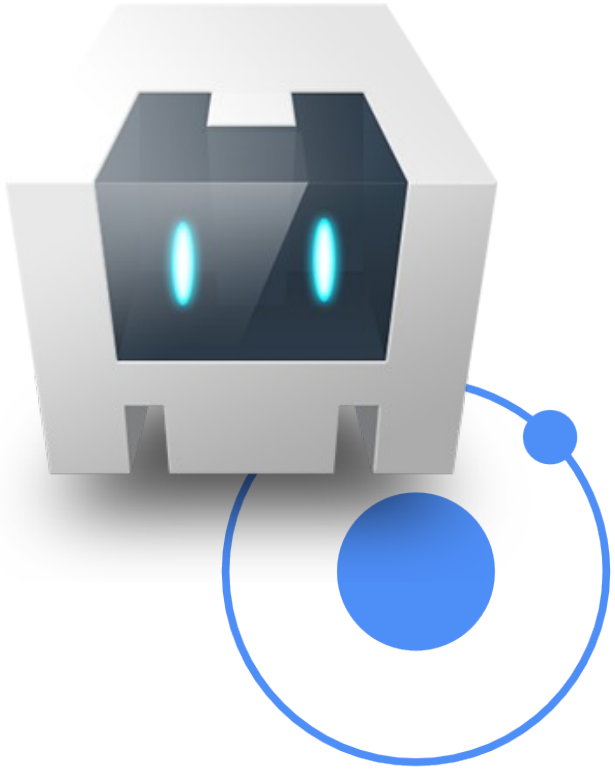


ionic

VS

AppGyver Steroids

Supersonic

About me

Marek Kalnik

@marekkalnik

CTO et Cofounder chez BAM
http://www.bamlab.fr

Dev web depuis 10 ans
freelance + Theodo

ionic

AppGyver Steroids

Backing

1,1 M levés

3,5M de levé des fonds (selon angel.co)

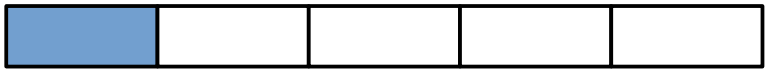
Historique

Créé en 2012

Créé en 2011

décembre 2014 – sortie de Supersonic

Nativité



Type d'outil

wrapper + CLI

wrapper + CLI + service de build

Compatiblité

iOS, Android, Windows Phone, Windows 8.1

iOS, Android

Langage

JavaScript

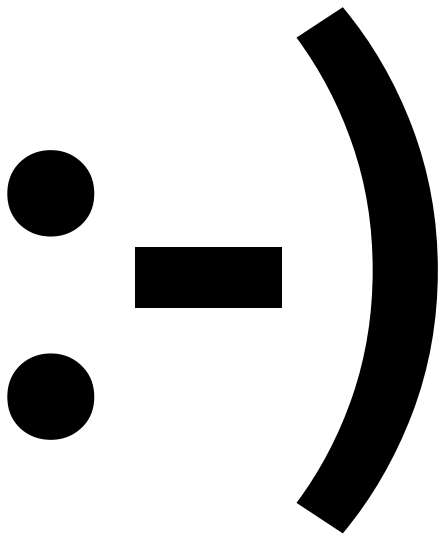
JavaScript

- framework javascript
- basé sur Angular
- éléments d'UI et de navigation

- framework javascript
- utilise les éléments graphiques d'Ionic (CSS)
- éléments d'UI et de navigation
- communication avec la couche native Steroids

- mise en route très rapide
- je peux coder comme je veux
- application en 2 semaines

