

# Développez des applications web desktop avec Electron de GitHub



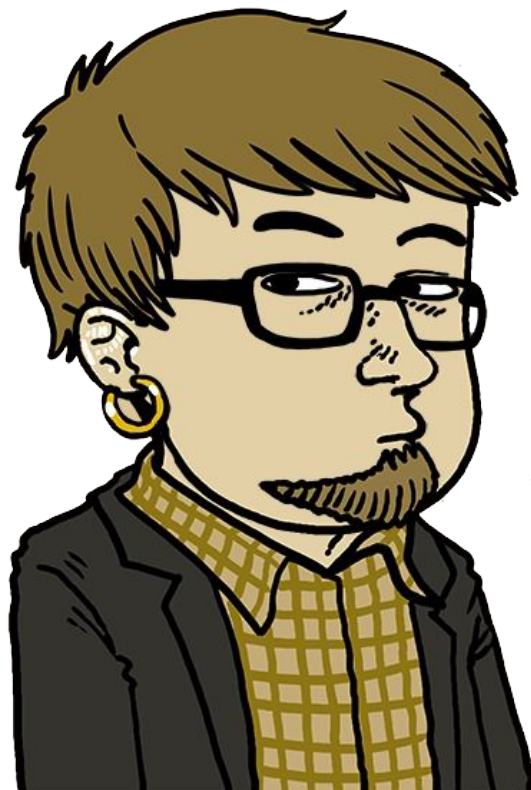
[ Matinale Zenika Nantes ]

# Yvonnick Frin

- Consultant Zenika Nantes
- Développeur Front-End

 @YvonnickFrin

 @frinyvonnick



# Eric Briand

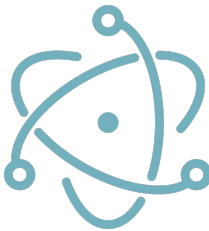
- Consultant Zenika Nantes
- Dev' touche à tout
- Co-organisateur du Docker Meetup Nantes

 @eric\_briand

 @ebriand



# Sommaire



## 01 Qu'est-ce qu'Electron ?

Les motivations du projet, son historique, ses fonctionnalités

## 02 Ecosystème

Son utilisation, activité, et concurrents

## 03 Architecture

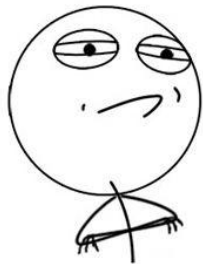
Qu'est-ce qui constitue une application Electron

## 04 Live coding

Du code, du vrai

## 05 Stratégie de build

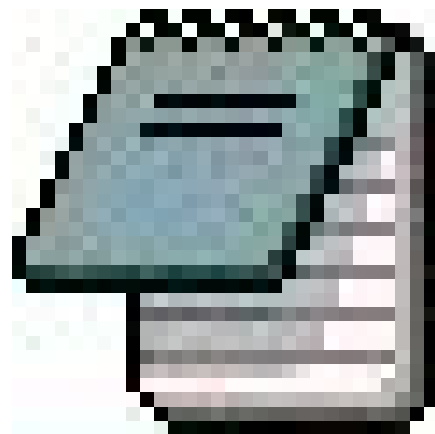
Testing, packaging, update...



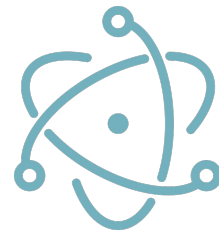
# Pourquoi faire une application desktop en 2016 ?



- Accéder aux fonctionnalités de l'OS hôte
- Confort d'utilisation (raccourcis clavier, menus...)
- Profiter des App Stores (OSX/Windows)
- Visibilité sur le poste



