶ (system)
–
61 251 52 %
2 675 056 17 %
3 961 104


DON'T RUN STUFF
IN THE MAIN
PROCESS


but what about...?


NO



WHAT THE MAIN PROCESS IS NOT

- "The Backend"
- "A Background Thread"
- "The Server"


THE MAIN
PROCESS IS FOR
ORCHESTRATION

THE MAIN PROCESS IS FOR ORCHESTRATION

Running code in the main process slows everything else down

Chromium uses IPC internally to do things, such as signaling window size changes

So when the main thread is busy, your app glitches!

THE MAIN PROCESS IS FOR ORCHESTRATION

ipc.send is asynchronous which is Better, but not enough!

The main process can still do a lot of work as a result of ipc.send, and block stuff

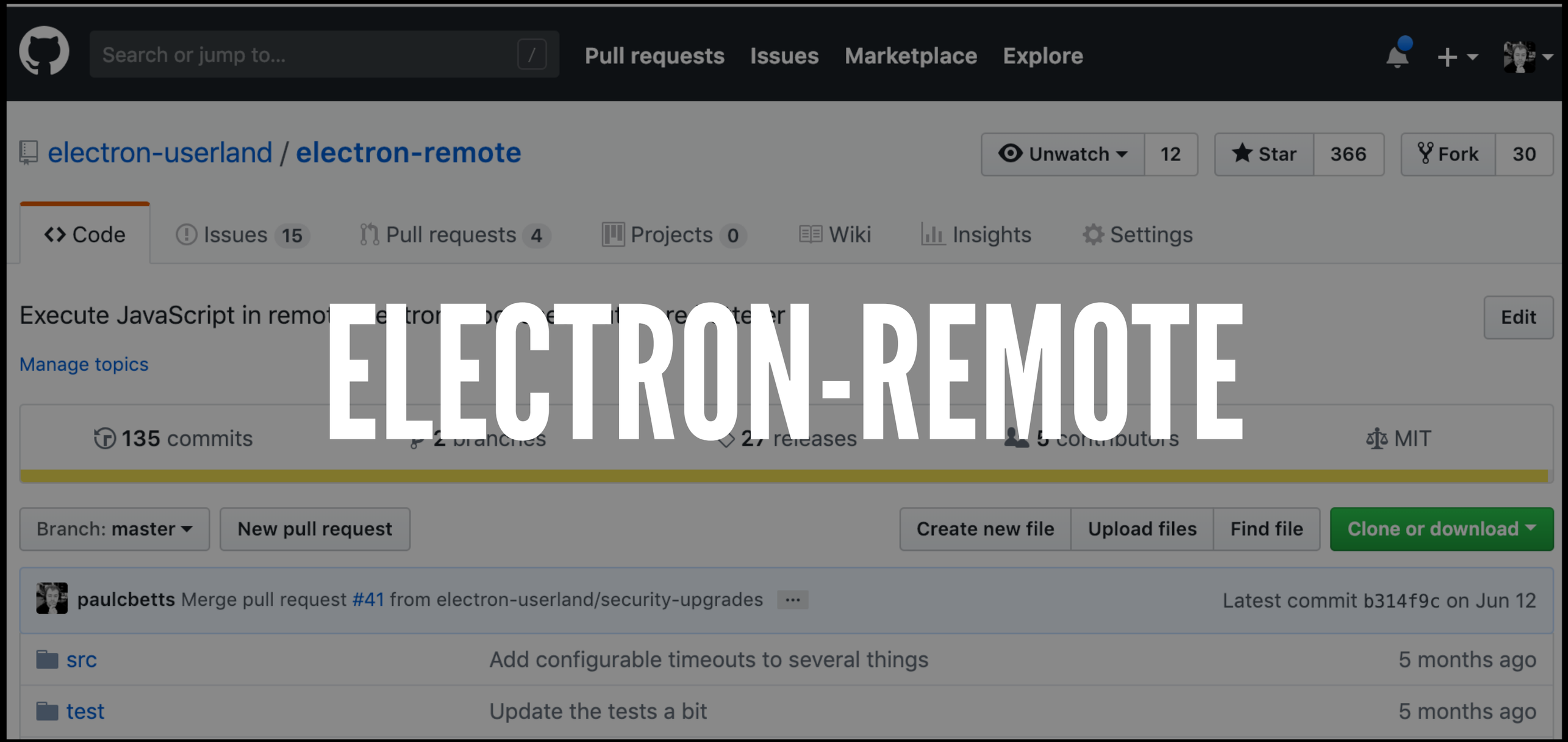
THE MAIN PROCESS IS FOR ORCHESTRATION

The main process should really only be used to tell other processes what to do

- Sending information between windows
- Signalling menu items and dock events
- Crash reporting and other APIs that only work in the main process


SO HOW CAN I DO
STUFF THEN??


