



# Développez des applications web desktop avec Electron de Github

Yvonnick Frin @YvonnickFrin

Eric Briand @eric\_briand

# 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





zenika

<animés par la passion>

# CODING THE WORLD



N°3 Great Place To Work des moins de 500 salariés

LILLE

SINGAPOUR

NANTES

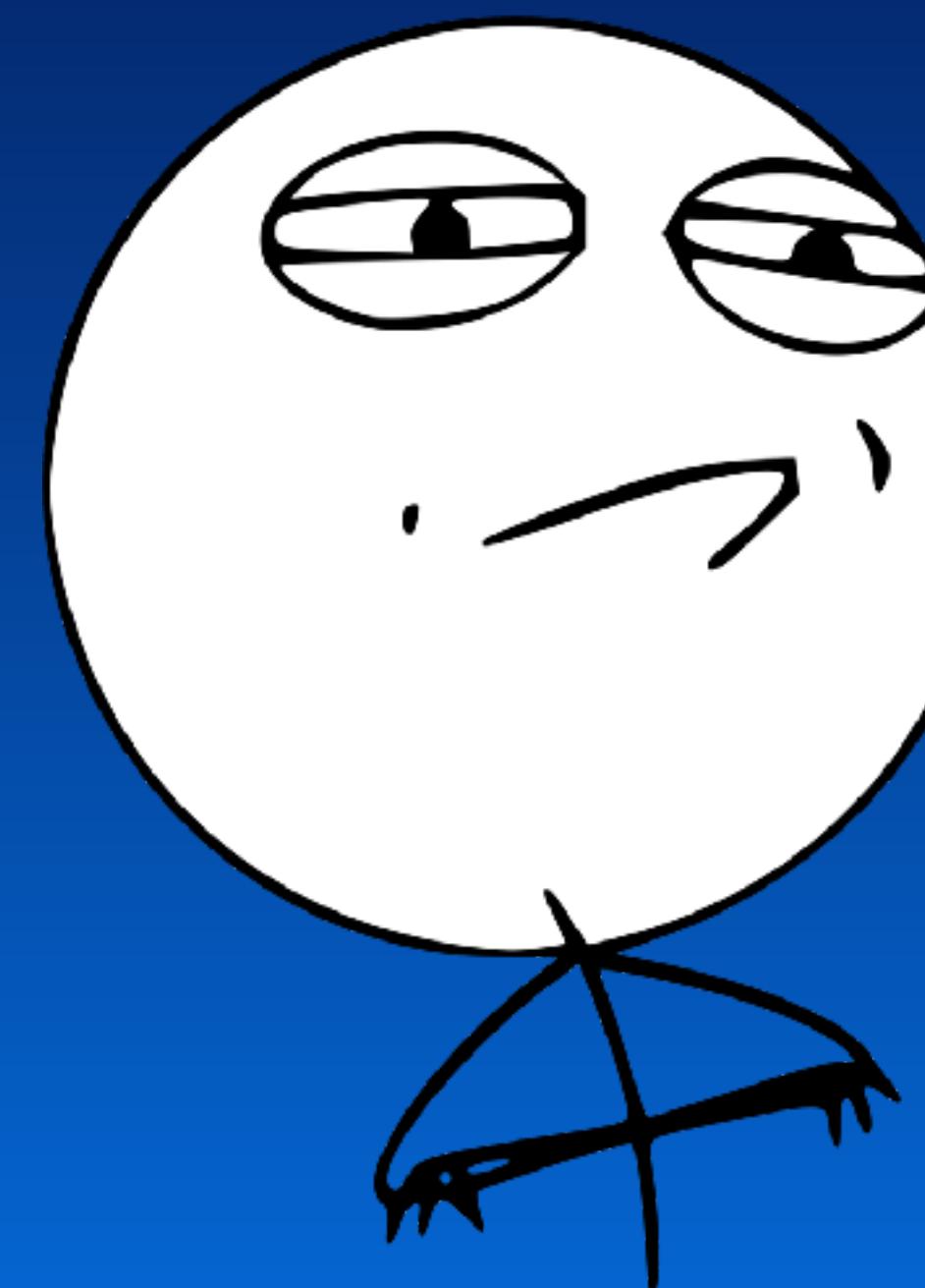
BORDEAUX

PARIS

RENNES

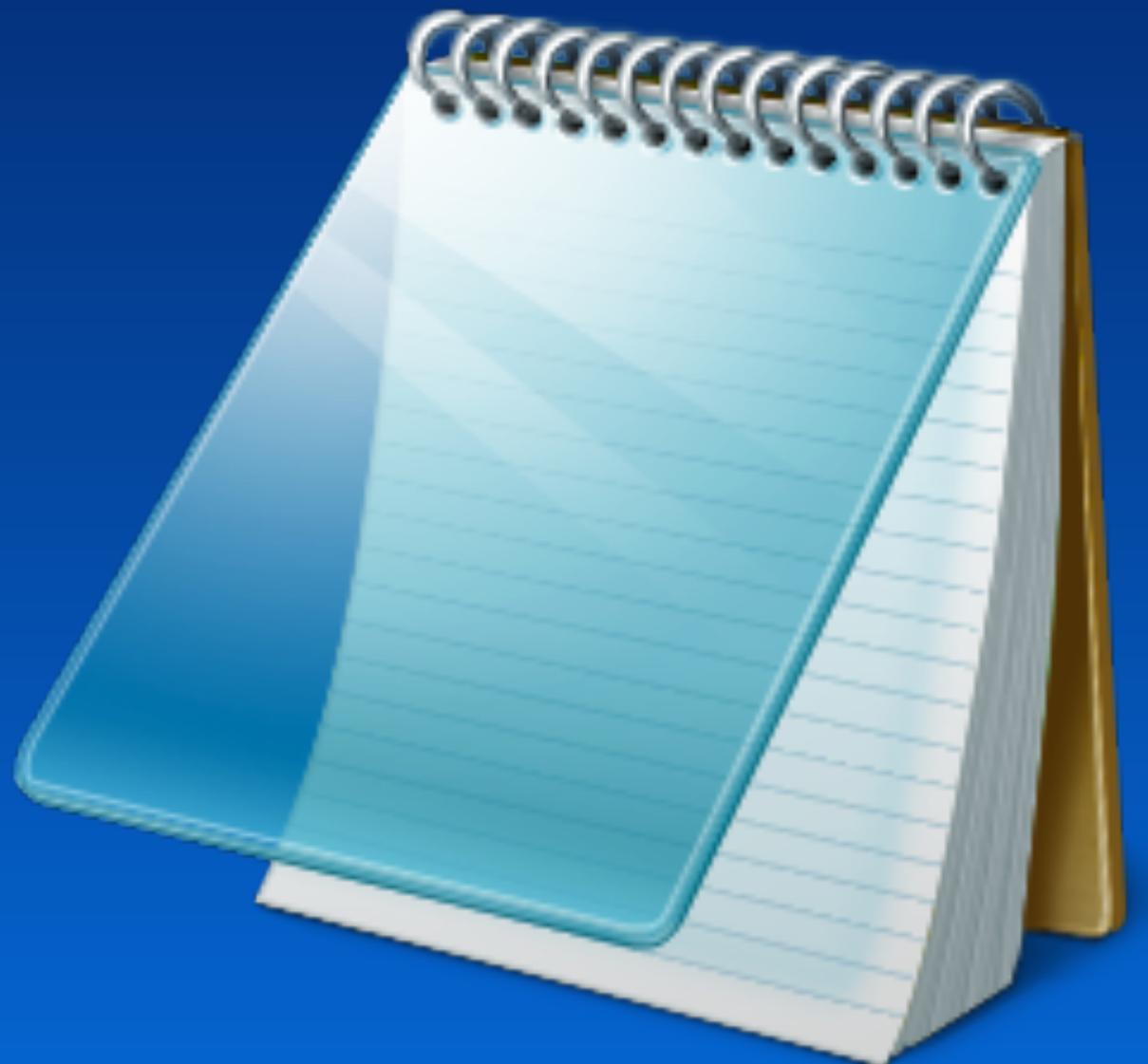
LYON

# Notre présentation



# Pourquoi faire du desktop en 2017 ?

- Accéder aux fonctionnalités de l'OS
- Confort d'utilisation
- Profiter des App Stores
- Visibilité sur le poste



# Des technos web pour du desktop ?

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



# Electron

- Framework pour coder des applications desktop avec des technologies web
- Développé initialement pour l'éditeur Atom sous le nom d'Atom Shell
- Crée par GitHub et open source depuis 2014



