ELECTRON

Build your first cross-platform desktop app using Electron

Kevin Bluer

Bluer & Company

Agenda

Agenda

- What is Electron?
- Who Uses It?
- Differences to NW.js
- When Should You Use It?
- Under The Hood
- Hello World
- Hello Universe
- Packaging + Deploying
- Next Steps

What is it?

What is Electron?

- "Build **cross-platform** desktop apps with web technologies"
- History
  - Created and maintained by **Github**
  - Formally "Atom Shell"
  - Open Sourced **May 2014**
  - Pretty much single-handedly built by Cheng Zhao

Core Features

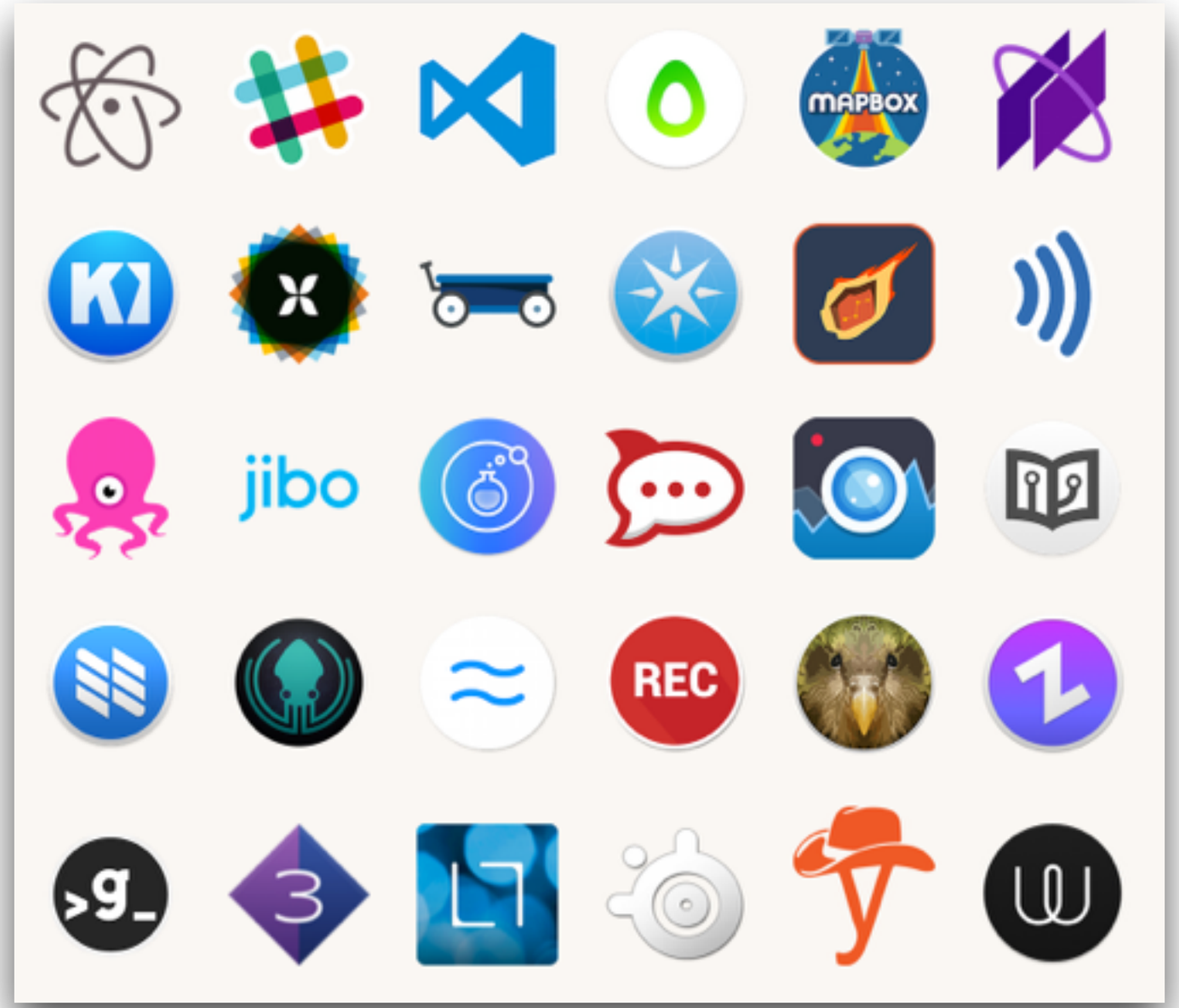
- **Web Tech**
  - “Use HTML, CSS, and JavaScript with Chromium and Node.js to build your app.”
- **Open Source**
  - “Electron is open source; maintained by GitHub and an active community.”
- **Cross Platform**
  - “Electron apps build and run on Mac, Windows, and Linux.”

