定制浏览器方案

定制浏览器方案

- 简介
  - Chromium
  - CEF
  - NW
  - Electron
- 市场调研
  - CEF 案例
  - NW 案例
  - Electron 案例
- 实战
  - NW => Hello, world!
  - Electron => Hello, world!
- GitHub关注度和活跃度
  - CEF
  - NW
  - Electron

最近研究了一下基于 `Chromium` 定制浏览器的方案，查阅了大量资料，具体方案也有了大概的蓝图。

在阐述方案之前，先要了解几个名词：

简介

Chromium

> Chromium是一个由Google主导开发的网页浏览器，以BSD许可证等多重自由版权发行并开放源代码。Chromium的开发可能早自2006年即开始。
>
> Chromium是Google为发展自家的浏览器Google Chrome而打开的项目，所以

> Chromium相当于Google Chrome的工程版或实验版（尽管Google Chrome本身也有β版），新功能会率先在Chromium上开放，待验证后才会应用在Google Chrome上，故Google Chrome的功能会相对落后但较稳定。 —— 维基百科

CEF

> Chromium Embedded Framework (CEF)是个基于Google Chromium项目的开源Web browser控件，支持Windows, Linux, Mac平台。除了提供C/C++接口外，也有其他语言的移植版。
>
> 因为基于Chromium，所以CEF支持Webkit & Chrome中实现的HTML5的特性，并且在性能上面，也比较接近Chrome。
> CEF还提供的如下特性：自定义插件、自定义协议、自定义JavaScript对象和扩展；可控制的resource loading, navigation, context menus等等 —— 百度百科

NW

> NW.js 是基于 Chromium 和 Node.js 运行的， 以前也叫nodeWebkit。这就给了你使用HTML和JavaScript来制作桌面应用的可能。在应用里你可以直接调用Node.js的各种api以及现有的第三方包。因为Chromium和 Node.js 的跨平台，那么你的应用也是可以跨平台的。 —— SegmentFault

Electron

> Electron（最初名为Atom Shell）是GitHub开发的一个开源框架。它允许使用Node.js（作为后端）和Chromium（作为前端）完成桌面GUI应用程序的开发。Electron现已被多个开源Web应用程序用于前端与后端的开发，著名项目包括GitHub的Atom和微软的Visual Studio Code。 —— 维基百科

所以， **CEF**、**nw**、**Electron** 都是基于 `Chromium` 的开源框架，可以实现所需的定制浏览器需求，准确的讲应该是用 `HTML5` 、 `CSS3` 、 `JavaScript` 来制作拥有漂亮界面的**桌面应用**。

就是一个本地客户端应用程序使用一个内置的浏览器内核渲染前端界面，另一方面还可以调用本地系统级API，实现本地应用程序的各种功能。

市场调研

通过查阅大量资料得知，以各企业的线上产品及使用的技术供参考。

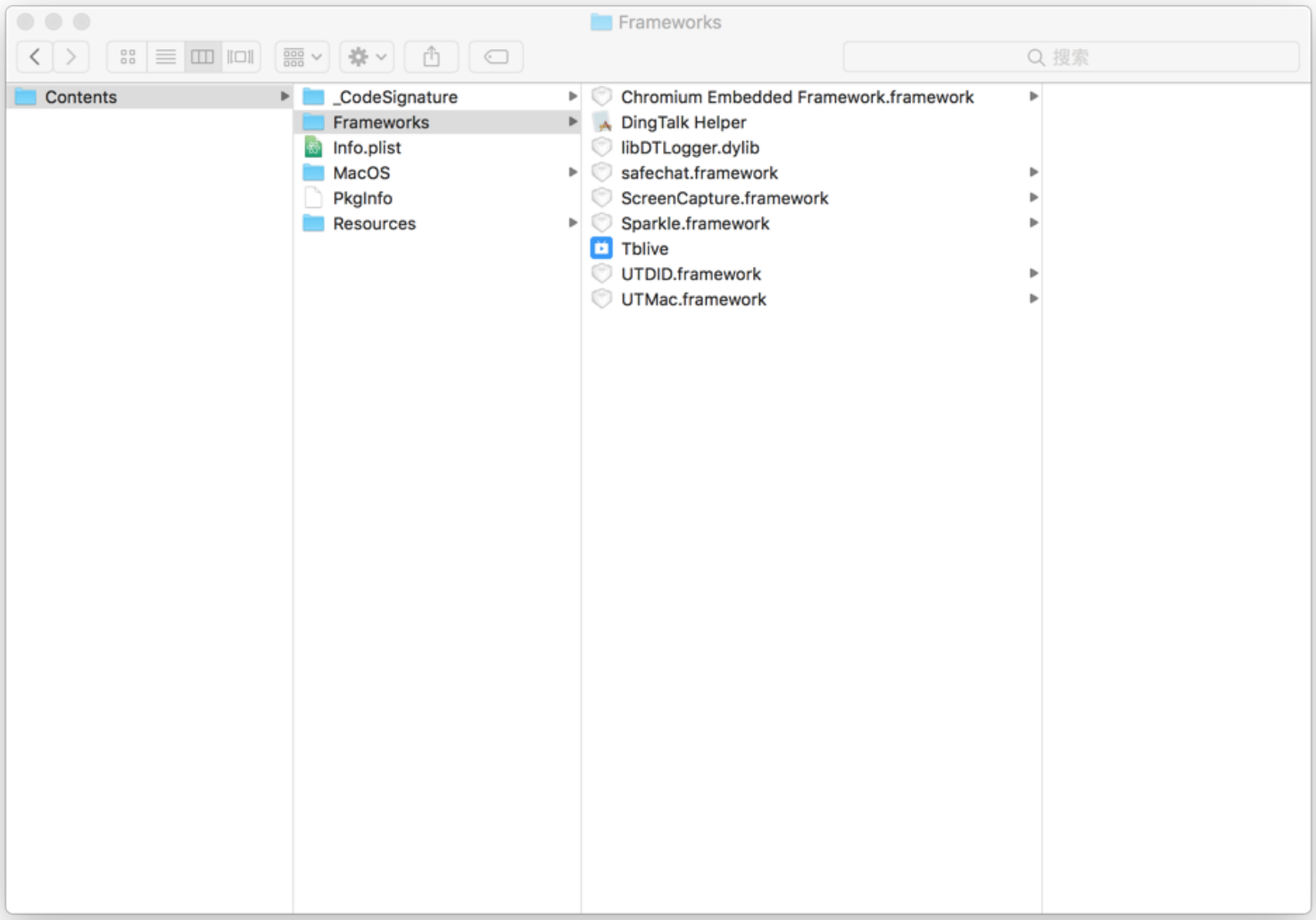
CEF 案例

据 **CEF** 官方介绍，以下（如图）桌面应用在使用 `CEF` 。

who is using CEF

国内桌面应用有：有道云笔记 *（网易）*、钉钉 *（阿里巴巴）*、QQ *（腾讯）* 等，查看安装后目录及文件，可以看出 **有道云笔记**、**钉钉** 是使用的是 `CEF` ，而 **钉钉** 界面是使用 `AngularJs` ，据了解后端应该用了 `C++` 和 `Python` 。