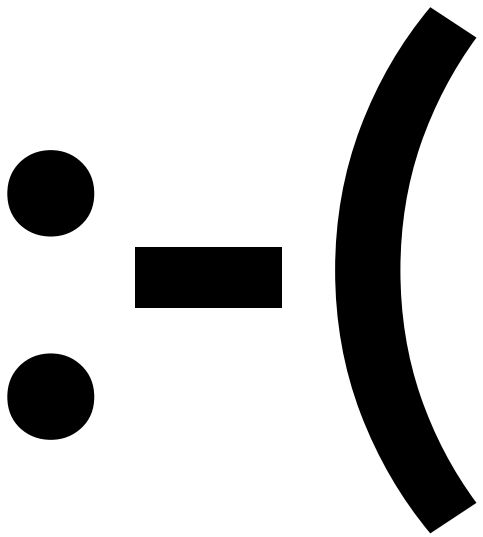
Résultat



super content

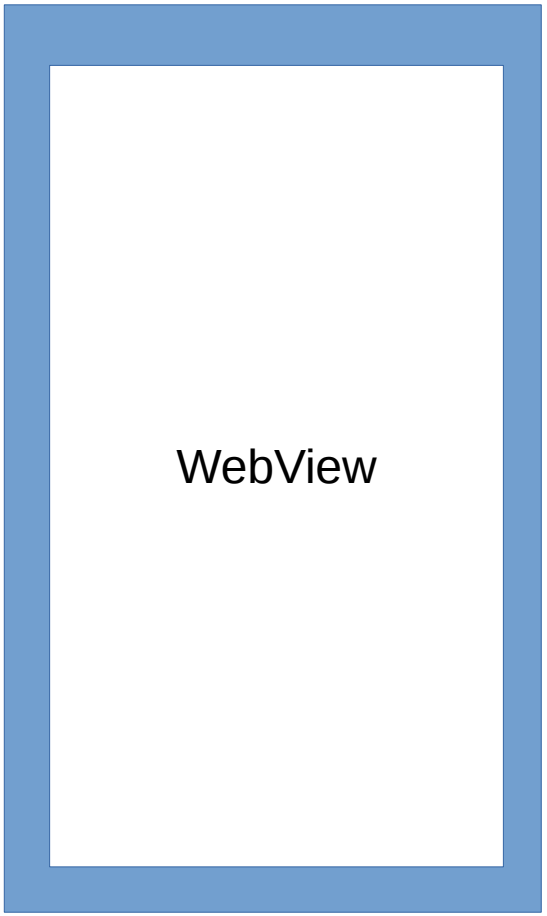
- mais qu'est-ce qui se passe avec les trasitions ?
- et c'est qoui cette fuite de mémoire sous iOS ?

Résultat

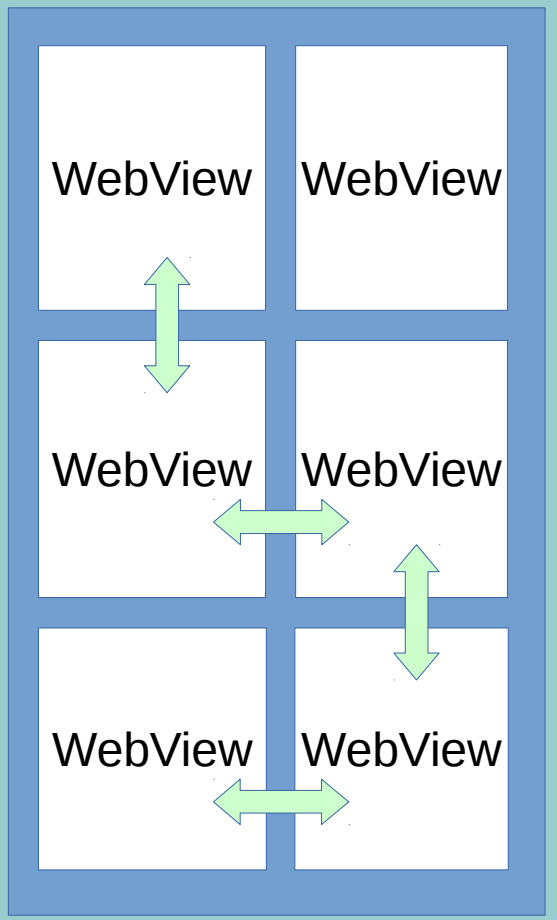


pas content

Les vues et navigation natives ?

