

# Master Digital Healthcare

## Web Technology Review 2016

---

FSDT1 -  
WS 15/16

Authors: Mario Schartner  
Anita Angerer  
Georg Schreiner  
Jakob Winkler  
Christian Nasel

Editor: Jakob Doppler

# Table of Content

- 1. Introduction ..... 3
- 2. Cordova – a Cross platform tools for Mobile Web Dev: Write your own CameraApp by *Georg Schreiner* ..... 4
- 3. Angular JS – a Data Binding Framework for Web by *Anita Angerer*..... 20
- 4. Run NodeJS on a microcontroller such as the Raspberry PI by Mario Schartner ..... 27
- 5. What is Unity? by *Anita Angerer, Georg Schreiner, Mario Schartner* ..... 46
- 6. Create a most simple Mobile Web App with the cross-platform tool Cordova. by *Christian Nasel*..... 62

## **1. Introduction**

This survey covers some advanced Web Technologies and their prototypical application. Each chapter is a research result of the respective owner and should give others a taste and feel of what latest Web Technologies can do.

Digital Healthcare Masters Course FSDT1 (WS 2015/16)

Jakob Doppler (Editor), 2015

## 2. Cordova – a Cross platform tools for Mobile Web Dev: Write your own CameraApp *by Georg Schreiner*

### 2.1 What is Cordova:

(source: <https://cordova.apache.org>)

Apache Cordova is an open-source mobile development framework. It allows you to use standard web technologies such as HTML5, CSS3, and JavaScript for cross-platform development, avoiding each mobile platforms' native development language. Applications execute within wrappers targeted to each platform, and rely on standards-compliant API bindings to access each device's sensors, data, and network status.

Apache Cordova graduated in October 2012 as a top level project within the Apache Software Foundation (ASF). Through the ASF, future Cordova development will ensure open stewardship of the project. It will always remain free and open source under the Apache License, Version 2.0. Visit [cordova.apache.org](https://cordova.apache.org) for more information.

Use Apache Cordova if you are:

- a mobile developer and want to extend an application across more than one platform, without having to re-implement it with each platform's language and tool set.
- a web developer and want to deploy a web app that's packaged for distribution in various app store portals.
- a mobile developer interested in mixing native application components with a *WebView* (special browser window) that can access device-level APIs, or if you want to develop a plugin interface between native and WebView components.

For further information visit <https://cordova.apache.org>.

### 2.2 How to install Cordova on Windows

(from: <https://evotthings.com/doc/build/cordova-install-windows.html>)

#### Follow these steps to install Cordova:

1. **Install Node.js:** Cordova runs on the Node.js platform, which needs to be installed as the first step. Download installer from: <http://nodejs.org>
2. Go ahead and run the downloaded installation file. It is recommended to use the default settings. Node.js needs to be added to the PATH environment variable, which is done by default.

3. To test the installation, open a command window (make sure you open a new command window to get the updated path settings made by the Node.js installation), and type:

```
node --version
```

If the version number is displayed, Node.js is installed and working!

**Install Git.** Git is a version control system, which is used by Cordova behind-the-scenes.

Download and install from: <https://git-for-windows.github.io/>. Default settings are recommended.

Add PATH to the console:

```
set PATH C:\Users\<InsertYourUsername>\AppData\Local\Programs\Git\bin;%PATH%
```



1. **Install Cordova.** Cordova is installed using the Node Package Manager (npm). Type the following in the command window to install:

```
npm install -g cordova
```

2. Test the Cordova install by typing:

```
cordova --version
```

If you see the version number, you have successfully installed Apache Cordova!

## Install Java

The Android SDK needs the Java Development Kit (JDK) to be installed (minimum version 1.6). Note that the Java Runtime Environment (JRE) is not sufficient, you will need the JDK. To check if you have the JDK installed already, type this on the command line:

```
javac -version
```

If you do not have the JDK installed, proceed as follows:

1. Download the Java SE JDK (SE = Standard Edition) from Oracle:[www.oracle.com/technetwork/java/javase/downloads/](http://www.oracle.com/technetwork/java/javase/downloads/). Click the Java SE Download to see the list fo download. Get the "Windows x86" download if you have 32-bit Windows, and "Windows x64" if you have 64-bit Windows. If you do not know which version you have, find out using the Control Panel by selecting "System and Security" and then "System", where you will find the "System type" saying if your Windows version is 32-bit or 64-bit.
2. Go along and run the downloaded installer file. Using the default selections should be fine, but take a note of the directory in which you install the JDK. You will need to add this to the PATH in a later step below.
3. Next, update your path to include the JDK. Open the **Control Panel**, click **System and Security**, click **System**, click **Change settings**, which will open the System Properties window. Select the **Advanced** tab, then click the **Environment Variables** button. *(Note Georg: In Windows 10 you have to open the control panel, click on System, then on Advanced system Settings on the left side and then on environment variables)*
4. In the list **User variables** select **PATH** and click the **Edit** button. (If there is no PATH entry in the list, click the New button to create one.)
5. At the end of the field **Variable value**, add a semicolon followed by the path to the bin directory of the JDK install. Here is an example (note that this must be the actual path used for the install on your machine):

```
;C:\Program Files\Java\jdk1.8.0_11\bin
```

An easy way to do this is to prepare the path to add in a text editor, then paste it at the end of the input field. When done click the **OK** button.

6. Next add the **JAVA\_HOME** variable if it is not present (and if it is in the list, you may need to update its value using the Edit button). Click the **New** button. In the field **Variable name** type:

```
JAVA_HOME
```

In the field **Variable value** enter the path to the directory where the JDK is installed, without the semicolon and the /bin subdirectory, for example:

```
C:\Program Files\Java\jdk1.8.0_11
```

Click the **OK** button.

7. Click the **OK** button again to close the Environment Variables window.
8. Now you are ready to test the install. Close any open command windows, and open a new command window and type:

```
javac -version
```

If you see a version number you are done with the JDK install!

## Install Ant

[Apache Ant](#) is a build system for Java, which is used by Cordova and the Android SDK. To install Ant, follow these steps:

1. Download Ant from here: [ant.apache.org/bindownload.cgi](http://ant.apache.org/bindownload.cgi). Get the zip download available at the page. Click the zip-file link for the most recent release, e.g. **apache-ant-1.9.4-bin.zip**, and save the file to your machine.
2. Unpack the zip file to the directory on your machine where you want Ant to be installed. You can pick any directory for the install. In this guide we use this as an example:

```
C:\Users\miki\ant
```

Note that the files in the ant package should go directly into this directory. Make a note of the directory as you will need to add it to the PATH.

3. To add Ant to the PATH, open the **Control Panel**, click **System and Security**, click **System**, click **Change settings**, click the **Advanced** tab, then click the **Environment Variables** button.
4. In the list **User variables** select **PATH** and click the **Edit** button.
5. At the end of the field **Variable value**, add a semicolon followed by the path to the bin directory of the Ant install. Here is an example:

```
;C:\Users\miki\ant\bin
```

Click the **OK** button.

6. Next add the **ANT\_HOME** variable. Click the **New** button. In the field **Variable name** type:

```
ANT_HOME
```

In the field **Variable value** enter the path to the directory where Ant is installed, without the semicolon and the /bin subdirectory, for example:

```
C:\Users\miki\ant
```

Click the **OK** button.

7. Click the **OK** button again to close the Environment Variables window.
8. Now test the install. Close any open command windows, and open a new command window and type:

```
ant -version
```

If you see a version number you have installed Ant successfully!

NOTE Georg: if this way doesn't work, you should try the following commands in the console:

```
set JAVA_HOME=C:\java\jdk\1.6.0_xx
set ANT_HOME=C:\ant
set ANT_ARGS=-lib C:\ant_xtralibs;C:\ant_testlibs
set PATH=%PATH%;%JAVA_HOME%\bin;%ANT_HOME%\bin;C:\cvsnt
```

After you did this, **reinstall** node.js.

## Install the Android SDK Tools

The SDK Tools for Android are used by Cordova to build Android apps. Follow these steps to install the SDK Tools:

1. Go to the page [developer.android.com/sdk](http://developer.android.com/sdk) scroll down the page and click "VIEW ALL DOWNLOADS AND SIZES". Under "SDK Tools Only", click the windows installer exe file and download it (this file is named e.g. **installer\_r23.0.2-windows.exe**).
2. When downloaded, run the installer. You should do fine to use the default settings used by the installer, but make a note of the directory in which the SDK is installed, as you will have to add this to the PATH in the next step.



3. To add the SDK Tools to the PATH, open the **Control Panel**, click **System and Security**, click **System**, click **Change settings**, click the **Advanced** tab, then click the **Environment Variables** button.
4. In the list **User variables** select **PATH** and click the **Edit** button.
5. At the end of the field **Variable value**, add a semicolon followed by the path to the **tools** and **platform-tools** directories of the Android SDK install. Here is an example of what to add (note that there are two paths in one line, separated by a semicolon):

```
;C:\Users\miki\AppData\Local\Android\android-  
sdk\tools;C:\Users\miki\AppData\Local\Android\android-sdk\platform-tools
```

You can prepare the path in a text editor, copy it and paste at the end of the input field. Click the **OK** button when done.

6. Click the **OK** button again to close the Environment Variables window.
7. Now test the install. Close any open command windows, open a new command window and type:

```
adb version
```

This should display the version of the Android Debug Bridge.

8. As the final step, you need to get the specific Android SDK version used by Cordova. This is done by running the **Android SDK Manager** by typing the command:

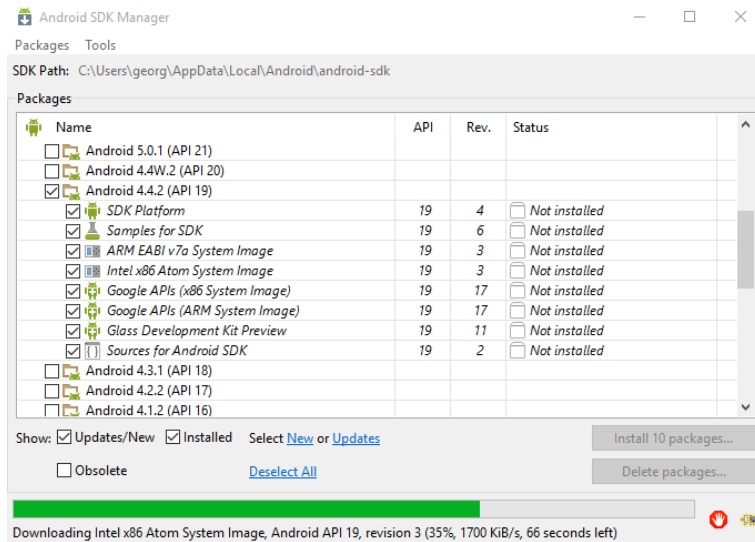
```
android
```

This launches a window where you can select to install specific Android SDKs.

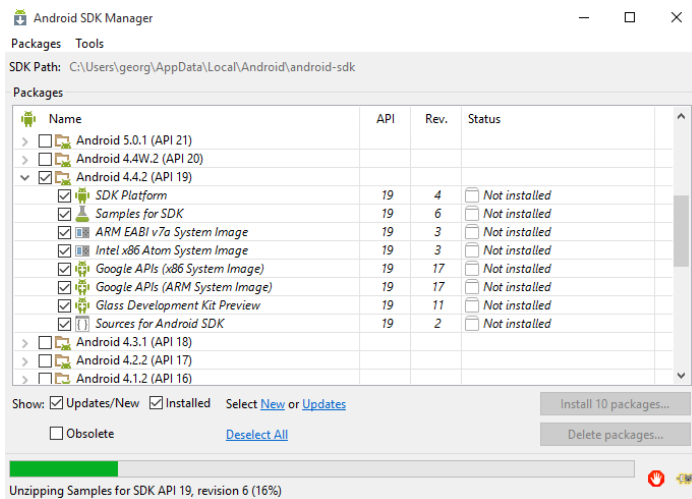
Note: if the command in the console does not work, you can use the following method: Under Windows 10 or 8 hit the windows-key and type “*sdk manager*” and click on it.