# Activité GitHub



44K



13K



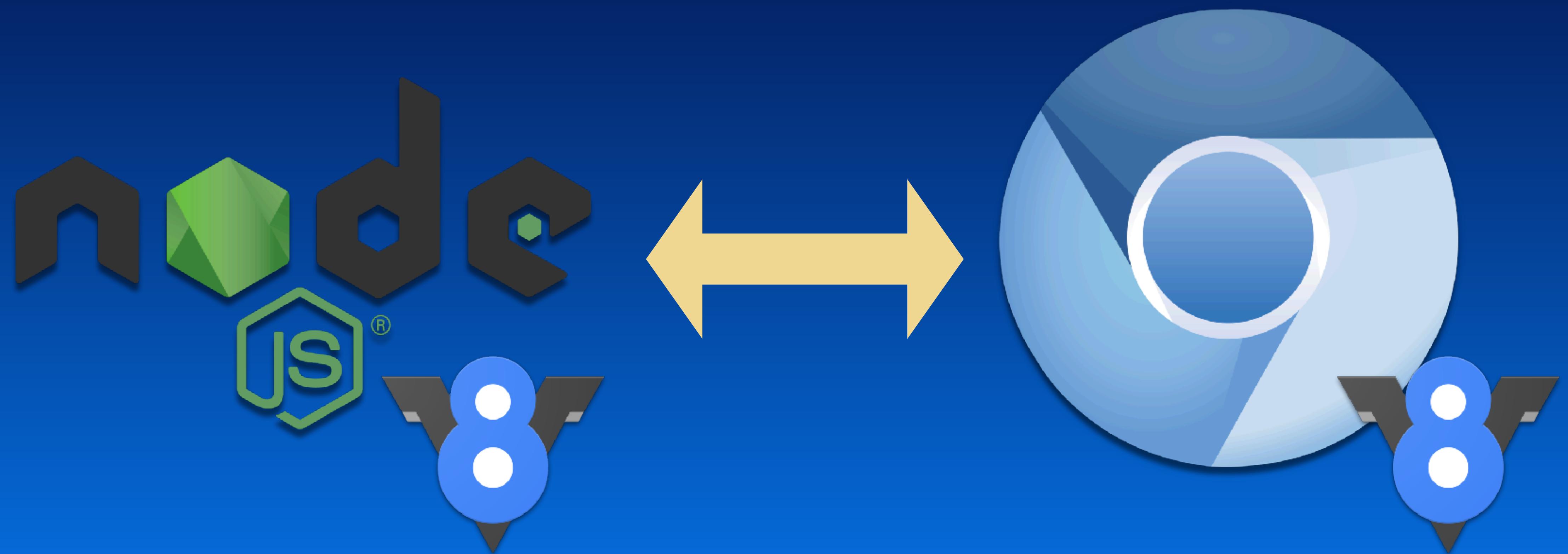
560

# Rythme des releases

- Plus de 270 releases depuis 2014
- 10 mai 2016 : 1.0
- Une release toutes les semaines et une majeure tous les mois
- Actuellement en 1.6.2 (enfin quand on a regardé après le café du matin...)