Makes the hard parts “easy”

- Automatic updates
- Crash reporting
- Windows installers
- Debugging and profiling
- Native menus and notifications

Who Uses It?

Who uses it?



More Specifically

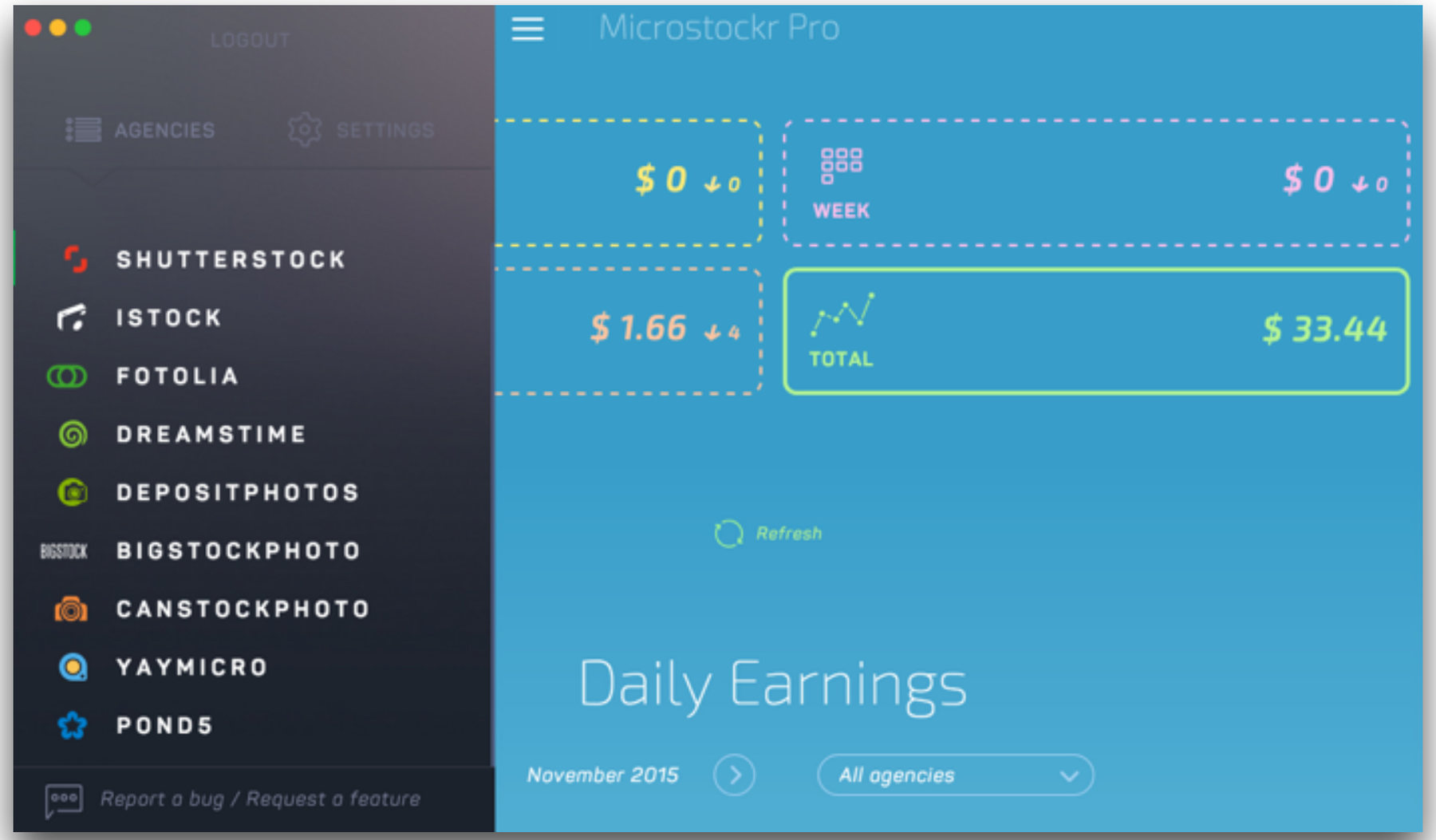
- Slack (messaging)
- WordPress (content-management)
- Mapbox (mapping)
- Avacode (client-side file manipulation)
- Microstockr (3rd party data / visualization)
- B00st (“evernote" for code)