QQ 很早之前就通过内嵌 **IE** 来实现一些功能和界面。从2013年开始，**QQ** 引入了 `CEF` ，对一些之前用 **IE** 的地方进行了替换。



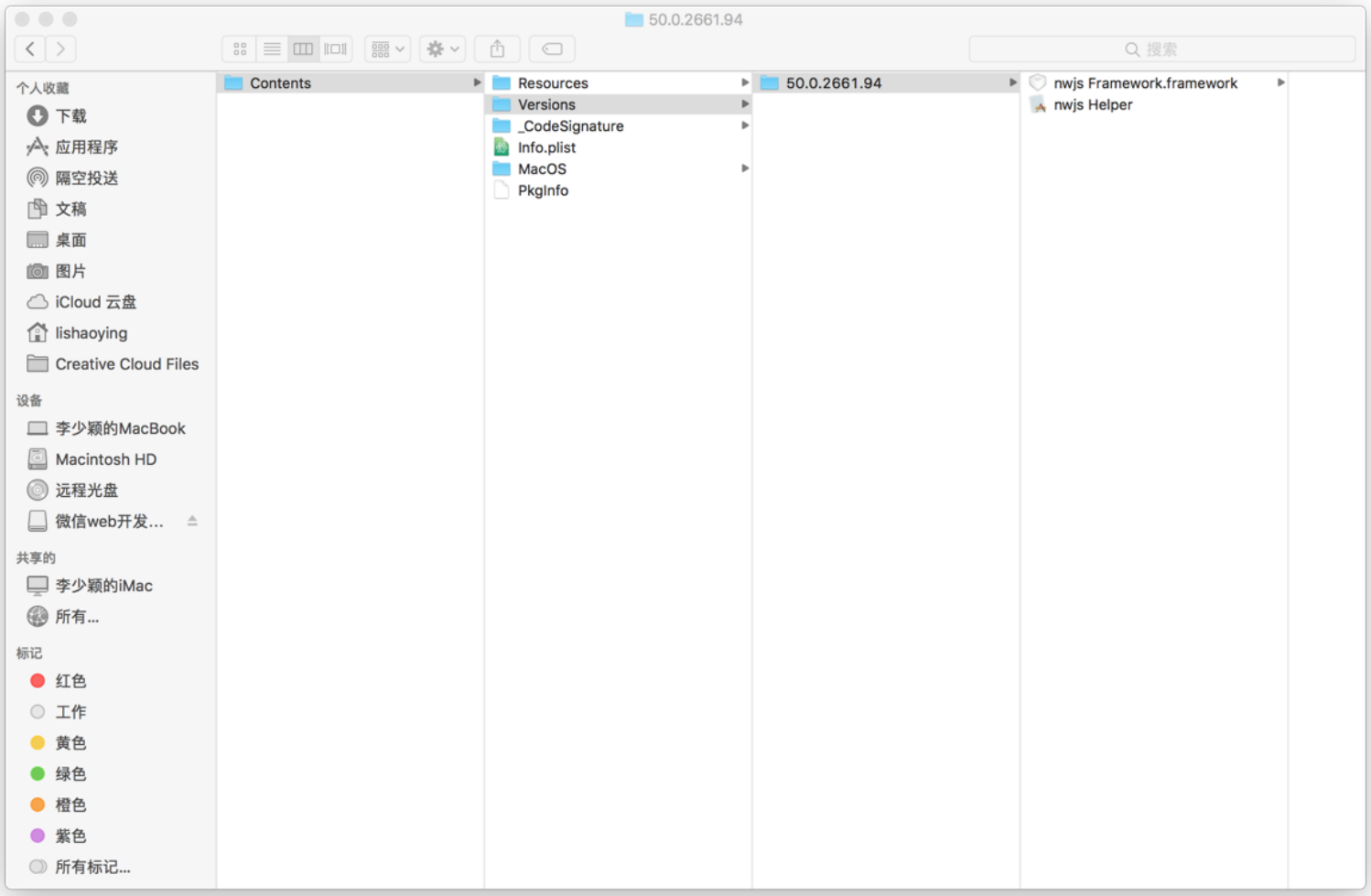
钉钉Mac版目录

钉钉Mac版应用界面

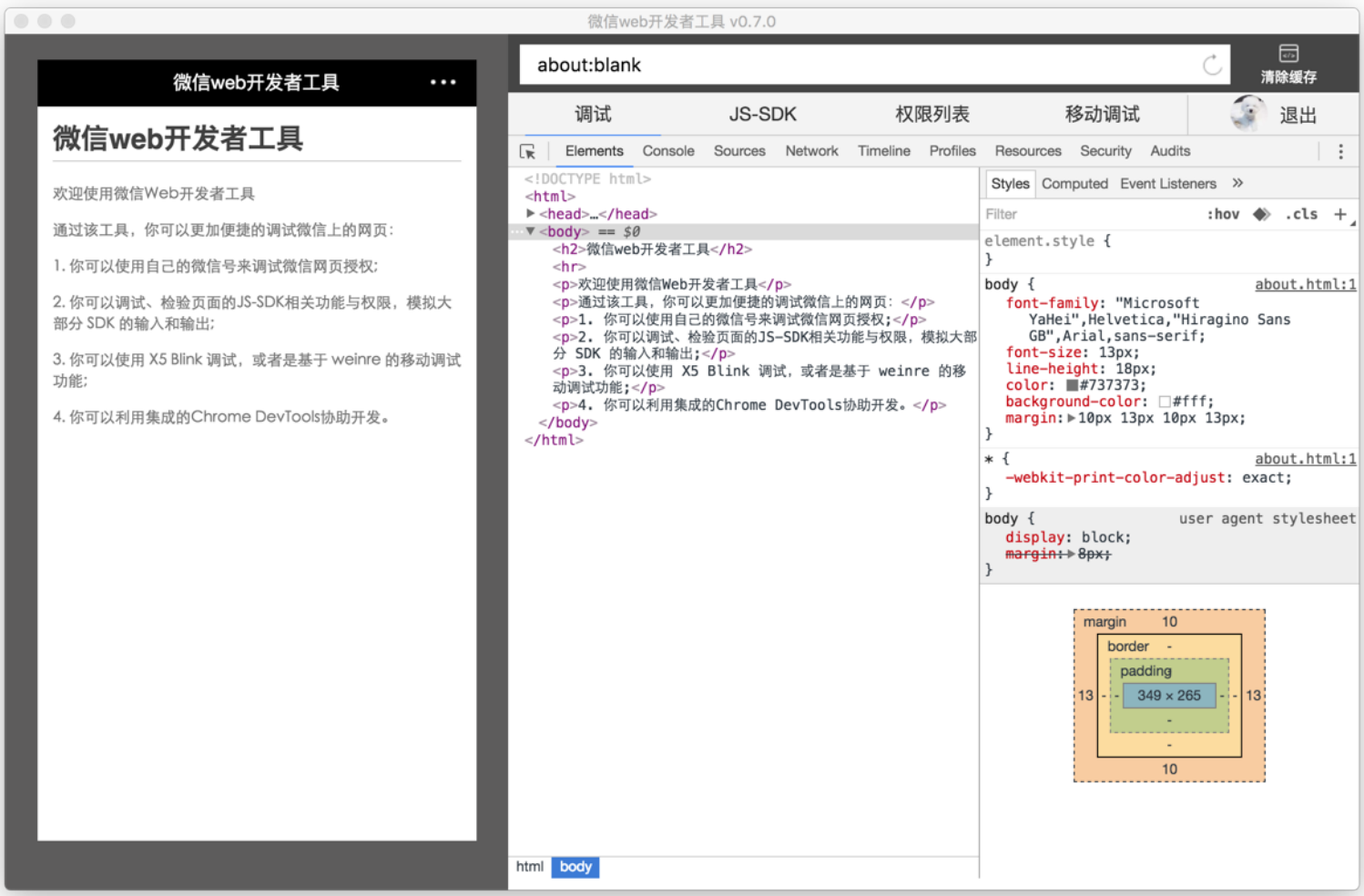
NW 案例

这个是 **NW** 官方给出的使用 `nw.js` 的应用列表：https://github.com/nwjs/nw.js/wiki/List-of-apps-and-companies-using-nw.js