# Vision macro



# “The hard parts made easy”



Updates  
automatiques



Menus natifs  
& notifications



App crash  
reporting



Debugging  
& profiling

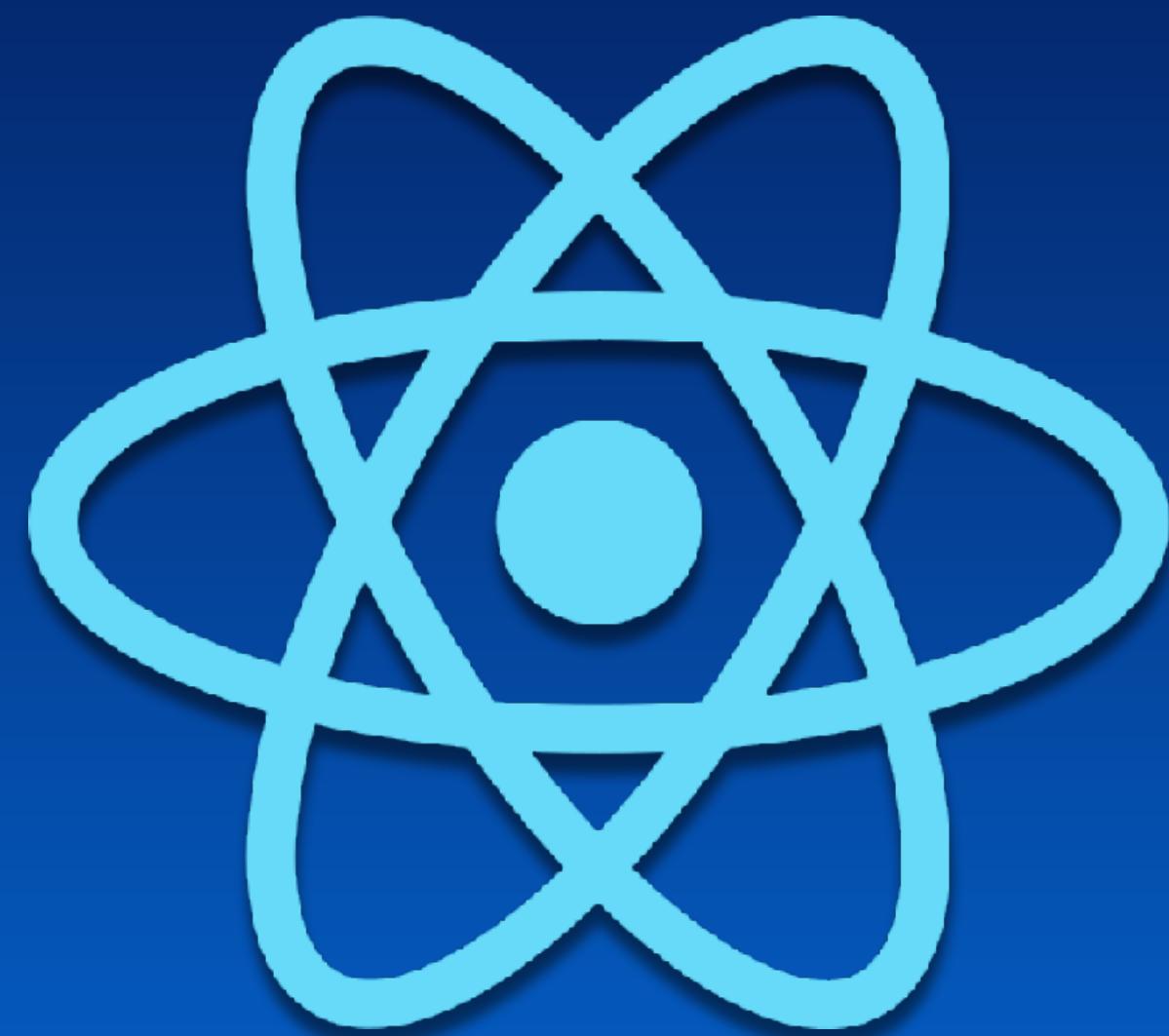


Installateurs  
Windows

# Alternatives

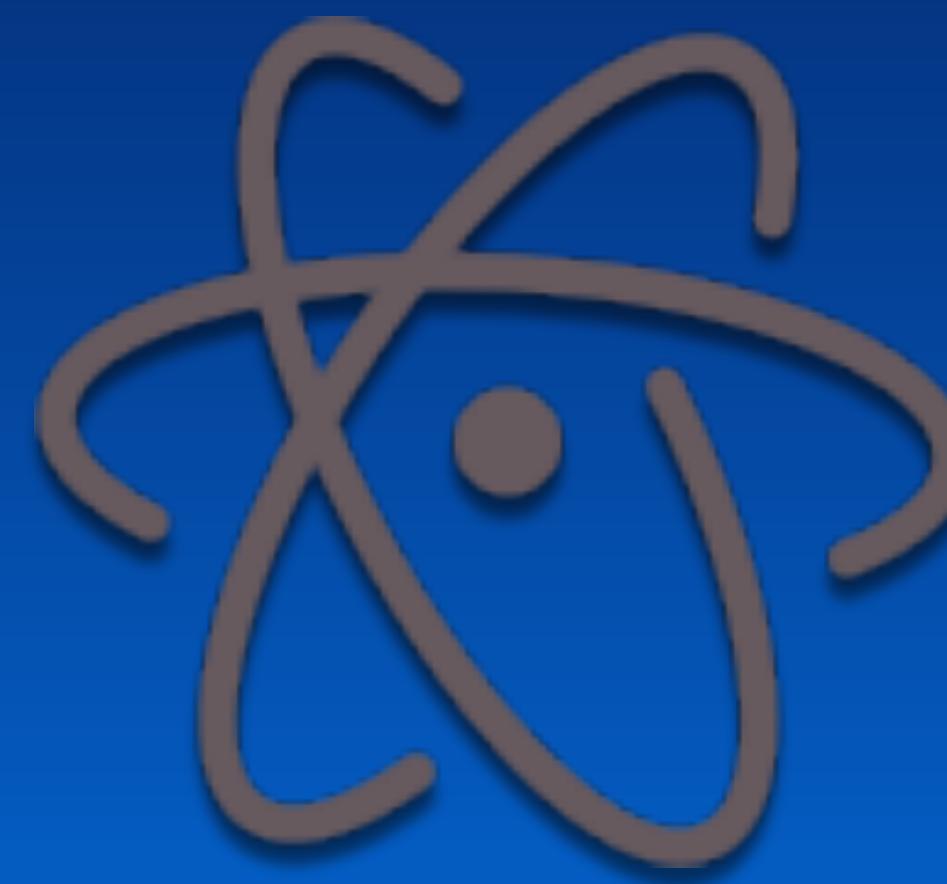


NW.js



Reactive Native

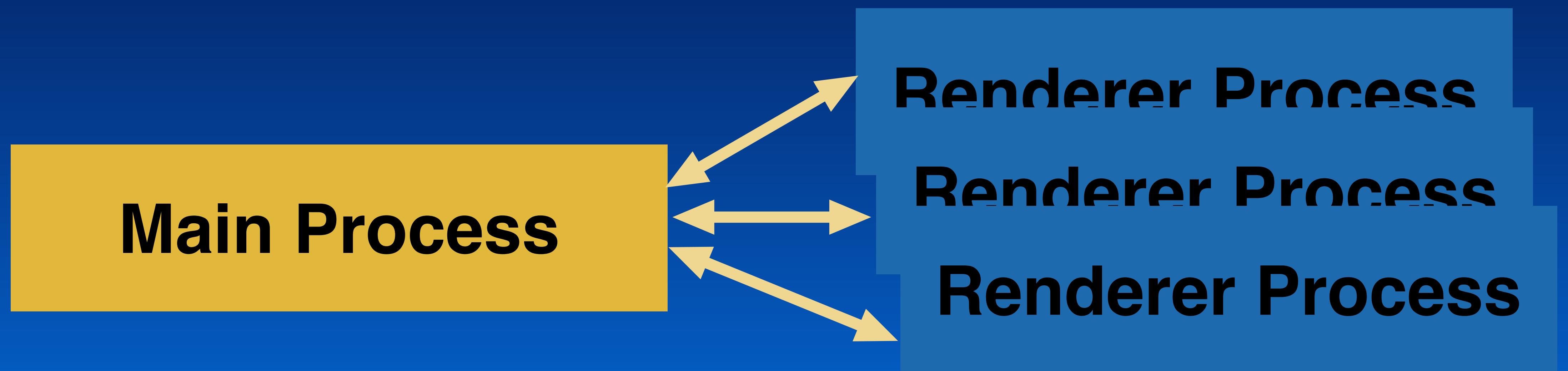