More Specifically

- Slack (messaging)
- WordPress (content-management)
- Mapbox (mapping)
- Avacode (client-side file manipulation)
- Microstockr (3rd party data / visualization)
- B00st (“evernote" for code)

Some Examples

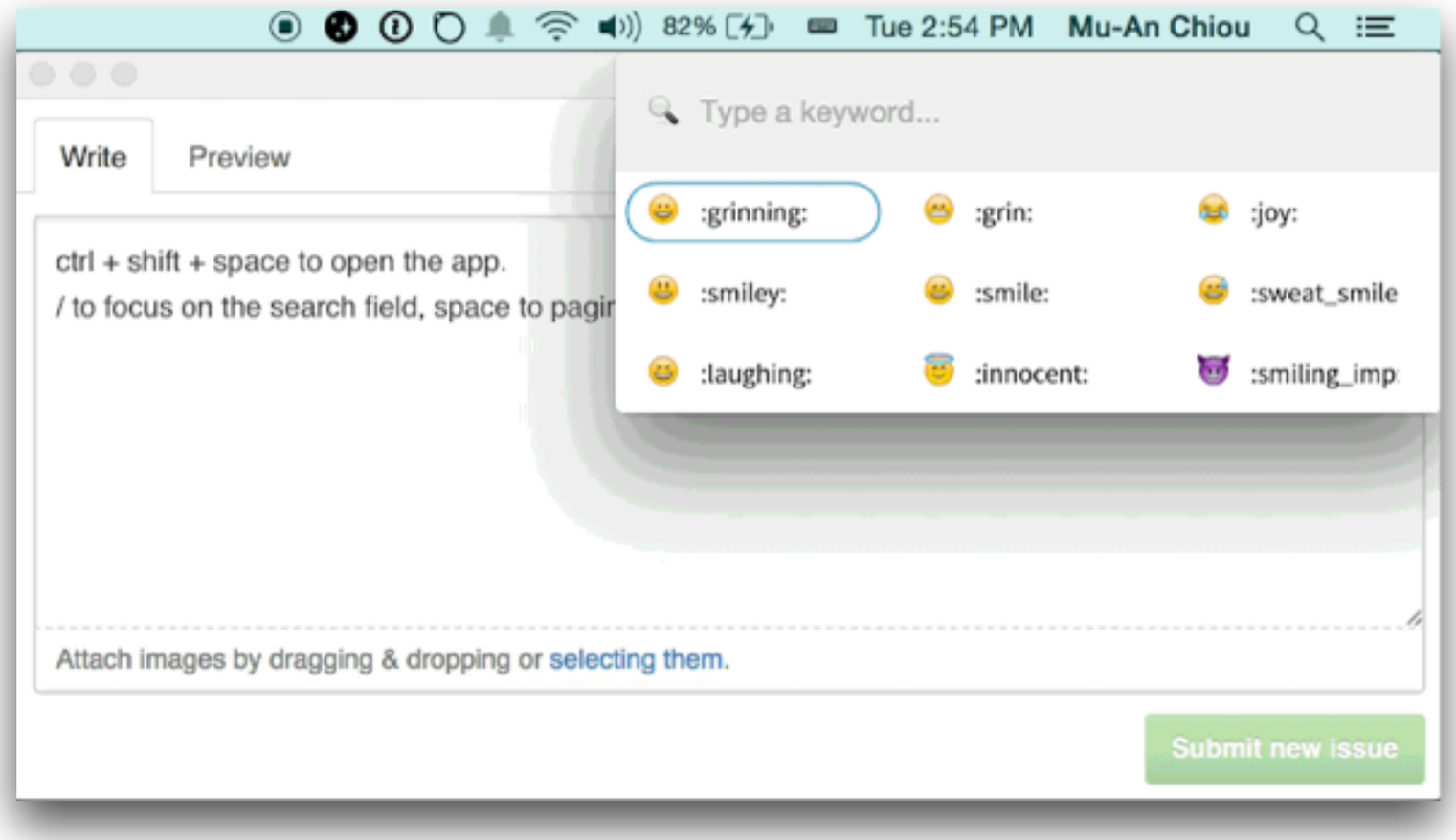
Custom UX


LOGOUT
AGENCIES
SETTINGS
SHUTTERSTOCK
ISTOCK
FOTOLIA
DREAMSTIME