In both ways the following window should appear:



- First time you launch the Android SDK Manager there will be preset selections. It is recommended to leave these untouched. Also select the entry "Android 4.4.2 (API 19)". This is the version used by the current Cordova 3.5 version. **Note that the Android SDK required by Cordova will change in the future, as new versions of Cordova and Android are released. When this happens, open the Android SDK Manager again, and install the required API version(s).**



(as mentioned above, search Android 4.4.2 (API 19) in the list and select it. Now click on the button “Install 10 packages” and wait till the procedure is over).

## Install Android Studio

<http://developer.android.com/sdk/index.html>

We will use Android Studio for programming the App.

## Install Genymotion

<https://www.genymotion.com/#!/download>

Click on the Button “Upgrade my plan now” → You should be able to download the free version of genymotion.

## If You Get Stuck

If you get stuck, consult the documentation at the respective web sites for Cordova, Java, Ant, and Android. The Cordova documentation specific for Android is found here: [cordova.apache.org/docs/en/3.6.0/guide\\_platforms\\_android\\_index.md.html](http://cordova.apache.org/docs/en/3.6.0/guide_platforms_android_index.md.html). (For all other questions, error messages, warnings, etc. ask google for help)

One thing to do is to inspect all the environment variables. You can do this from a command window (note that you have to open a new command window after updating environment variables for updated values to be available). This displays the system PATH:

```
echo %PATH%
```

Here is how to inspect the values of JAVA\_HOME and ANT\_HOME:

```
echo %JAVA_HOME%  
  
echo %ANT_HOME%
```

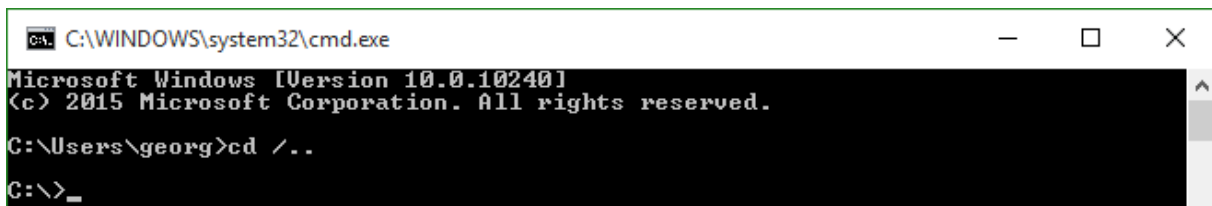
## 2.3 Creating a Cordova Project (IOS and Android)

In this case, we will write a Camara-App.

Here is how to create a new Cordova project (procedure is the same for all platforms):

1. Open a command-line window
2. Go to a folder where you want the project subfolder to be created.

Note Georg: with the command "cd /.." you can change in the main directory of your Drive (C:\). In step 3, the new project folder will be stored in the main-directory (C:\)



```
C:\WINDOWS\system32\cmd.exe  
Microsoft Windows [Version 10.0.10240]  
(c) 2015 Microsoft Corporation. All rights reserved.  
C:\Users\georg>cd /..  
C:\>_
```

3. Use the **cordova create** command to create a project. This command has the following form:

```
cordova create project-folder app-identifier app-name
```

Here is an example that will create a folder called "CamaraApp" with project template files:

```
cordova create CameraApp com.revivalx.cordova.camera Camera-App
```



```
ca. Command Prompt
C:\>cordova create CameraApp com.revivalx.cordova.camera Camera-App
Creating a new cordova project with name "Camera-App" and id "com.revivalx.cordova.camera" at location "C:\CameraApp"
C:\>
```

Note that creating the project may take a while.

4. To issue further Cordova commands, move into the project folder, e.g. by typing:

```
cd CameraApp
```



```
ca. Command Prompt
C:\>cd CameraApp
C:\CameraApp>
```

## Install cordova camera plugin

```
cordova plugin add https://github.com/apache/cordova-plugin-camera
```

**NOTE:** In my case I had to execute this command in GIT (We've already installed it above). You have to run the "Git Cmd.exe". I will use GIT from now on instead of the windows console.

## Building for Android

Here is how you build your new Cordova project for Android:

1. Make sure you are in the project folder in a terminal window
2. Add the Android platform to the project. Note that this only needs to be done once for each project. (Try to use the newest android version → look for it in google. Or cordova.apache.org)

```
cordova platform add android@5.0.0
```

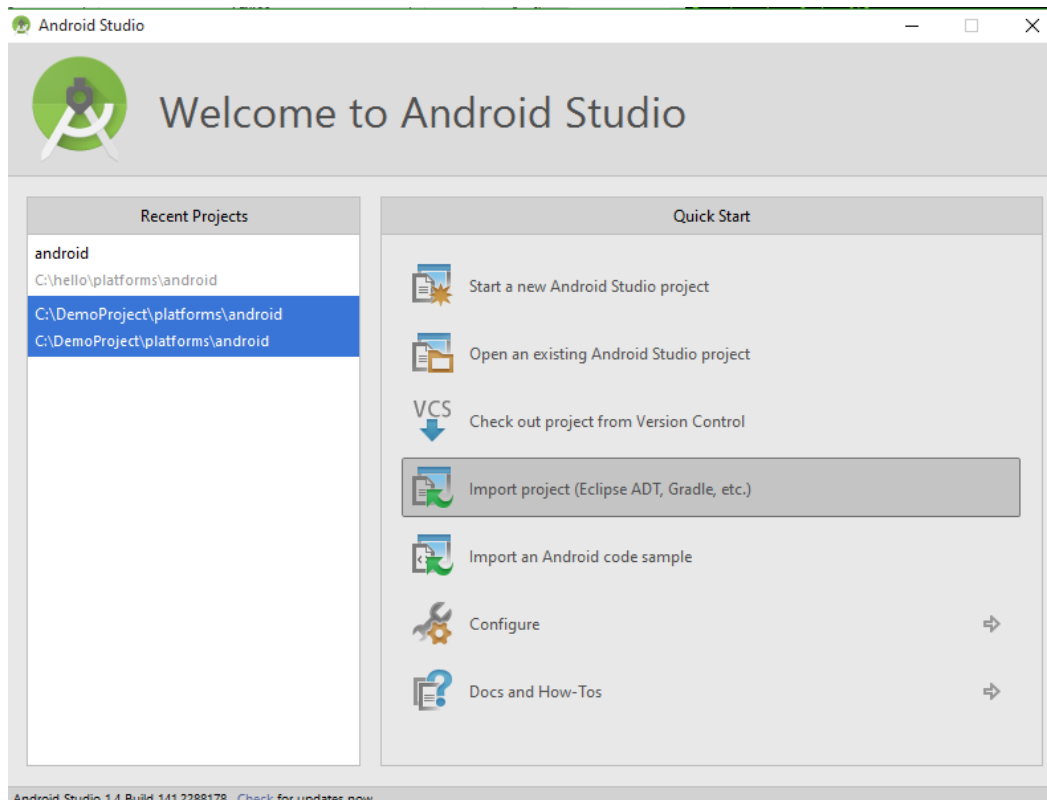
```
C:\CameraApp>cordova platform add android@5.0.0
Adding android project...
Creating Cordova project for the Android platform:
  Path: platforms\android
  Package: com.revivalx.cordova.camera
  Name: Camera_App
  Activity: MainActivity
  Android target: android-23
Android project created with cordova-android@5.0.0
Installing "cordova-plugin-camera" for android
C:\CameraApp>
```

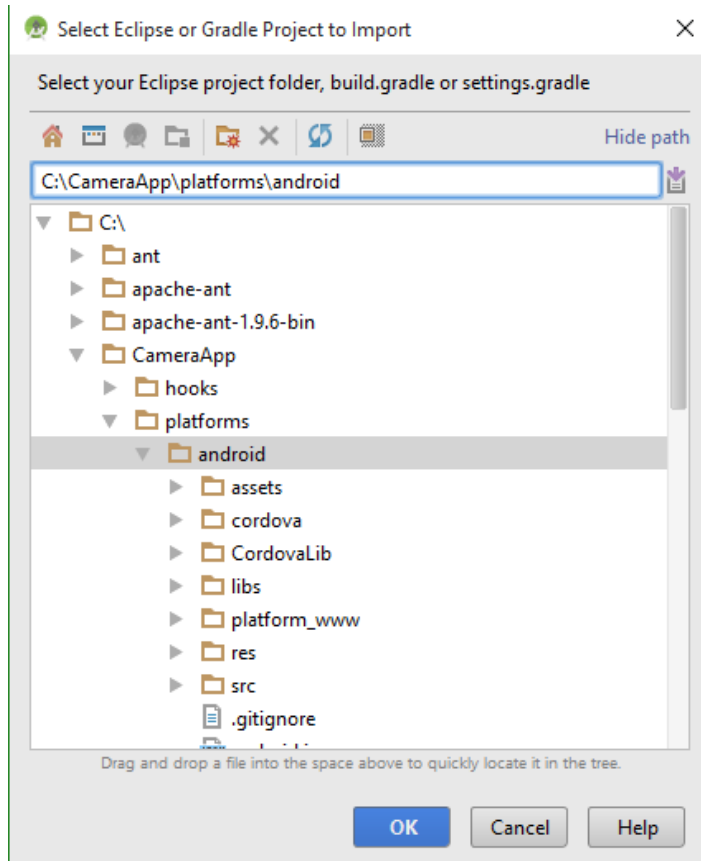
**Note:** if there an error message appears, you may have to install other sdk packages within the sdk manager. Therefore you have to visit <https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html> to look for the newest requirements. If you have updated the SDK-Manager I suggest a restart, before you reenter the “cordova platform add android” command into the console.

With this command you will download and install the cordova plugin from the specific website.

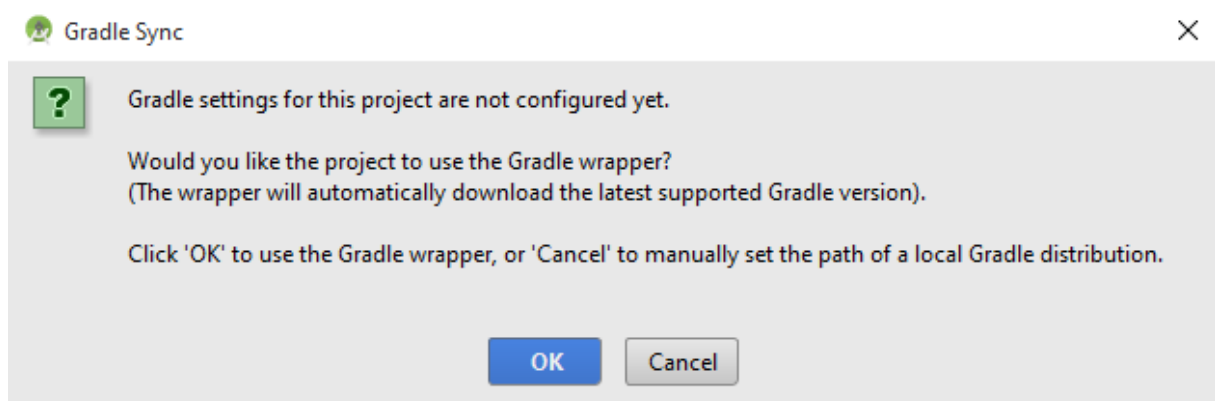
## 2.4 Implementing the Camera-App with Android Studio

1. Open Android Studio
2. Import your cordova project into Android Studio.





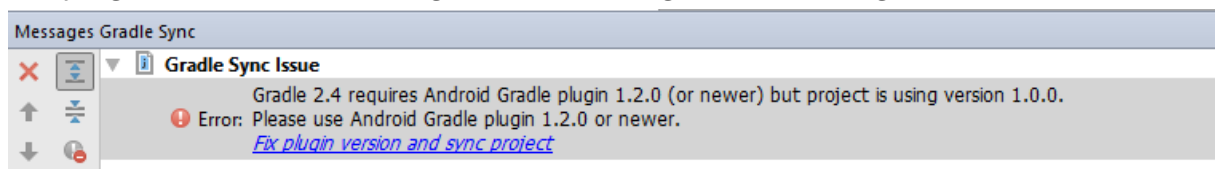
Navigate to C:\CameraApp\platforms\android and click the ok button.