# Quelles applications l'utilisent ?



# Quelles applications l'utilisent ?



# Architecture d'une application



# Main Process

---

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

# Renderer Process

---

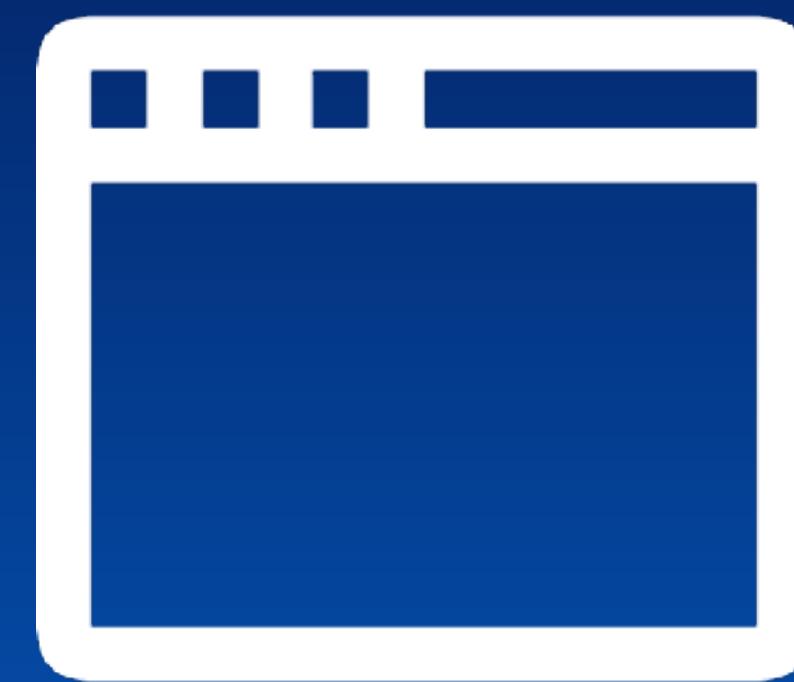
- Gère l'affichage des pages
- Gère les interactions utilisateurs
- Un renderer process par fenêtre