而国内的有，比如微信开发工具等，是基于 `nw.js` 开发的。



微信开发工具Mac版目录

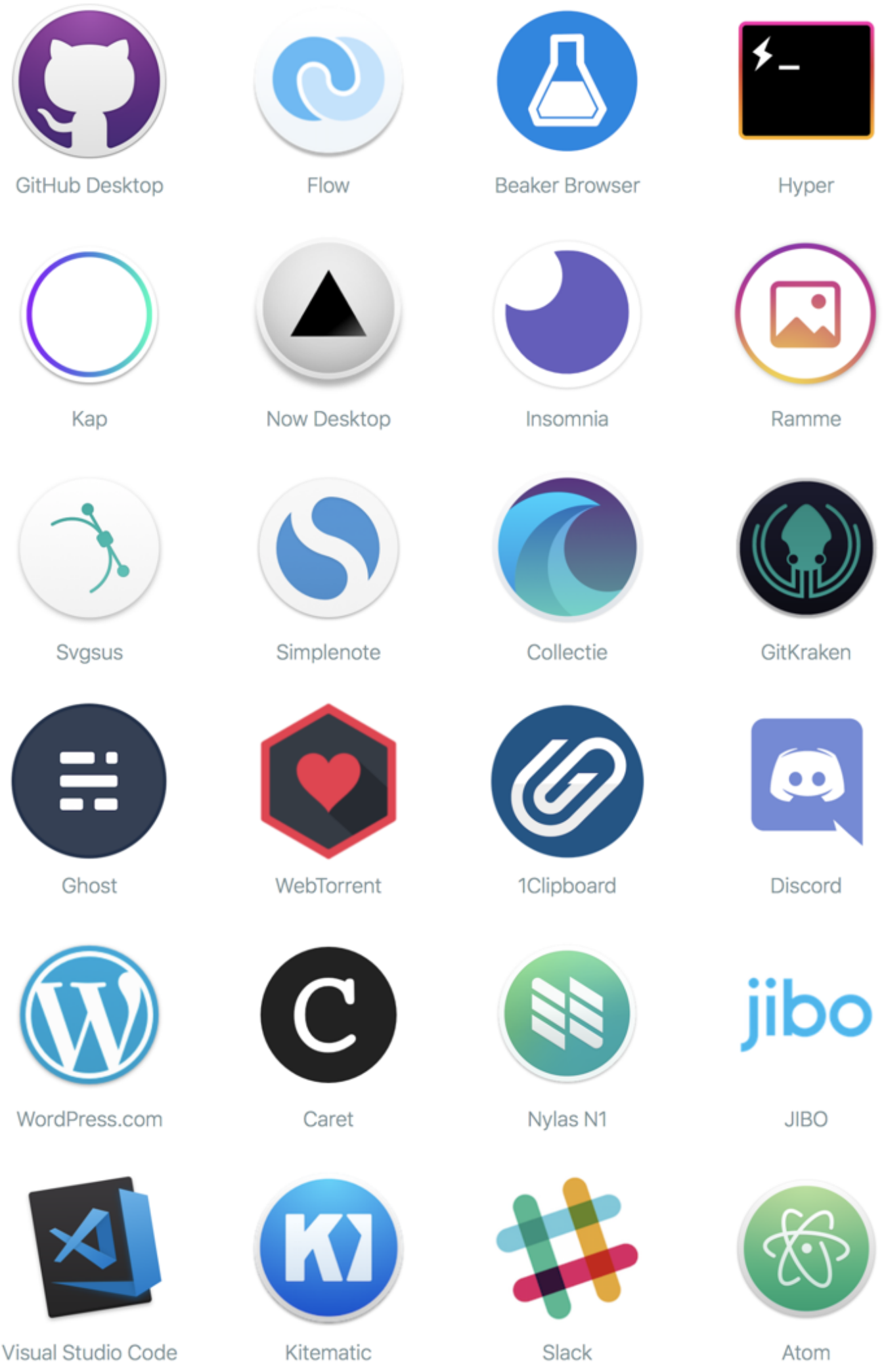


微信开发工具Mac版

Electron 案例

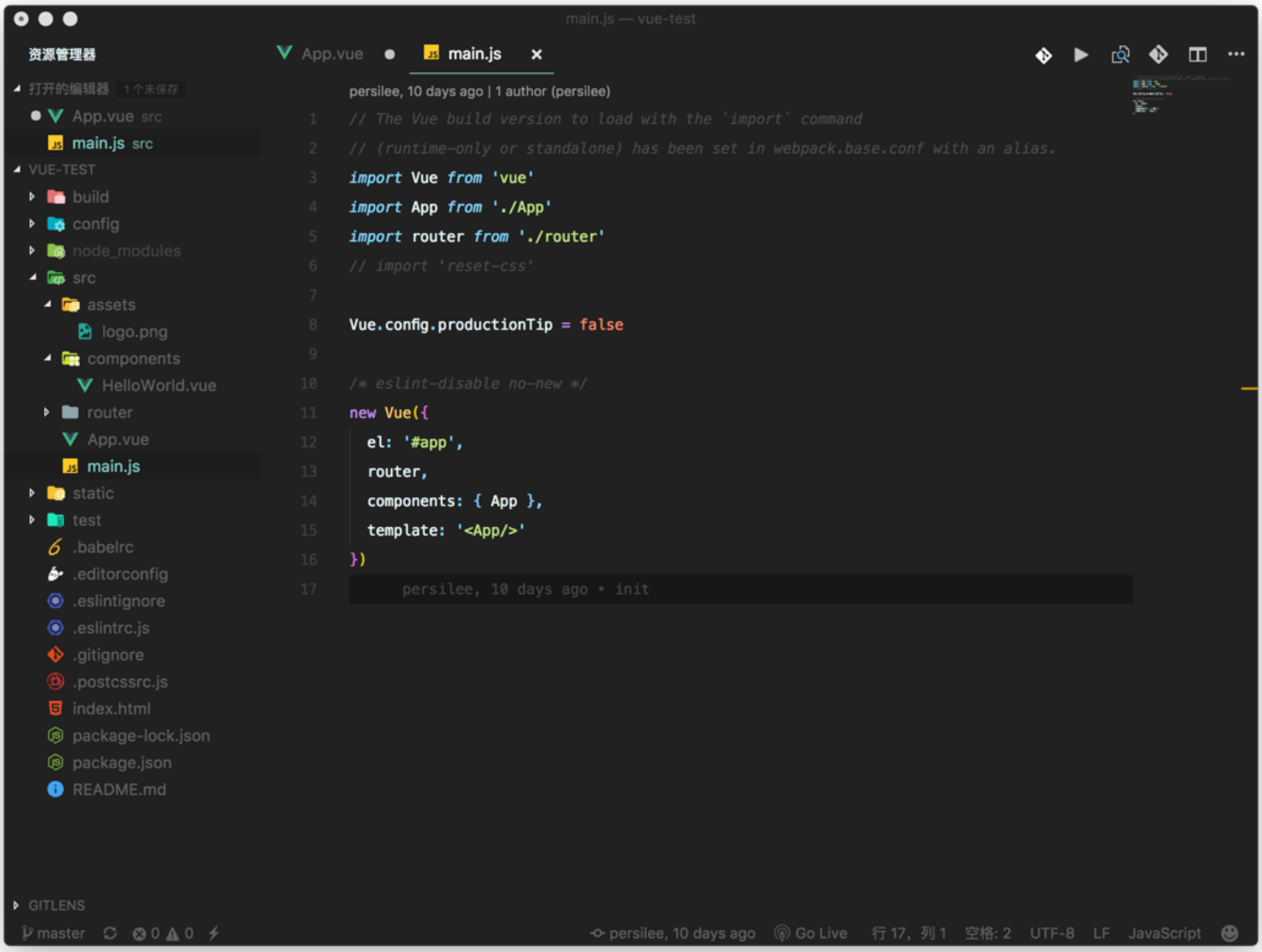
这个是 **Electron** 官方给出的是用 `electron` 的应用列表：https://electronjs.org/apps

如图，**Electron** 已被像 **微软**、 **Facebook**、 **Slack** 和 **Docker** 这样的公司用于创建应用程序。



electron 应用

我所用的编辑器 **Visual Studio Code** 就是基于 `electron` 开发的



VS Code Mac版

实战

下面会分别用 `nw.js` 和 `electron` 做一个简单的 **Dome**。

由于 **CEF** 文档资料少且原生是 `C\C++` ， 虽然官方给出了 `java` 版的 **JCEF** ， 开发起来效率较低，故此不知演示。

这个是 **CEF** 官网，在 *External Projects* 章节列出支持语言：

> - Net (CEF3) - https://github.com/cefsharp/CefSharp
> - Net (CEF1) - https://bitbucket.org/fddima/cefglue
> - Net/Mono (CEF3) - https://bitbucket.org/xilium/xilium.cefglue
> - Net (CEF3) - https://bitbucket.org/chromiumfx/chromiumfx
> - Delphi (CEF1) - http://code.google.com/p/delphichromiumembedded/

- Delphi (CEF3) - https://github.com/hgourvest/dcef3
- Delphi (CEF3) - https://github.com/salvadordf/CEF4Delphi
- Go - https://github.com/CzarekTomczak/cef2go
- Java - https://bitbucket.org/chromiumembedded/java-cef
- Java - http://code.google.com/p/javacef/
- Python - http://code.google.com/p/cefpython/

NW => Hello, world!

从一个简单的例子来让我们看看如何编写一个 **NW** 应用。

- **第一步** 创建 `package.json` 配置文件

```
{
    "name": "helloworld",
    "main": "index.html",
    "icon": "img/app.png",
    "window": {
        "icon": "img/app.png"
    }
}
```

`main` 配置应用打开首页， `name` 配置应用的名称。

- **第二步** 创建 `index.html`

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-s
cale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Holle NW</title>
</head>
<style>
    html,
    body {
        height: 100%;
        margin: 0;
```

```
    }

    .box {
        height: 100%;
        display: flex; /* css3 弹性盒子 */
        justify-content: center;
        align-items: center;
    }
</style>