Hit OK (and pray that everything is working).

### 3. Error Handling:

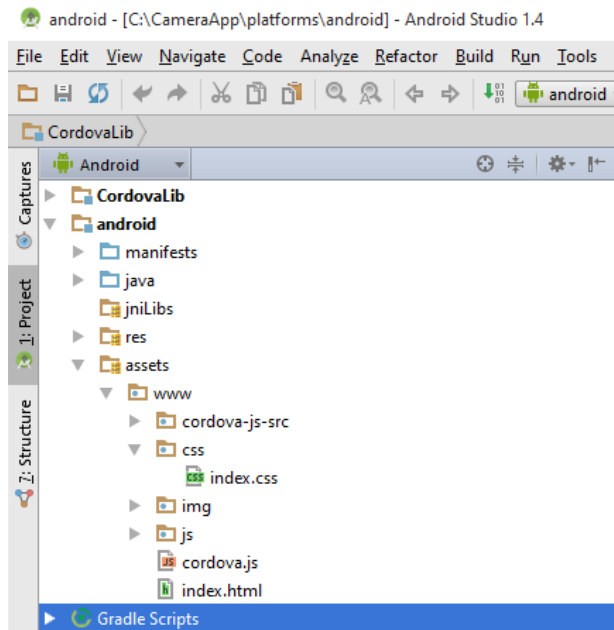
Praying was useless☺ I've got the following error message:



Click on the "Fix plugin version and sync project" button. Everything is working correct at the moment. Yeah, great.

### 4. Delete CSS and Image folders

Go to android→assets→www and delete the 2 folders (right click→delete), because they will not be needed in this tutorial.



**5. Open index.html in the same folder and replace the text with the following:**

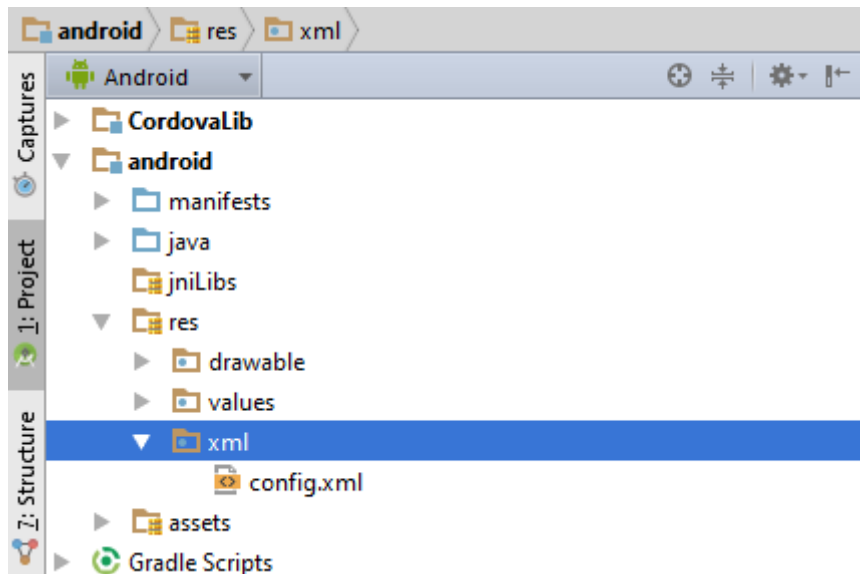
```
<!DOCTYPE html>

<html>
<head>
  <meta charset="utf-8">
  <meta content="telephone=no" name="format-detection">
  <!-- WARNING: for iOS 7, remove the width=device-width and
  height=device-height attributes. See
  https://issues.apache.org/jira/browse/CB-4323 -->
  <meta content=
    "user-scalable=no, initial-scale=1, maximum-
    scale=1, minimum-scale=1, width=device-width, height=device-height,
    target-densitydpi=device-dpi"
    name="viewport">
  <script src="cordova.js" type="text/javascript"></script>
  <script src="js/index.js" type="text/javascript"></script>
  <title>Camera Cordova Plugin</title>
</head>

<body>
<button onclick="capturePhoto();">Capture Photo</button><br>
<button onclick="getPhoto(pictureSource.PHOTO_LIBRARY);">From Photo
  Library</button><br>
<img id="image" src="" style="display:none;width:100%;">
</body>
</html>
```

**6. Open up your config.xml file (android→res→xml) and replace it with the following code:**



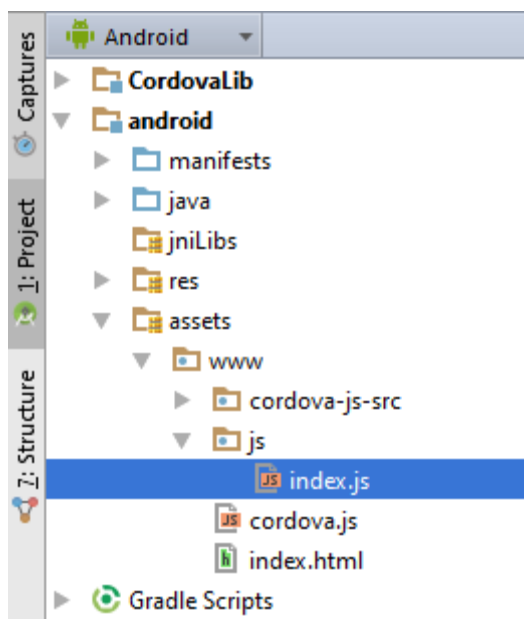


```

<?xml version='1.0' encoding='utf-8'?>
<widget id="com.revivalx.cordova.camera" version="0.0.1"
xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
  <feature name="Camera">
    <param name="android-package"
value="org.apache.cordova.camera.CameraLauncher" />
  </feature>
  <name>CameraCordova</name>
  <description>
    This source code provides example for taking pictures and for
    choosing images from the system's image library.
  </description>
  <content src="index.html" />
  <access origin="*" />
</widget>

```

**7. Open index.js file in js folder and replace the code with the following**



## Master Digital Healthcare - Web Technology Review 2016

```
var pictureSource; // picture source
var destinationType; // sets the format of returned value
// Wait for device API libraries to load
//
document.addEventListener("deviceready", onDeviceReady, false);
// device APIs are available
//

function onDeviceReady() {
pictureSource = navigator.camera.PictureSourceType;
destinationType = navigator.camera.DestinationType;
}
// Called when a photo is successfully retrieved
//

function onPhotoDataSuccess(imageURI) {
// Uncomment to view the base64-encoded image data
console.log(imageURI);
// Get image handle
//
var cameraImage = document.getElementById('image');
// Unhide image elements
//
cameraImage.style.display = 'block';
// Show the captured photo
// The inline CSS rules are used to resize the image
//
cameraImage.src = imageURI;
}
// Called when a photo is successfully retrieved
//

function onPhotoURISuccess(imageURI) {
// Uncomment to view the image file URI
console.log(imageURI);
// Get image handle
//
var galleryImage = document.getElementById('image');
// Unhide image elements
//
galleryImage.style.display = 'block';
// Show the captured photo
// The inline CSS rules are used to resize the image
//
galleryImage.src = imageURI;
}
// A button will call this function
//

function capturePhoto() {
// Take picture using device camera and retrieve image as base64-encoded string
navigator.camera.getPicture(onPhotoDataSuccess, onFail, {
quality: 30,
targetWidth: 600,
targetHeight: 600,
destinationType: destinationType.FILE_URI,
saveToPhotoAlbum: true
});
}
// A button will call this function
//
```

```
function getPhoto(source) {  
  // Retrieve image file location from specified source  
  navigator.camera.getPicture(onPhotoURISuccess, onFail, {  
    quality: 30,  
    targetWidth: 600,  
    targetHeight: 600,  
    destinationType: destinationType.FILE_URI,  
    sourceType: source  
  });  
}  
// Called if something bad happens.  
//  
function onFail(message) {  
  //alert('Failed because: ' + message);  
}
```

### 8. Save the Project

Hint: You can refer full documentation on this

site, [http://docs.phonegap.com/en/3.3.0/cordova\\_camera\\_camera.md.html](http://docs.phonegap.com/en/3.3.0/cordova_camera_camera.md.html).

### Build the Android Project

```
C:\CameraApp>cordova build android
```

```
BUILD SUCCESSFUL  
Total time: 2 mins 30.24 secs  
Built the following apk(s):  
   C:\CameraApp\platforms\android\build\outputs\apk\android-debug.apk  
C:\CameraApp>
```

# THE END

Comments from the author: successfully tested with emulator genymotion custom phone 5.1.0 API 22, 768 x 1280 and OnePlus One

I uploaded the app on the following link:

<https://mahara.fhstp.ac.at/artefact/artefact.php?artefact=2998&view=162>

You can try it out if you want to.

## 3. Angular JS – a Data Binding Framework for Web

*by Anita Angerer*

### 3.1 What is AngularJS ?

HTML is a great language for building static documents, but it's not really made for creating dynamic web applications. So creating dynamic web applications only with html needs a lot of "tricking the browser". AngularJS is a structural framework for the development of dynamic web applications, mainly maintained by Google. It's open source and lets you extend your HTML vocabulary for your application. Therefore AngularJS also is a very helpful tool for the creation of single-page applications.

### 3.2 Adding AngularJS to your web page

AngularJS is a JavaScript framework, so it is written in JavaScript and distributed as a JavaScript file. Therefore it can be added to your web page with a script tag as the following:

```
<script  
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">  
</script>
```

You can find the right code snippet by browsing the Internet for "Google hosted libraries". This will lead you to <https://developers.google.com/speed/libraries/>, here you can find the code snippet as well as the different stable versions.

### 3.3 AngularJS directives extends HTML

AngularJS directives are HTML attributes with **ng** prefix. AngularJS extends HTML with these ng-directives, so they are simply written in the HTML tags as an additional adjective. For example:

```
<div ng-app=""></div>
```

```
<div data-ng-app=""></div>
```



*“data-ng-“ can be used instead of “ng-“ for making page HTML valid. Both has the same meaning for the functionality of your code and does the same. The only difference is, that some page validating tools don't know “ng-“ and therefore give back an error, while they don't see an error when “data-ng-“ is used. So if you want to validate your website using a validation tool, always using “data-“ before ng-directives might spare you some complications.*

## ***Data Binding***

Data binding in AngularJS is an automatic way of updating the view whenever the model is changed and the other way round updating the model whenever the view is changed. That way the worry about how to manipulate the DOM tree is eliminated. How to bind data with ng-directives is shown in the next step, an easy example application.

### ***A most simple example application using ng-directives and data binding***

Following we're going to use some basic ng-directives for a most simple example application.