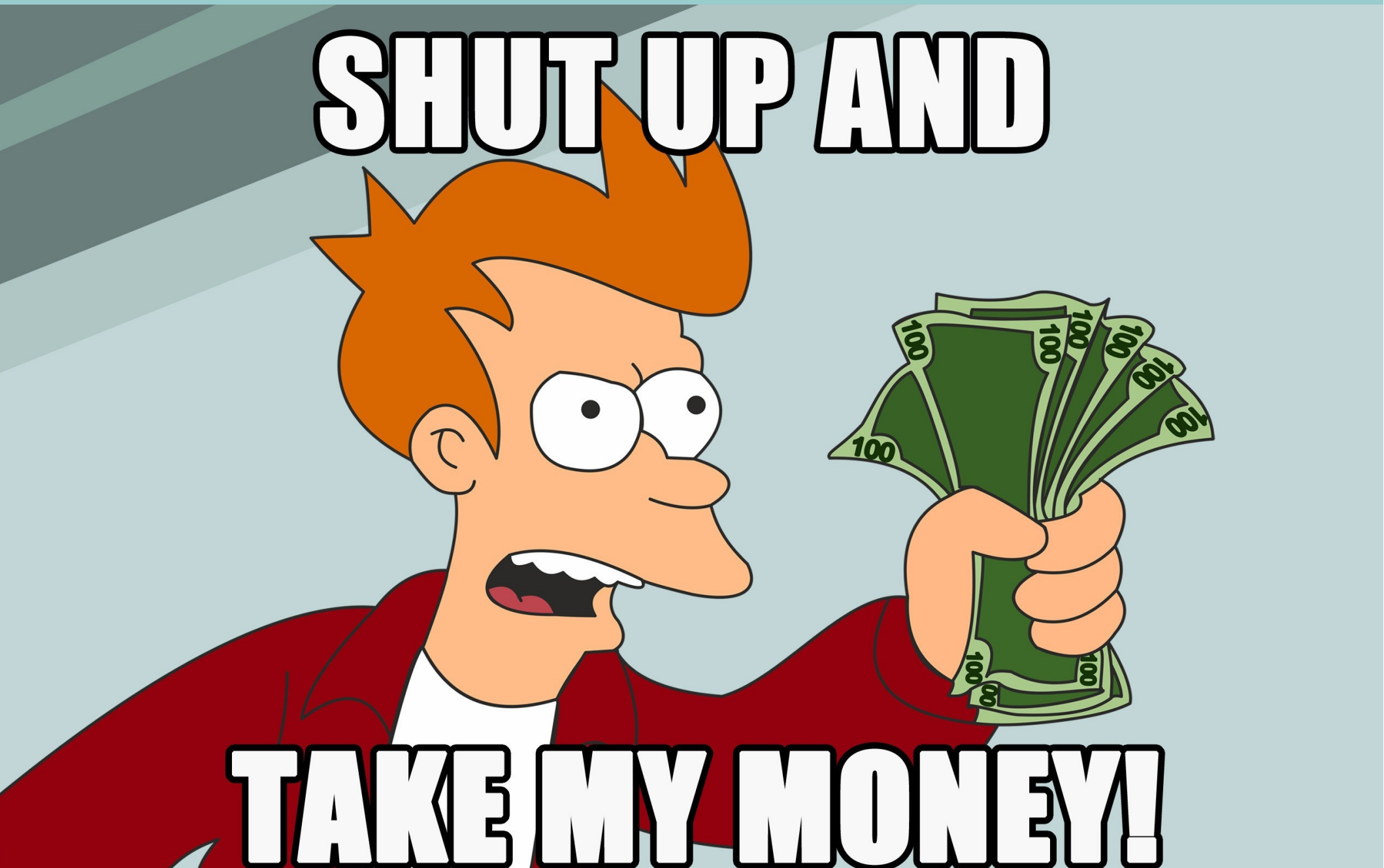


App



App

Les vues et navigation natives ?