DEPOSITPHOTOS
BIGSTOCKPHOTO
CANSTOCKPHOTO
YAYMICRO
POND5
Report a bug / Request a feature
Microstockr Pro
$ 0 0
WEEK
$ 0 0
$ 1.66 4
TOTAL
$ 33.44
Refresh
Daily Earnings
November 2015
All agencies

ELECTRON

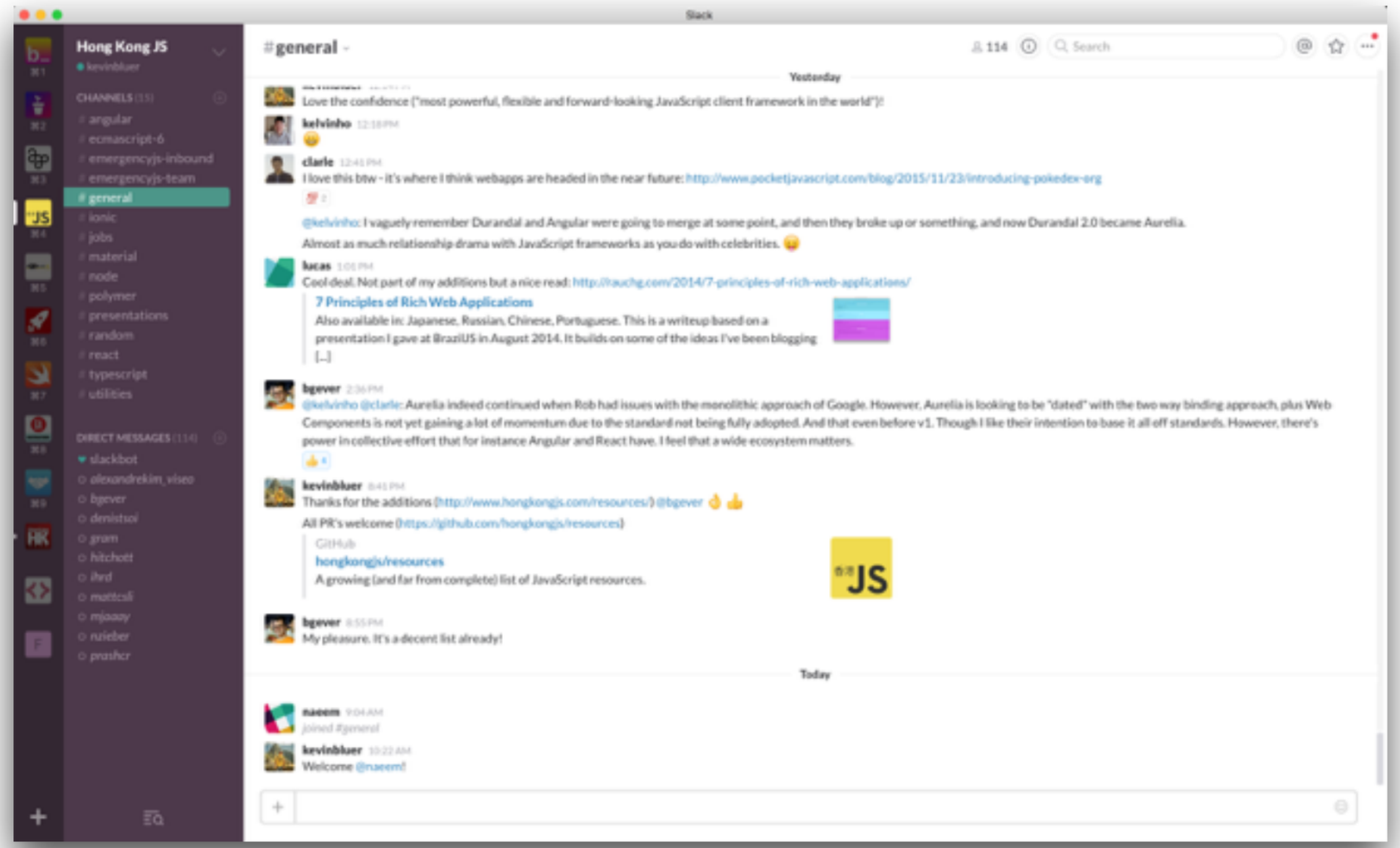
Automatic Updating

Menubar App



- https://github.com/muan/mojibar

And you may have heard of this...