WHAT IF WE CREATE A BROWSERWINDOW
BUT DIDN'T SHOW IT?



ELECTRON-REMOTE

ELECTRON-REMOTE, DOING WORK FROM THE MAIN PROCESS

```
import { createProxyForRemote } from 'electron-remote';

// myWindowJs is now a proxy object
// for myWindow's `window` global object
const myWindowJs = createProxyForRemote(myWindow);

// Functions suffixed with _get
// will read a value
userAgent = await myWindowJs.navigator.userAgent_get()
```

window.requestIdleCallback()

Languages

Jump to: Syntax Example Specifications Browser compatibility See also

Web technology for developers >
Web APIs /Window
window.requestIdleCallback()

Related Topics

Window

- Properties
  - applicationCache
  - caches
  - closed
  - console
  - controllers
  - crypto

This is an experimental technology
Check the Browser compatibility table carefully before using this in production.

The `window.requestIdleCallback()` method queues a function to be called during a browser's idle periods. This enables developers to perform background and low priority work on the main event loop, without impacting latency-critical events such as animation and input response. Functions are generally called in first-in-first-out order; however, callbacks which have a timeout specified may be called out-of-order if necessary in order to run them before the timeout elapses.

You can call `requestIdleCallback()` within an idle callback function to schedule another callback to take place no sooner than the next pass through the event loop.

A `timeout` option is strongly recommended for required work, as otherwise it's possible multiple seconds will elapse before the callback is fired.

REQUESTIDLECALLBACK IS SUPER COOL

REQUESTIDLECALLBACK IS SUPER COOL

Like setTimeout but only runs once the UI is no longer busy

The callback allows you to repeatedly schedule requestIdleCallback to do work in a loop

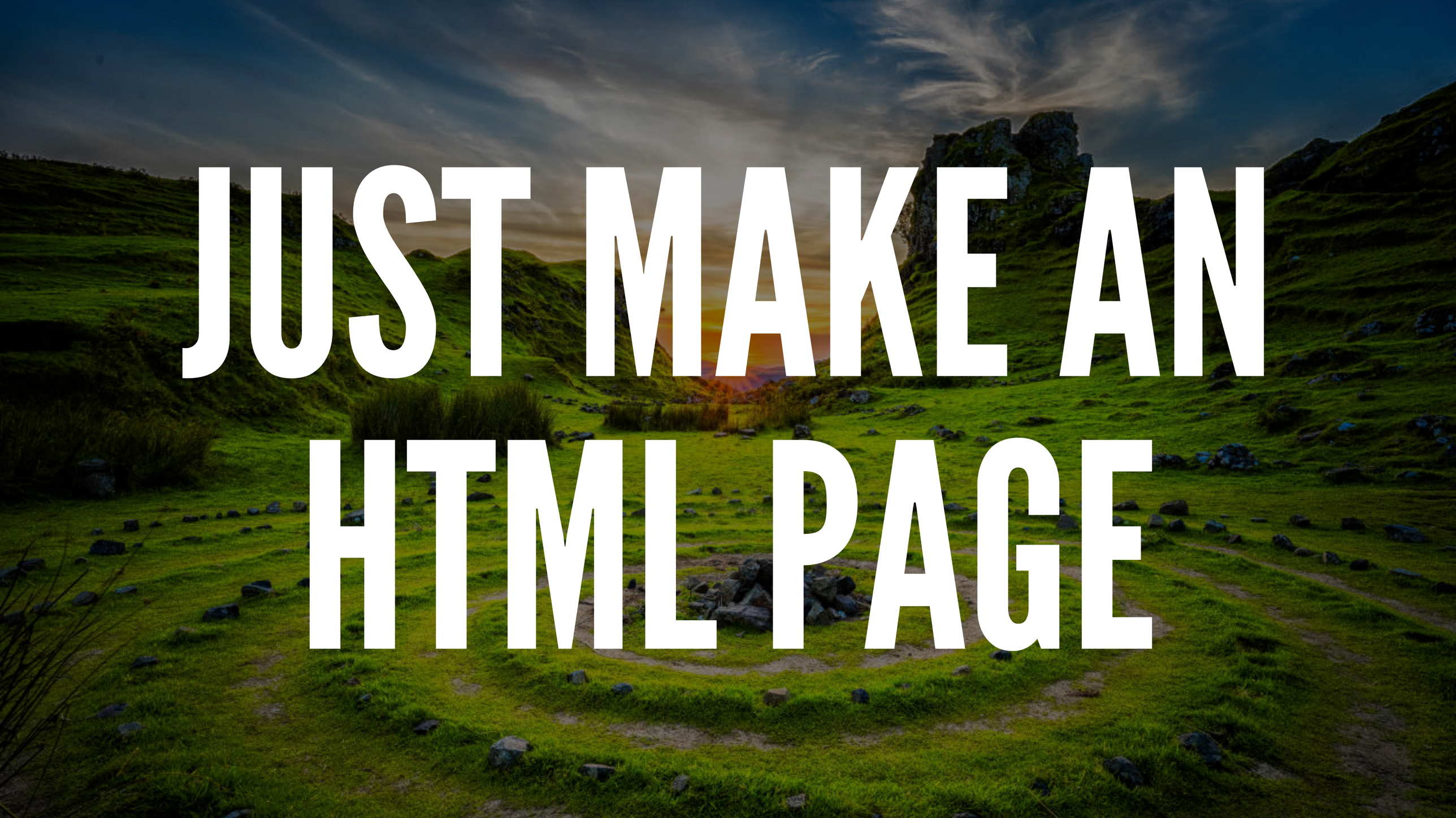
Writing App Data in the background is a great place to use requestIdleCallback