**(data-) ng-app** defines an AngularJS application. You always need to start with ng-app so HTML knows you're using AngularJS within this part of your document, so it also couldn't solve your expressions. In the following example "ng-app" tells AngularJS that there is an AngularJS application in this <div> element.

```
<div ng-app=""></div>
```

**(data-) ng-model** binds values from HTML controls (input fields, text areas, selections) to application data /to an application variable. The following example shows how to bind the data received from the <input> field to an application variable, here called "variableName".

```
<p>Name: <input type="text" ng-model="variableName"></p>
```

**(data-) ng-bind**, as the name indicates, binds application data to your HTML view. In the following example the innerHTML is bound to the same variable "variableName". So the text the user writes in the input field from above is simultaneously written in your <p> element.

```
<p ng-bind="variableName"></p>
```

**(data-) ng-init** initializes AngularJS application variables. So if we add ng-init to our <div> element we initialize our variable "variableName" with the value 'Replace it with your name'.

```
<div ng-app="" ng-init="variableName='Replace it with your name'"></div>
```

**(data-) ng-controller** defines the application controller as follows

```
<div ng-app="myApp" ng-controller="myCtrl"></div>
```

## **3.4 AngularJS Expressions**

AngularJS expressions are written in double braces like {{expression}}. Expressions output the data exactly where they are written and are a lot like JavaScript expressions, they can contain variables, literals and operators. Still they are not the same, because they do also support filters and can be written inside HTML, and do not support conditionals, exceptions and loops. They bind data the same way as the ng-bind directive. Added to our HTML document the following line puts out the content of our variable, just as ng-bind does.

```
<p>Your first expression: {{ variableName }}</p>
```

 *AngularJS numbers, strings, objects and arrays are similar to JavaScript.*

If you put the above lines together in a HTML document (**Attention:** first add the script tag for AngularJS) you'll get an HTML page with an input field, where the content that you put in is written in your HTML console as you enter it in the input field. Dynamic DOM tree manipulation for the win! 😊

### 3.5 AngularJS Applications

An AngularJS Applications basically consists of

- View, which is the HTML
- Modules define AngularJS applications
- Controllers control AngularJS applications (JavaScript function that makes /changes /removes /controls the data)

#### **Modules**

The module defines the application; it is the container that contains the different application parts and controllers.

A module is created with the AngularJS function `angular.module` and referred to an HTML element in which the application runs/will run.

```
<div ng-app="myApp"></div>

<script>
  var app = angular.module("myApp", []);
</script>
```

It's important to notice that without a module, no other Angular JS elements as controllers, directives, etc. can be added.

#### **Controllers**

Controllers control the data of AngularJS applications and always belong to a module. An AngularJS controller is a JavaScript object and therefore created by a JavaScript constructor.

#### **Scope**

The `$scope` object is available for both, the controller and the view. Properties added to the `$scope` object in the controller can be accessed by the view. In the view it is referred to these properties with the property name like `{{firstName}}`.

### ***Example that shows the use of modules, controllers and \$scope***

The following example shows an AngularJS application (defined by ng-app="myApp") that runs inside the <div>. The **ng-controller="myCtrl"** directive defines a controller.

The myCtrl function is a JavaScript function. AngularJS will invoke the controller with a \$scope object, the \$scope object is passed as an argument whenever creating a controller. \$scope is the application object (the owner of application variables and functions). The controller creates two variables in the scope (firstName and lastName). The ng-model directives bind the input fields to the controller properties (firstName and lastName).

```
<div ng-app="myApp" ng-controller="myCtrl">
First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}
</div>

<script>
var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>
```

## **3.6 Events**

The **ng-click** directive defines AngularJS that will be executed when the HTML element is clicked. The syntax for ng-click is the following; it must not urgently be used with a <button> element.

```
<button ng-click="count = count + 1">Click Me!</button>
```

## **3.7 Filters**

- To format data filters can be added to AngularJS. Filters can be added to expressions and directives by using the pipe | followed by a filter like |uppercase.

```
<p>The name is {{ lastName | uppercase }}</p>
```

```
<li ng-repeat="x in names | orderBy:'country'"></li>
```

### ***Filters***

- **currency** formats a number to a currency format
- **date** formats a date to a specified format
- **filter** select a subset of items from an array
- **json** formats an object to a JSON string
- **limitTo** limits an array/string, into a specified number of elements/characters
- **lowercase** formats a string to lower case
- **number** formats a number to a string
- **orderBy** orders an array by an expression
- **uppercase** formats a string to upper case

**Now it's time for you to try, the full code examples can be found in the appendix.**

 *It should be noted that this research only includes the basics of AngularJS, of course AngularJS contains a lot more and can do a lot more.*



## 3.8 Appendix

### *mostSimpleDataBindingApp.html*

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"><
/script>
<body>

<div ng-app="" ng-init="variableName='Replace it with your name'">

<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="variableName"></p>
<p ng-bind="variableName"></p>

<!--Expression:-->
<p>Your first expression: {{ variableName }}</p>

</div>

</body>
</html>
```

### *modulesAndController.html*

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"><
/script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>

</body>
</html>
```

### ***ngClick.html***

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"><
/script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