<body>
    <div class="box">
        <h1>Holle NW!</h1>
    </div>
</body>

</html>
```

这是一个简单的 **HTML** 文件，加入了一点CSS，目的是让 **Holle NW!** 水平垂直居中。

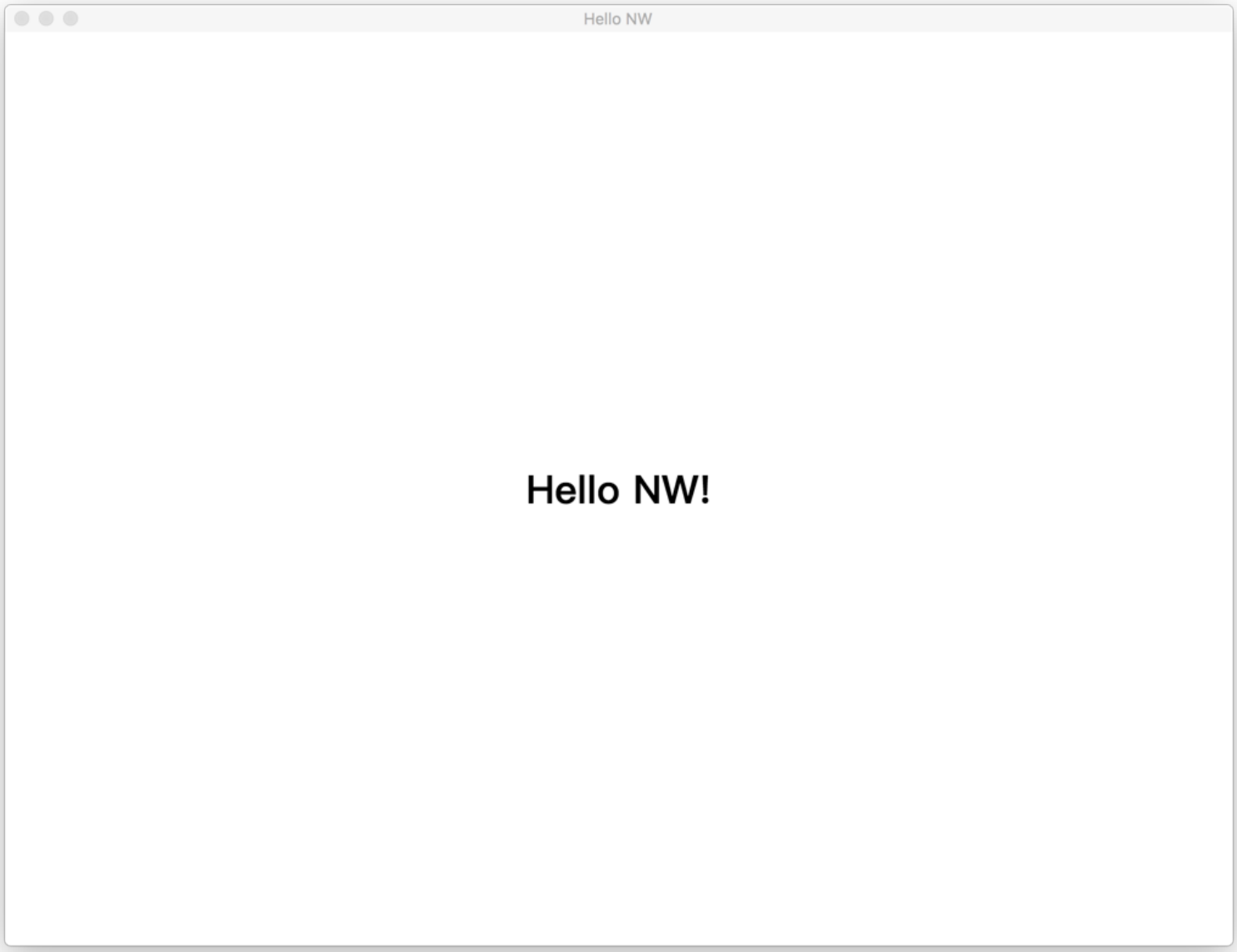
- **第三步** 打包应用

这里我只测试了 **Mac** 和 **Windows** 的打包，**Linux** 没有测试。

Mac打包应用： 在项目根目录执行以下命令，把所有文件压缩成 `app.nw` 文件。

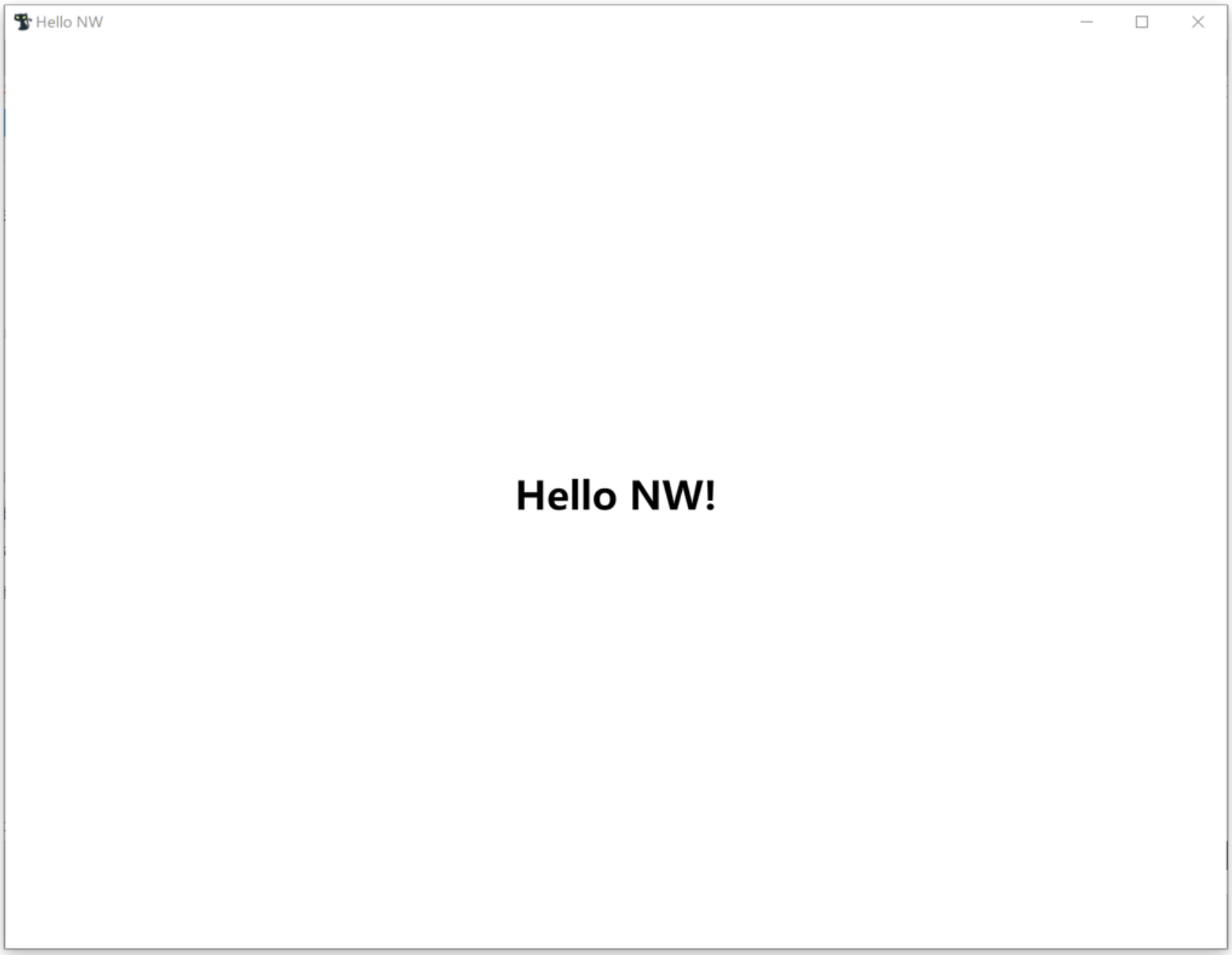
```
zip -r app.nw *
```

然后把`app.nw` 文件放到 `nwjs.app/Contents/Resources/` 目录下即可，效果如图:



Mac下运行效果

Windows打包应用： 将应用的所有相关文件打成一个名为`package.nw`的压缩包，将`package.nw`与**NW**可执行文件放到相同目录即可，效果如图：



Windons下运行效果

Electron => Hello, world!

Electron 可以让你使用纯 `JavaScript` 调用丰富的原生(操作系统) **APIs** 来创造桌面应用。

只需3个文件就可以构建一个简单的应用