ELECTRON-REMOTE, TASKPOOL

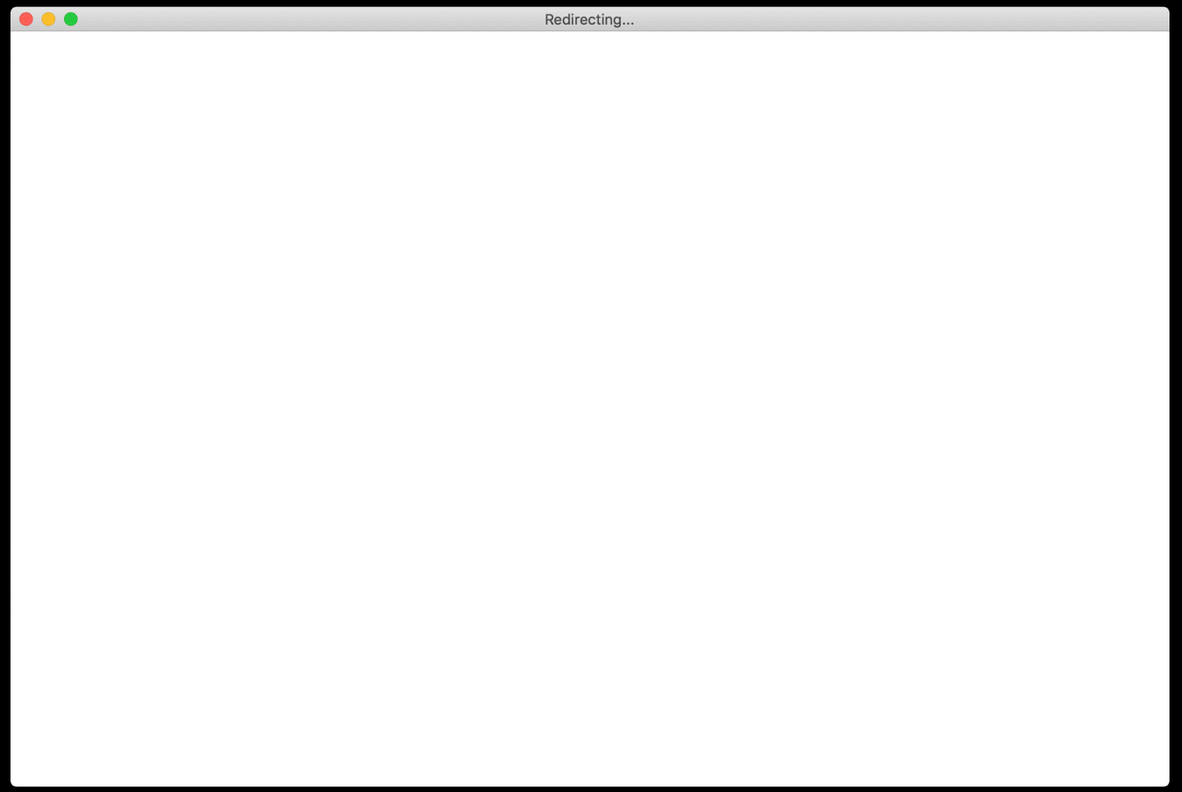
```
import { requireTaskPool } from 'electron-remote';

const myCoolModule = requireTaskPool(
  require.resolve('./my-cool-module'));

// This method will run synchronously,
// but in a background BrowserWindow process
// so that your app will not block
let result = await myCoolModule.calculateDigitsOfPi(100000);
```


JUST MAKE AN
HTML PAGE

THIS EXPERIENCE IS A DRAG.