# Pourquoi utiliser les technologies web pour du desktop ?

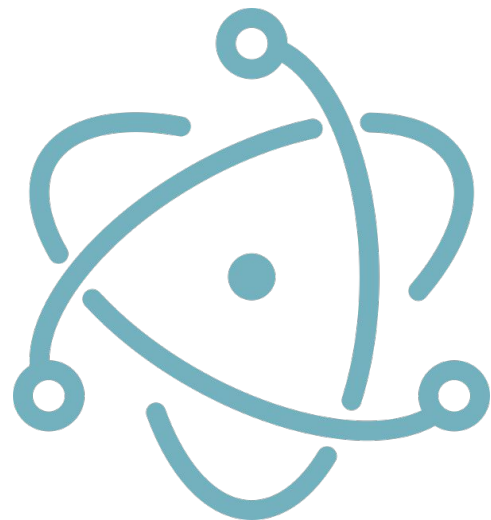


- Ecosystème riche
- Forte communauté
- Performances comparables au natif avec V8
- Simplicité de développement



# Qu'est-ce qu'Electron ?

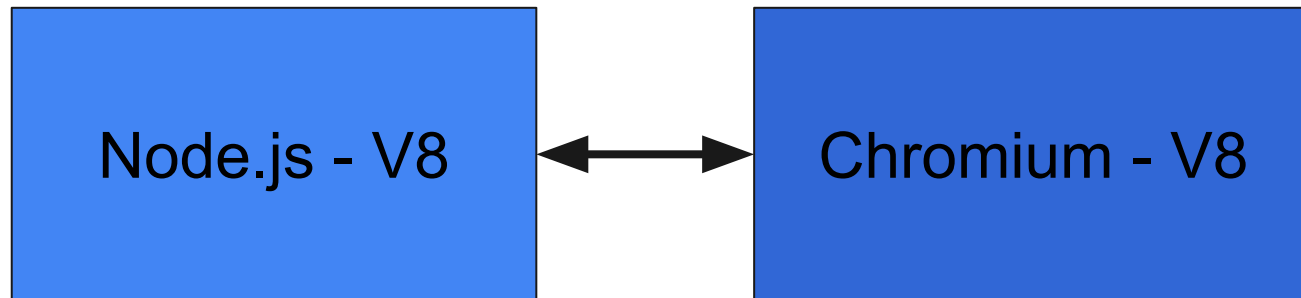
- Développé initialement pour les besoins de l'éditeur Atom
- Créé par GitHub
- Framework pour créer des applications natives cross-platform avec des technologies web
- Open source depuis juillet 2014