Differences to NW.js

Differences to NW.js*

- Node Webkit (sponsored by Intel) has been around since 2011
- Supports Windows XP, whereas Electron is Win7+
- NW.js is more browser orientated (e.g. the entry point is a HTML file) vs Electron's more Node.js approach
- Slightly poor performance in NW.js (although debatable)

* Also check-out MacGap (http://docs.macgap.com/)

When Should You Use + Potential?

When should you use it?

- **Why shouldn't I just create a web app?**
- Emphasis on one or more of the following:
  - Continual Network Connectivity (Slack)
  - Data Intensiveness (Mapbox, Microstockr)
  - Sophisticated client needs (Avocode, B00st)
- Opens up new set of considerations given it's now "easy" :)
- Similar considerations to that of hybrid mobile

Potential

- Blur the lines between web and desktop (and mobile), with the caveat that there are always going to be situations when it's not appropriate
- Enables you to create desktop experiences when logical (with minimal technical overhead)

Under the Hood

Chromium + Node.js

- “Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine”
- “Chromium is the open-source web browser project from which Google Chrome draws its source code”
- Essentially creates a browser instance + node.js runtime in a desktop container

Source

```
Electron
├─atom - Source code of Electron.
|  ├─ app - System entry code.
|  ├─ browser - The frontend including the main window, UI, and all of the
|  |   main process things. This talks to the renderer to manage web pages.
|  |   ├─ lib - Javascript part of the main process initialization code.
|  |   ├─ ui - Implementation of UI stuff for different platforms.
|  |   |   ├─ cocoa - Cocoa specific source code.
|  |   |   ├─ gtk - GTK+ specific source code.
|  |   |   └─ win - Windows GUI specific source code.
|  |   ├─ default_app - The default page to show when Electron is started
|  |   |   without providing an app.
|  |   ├─ api - The implementation of the main process APIs.
|  |   |   └─ lib - Javascript part of the API implementation.
|  |   ├─ net - Network related code.
|  |   ├─ mac - Mac specific Objective-C source code.
|  |   └─ resources - Icons, platform-dependent files, etc.
|  ├─ renderer - Code that runs in renderer process.
|  |   ├─ lib - Javascript part of renderer initialization code.
|  |   └─ api - The implementation of renderer process APIs.
|  |       └─ lib - Javascript part of the API implementation.
|  └─ common - Code that used by both the main and renderer processes,
|      including some utility functions and code to integrate node's message
|      loop into Chromium's message loop.
|      ├─ lib - Common Javascript initialization code.
|      └─ api - The implementation of common APIs, and foundations of
|          Electron's built-in modules.
|          └─ lib - Javascript part of the API implementation.
├─ chromium_src - Source code that copied from Chromium.
├─ docs - Documentations.
├─ spec - Automatic tests.
├─ atom.gyp - Building rules of Electron.
└─ common.gypi - Compiler specific settings and building rules for other
   components like `node` and `breakpad`.
```

Processes

- Main (essentially node.js)
  - All the core modules and desktop communication (as well as creating the browser window)
  - Also all the native controls / menus / desktop integrations / etc
- Renderer (essentially the web pages themselves)
- *ipcMain and ipcRenderer can be used to send messages between the two respectively*

Hello World

Follow Along?

- Join / Open HKJS Slack (slack.hongkongjs.com)

Hello World Part #1

- npm init
- npm i electron-prebuilt --save-dev
- *Note the electron binary in npm_modules/*
- Update package.json start script:

```
"scripts": {
  "start": "electron ."
},
```

- Create index.js and add the following.
- Run “npm start”
- *You should see a blank browser window*