<button ng-click="count = count + 1">Click Me!</button>

<p>{{ count }}</p>

</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.count = 0;
});
</script>

</body>
</html>
```

### **3.9 Used references**

<https://angularjs.org/>

<http://www.w3schools.com/angular/default.asp>

<https://www.codecademy.com/learn/learn-angularjs>

## 4. Run NodeJS on a microcontroller such as the Raspberry PI

by Mario Schartner

Time - install nodeJS: 45min – install from zero: 9h – complete installing with docu: 17h

### 4.1 Introduction

The Raspberry PI is a small factor SoC (System on a chip) board with low power consumption, a mini computer board with common interfaces and connectors like our smartphones contain. Because the board is very tiny and consumes low power there is a big potential for creativity and many possibilities are arising out of this concept.

So you can choose whether you want to use the USB ports for connecting harddrive, mouse or keyboard, network connection for doing some server – client stuff or a video/HDMI connector for running GUI (graphical user interface) supported software using the Raspberry PI as a simple mini computer, do-it-yourself tablet PC or media center for streaming your film library from your already configured home NAS (network attached server) shares. In most cases these ARM boards (they have no x86 board architecture like a common desktop PC or Laptop) use operating systems, which are special prepared to run on the certain board. With the Raspberry PI it is possible to choose between various performance-driven OS' (operating systems) aiming on the already mentioned required purpose.

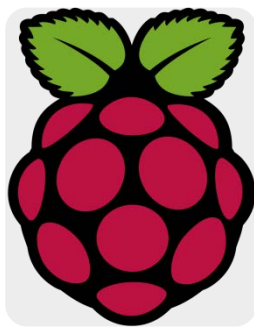


Fig. 5.1.1 Raspberry PI Logo

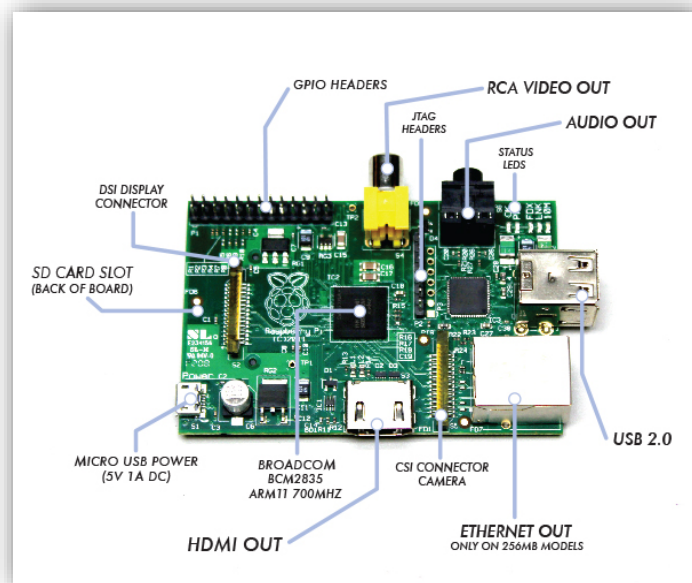


Fig. 5.1.2 Raspberry PI board

## 4.2 Preparing the PI

With the PI we have the hardware. But like any other computational device the hardware needs an operating software to run - the so called operating system - which has the board's drivers included for communicate with the hardware and let it work with most performance. For this purpose the PI has an SD card slot. We can now download the required OS for our purpose, in our case we take the standard OS for the PI with the name Raspbian (because it's a modified version of the famous Debian Linux on which many Linux distributions rely on).

For more details on what is Linux, which differences are between the distributions and the GUI there are many info's on the internet on how to join the world of Linux and with which easy distribution you can begin learning about it.

We now download the Raspbian image from <https://www.raspberrypi.org/downloads/>.

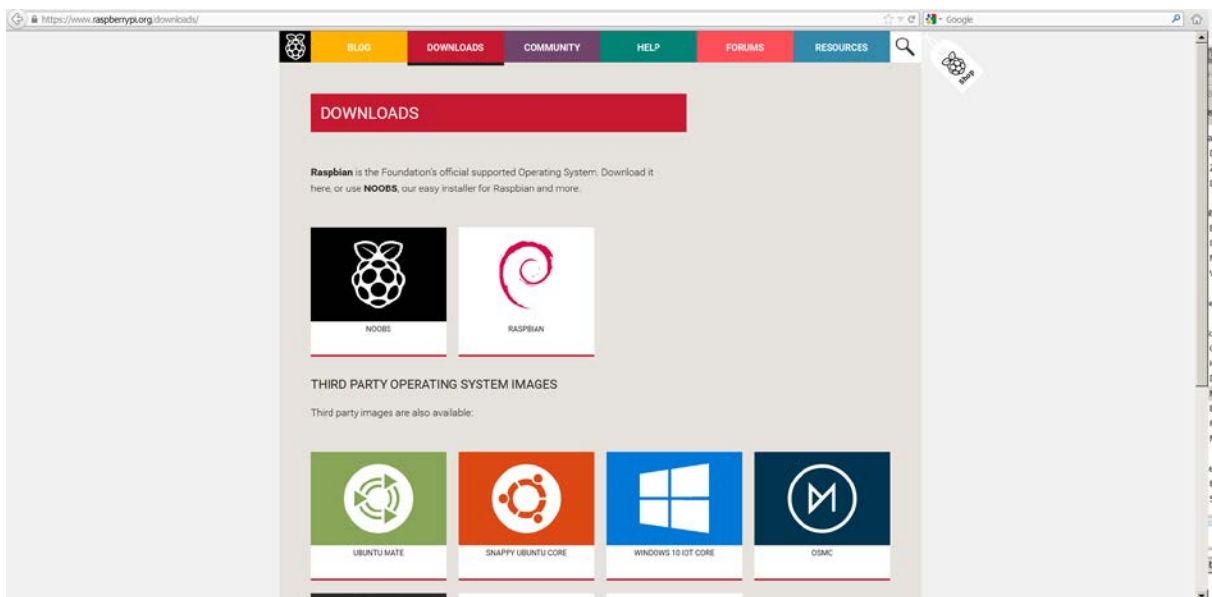


Fig. 4.2.1 Download link for Raspbian <https://www.raspberrypi.org/downloads/>

After the image download has finished we want to get the OS to a SD card. It is recommended to take an 8 GB card and pre-format it to FAT32. For installing the Image to the SD card you need a computer with internal multimedia card reader or a card reader device which you can plug to the PC via USB.



Fig. 4.2.2 Multimedia Card



reader for all platforms (OS)

Now with the needed equipment we can begin to work. We connect the SD card to a free SD slot and should see the device. In general the card is preformatted to the required FAT32 if it's new. If not, formatting devices is possible with any OS.

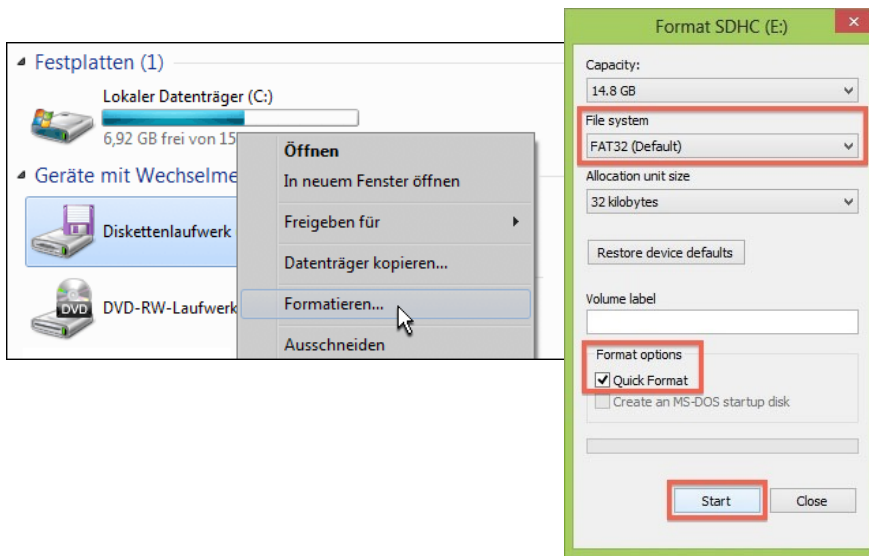


Fig. 4.2.3 Formatting the SD card in Windows

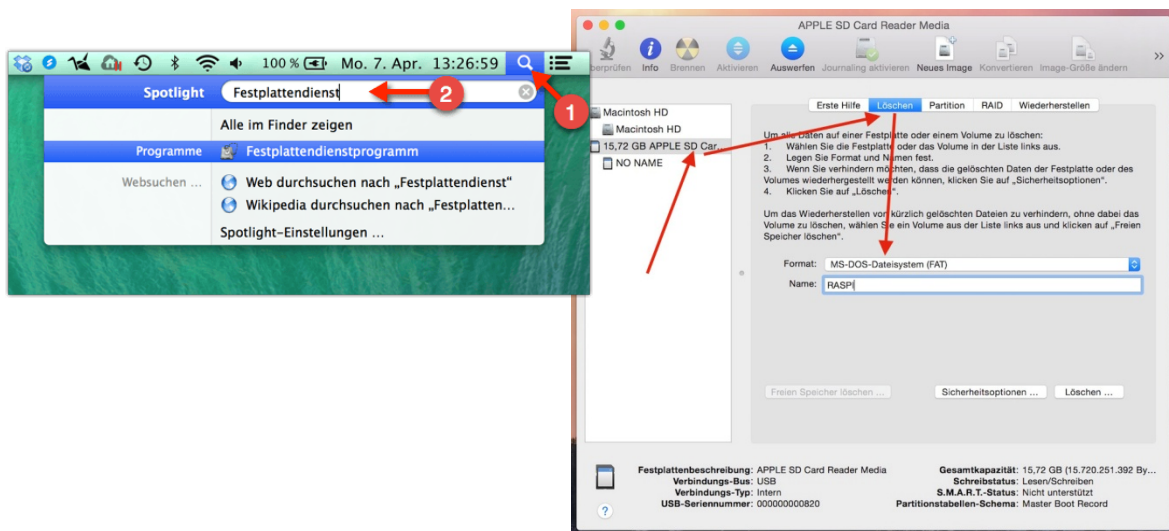


Fig. 4.2.4 Formatting the SD card in Mac OSX

When formatted, the easiest way to get the Image onto the card (in windows) is with the Win32DiskImager. This tool will make the stick bootable, that the PI hardware can find the

correct media with the OS. There are also other tool to get an Image File onto a media, also for MAC and Linux

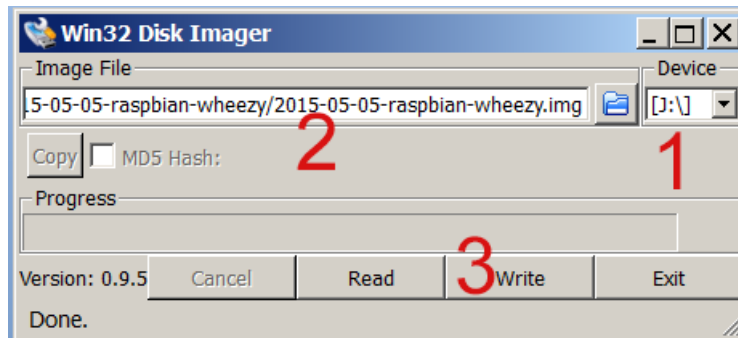


Fig. 4.2.5 Formatting the SD card in windows

Figure 5.2.5 shows the click-order. You choose the device with the Raspbian image and the affiliated path. If you're sure you configured that right you only have to click the "Write" Button for unpacking and transferring the image to the card. This could need some minutes, so stay calm 😊.



Fig. 4.2.6 Waiting until formatting is done

OK! The copy process has finished because the Imager says "Done" in the left lower corner. We're now ready to eject the SD card from the slot and feed it to the PI.

We now connect the PI to a router/switch or another network device (I did all following steps under consideration of using the router/switch).

When connected to the network device there's only the power supply left. We plug it into a socket and the Raspberry should immediately start to work which we can see on lighting and blinking LEDs. We can now start our first connection to the PI and the fresh new OS.

If the PI stays dark, you have to check the power supply and the recommendations on the website for technical specification you have to pay attention on.

### 4.3 Installing needed software

For working with the PI there are some different options depending on your knowledge and your technical infrastructure you have available.


For this documentation I have no monitor available for the PI. For a more transparent documentation I will prepare the PI to get a GUI via network connection.

#### Initial setup of the PI

With the PI running, first we have to check if we can get a connection to the PI via network. We therefore need another small tool which is the basic software for administer servers and the whole network infrastructure. The application is called putty and freeware which you can download from <http://www.putty.org/>. You can download it to the desktop, because it is standalone software so there is no installation needed.

We are ready to go! Now depending on the configuration of your network device we have to look if we got an IP address over DHCP or we are in a static configuration. If the following steps don't work you have to check you network configuration to get the needed information. In general all devices are preconfigured in DHCP and normally get an IP address from the router/switch. If there is a static configuration, someone already has configured one of the devices we are using to communicate with themselves.

Therefore we look up for the IP with our administration device (Laptop, PC, etc.).

In windows start the command line via the startup menu (ger: Zubehör-Eingabeaufforderung) or more simple  + R and in the appearing field we write "cmd".

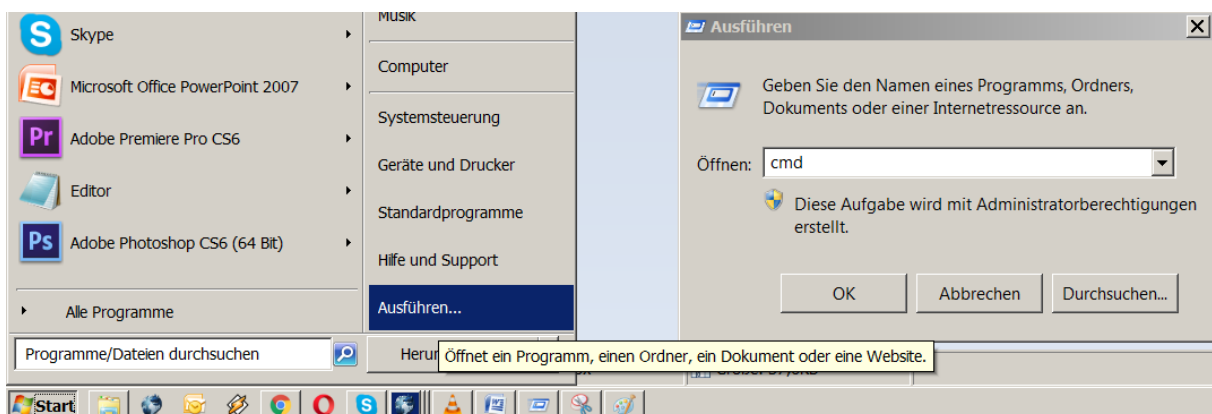


Fig. 4.3.1 Running the command line

We now enter the command *"ipconfig /all"* (Mac OSX: *"ifconfig"* in the terminal/console) to show all address parameters of all devices.

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\scharti>ipconfig /all

Windows-IP-Konfiguration

    Hostname . . . . . : schartiWORK
    Primäres DNS-Suffix . . . . . :
    Knotentyp . . . . . : Hybrid
    IP-Routing aktiviert . . . . . : Nein
    WINS-Proxy aktiviert . . . . . : Nein

Ethernet-Adapter LAN-Verbindung:

    Verbindungsspezifisches DNS-Suffix:
    Beschreibung. . . . . : Intel(R) 82579LM Gigabit Network Connecti
on
    Physikalische Adresse . . . . . : B4-B5-2F-CB-3D-D0
    DHCP aktiviert. . . . . : Nein
    Autokonfiguration aktiviert . . . : Ja
    Verbindungslokale IPv6-Adresse . . : fe80::f447:dd66:31cd:c4c8%13(Bevorzugt)
    IPv4-Adresse . . . . . : 192.168.1.30(Bevorzugt)
    Subnetzmaske . . . . . : 255.255.255.0
    IPv4-Adresse . . . . . : 192.168.2.6(Bevorzugt)
    Subnetzmaske . . . . . : 255.255.255.0
    Standardgateway . . . . . : 192.168.1.1
    DHCPv6-IAID . . . . . : 280278319
    DHCPv6-Client-DUID. . . . . : 00-01-00-01-18-82-87-CB-B4-B5-2F-CB-3D-D0

    DNS-Server . . . . . : 192.168.1.1
    . . . . . : 195.3.96.67
    NetBIOS über TCP/IP . . . . . : Aktiviert

Ethernet-Adapter LAN-Verbindung 4:

    Medienstatus. . . . . : Medium getrennt
    Verbindungsspezifisches DNS-Suffix:
    Beschreibung. . . . . : TeamViewer UPN Adapter
    Physikalische Adresse . . . . . : 00-FF-52-57-92-E5
    DHCP aktiviert. . . . . : Ja
    Autokonfiguration aktiviert . . . : Ja

Ethernet-Adapter Hamachi:
```

Fig. 4.3.2 Formatting the SD card in windows

If you are connected via Ethernet you have to keep the Ethernet Adapter address in mind otherwise via W-LAN the IP address from the wireless adapter is the correct one for you.

In general, if you have an address from DHCP, there should be at least an IP like 192.168.x.x and you can see it in the adapter listing *"DHCP activated"*, but we will give it a try and see what we get.

Initially the PI also has DHCP activated and has now assigned an IP from the router.

If this is not working you have to know the IP of the router and connect to it per SSH with putty if this option is activated or with web-GUI via browser to look for connected devices and get their IP address, in our case the PI's.



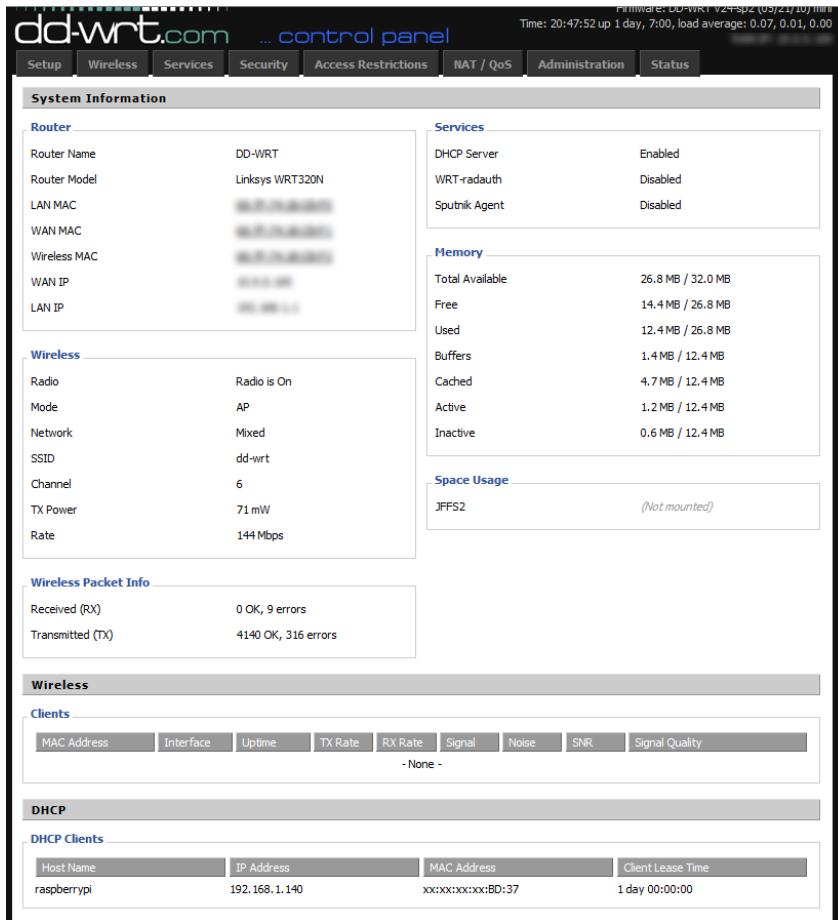


Fig. 4.3.3 Router configuration with connect devices

We can now communicate with the PI. For this, we open the putty application and enter the address we have for the PI and connect to it.