# Vision macro d'Electron

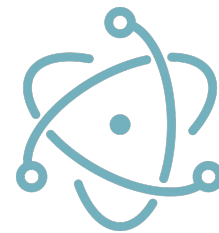


Electron





# Principales fonctionnalités



Automatic  
updates



Native menus  
& notifications



App crash  
reporting



Debugging  
& profiling



Windows  
installers

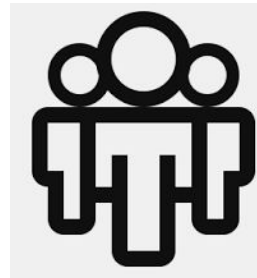
# Activité GitHub



35k+



4k+

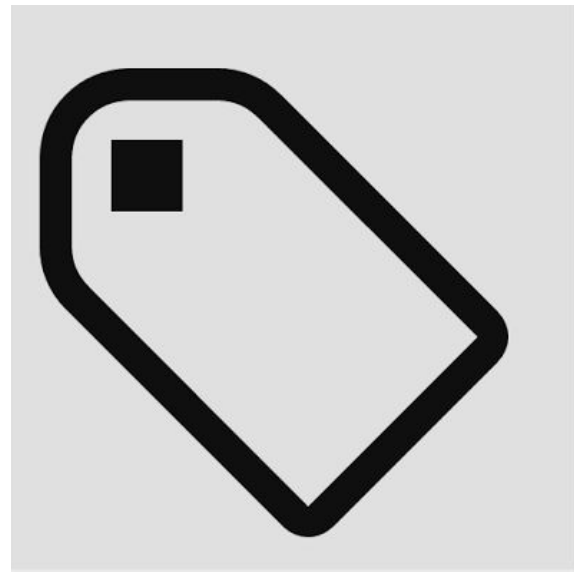


500+

# Rythme des releases



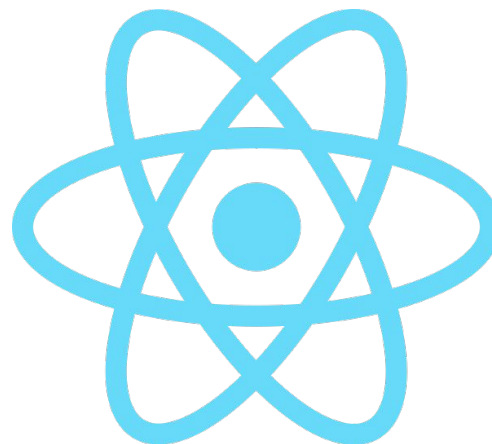
- Plus de 250 releases depuis juillet 2014
- 10 mai 2016 : 1.0
- Une release toutes les semaines et une majeure tous les mois
- Actuellement en 1.4.11 (enfin quand on a regardé après le café du matin...)



# Alternatives



**NW.JS**

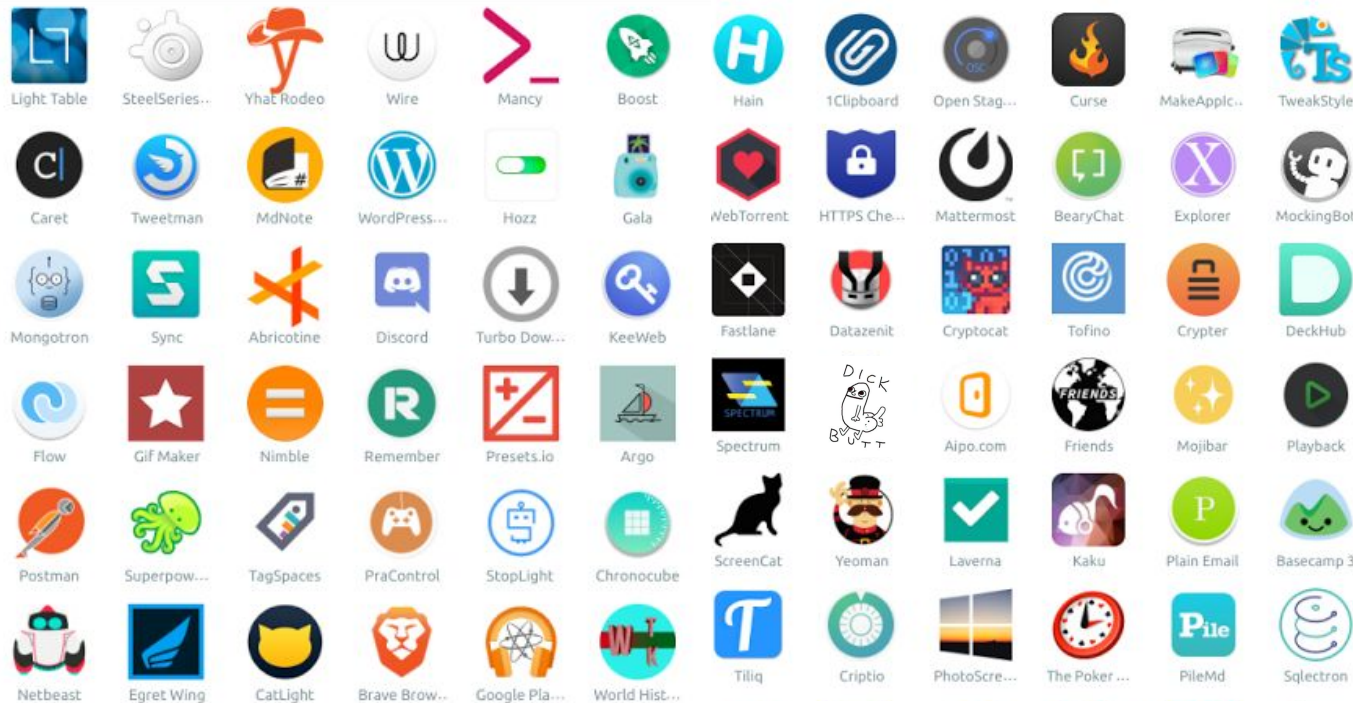
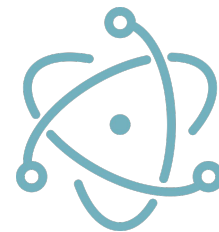