```
your-app/
   ├──── package.json
   ├──── main.js
   └──── index.html
```

- **第一步** 创建配置文件

首先需要安装 **Node** 环境，用 `npm` 来创建一个应用的配置文件 `package.json`

```
npm init
```

在 `package.json` 里新增启动命令 `start`

```
{
    "name": "your-app",
    "version": "0.1.0",
    "main": "main.js",
    "scripts": {
      "start": "electron ."
    }
  }
```

- **第二步** 创建入口文件 `main.js`

```
const {app, BrowserWindow} = require('electron');
const path = require('path')
const url = require('url')

function createWindow() {
    win = new BrowserWindow({
        width: 1008,
        height: 759
    })

    win.loadURL(url.format({
        pathname: path.join(__dirname, 'index.html'),
        protocol: 'file:',
        slashes: true
    }))
}

app.on('ready', createWindow)
```

代码已经很清晰直观，`createWindow` 创建一个桌面窗口，而大小由 `width`、`height` 控制，`win.loadURL` 用来加载页面。

- **第三步** 创建展示文件 `index.html`

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-s
cale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Hello Electron</title>
</head>
<style>
    html,
    body {
        height: 100%;
        margin: 0;
    }

    .box {
        height: 100%;
        display: flex;
        /* css3 弹性盒子 */
        justify-content: center;
        align-items: center;
    }
</style>

<body>
    <div class="box">
        <h1>Hello Electron!</h1>
    </div>
</body>

</html>
```

这是一个简单的 **HTML** 文件，加入了一点CSS，目的是让 **Holle NW!** 水平垂直居中。

- **第四步** 打包应用

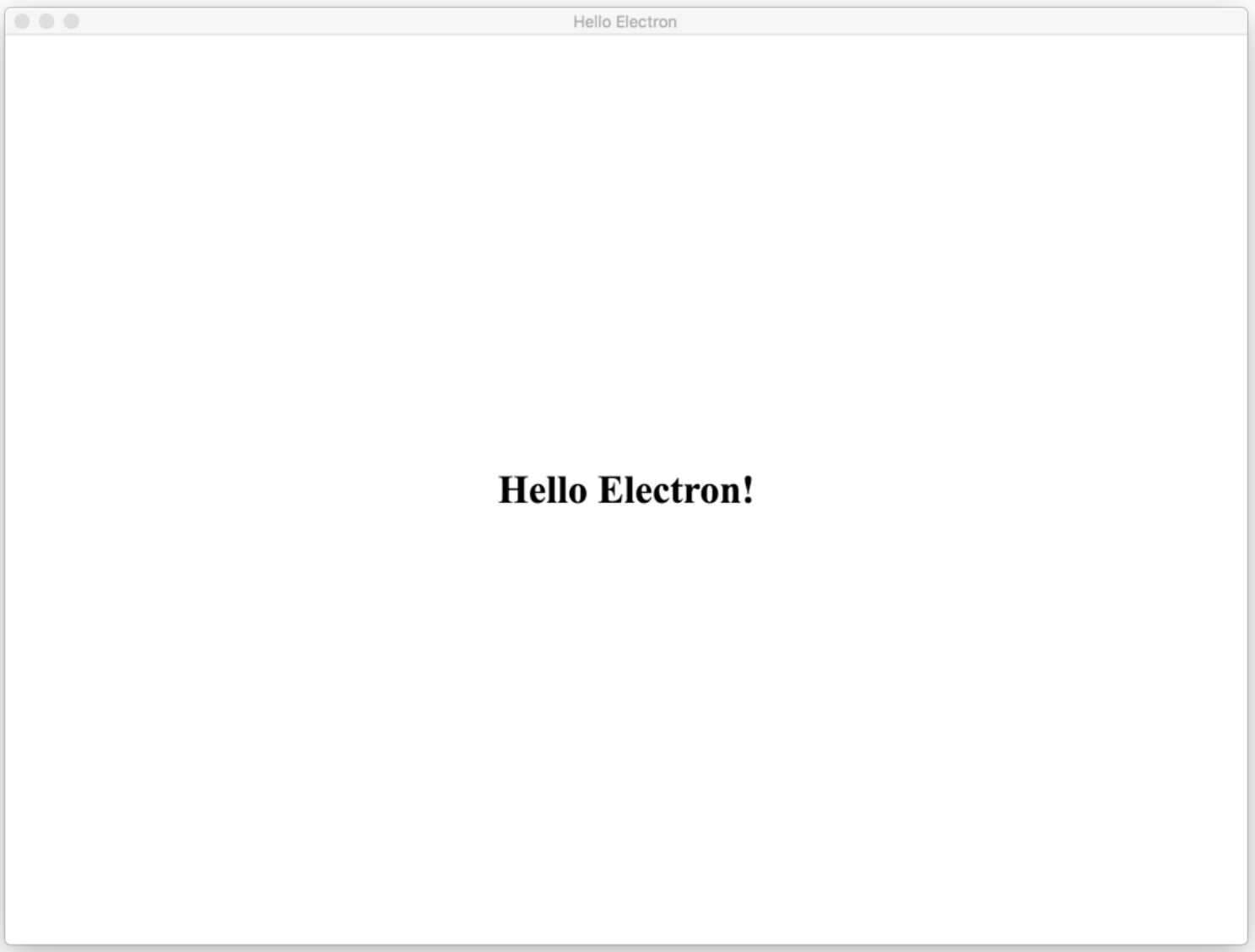
打包应用： 打包应用可以是用 `electron-packager` 工具进行打包，需要在 `package.json` 配置以下命令

```
"scripts": {
    "start": "electron .",
    "packager": "electron-packager ./ HelloElectron --all --out
./OutApp --version 0.0.1 --overwrite --ignore=node_modules --ico
n=./app/img/app.ico"
```

