Lab 9: Hybrid applications

Design and Development of Mobile Applications
E. De Coninck, S. Leroux, P. Simoens
2015-2016

1 Introduction

In the previous assignment you created a basic HTML5 webapplication using Bootstrap. Bootstrap made developing a responsive application straight-forward. New HTML5 and Javascript technologies make it now possible to include functionality such as persistent storage, location awareness and camera access to webapplications. Other functionality such as scheduling notifications is still restricted to native applications. A hybrid application allows us to package a webapplication in a native container. Specific native functionality is made available through Javascript libraries. In this lab session we will use Apache Cordova (`https://cordova.apache.org/`), one of the many frameworks available to create hybrid applications. We will transform the web application from the previous assignment to a hybrid application. This will allow us to schedule notifications when deadlines are about to expire.

2 Install Cordova

- Check if Cordova is installed by executing the "cordova" command.
- Execute the following commands if cordova is not installed:

```
npm install -g cordova
set PATH=%PATH%;C:\Users\nw4\AppData\Roaming\npm
set PATH=%PATH%;C:\android\sdk\tools
set JAVA_HOME="C:\Program Files\Java\jdk1.8.0_60"
```

- Note that you need to set the environment variables every time you start a new console session.

3 A new hybrid application

- We will follow the steps from: `http://cordova.apache.org/docs/en/5.4.0/guide/cli/index.html`).
- To speed up development time you should only create an Android version of your app. If you have access to an IOS or Windows phone you can try to build a version for your platform but note that you may need to install additional SDKs.
- Only the Android 5.0 SDK is installed on the lab PCs. To make sure that Cordova uses this SDK you should use *cordova platform add android@5.0.0* instead of *cordova platform add android*.
- Build your application. Make sure your *PATH* variable contains the "android" executable .

- Run your app in the default Android emulator. Note: You do not need to explicitly build your app before running your app in the emulator, the *emulate* command will build the application for you.

Figure 1: Default Cordova app

4 A hybrid version of your web app

- Add your html, css and Javascript files from the previous assignment to the *www* folder of your cordova app
- Build your application and test if everything is still working.
- Try to install your application on a physical device.

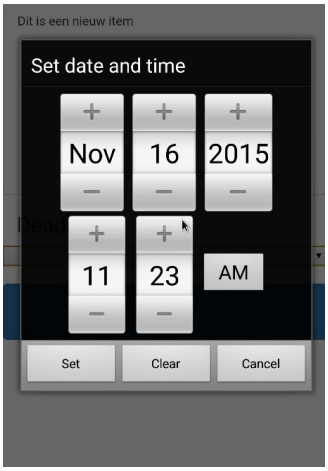

Figure 2: The datetime picker on the emulator.

5 Scheduling native notifications

- Cordova has a powerful plugin system that allows developers to extend the functionality of Cordova.
- We will use the Cordova Local-Notification Plugin `https://github.com/katzer/cordova-plugin-local-noti` to schedule notifications.
- Add this plugin to your Cordova app and implement the functionality to show notifications (with sounds, blinking LEDs, ...)

- Make sure you include the *<script type="text/javascript" src="cordova.js"></script>* tag in your html file. You do not need to add this file explicitly to your www folder. It will be added automatically when you build your app.

6 Debugging Cordova applications

- Webapplications are easy to debug thanks to the powerfull developer tools included in modern browsers such as firefox and chrome.
- Chrome also allows remote debugging. You can debug an application running on your phone through a google Chrome instance on your desktop. The only thing you need is a phone with android version 4.4+ connected to your desktop through usb. (The emulator works as well)
- See this webpage for an overview of the remote debugging functionality: `https://developer.chrome.com/devtools/docs/remote-debugging`
- This remote debugging functionality is not restricted to Cordova apps but also allows you to debug websites running on a physical device (or emulator).

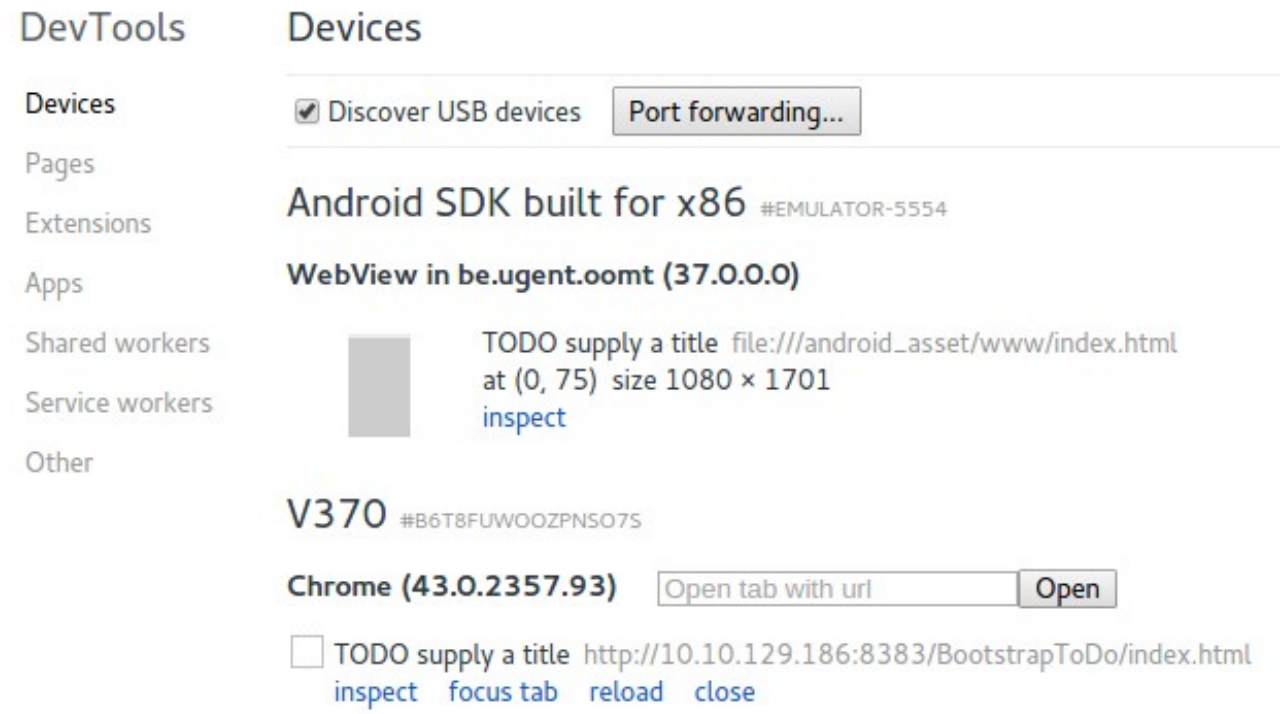

Figure 3: Remote debugging in the Chrome browser.