**React Native**

# Quelles applications l'utilisent ?



# Et bien d'autres !



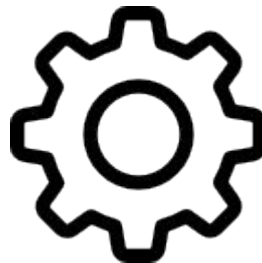
# Plugins



Visuals



Packagers



Tools



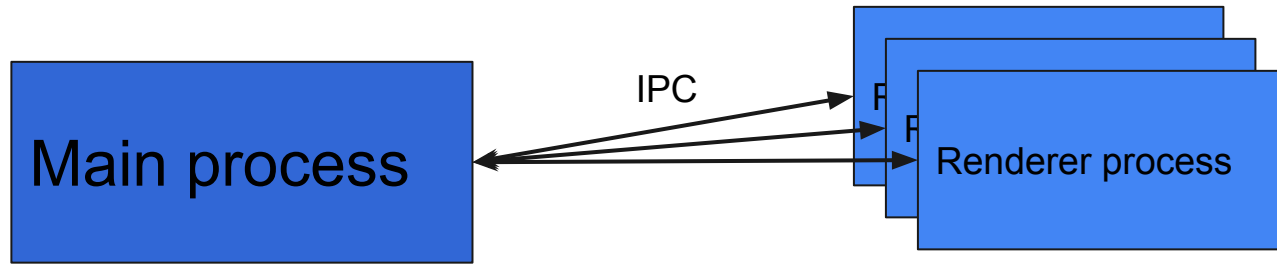
Updaters

<https://github.com/sindresorhus/awesome-electron>

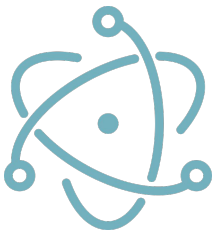
# Architecture logicielle d'une application Electron



## Electron

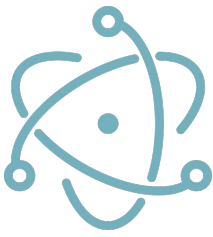






# Main Process

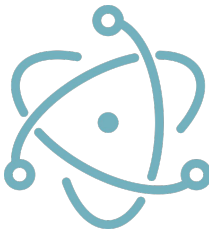
- Gère les instances des fenêtres
- Accède aux fonctionnalités du système
- Réalise les opérations coûteuses



# Renderer Process

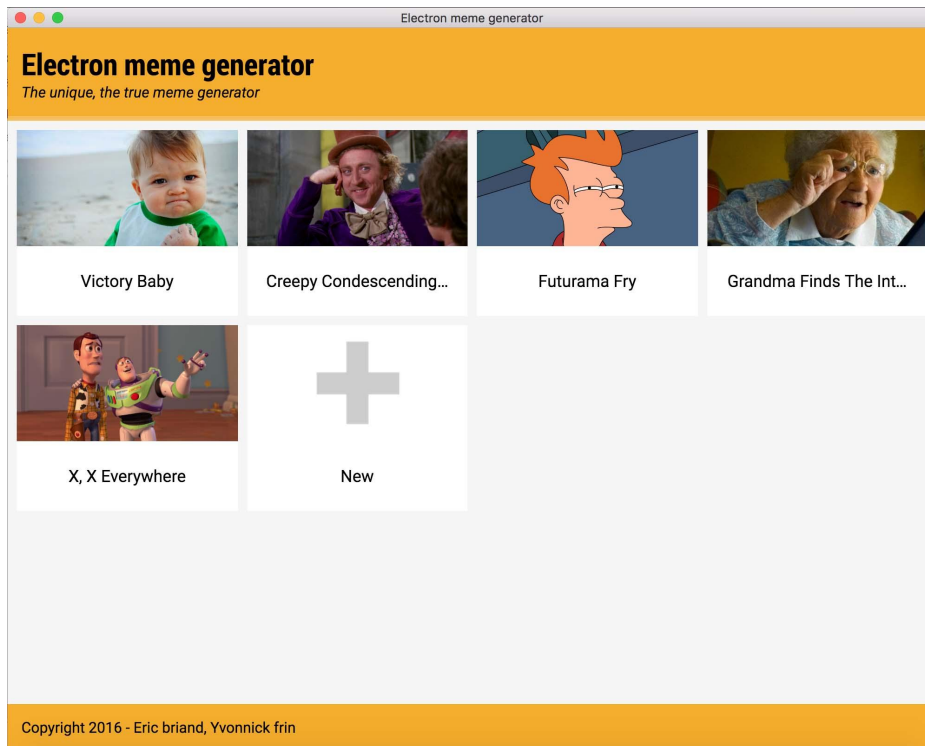
- Gère l'affichage des pages
- Gère les interactions utilisateurs
- Un renderer process par fenêtre