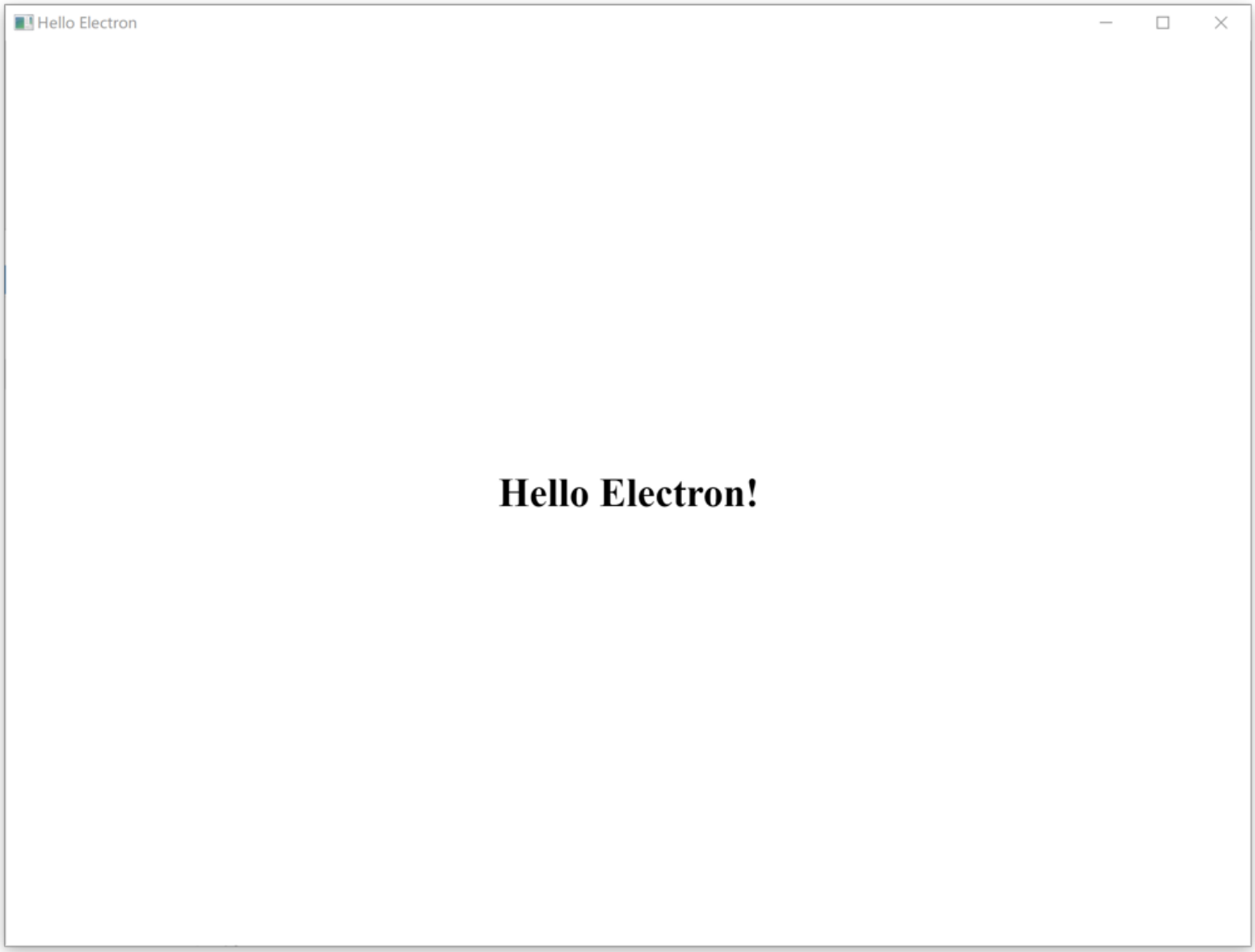
```
    },
```

然后，运行在终端执行命令 `npm run packagerMac` 即可打包 `linux`、`Mac`、`windows` 三大平台应用包，效果如图:

打包后的应用



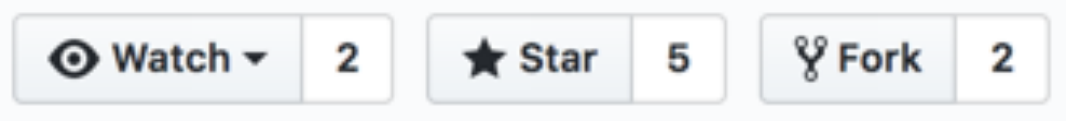
Mac下运行效果

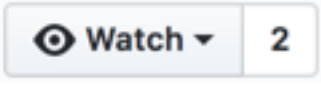


Windows下运行效果

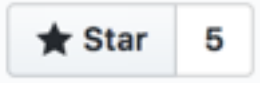
GitHub关注度和活跃度

首先我们需要先了解一下 **GitHub** 的以下三个状态的意思，





：表示你以后会关注这个项目的所有动态，这个项目以后只要发生变动，如被别人提交了 `pull request`、被别人发起了`issue` 等等情况，你都会在自己的个人通知中心，收到一条通知消息，如果你设置了个人邮箱，那么你的邮箱也可能收到相应的邮件。



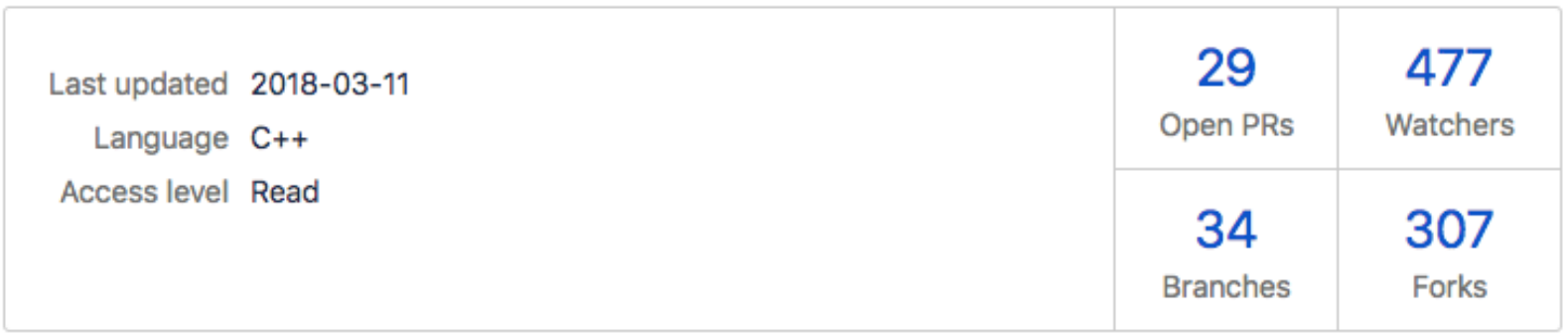
：表示你喜欢这个项目或者通俗点，可以把他理解成朋友圈的点赞，表示对这个项目的支持。