# IPC

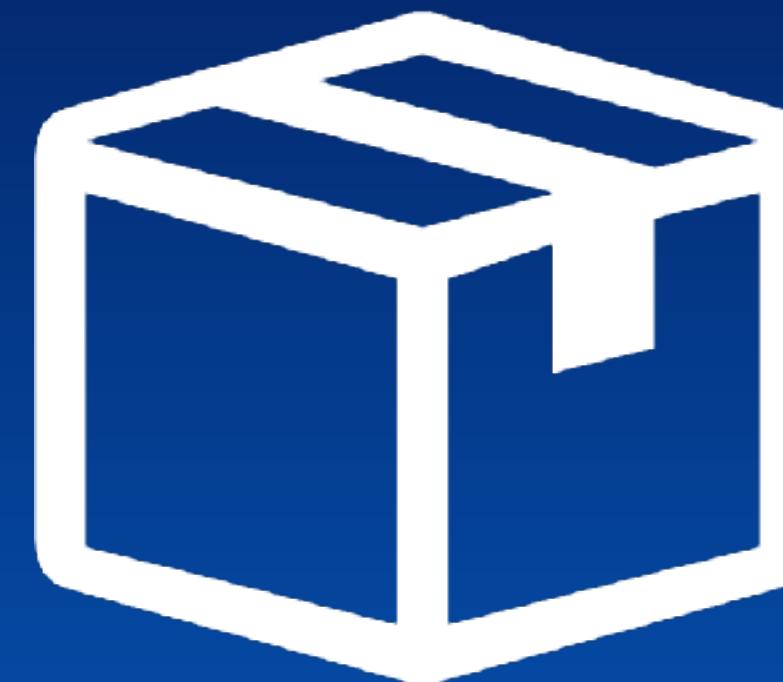
---

- 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

# Plugins



Visuals



Packagers



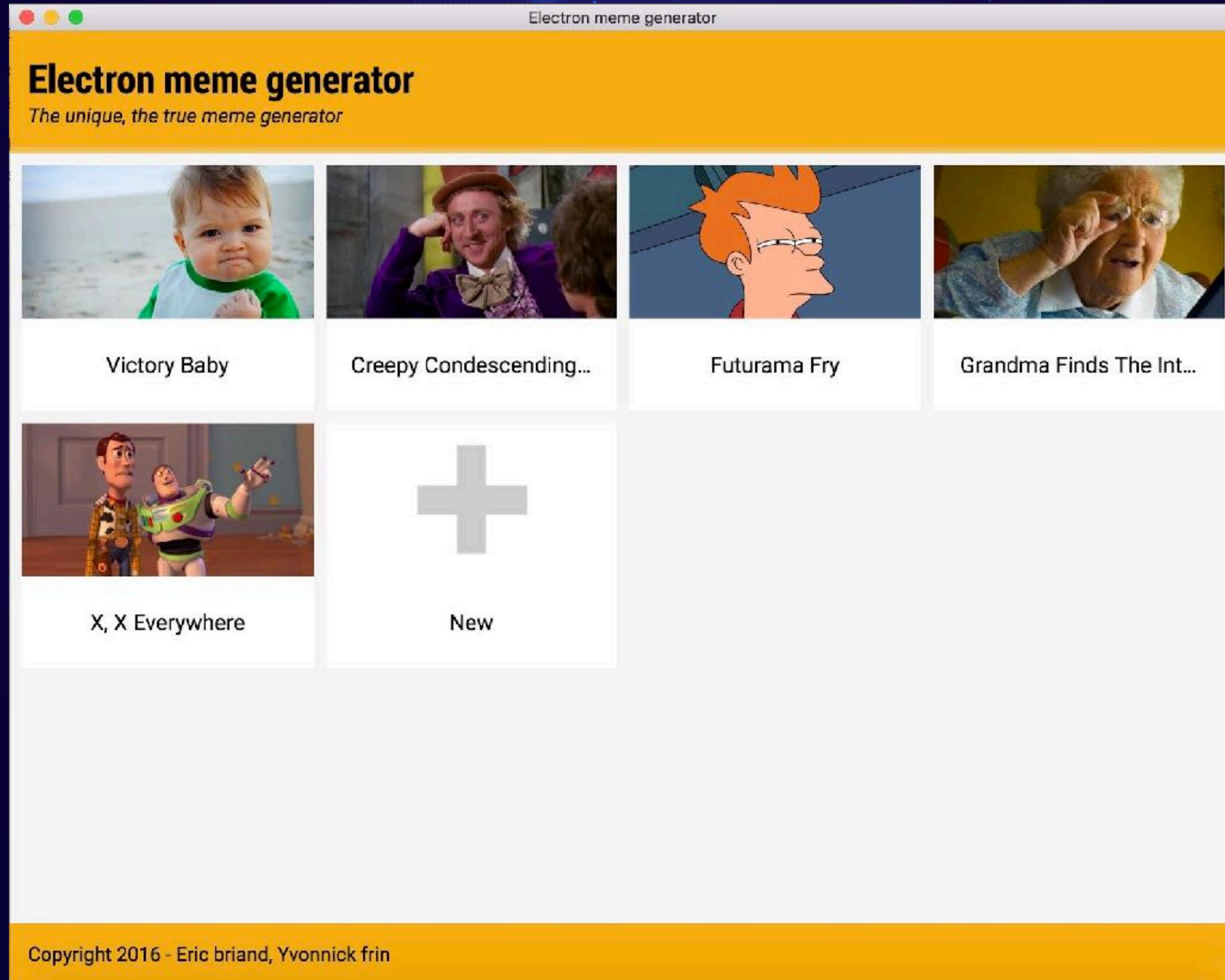
Tools



Updaters

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

# 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

# Packaging

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



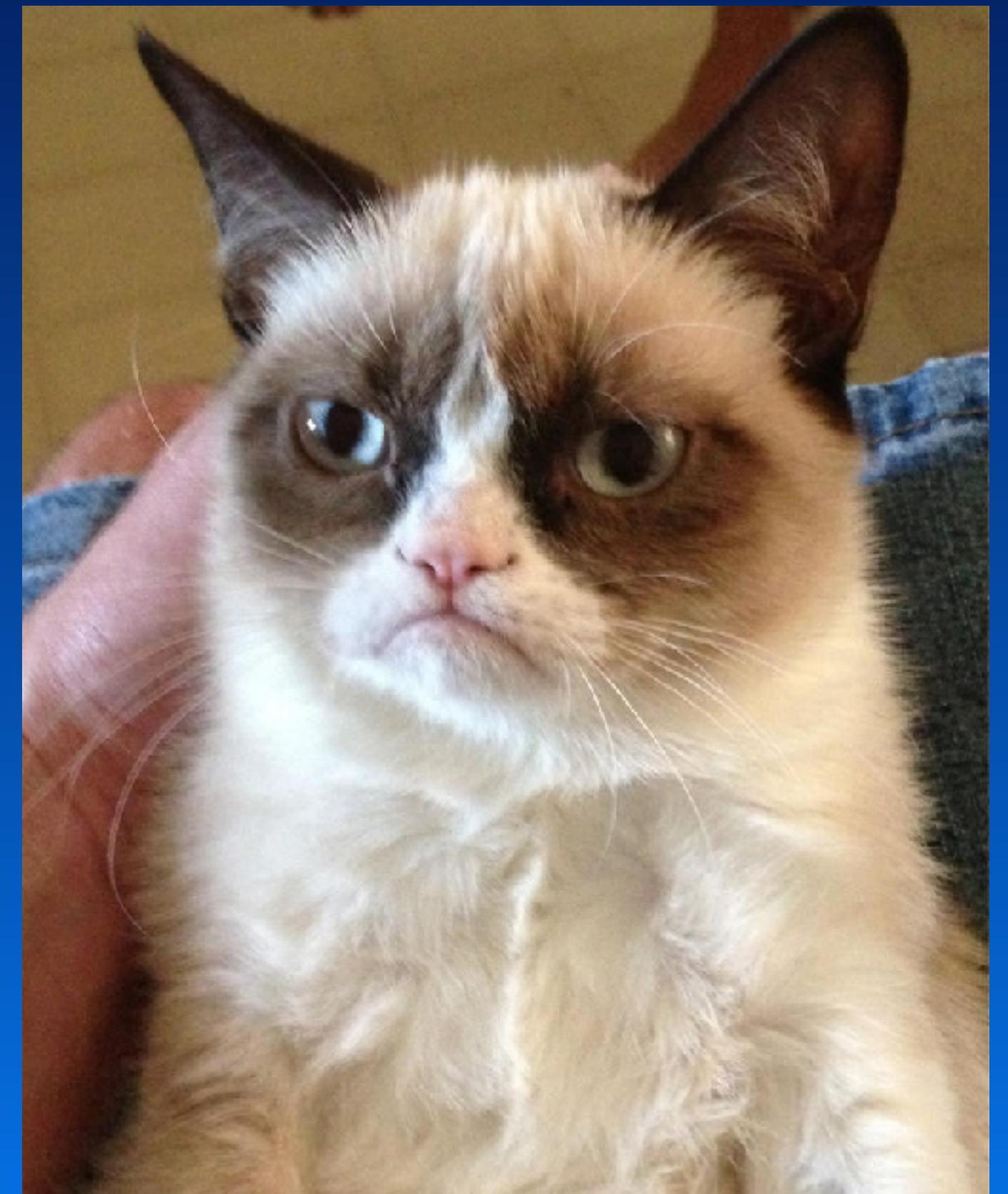
# Mise à jour

- 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 moins aimé

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



# 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>

The screenshot shows the Electron API Demos website. The left sidebar contains a navigation menu with sections like WINDOWS, MENUS, NATIVE USER INTERFACE, COMMUNICATION, SYSTEM, and MEDIA. The 'WINDOWS' section is currently selected, with its sub-section 'Create and manage windows' highlighted. The main content area on the right is titled 'Create and Manage Windows'. It explains that the `BrowserWindow` module allows creating new browser windows or managing existing ones. It notes that each window is a separate process (renderer process) controlled by the main process. A link to the full API documentation is provided. Below this, three specific examples are listed: 'Create a new window', 'Manage window state', and 'Create a frameless window', each with a brief description and compatibility information.