# IPC

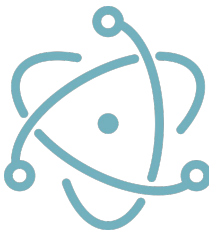


- Gère la communication entre MainProcess et les instances de RendererProcess
- Basé sur l'IPC de chromium
- Pour communiquer entre les instances de RendererProcess, il faut passer par le MainProcess

# Live coding : meme generator



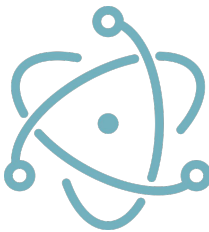
# Spectron



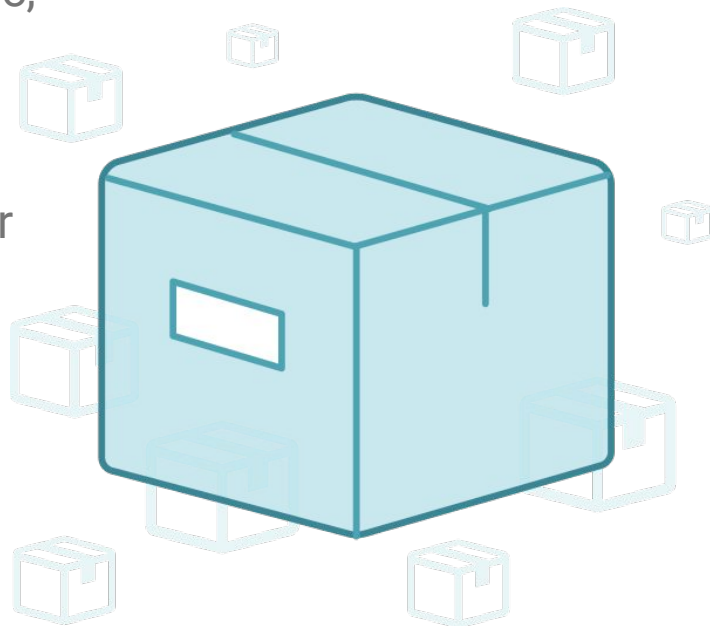
- Framework de test pour electron
- Basé sur le ChromeDriver
- Permet de gérer facilement le cycle de vie de l'appli
- Agnostique du framework de test utilisé

```
it('opens a window displaying the memes', function () {  
  return app.client.getWindowCount().should.eventually.equal(1)  
    .browserWindow.isMinimized().should.eventually.be.false  
    .browserWindow.isDevToolsOpened().should.eventually.be.false  
    .browserWindow.isVisible().should.eventually.be.true  
    .browserWindow.isFocused().should.eventually.be.true  
    .browserWindow.getBounds().should.eventually.have.property('width').and.be.above(0)  
    .browserWindow.getBounds().should.eventually.have.property('height').and.be.above(0)  
    .getTitle().should.eventually.equal('Electron meme generator')  
})
```

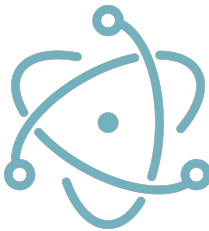
# Packaging



- Cross-platform en 32 ou 64 bits : windows, mac, linux
- Package out of the box avec electron-packager
- Création d'installateur windows
- Possibilité de packager pour les stores mac & windows