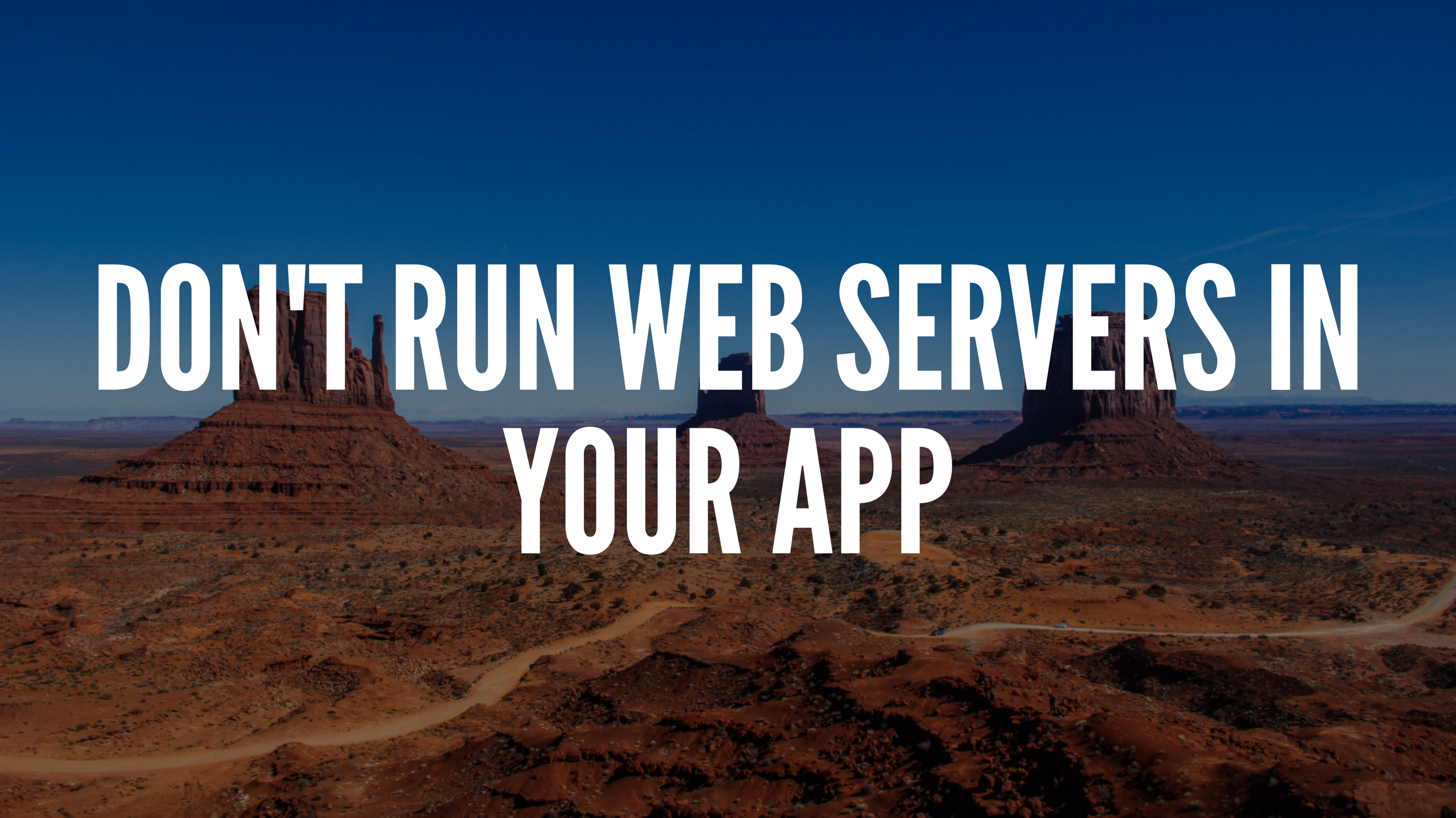

JUST MAKE AN HTML PAGE
Putting a website into an Electron frame is easy, but not great for Users
Offline Mode is way easier
Your app will start Really Fast
Starting with a Desktop Mindset will make your app feel like an app

HTML PAGES ARE MORE SECURE

Designing a hybrid app is Very Security Sensitive, so that you don't accidentally give Desktop Powers™ to remote content

When all of the code for your app is local, you remove this possibility altogether

XSS is still extremely important to watch out for!


DON'T RUN WEB SERVERS IN
YOUR APP


DON'T RUN WEB SERVERS IN YOUR APP
...cause like, what if more than one user uses your app?
Your web service now a great way to move data between different users
If you run as Admin, it's now a great way to local EoP
...or if you're really unlucky, have arbitrary websites run Desktop code

Electron Forge

The command line interface for ambitious Electron applications

USE ELECTRON-FORGE

```
# Launch your app
$ cd my-new-project
$ electron-forge start
```

Ready for a closer look? Dive into the CLI documentation

Opinionated Electron development

Modern language compilation, one-step builds for all platforms,

USE ELECTRON-FORGE

electron-forge handles all of the things you might want to use Express or Webpack for, like Hot Module Reload

It handles Babel/TypeScript/LESS/Sass via hooking Electron and compiling on-the-fly during development

electron-forge does all of the packaging and compilation work too


BUT I LIKE WEBPACK!