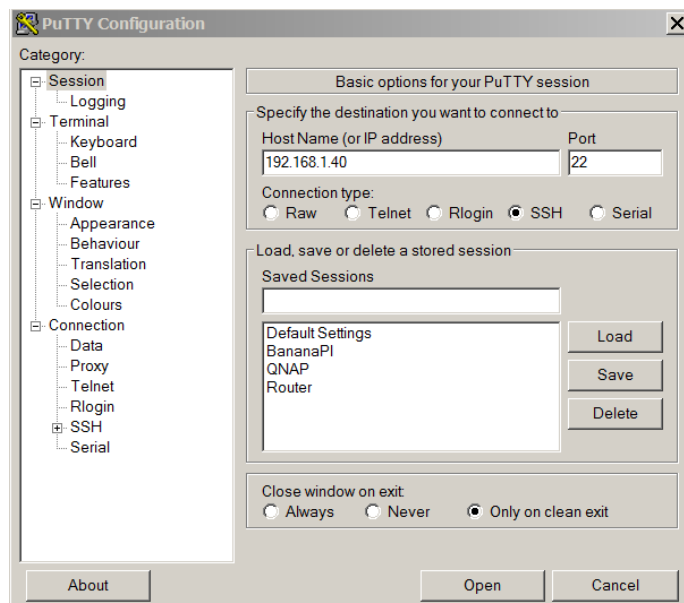


Fig. 4.3.4 Connecting to the PI with putty

We now got our terminal window asking for username (initial: pi) and password (initial: raspberry).

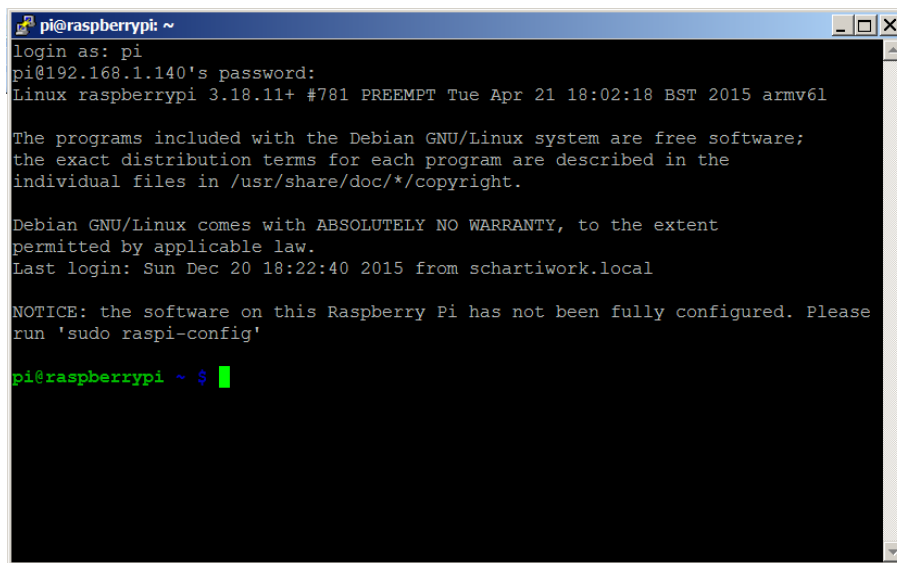


Fig. 4.3.5 Welcome Screen after Login

Because we want to use the PI as server which has to be reachable 24/7 it needs a static IP address in the same subnet as at the moment. This is important because if the router DHCP client address list gets cleared and a new IP is assigned to the PI, every PI depending application and server the PI communicated with, will not find it anymore.

With the command:

```
sudo nano /etc/network/interfaces
```

we start the command line “nano” editor.

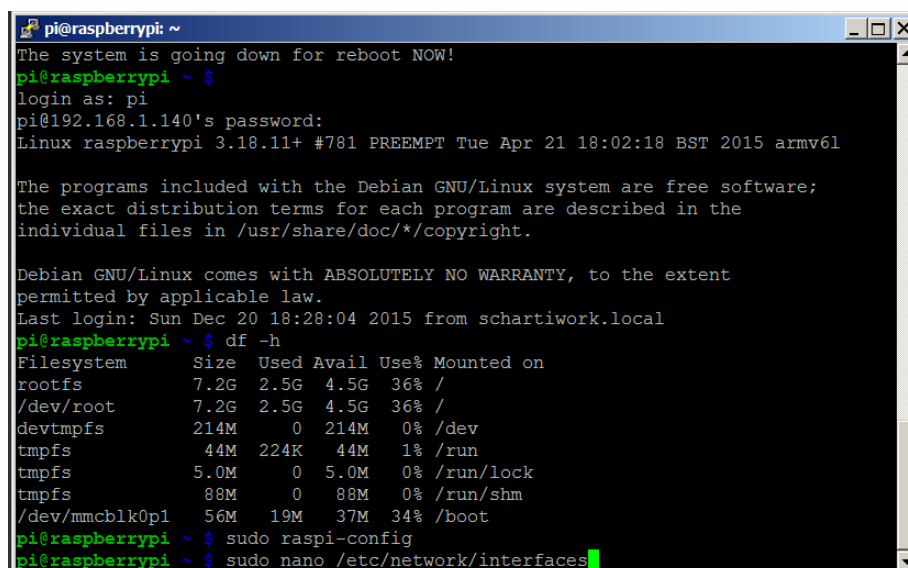
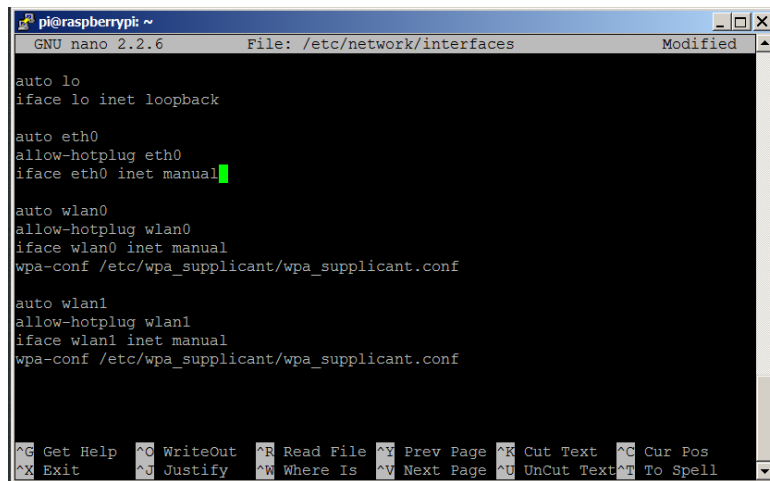


Fig. 4.3.6 Start nano editor and open the interfaces configuration text file

Now we can edit the interfaces file as follows:



```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/network/interfaces Modified
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet manual

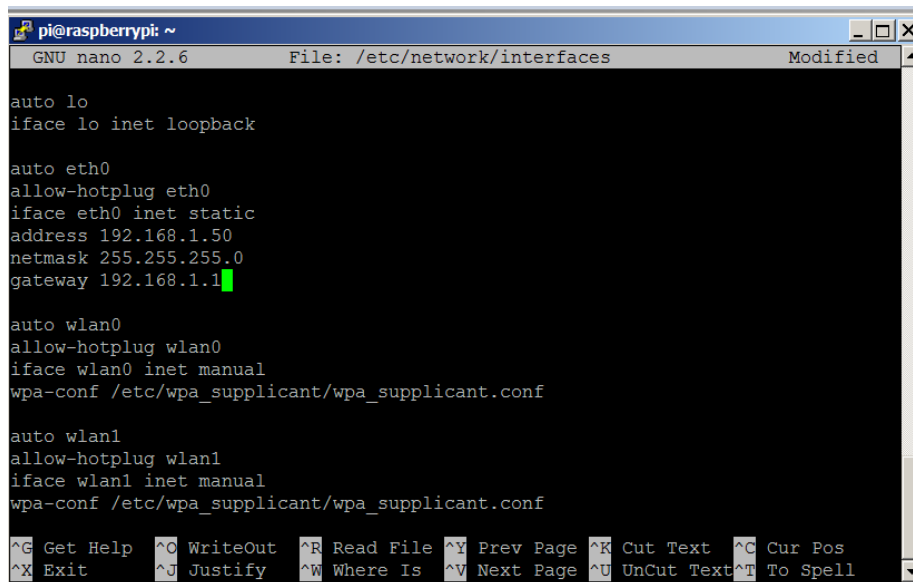
auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

auto wlan1
allow-hotplug wlan1
iface wlan1 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Fig. 4.3.7 Original file entries

We edit the LAN interface to the desired IP address of the PI, the standard subnet mask and the correct gateway (router address) to get connection to the internet for further software package downloading.



```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/network/interfaces Modified
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet static
address 192.168.1.50
netmask 255.255.255.0
gateway 192.168.1.1

auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

auto wlan1
allow-hotplug wlan1
iface wlan1 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Fig. 4.3.8 corrected file entries for static IP

We finally quit the nano editor by pressing “*Ctrl+X*”, then “*Y*” and *Enter* and are back in the shell.

The last step to make the changes work is to restart the network service by entering the following command:

```
sudo /etc/init.d/networking restart
```

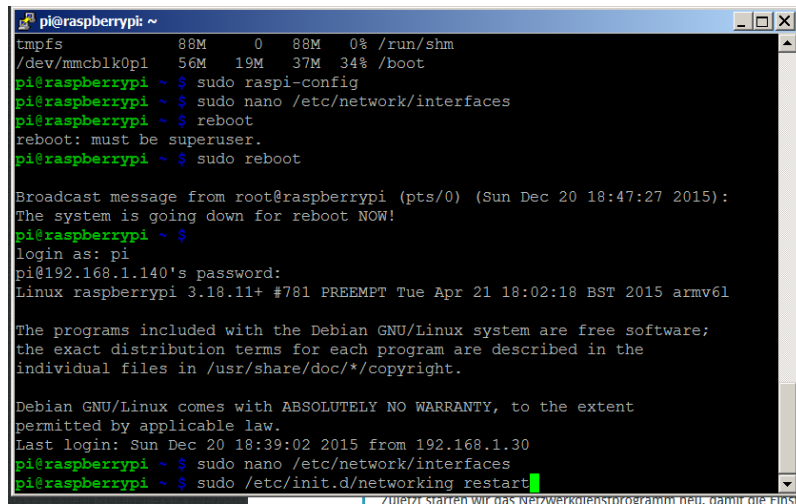


Fig. 4.3.9 Restarting the networking service

With the now static configured IP, we can do a small tweaking in the PI configuration. Because the downloaded initial image doesn't use 8 GB space on the SD card, we can now extend the storage to the full 8 GB to have the possibility to install more applications.