# Mise à jour



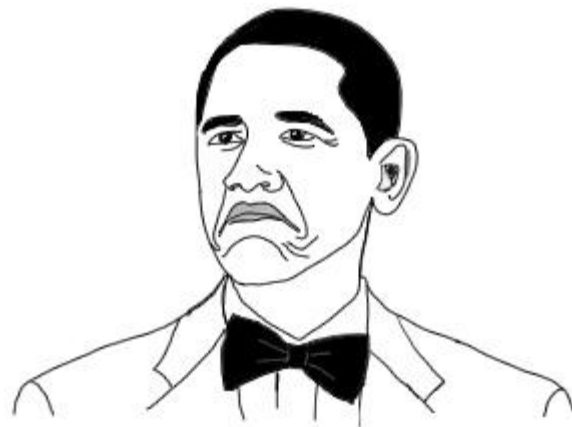
- Appel d'un service REST distant pour avoir les infos sur les dernières versions
- Peut être basée sur les GitHub releases
- electron-release-server
- Auto update via Squirrel



# On a aimé



- Documentation très bien faite
- Courbe d'apprentissage
- Ecosystème Node.js + web à dispo
- Un seul navigateur à supporter
- TRES actif



**NOT BAD**



# <https://github.com/electron/electron-api-demos>



ELECTRON API DEMOS

WINDOWS

Create and manage **windows**

Handling window **crashes and hangs**

MENUS

Customize **menus**

Register keyboard **shortcuts**

NATIVE USER INTERFACE

Open **external links** or system **file manager**

Use system **dialogs**

Put your app in the **tray**

COMMUNICATION

Communicate between the **two processes**

SYSTEM

Get app or user **system information**

Copy and paste from the **clipboard**

Launch app from **protocol handler**

MEDIA

**Print** to PDF

Take a **screenshot**

About

<> with ❤️ by GitHub

Create and Manage Windows

The `BrowserWindow` module in Electron allows you to create a new browser window or manage an existing one.

Each browser window is a separate process, known as the renderer process. This process, like the main process that controls the life cycle of the app, has full access to the Node.js APIs.

Open the [full API documentation](#) in your browser.

Create a new window

SUPPORTS: WIN, OS X, LINUX | PROCESS: MAIN

Manage window state

SUPPORTS: WIN, OS X, LINUX | PROCESS: MAIN

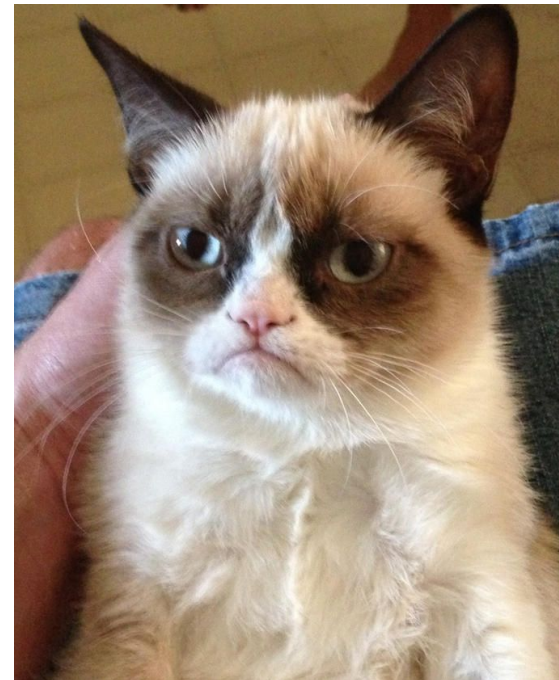
Create a frameless window

SUPPORTS: WIN, OS X, LINUX | PROCESS: MAIN

# On a moins aimé



- N'abstrait pas totalement les spécificités OS
- Privilégie les fonctionnalités communes à tous les OS
- Taille du “livrable”
- Pas de support mobile





# Merci !

Yvonnick Frin (@YvonnickFrin) Eric Briand (@eric\_briand)

<http://electron.atom.io>

<https://github.com/sindresorhus/awesome-electron>

<https://github.com/frinyvonnick/electron-meme-generator>

<https://github.com/ebriand/electron-meme-generator-ou-pas>