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.

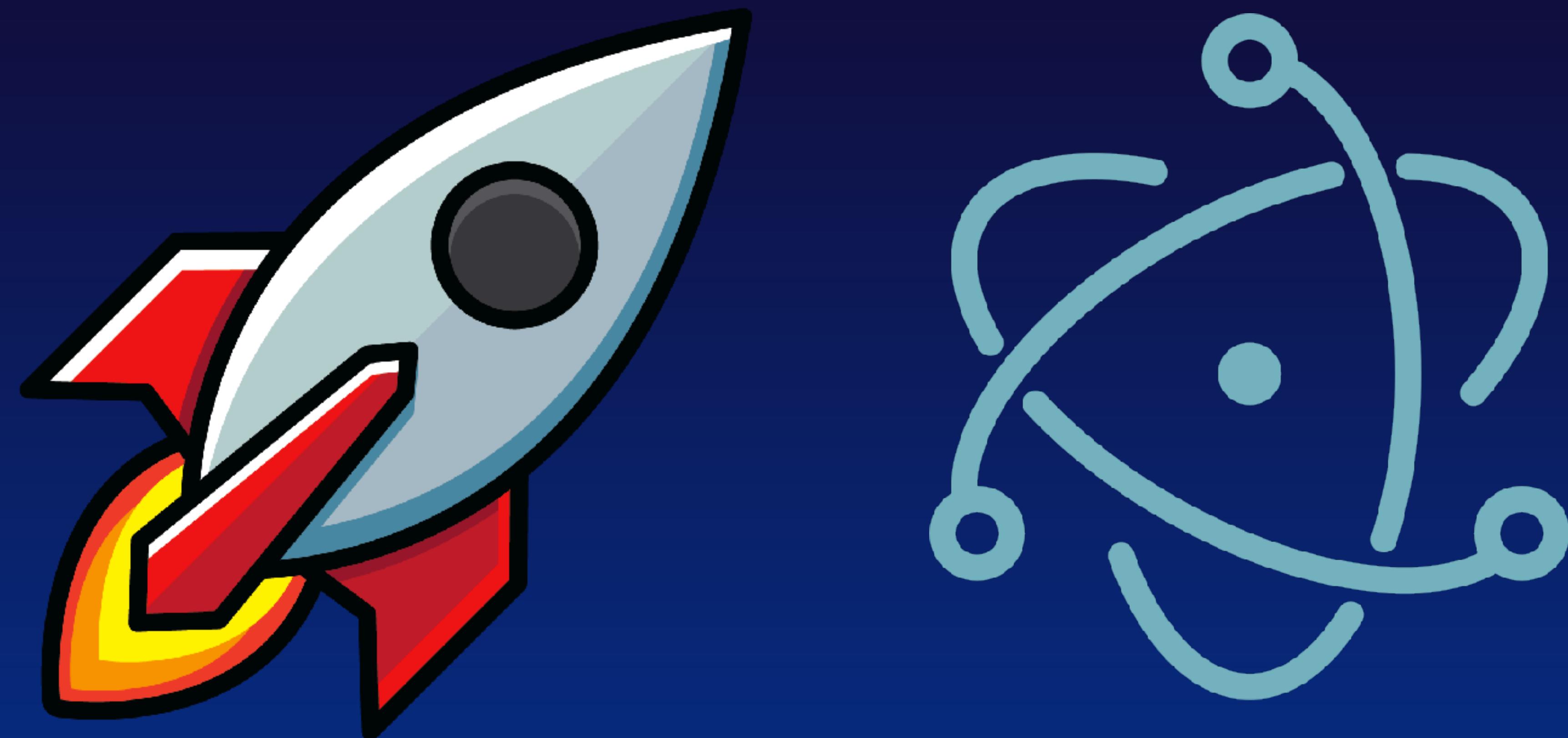
Each browser window is a separate process, known as the renderer process. This process, like the main process that controls the life cycle of the app, has full access to the Node.js APIs.

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





# Merci !

Yvonnick Frin @YvonnickFrin  
Eric Briand @eric\_briand