：当选择 `fork`，相当于你自己有了一份原项目的拷贝，当然这个拷贝只是针对当时的项目文件，如果后续原项目文件发生改变，你必须通过其他的方式去同步。

*（一般用于修改**bug**和优化项目或者在此项目上开发新功能等）*

CEF

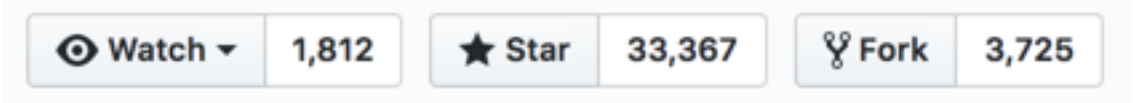
CEF 在 **GitHub** 找不到项目，这个官网 提供的数据，如图*（由于在**GitHub** 没有项目，相关数据无法准确统计）*。



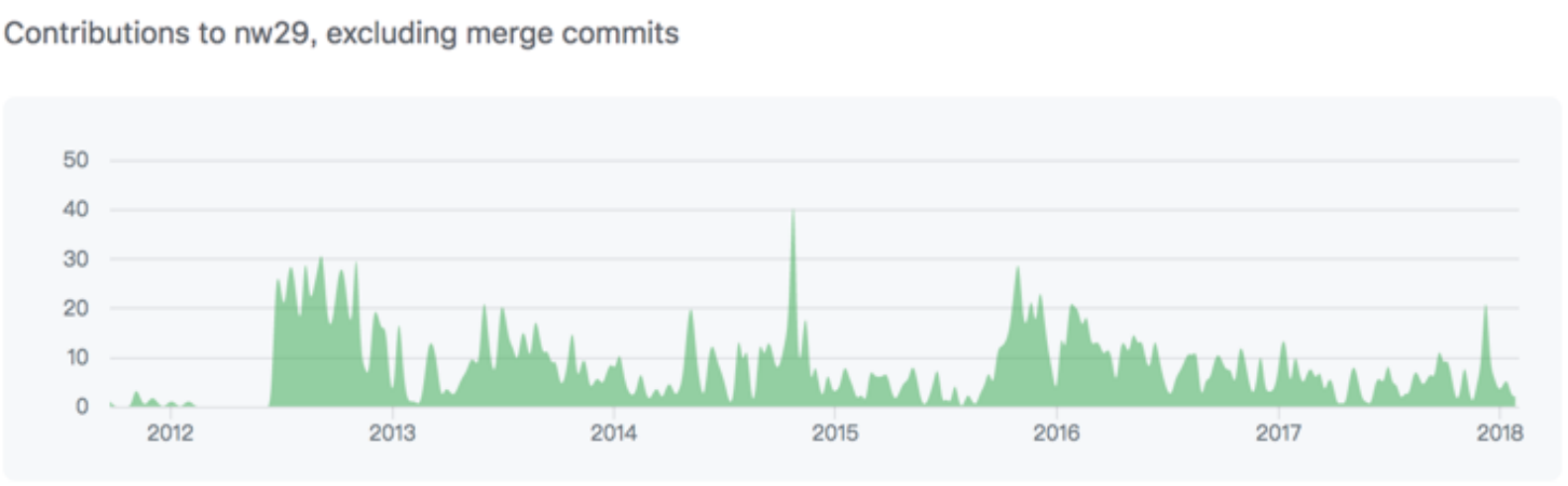
CEF关注度

NW

NW关注度：



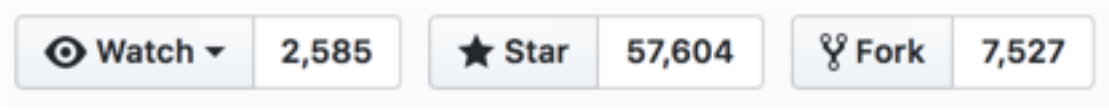
NW活跃度：如图



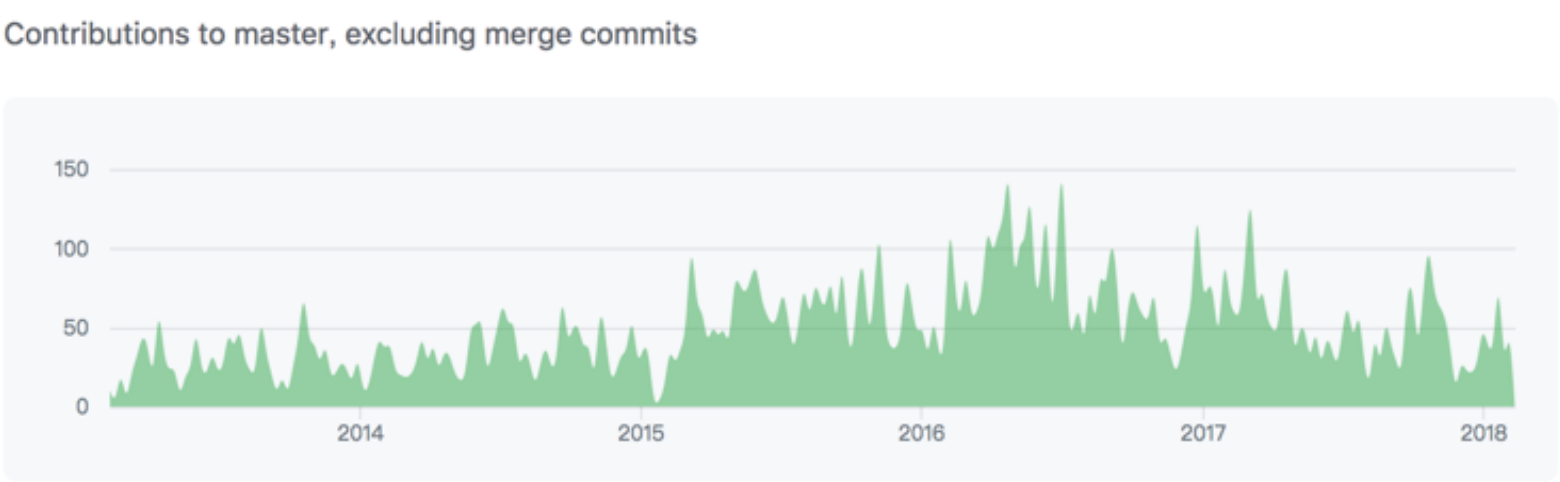
2011年~2018年提交量

Electron

Electron关注度：



活跃度： 如图



electron2013年~2018年提交量

通过以上的 **市场调研**、**实战**、**GitHub关注度和活跃度** 等 **Electron** 都占有优势，如下

- 市场案例较多，各大型企业都在使用
- 开发实战代码更直观，容易理解和维护，各种文档健全、网络资料较多且质量较高，周边辅助工具齐全，开发效率可大大提高
- **GitHub** 关注度和活跃度持续攀升

所以结合以上情况，之后会用以下技术栈基于 **branch** 做一个完善的案例

- **跨平台桌面应用框架：** `electron` *（Chromium + Node.js）*
- **UI库：** `iView`
- **js框架：** `Vue.js`
- **自动化构建工具：** `webpack`
- **HTML5、CSS3、ES6**