Before changing the configuration we enter the following command to visualize that the SD card shows less than 8 GB.

```
df -h
```

We now enter the command mentioned in the welcome screen to configure the PI again with super user rights – therefore "sudo":

```
raspi-config
```

Beside other menus we straight choose the first point for expanding the file system to ensure using the whole capacity

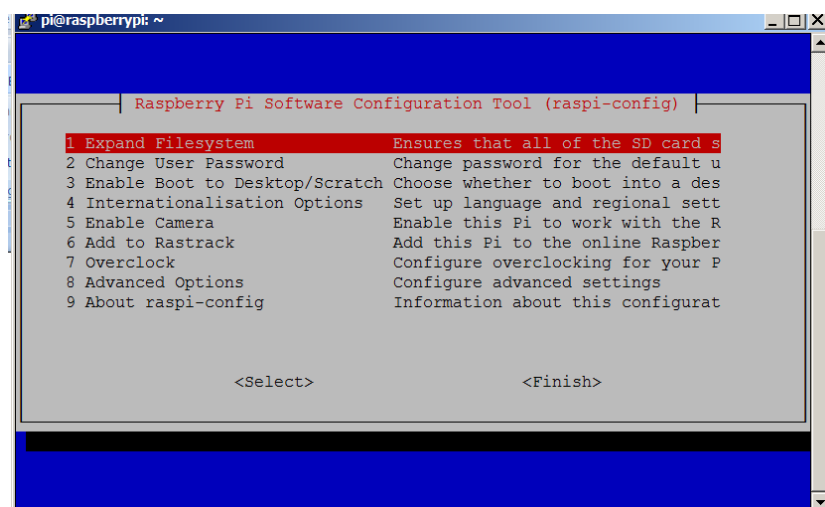
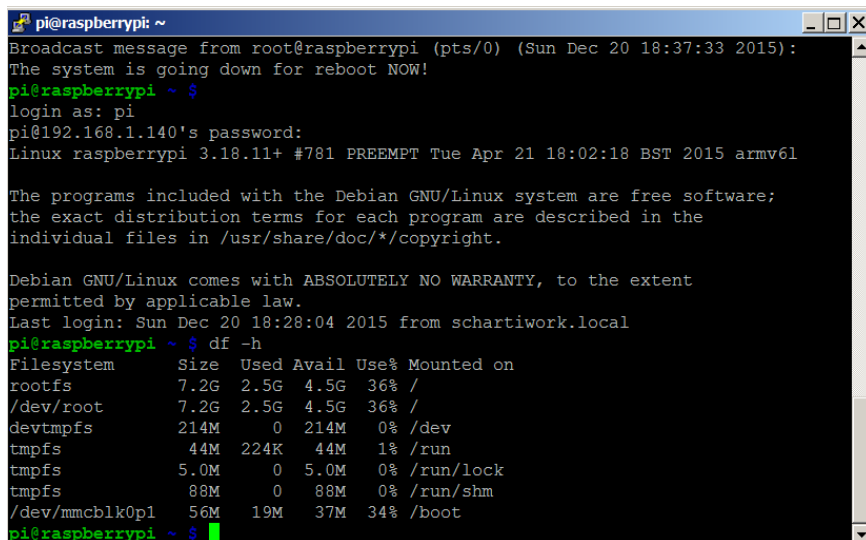


Fig. 4.3.10 Expanding the SD card file system to the whole usable capacity

Now we can see the PI is using the full capacity of the SD card.

With the command “df -h” we can see the all drives with their capacity

A terminal window on a Raspberry Pi showing the output of the 'df -h' command. The terminal displays system boot messages, login information, and the disk usage table. The disk usage table shows the following data:

Filesystem	Size	Used	Avail	Use%	Mounted on
rootfs	7.2G	2.5G	4.5G	36%	/
/dev/root	7.2G	2.5G	4.5G	36%	/
devtmpfs	214M	0	214M	0%	/dev
tmpfs	44M	224K	44M	1%	/run
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	88M	0	88M	0%	/run/shm
/dev/mmcblk0p1	56M	19M	37M	34%	/boot

Fig. 4.3.11 Checking the size change

## Installing tightVNC

Although we have no screen it is not absolutely necessary to install a Remote Desktop Viewer but for beginners it’s quite a good advantage to see what happens while we are using the console/terminal.

We have to use the package management command in console to get and install the desired software. The command is the following:

```
sudo apt-get install tightvncserver
```

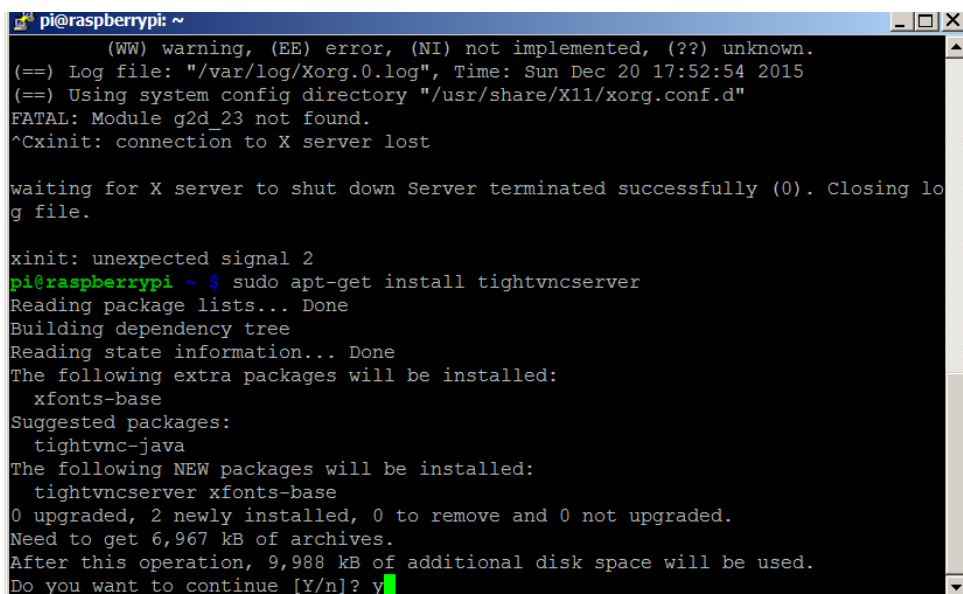
A terminal window on a Raspberry Pi showing the installation of 'tightvncserver'. The terminal displays X server error messages, followed by the execution of 'sudo apt-get install tightvncserver'. The output shows the package lists, dependency tree, and the packages to be installed. The following extra packages will be installed: xfonts-base. Suggested packages: tightvnc-java. The following NEW packages will be installed: tightvncserver xfonts-base. 0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded. Need to get 6,967 kB of archives. After this operation, 9,988 kB of additional disk space will be used. Do you want to continue [Y/n]? y

Fig. 4.3.12 Installing the RDP Server application

After the package manager checked the dependencies he is asking to execute the installation. Because we for sure want to install this software we confirm the operation and press the “Y” key.

If you’re brand-new to Linux you initially have to know that “*sudo*” (SuperUserDO) runs the command as administrator with more user rights. “*apt-get*” is the name of the package management software from the Linux Debian distribution. “Install” means the command which we want to do and is followed by the application name we want to install. This is the basic structure in the shell (Linux command line) and can be used in all Debian – like distributions.

Additionally we can install preload, because it accelerates the GUI. If we use the PI only as server and rarely the GUI, there’s no need for preload and you can jump to the next block without entering the following two commands.

```
sudo apt-get install preload
```

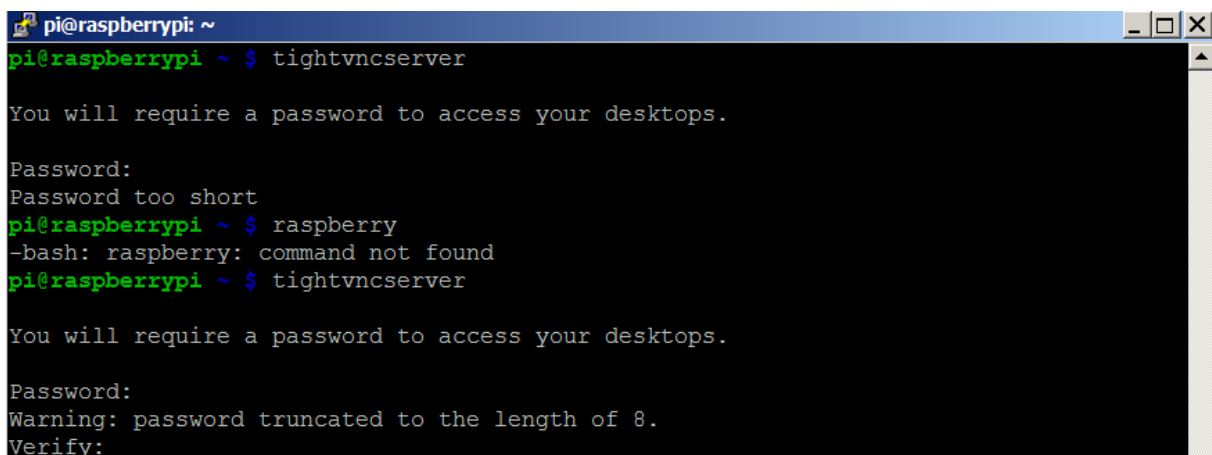
and

```
sudo sed -i 's/sortstrategy = 3/sortstrategy = 0/g' /etc/preload.conf
```

Without installed preload we enter here again and first time start our VNC RDP (remote desktop) session with entering:

```
tightvncserver
```

The first time starting the service you will be asked for a password for following sessions. Because the password is limited to 8 characters I entered “*raspberr*”.

A terminal window titled 'pi@raspberrypi: ~' showing the execution of 'tightvncserver'. The output includes a prompt for a password, a warning that the password is too short, and the user entering 'raspberr'. This results in a '-bash: raspberr: command not found' error. The user then re-enters 'tightvncserver', which prompts for a password again, showing a warning that the password is truncated to 8 characters and a 'Verify:' prompt.

```
pi@raspberrypi ~ $ tightvncserver
You will require a password to access your desktops.
Password:
Password too short
pi@raspberrypi ~ $ raspberr
-bash: raspberr: command not found
pi@raspberrypi ~ $ tightvncserver
You will require a password to access your desktops.
Password:
Warning: password truncated to the length of 8.
Verify:
```

Fig. 4.3.13 Starting the RDP Server and initially set a password

We have now started the GUI RDP Server on our PI but of course need a tool to contact the PI from our Desktop PC/Mac or laptop/MacBook. Therefore you can download and install the tightVNC Viewer for Windows (<http://www.tightvnc.com/download.php>) and the RealVNC Viewer for Mac

(<https://www.realvnc.com/products/vnc/documentation/5.0/installing-removing/macosx>).

You can now join the PIs server desktop by starting the downloaded and installed client VNC viewer app, entering the PIs static configured IP address with port 5901 and enter the before assigned password and hit the “OK” button.

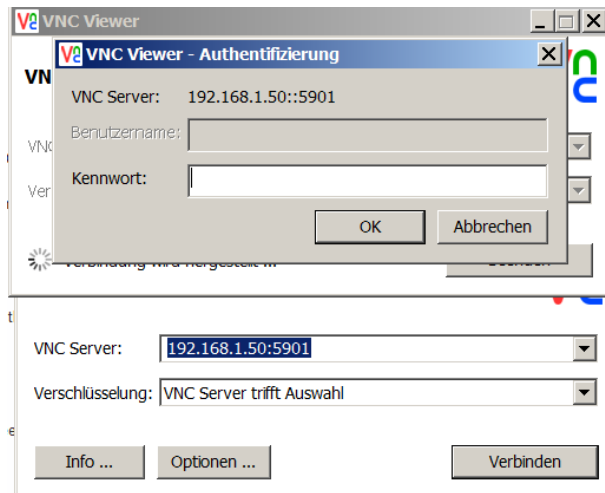


Fig. 4.3.14 Connecting to the Server GUI from our machine

Now you're entering the server desktop!