Hello World Part #2

- Add the following after the mainWindow declaration

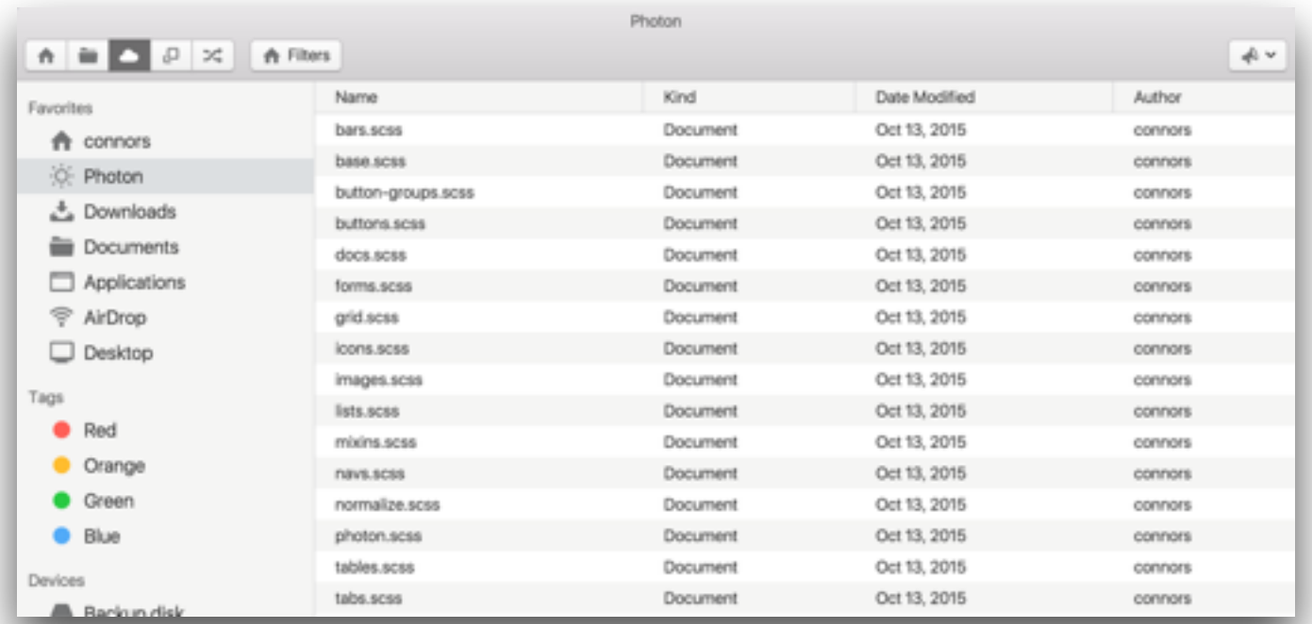
```
mainWindow.loadURL('file://' + __dirname + '/index.html');
```

- Create an index.html file and add “Hello World”
- Run *npm start (you may need to Ctrl + C first)*

Demo #2

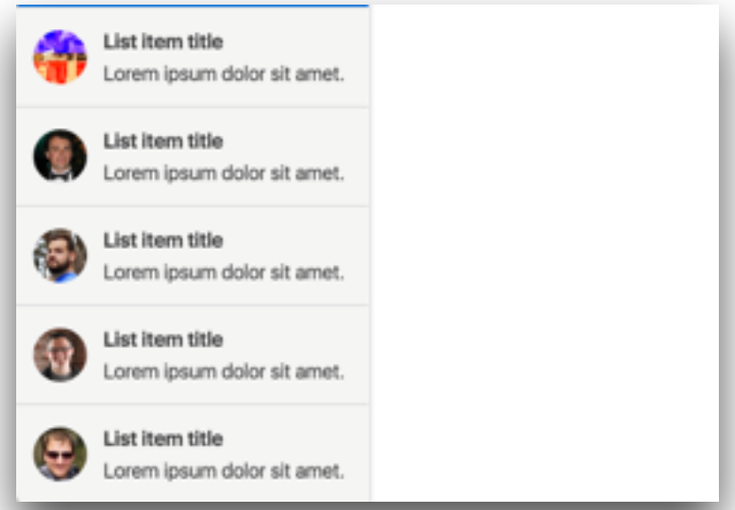
Introducing Photon

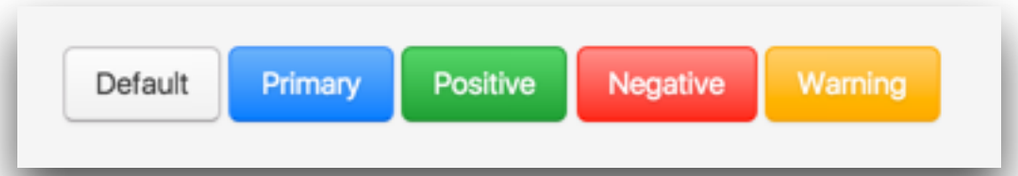
- "The fastest way to build beautiful Electron apps using simple HTML and CSS"
- More details at http://photonkit.com
- "Bootstrap of Desktop Development :)"