BUT I LIKE WEBPACK!

Trying to interact with Electron itself gets Weird because now there are two separate module systems

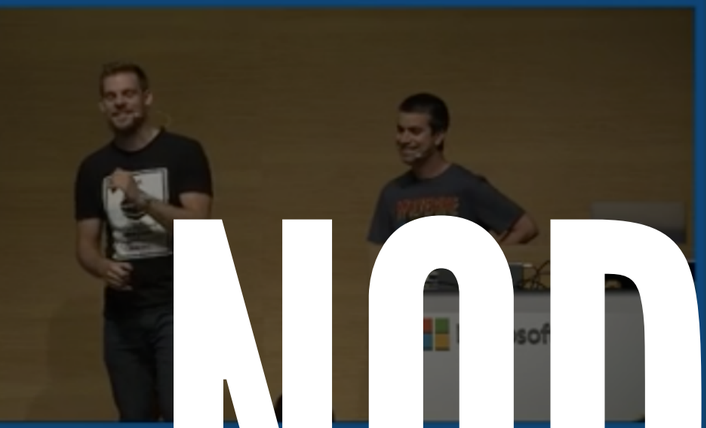
Native node modules are a pain with Webpack, both at runtime and on the build side

Packaging becomes way more complex

PERFORMANCE, BLAHHH


SECURITY,
BORINGGGG


MEMORY,
UGHHHHHH



Microsoft

T6976

Node.js Using Native Windows Features from Node.js and Electron

Senior Software Engineer

Senior Desktop Engineer @ Slack

#MSBuild

NODE-RT IS COOL

Microsoft Build 2017

CALL WIN10 APIS FROM ELECTRON SUPER EASILY

SOME COMPELLING EXAMPLES:

- Windows.Devices.Display
- Windows.Devices.Geolocation
- Windows.Media.Capture
- Windows.Media.OCR
- Windows.System.Power

WHAT ABOUT MACOS?


¯_(ツ)_/¯


¯_(ツ)_/¯
There's no easy way to call macOS APIs from Electron, you have to write a Native Node Module.
You can do very simple things with node-ffi, but more complicated things will lead to Segfault City

HOW CAN I FIGURE OUT WHAT I CAN DO?

New Project

Recent

Installed

Visual C#

Get Started

Windows Universal

Windows Desktop

.NET Core

.NET Standard

Test

Visual Basic

Visual C++

JavaScript

Other Project Types

Dependency Validation

Online

Not finding what you are looking for?

Open Visual Studio Installer

Sort by: Default

Search (Ctrl+E)

Blank App (Universal Windows)	Visual C#
Class Library (Universal Windows)	Visual C#
Windows Runtime Component (Universal Windows)	Visual C#
Optional Code Package (Universal Windows)	Visual C#
Unit Test App (Universal Windows)	Visual C#
Coded UI Test Project (Universal Windows)	Visual C#
Windows Application Packaging Project	Visual C#

Type: Visual C#

A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.

Name: App1

Location: C:\Users\me\source\repos

Browse...