Premier test

Importer une application Cordova existante dans Steroids

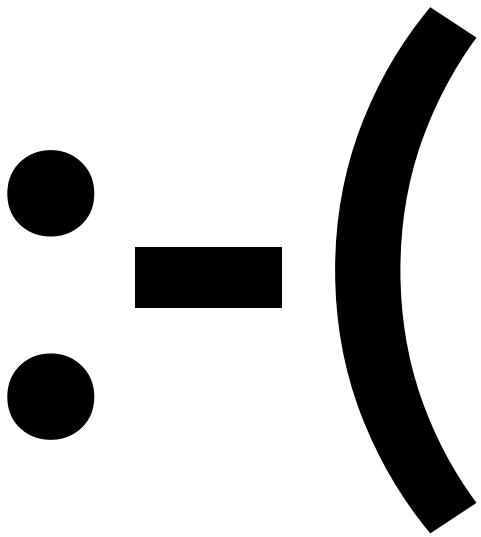
1) créer une application sterois single-page
2) copier toutes les sources dans www/
3) inclure les fichiers javascript de steroids dans l'index.html

Résultat

Aucun avantage de Steroids et tous les inconvenients

1) build uniquement par le cloud Steroids
2) besoin de faire quelques hacks pour que ça marche
   steroids.view.navigationBar.hide();
3) Grunt ;-)
4) si on veut vraiment coder « Steroids way », on est obligé d'utiliser l'application Scaner
5) aucune amélioration visible niveau performance

Résultat



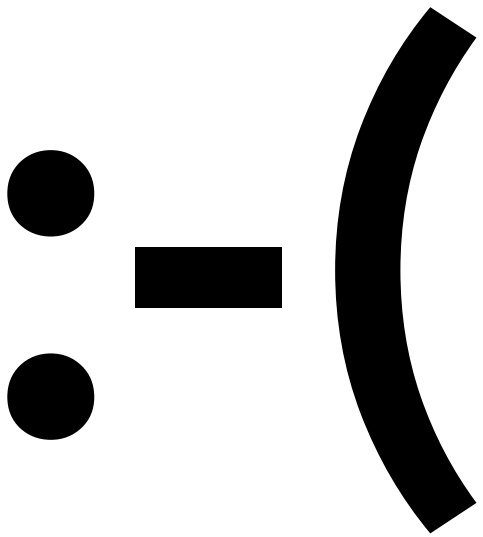
ça ne sert à rien

Deuxième test

Recréer une application Ionic en Steroids Multi Page

1) créer une application Steroids multi-page
2) importer l'application Ionic dans le dossier app
3) créer le fichier structure.coffee avec la structure de l'application
4) réorganiser le projet selon la manière qu'utilise Steroids
5) remplacer tous les <ion-view> par de divs avec ng-controller
6) remplacer les éléments ionic qui ont des équivalents dans steroids
   (ex. <ion-nav-title> par <super-navbar-title>)
7) remplacer toutes les references vers ui-routing par supersonic-navigation

Résultat



c'est dur !

Résultat

Résultat

bref, enfin, au niveau de la navigation...

Avangates

- on organise l'environement et le processus de travail comme on veut

Inconvenients

- gestion de mémoire parfois très complexe, ça serait plus simple de simplement tuer la webview

Avangates

- éléments natifs super fluides
- possibilité de préloader une vue
- la WebView est fermé une fois qu'on ferme la page
- build crosswalk facile à faire
- plus facile de builder sous iOS
- build steroids (si on en a besoin...)
- cloud mobile prêt à utiliser

Inconvenients

- communication entre les vues plus complexe (système de messages, local Storage etc)
- structure rigide et pas super bien documenté
- documentation
- implementation Android pas mature

Reco

- tout le reste

- projets petits et legers ou on veut avoir une navigation super fluide
- autres frameworks que Ionic