More Photon

- Heavily OS X “inspired” :)
- Headers, Lists, Navigation, Buttons, Forms, Icons





Steps

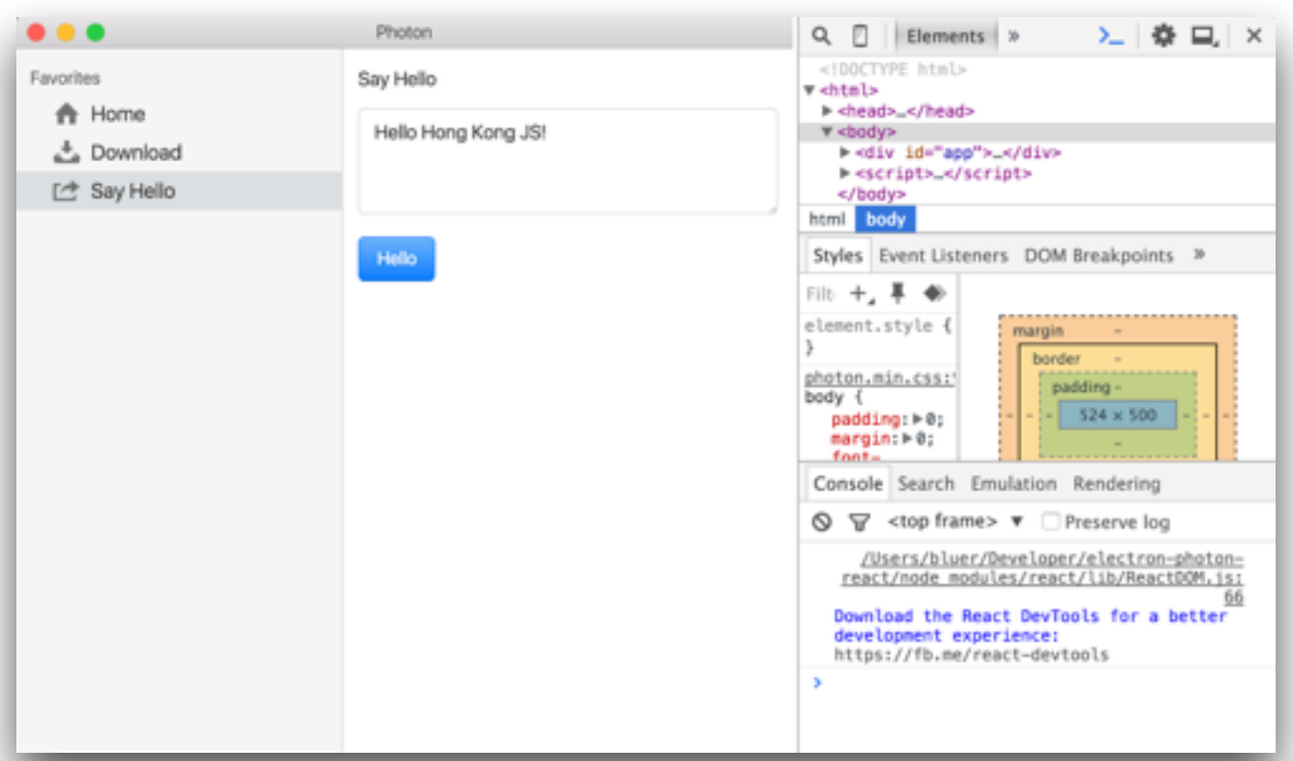
- Download (optional) at the following:
  - https://github.com/hongkongjs/electron-photon-react
  - "*npm start*" to run
- Thing to Note
  - Using ReactJS for the views (although obviously any front-end framework can be used)
  - Additional parameters passed to BrowserWindow
  - Use of the native menu + right-click
  - menu.js for right-click handling across the entire app

Debugging

Debugging (Client)