Solution: Create new solution

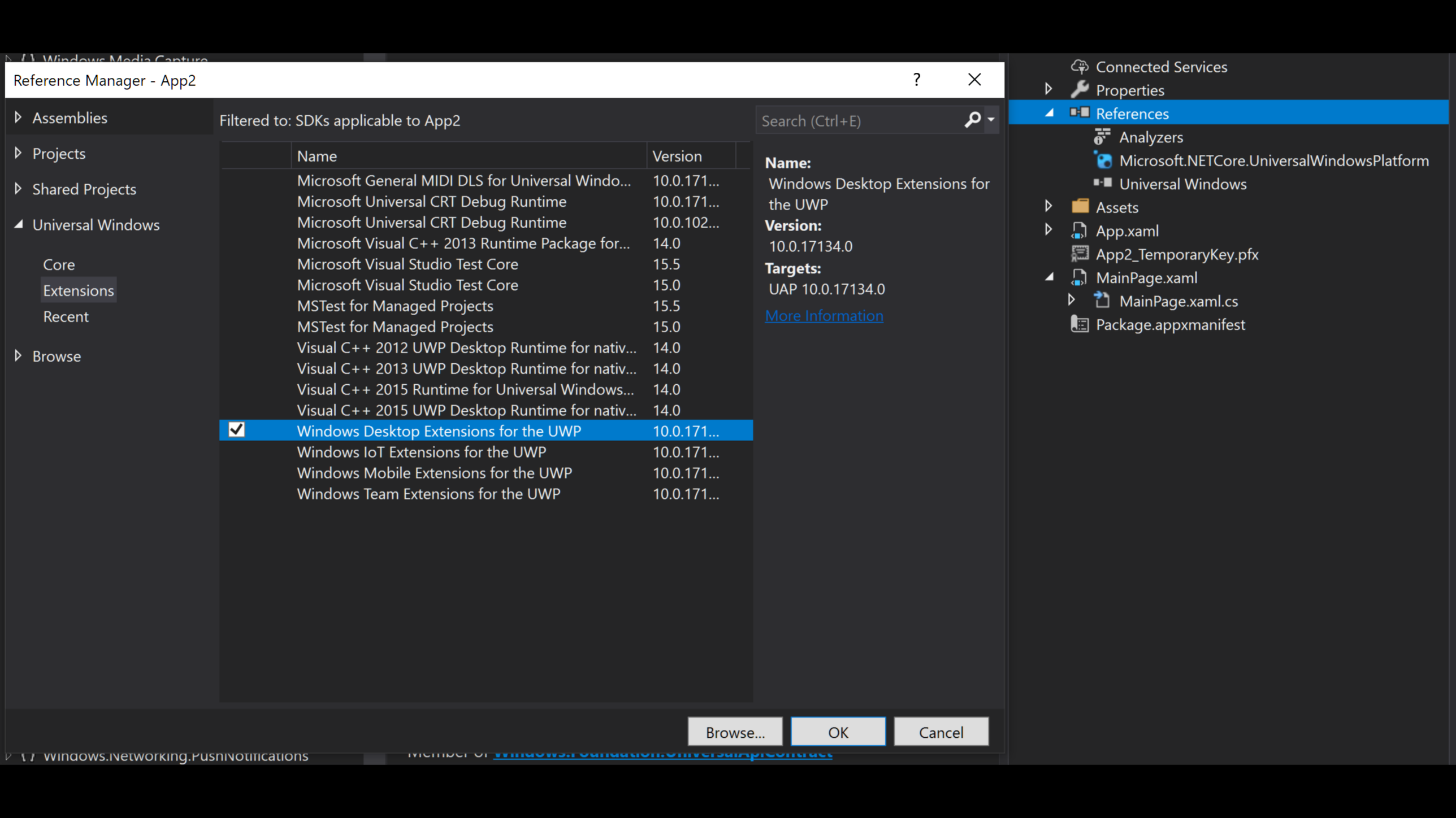
Solution name: App1

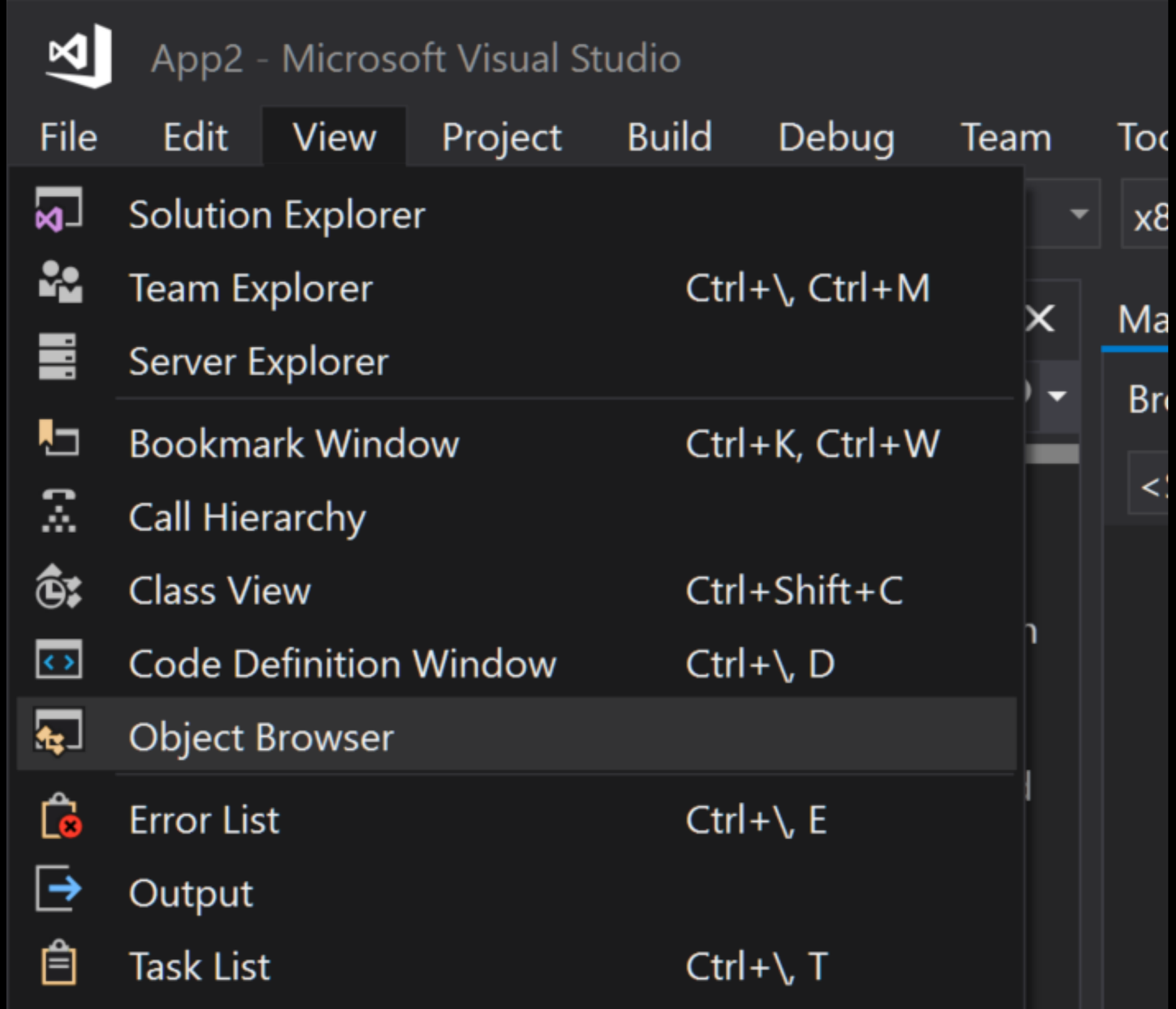
☑ Create directory for solution

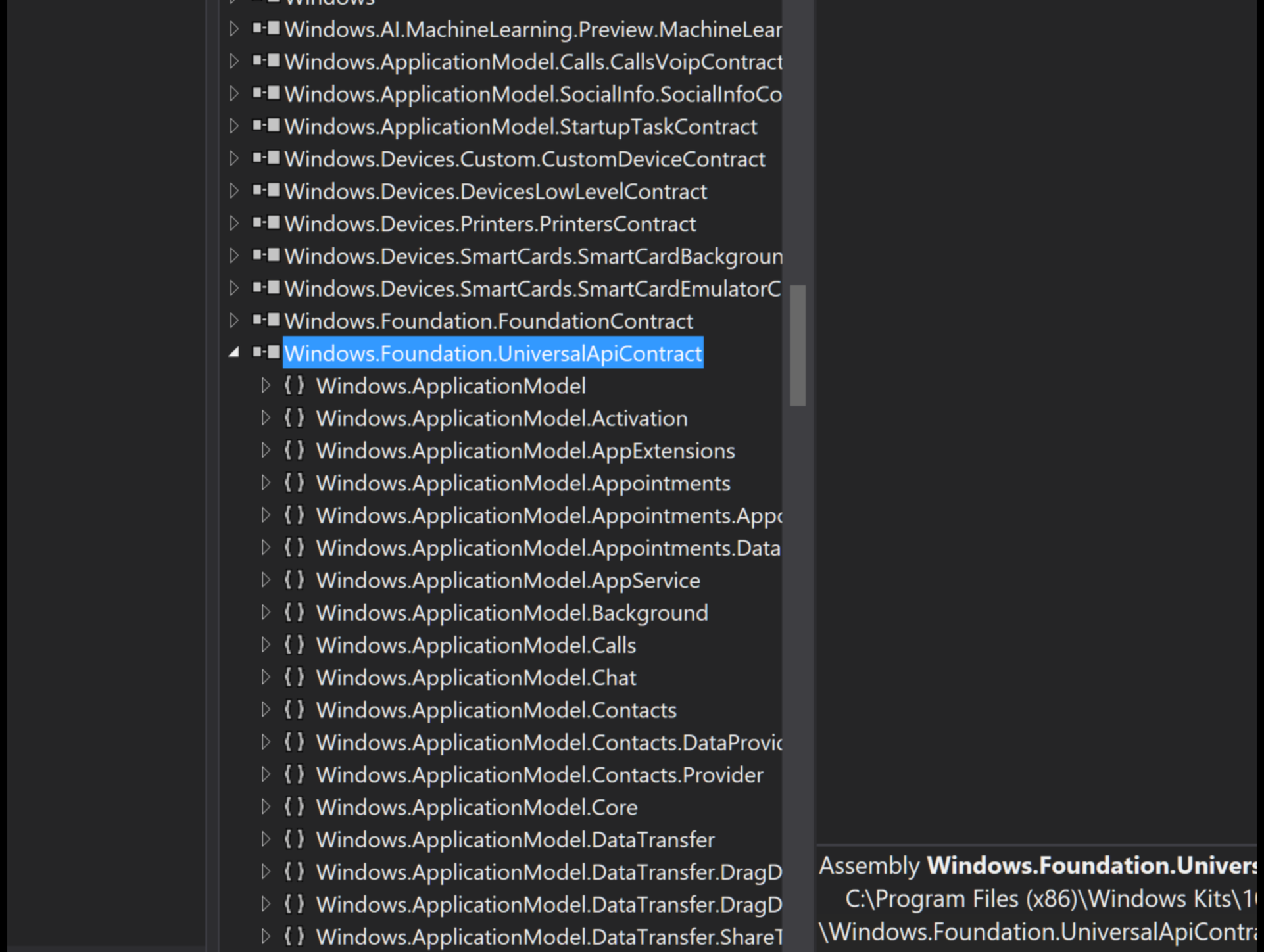
☐ Create new Git repository

OK

Cancel


Reference Manager - App2
Assemblies
Projects
Shared Projects
Universal Windows
Core
Extensions
Recent
Browse
Filtered to: SDKs applicable to App2
Search (Ctrl+E)
Name
Version
Microsoft General MIDI DLS for Universal Windo... 10.0.171...
Microsoft Universal CRT Debug Runtime 10.0.171...
Microsoft Universal CRT Debug Runtime 10.0.102...
Microsoft Visual C++ 2013 Runtime Package for... 14.0
Microsoft Visual Studio Test Core 15.5
Microsoft Visual Studio Test Core 15.0
MSTest for Managed Projects 15.5
MSTest for Managed Projects 15.0
Visual C++ 2012 UWP Desktop Runtime for nativ... 14.0
Visual C++ 2013 UWP Desktop Runtime for nativ... 14.0
Visual C++ 2015 Runtime for Universal Windows... 14.0
Visual C++ 2015 UWP Desktop Runtime for nativ... 14.0
Windows Desktop Extensions for the UWP 10.0.171...
Windows IoT Extensions for the UWP 10.0.171...
Windows Mobile Extensions for the UWP 10.0.171...
