Cognitive defusion application

Developing a single page application for android and iOS
Overview

• Cognitive defusion in short
• Bird’s eye view
• Cordova
• Ionic/AngularJS
• Testing and calabash-android
• Live presentation
Cognitive defusion

In very short

Decouple thoughts from reality by putting them into surreal situations.

«Imagine your mind is a parrot sitting on your shoulder and squaking your thought in your ear over and over again»
Bird’s eye view
Model/View/Controller

Abstraction Layer

Cordova
- media
- local-notification
- console
- file
- device
- keyboard
- inapp-browser

Operating System

iOS/Android
App/Cordova

- Ionic
- AngularJS

CSS
JavaScript
HTML
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no, width=device-width">
  <title>Kognitive Defusion</title>
  <!-- for customization purpose -->
  <link rel="stylesheet" href="lib/ionic/css/ionic.css">
  <link rel="stylesheet" href="css/style.css">
  <!-- pouchDB script for data storage and data synchronization -->
  <script src="lib/pouchdb/pouchdb-3.3.0.min.js"></script>
  <!-- uuid generator -->
  <script src="lib/uuid/uuid.js"></script>
  <!-- ionic/angularjs -->
  <script src="lib/ionic/js/ionic.bundle.js"></script>
  <!-- cordova script (this will be a 404 during development) -->
  <script src="cordova.js"></script>
  <!-- your app's js -->
  <script src="js/classes.js"></script>
  <script src="js/app.js"></script>
  <script src="js/controllers.js"></script>
  <script src="js/services.js"></script>
  <script src="js/language.js"></script>
</head>
<body ng-app="starter" ng-controller="AppController">
  <div ng-class="show-text">Filled dynamically</div>
</body>
</html>
App/Cordova

- Ionic/AngularJS

Look and Feel of the app

```html
<ion-view name="overview" title="{{wording[lang].overview.title}}">
  <ion-nav-buttons side="right">
    <a class="button button-outline icon-left ion-arrow-return-left" ng-click="back()"></a>
  </ion-nav-buttons>

  <ion-content>
    <div class="card">
      <div class="text-center item item-text-wrap item-text-center">
        {{wording[lang].overview.quote.text}}<span class="author">{{wording[lang].overview.quote.author}}</span>
      </div>
    </div>
  </ion-content>
</ion-view>
```
App/Cordova

- CSS
- JavaScript
- HTML

Ionic/AngularJS

Look and Feel for the app
Model/View/Controller

```html
<body ng-app="starter" ng-controller="AppController">
  <div id="show" class="hide">
    <div class="show-content"><span id="show-text"></span></div>
  </div>
  <ion-nav-bar align="left" ng-hide="hide_header" name="header">
    <ion-nav-view></ion-nav-view>
  </ion-nav-bar>
</body>
```

```javascript
stateProvider.state("splash", {
  url: "/splash",
  controller: "SplashCtrl",
  templateUrl: "templates/splash.html",
  data: {
    rule: function(User) {
      //Splashscreen should only appear if User wasn't
      //initialized yet ergo at the very beginning.
      if(!User.initialized) {
        return { access: false, alternateRoute: "exit" };
      }
    return { access: true, alternateRoute: undefined};
    }
  }
});
```
```javascript
$stateProvider.state("splash", {  
  url: "/splash" ,  
  controller: "SplashCtrl" ,  
  templateUrl: "templates/splash.html" ,  
  data: {  
    rule: function(User) {  
      // Splash screen should only appear if User wasn't initialized yet ergo at the very beginning.  
      if(User.initialized) {  
        return { access: false, alternateRoute: "exit" } ;  
      }  
    return { access: true, alternateRoute: undefined} ;  
    }  
  }  
});
```
Two ways of changing states

ui-sref

```
<a class="button button-outline icon-left ion-home" ui-sref="overview"></a>
```

programmatically

```
$state.go("exit");
```
State machine of my application
PouchDB

At the start of the application

Everytime the model changes
Design choice for storage

• No live binding between model and permanent storage
  • Model is loaded completely at the beginning from the permanent storage
  • Changes in the model must always be stored in the model and in the permanent storage separately
• Advantages: fast-access to the model, simple manipulation in tests with no need to manipulate permanent storage, promises have to be resolved only once at the beginning of the application (less error prone)
• Disadvantages: each change has to be stored twice (more error prone), with big data it uses large portions of the working memory
Automated Acceptance Testing
Gherkin and step definitions in calabash

Gherkin and Step Definitions in Calabash

Gherkin

Keywords: feature, scenario outline, scenario, background, given, when, then, and, but

Gherkin

Keywords: feature, scenario outline, scenario, background, given, when, then, and, but

Step Definitions

Gherkin step, may consist strong ruby pattern matching, e.g. `(.*?)`
Ruby Automation Code

```ruby
When(/^I touch home button$/)
  go_home

def go_home()
  button = nil
  # This button can only be found if
  # the home button is belongs to class "ion-home".
  # Note: This assumes the button is the only button
  # found with the class "ion-home".
  results = query("cordovaWebView css:'.ion-home'")
  button = nil
  
  if(results.size() == 1)
    button = results[0]
  elsif (results.size() > 1)
    fail("Ambiguous buttons found. Couldn't find out which one is home button. Found #{results.size()} buttons.
  end
  
  not button.nil?

  touch(button)
end
```
Live App Demonstration
Conclusion

• Each framework fulfilled a specific requirement/solved a specific problem
• Tests (cucumber/calabash) help to communicate requirements and are a nice way of documentation
• Successful application which worked on iOS and Android