SCIENCES SORBONNE UNIVERSITÉ

Android Web App

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Why developing a WebApp

If you already have a website

- Easy to develop of Proof-of-Concept for your application

If you want to display third party contents

- It may be useful to connect with webviews

If you want to use javascript

- For using existing framework

If you want a quick display of some information

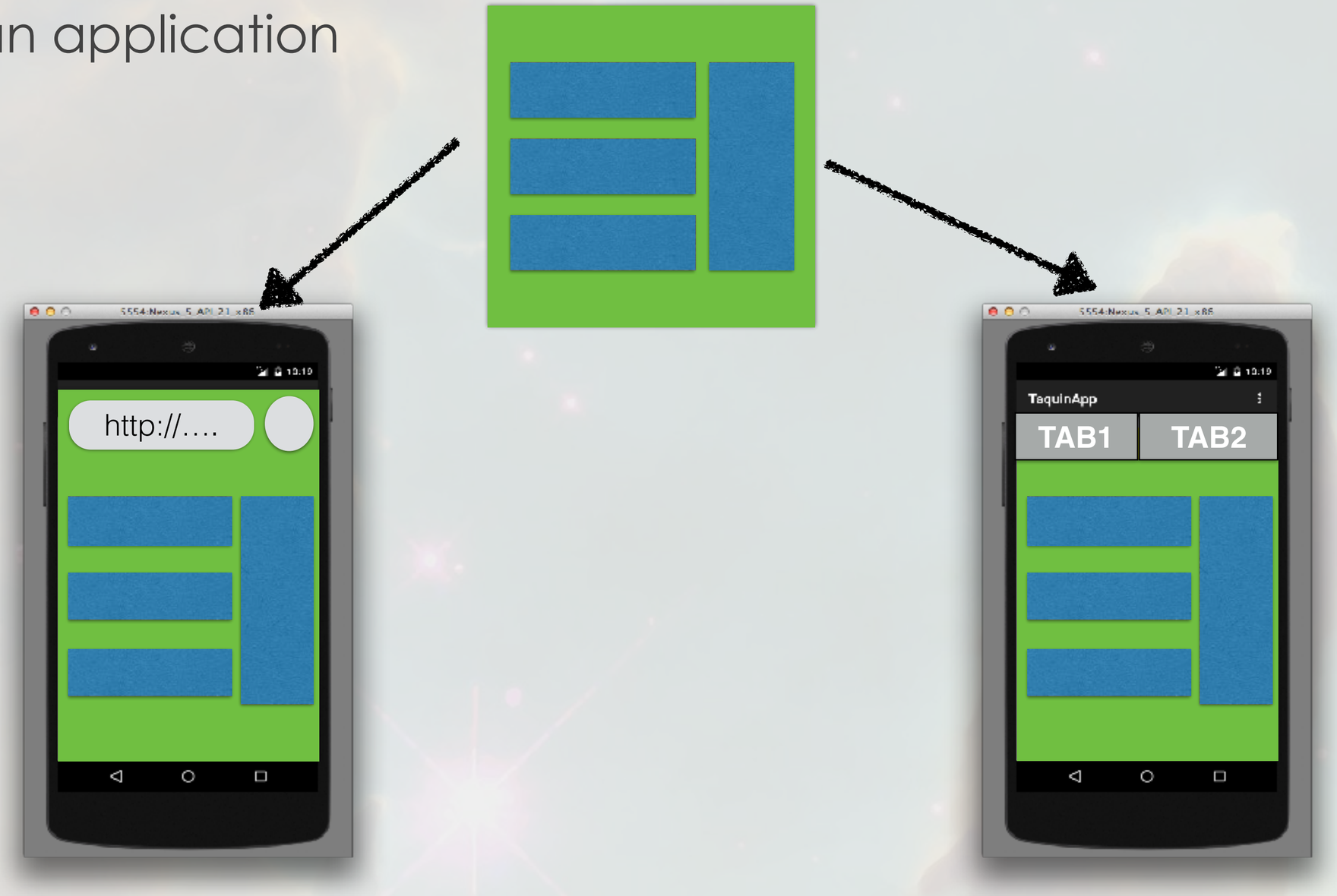
- Displays array for instance

If you want to offer the ability to surf on web while staying in your application

How to use WebViews

A webpage can be display

- on an explorer
- on an application



Require an access to the internet

- Declare it in the AndroidManifest.xml

```
<manifest ... >
 <uses-permission android:name="android.permission.INTERNET" />
 ...
</manifest>
```

Load an URL

```
webView = (WebView) findViewById(R.id.webView);
webView.setWebViewClient(new WebViewClient());
webview.loadUrl("http://google.com");
```

Loading Static HTML

Load a static HTML

```
String summary = "<html><body>You scored <b>192</b> points.</
body></html>";
webView.loadData(summary, "text/html", null);
```

You can also write into a file a load directly this file

```
WebView webView = (WebView)findViewById(R.id.webView);
File file = new File(Environment.getExternalStorageDirectory()
                        + "<FOLDER_PATH_TO_FILE>/<FILE_NAME>");
webView.loadUrl("file:///" + file.getAbsolutePath());
```

OR

```
webView.loadUrl("file:///android_asset/filename.html");
```

Using Javascript (1/2)

Allows the web view to use Javascript

- Warning: javascript may be memory/energy consumer

```
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

Connect Javascript and Java

- Annotate your API with **@JavascriptInterface**

```
public class WebAppInterface {
   Context mContext;
   WebAppInterface(Context c) {mContext = c; }

   @JavascriptInterface
   public void showToast(String toast) {
      Toast.makeText(mContext, toast, Toast.LENGTH_SHORT).show();
   }
}
```

Using Javascript (2/2)

Register the interface to the web view

- First parameter: the Java class
- Second parameter: the name that can be used from HTML webpages

```
webView.addJavascriptInterface(
        new WebAppInterface(this),
       "Android"
);
```

Build an HTML using this interface

```
<input type="button" value="Say hello »
        onClick="showAndroidToast('Hello Android!')" />
<script type="text/javascript">
    function showAndroidToast(toast) {
        Android.showToast(toast);
    }
</script>
```

Restrictions (1/2)

The app must restrict accessibles URL

- Capture outgoing links
  - So that the user cannot escape your application
- Define your own WebViewClient

```
private class MyWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view,
                                            String url) {
      if (Uri.parse(url).getHost().equals("www.example.com")) {
         return false;
      }
      Intent intent =
                new Intent(Intent.ACTION_VIEW,Uri.parse(url));
      startActivity(intent);
      return true;
    }
}
```

Restrictions (2/2)

Pass then this client to the web view

```
myWebView.setWebViewClient(new MyWebViewClient());
```

History management

- canGoBack; goBack; canGoForward; goForward

Capture keydown for implementing back

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK) &&
          myWebView.canGoBack()) {
        myWebView.goBack();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

Summary

Building applications based on web contents

- is easy
- helps to offer better user experience than if we only use an explorer
  - **Use of javascript**
  - **Use of CSS**
- Still compatible with Android mechanisms

How to debug such can application

- Use WebChromeClient
- Use onConsoleMessage
- Logcat will then display your logs
- OR your can define your own API