Windows Team Extensions for the UWP 10.0.171...
Name:
Windows Desktop Extensions for the UWP
Version:
10.0.17134.0
Targets:
UAP 10.0.17134.0
More Information
Browse...
OK
Cancel
Connected Services
Properties
References
Analyzers
Microsoft.NETCore.UniversalWindowsPlatform
Universal Windows
Assets
App.xaml
App2_TemporaryKey.pfx
MainPage.xaml
MainPage.xaml.cs
Package.appxmanifest


App2 - Microsoft Visual Studio
File
Edit
View
Project
Build
Debug
Team
Solution Explorer
Team Explorer Ctrl+\, Ctrl+M
Server Explorer
Bookmark Window Ctrl+K, Ctrl+W
Call Hierarchy
Class View Ctrl+Shift+C
Code Definition Window Ctrl+\, D
Object Browser
Error List Ctrl+\, E
Output
Task List Ctrl+\, T


Windows.AI.MachineLearning.Preview.MachineLear
Windows.ApplicationModel.Calls.CallsVoipContract
Windows.ApplicationModel.SocialInfo.SocialInfoCo
Windows.ApplicationModel.StartupTaskContract
Windows.Devices.Custom.CustomDeviceContract
Windows.Devices.DevicesLowLevelContract
Windows.Devices.Printers.PrintersContract
Windows.Devices.SmartCards.SmartCardBackgroun
Windows.Devices.SmartCards.SmartCardEmulatorC