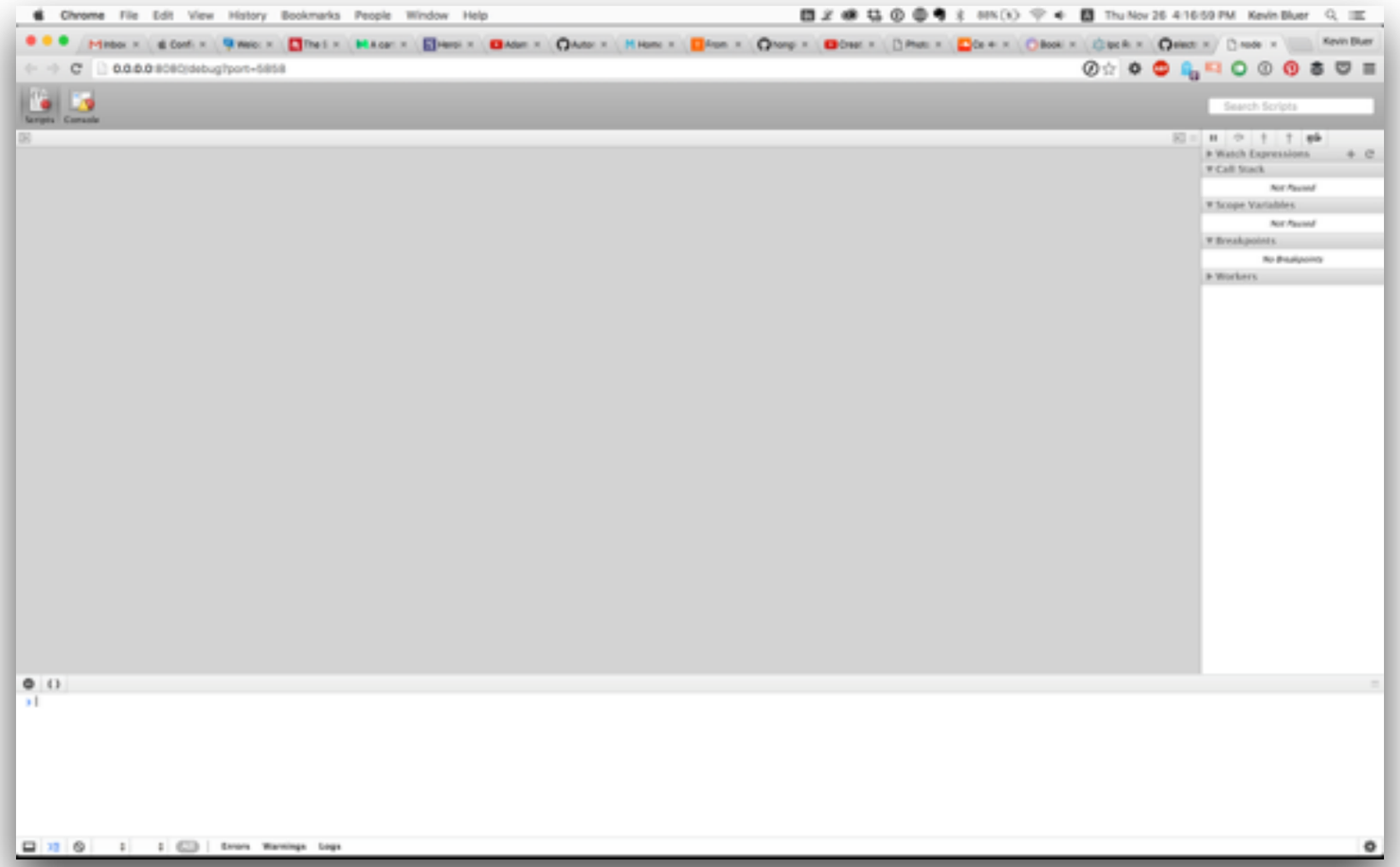
- To show on Chromium's debug tools (or Cmd+Opt+I)

```
33    // Open the DevTools.
34    mainWindow.openDevTools();
35
```



Debugging (Server)

- Via node-inspector (and run the app in debug mode *“electron --debug=5858 app.js”*)



Packaging for Distribution

Packaging for Distribution

- Some useful packages
  - maxogden/electron-packager
  - loopline-systems/electron-builder
- Support for Mac App Store (assuming you want to go that route) and Windows Store (I think)

Next Steps

Next Steps

- Check out the “interesting” open source apps

- Checkout: https://github.com/sindresorhus/awesome-electron
- Keep an eye on: https://github.com/gabrielbull/react-desktop
- Discuss (slack.hongkongjs.com)

Q&A