Windows.Foundation.FoundationContract
Windows.Foundation.UniversalApiContract
Windows.ApplicationModel
Windows.ApplicationModel.Activation
Windows.ApplicationModel.AppExtensions
Windows.ApplicationModel.Appointments
Windows.ApplicationModel.Appointments.Appo
Windows.ApplicationModel.Appointments.Data
Windows.ApplicationModel.AppService
Windows.ApplicationModel.Background
Windows.ApplicationModel.Calls
Windows.ApplicationModel.Chat
Windows.ApplicationModel.Contacts
Windows.ApplicationModel.Contacts.DataProvic
Windows.ApplicationModel.Contacts.Provider
Windows.ApplicationModel.Core
Windows.ApplicationModel.DataTransfer
Windows.ApplicationModel.DataTransfer.DragD
Windows.ApplicationModel.DataTransfer.DragD
Windows.ApplicationModel.DataTransfer.ShareT
Assembly Windows.Foundation.Univers
C:\Program Files (x86)\Windows Kits\1
\Windows.Foundation.UniversalApiContra

THANKS!

@PAULCBETTS (GITHUB, TWITTER)

ELECTRON PRO TIPS

Red Threads:

- Performance And Memory Usage Matters
- Users Care about Memory Usage, so you should too - you have great tools to debug it!
- Just Load Less Stuff - module load time is super easy to debug in Perf tools