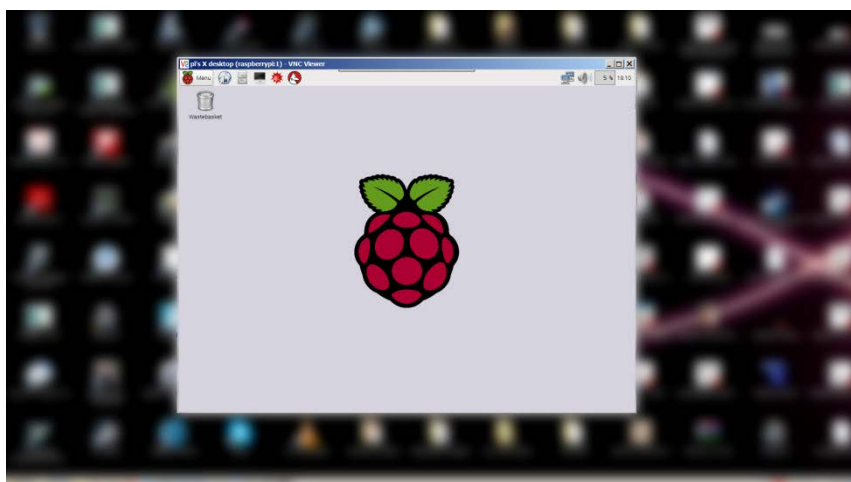


Fig. 4.3.15 Raspberry PI desktop

E voila! You've joined the graphical interface of the PI and can use some preinstalled applications. But you have to know that RDP connections are slower and have more latency

than locally using a monitor and working with the server. It is not useful to do some graphic – sophisticated stuff like watching videos or playing graphic – intensive games because the delay when transferring the picture will display only every 25<sup>th</sup> frame estimated.

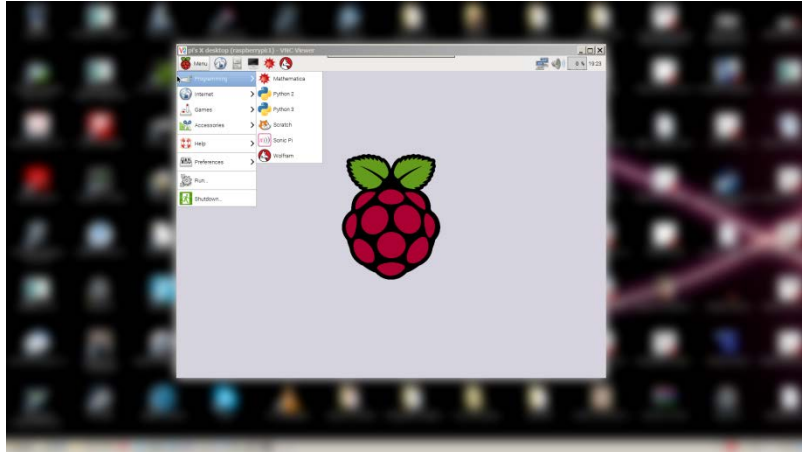


Fig. 4.3.16 Raspberry PI desktop

Congratulations, you have installed your first application onto the PI.

If you want to automatically start the VNC server on every startup you have to write a shell script (everywhere on the internet), otherwise you have to use the *"tightvncserver :1"* command to start the GUI server. If you use the PI as server or for testing purposes like we do, it is not necessary to do this, because the server in general runs 24/7 without any restarts, and for testing purposes we want to use the vnc server start command to remember it. To kill the service we run this command:

```
vncserver -kill :1
```

## 4.4 Installing nodeJS

We are now ready to go over to the main target that we want to accomplish – installing the Node JS Server application.

Actually Node JS is preferred because it is the solution for server based Java scripting and perfect for working off parallel server requests very fast.

When installing a new application we first of all have to check and upgrade our current package database with:

```
sudo apt-get update && sudo apt-get upgrade
```

The PI checks and upgrades every previous installed application and corrects possible errors in the package database.



```
pi@raspberrypi ~$ sudo apt-get update && sudo apt-get upgrade
Hit http://raspberrypi.collabora.com wheezy Release.gpg
Get:1 http://mirrordirector.raspbian.org wheezy Release.gpg [490 B]
Hit http://raspberrypi.collabora.com wheezy Release
Get:2 http://mirrordirector.raspbian.org wheezy Release [14.4 kB]
Get:3 http://archive.raspberrypi.org wheezy Release.gpg [473 B]
Get:4 http://archive.raspberrypi.org wheezy Release [17.6 kB]
Hit http://raspberrypi.collabora.com wheezy/rpi armhf Packages
Get:5 http://mirrordirector.raspbian.org wheezy/main armhf Packages [6,909 kB]
Get:6 http://archive.raspberrypi.org wheezy/main armhf Packages [135 kB]
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en_GB
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en
Ign http://archive.raspberrypi.org wheezy/main Translation-en_GB
Ign http://archive.raspberrypi.org wheezy/main Translation-en
Get:7 http://mirrordirector.raspbian.org wheezy/contrib armhf Packages [23.6 kB]
Get:8 http://mirrordirector.raspbian.org wheezy/non-free armhf Packages [49.3 kB]
Get:9 http://mirrordirector.raspbian.org wheezy/rpi armhf Packages [592 B]
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en
100% [5 Packages xz 0 B] 11.6 kB/s 0s
```

Fig. 4.4.1 Upgrading installed PI applications

You have to confirm the upgrade by pressing the “Y” key, then the PI installs the downloaded upgrade packages.

```
Get:38 http://mirrordirector.raspbian.org/raspbian/ wheezy/main ghostscript armhf 9.0
5-dfsg-6.3+deb7u2 [79.7 kB]
Get:39 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libgs9 armhf 9.05-dfs
g-6.3+deb7u2 [1,545 kB]
Get:40 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libgs9-common all 9.0
5-dfsg-6.3+deb7u2 [1,977 kB]
Get:41 http://mirrordirector.raspbian.org/raspbian/ wheezy/main openssl armhf 1.0.1e-
2+rvvt+deb7u18 [702 kB]
Get:42 http://mirrordirector.raspbian.org/raspbian/ wheezy/main sudo armhf 1.8.5p2-1+
rmu3 [838 kB]
Get:43 http://mirrordirector.raspbian.org/raspbian/ wheezy/main unzip armhf 6.0-8+deb
7u5 [195 kB]
Get:44 http://mirrordirector.raspbian.org/raspbian/ wheezy/main wpasupplicant armhf 1
.0-3+deb7u3 [542 kB]
Get:45 http://mirrordirector.raspbian.org/raspbian/ wheezy/main wpagui armhf 1.0-3+de
b7u3 [361 kB]
Get:46 http://archive.raspberrypi.org/debian/ wheezy/main libraspberrypi-bin armhf 1.
20151118-1 [233 kB]
43% [Working] 1,038 kB/s 1min 12s
```

Fig. 4.4.2 Upgrading installed PI applications

When the upgrading process is completed, we prepare the PI with libraries on which the NodeJS application is built on with

```
sudo apt-get install git-core build-essential python libssl-dev nano
```

After that you have to decide which nodeJS version you need. Higher than version 0.9.x nodeJS is ready to download as precompiled binaries. Earlier than 0.9.x you have to precompile it separately on your own local PI and therefore some more steps to go through (see link for precompiling).

Now finally we can grab our desired nodeJS package from the web.

We change the folder as working directory where we want to download the package

```
cd /opt
```

and get our package from the web with

```
sudo wget http://nodejs.org/dist/v0.10.8/node-v0.10.8-linux-arm-pi.tar.gz
```

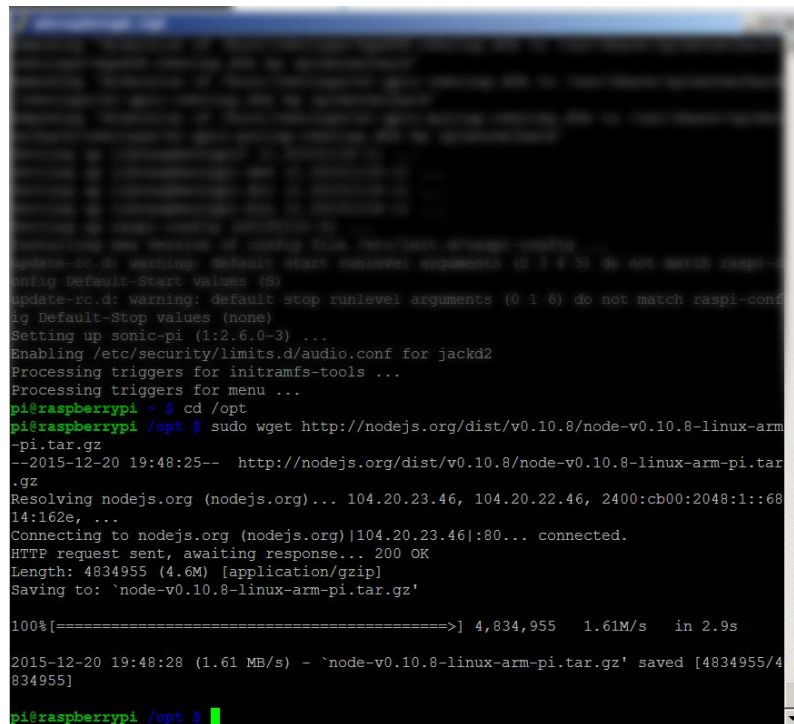


Fig. 4.4.3 Download the NodeJS Package to /opt

After that you can check the position of the \*.tar archive package via the GUI in the File Manager.

We trust the command line and unpack the \*.tar archive with

```
sudo tar xfvz node-v0.10.8-linux-arm-pi.tar.gz
```

then we delete the \*.tar archive with

```
sudo rm node-v0.10.8-linux-arm-pi.tar.gz
```

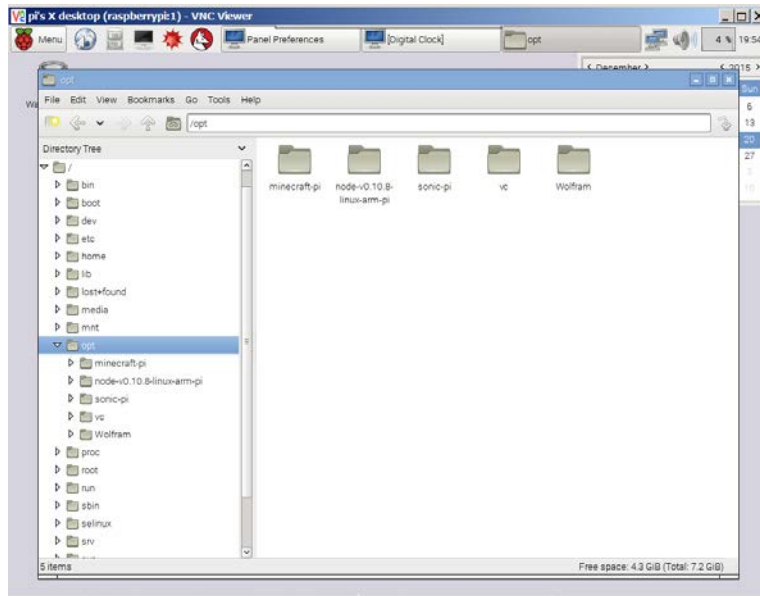


Fig. 4.4.4 Check the unpacking and removing \*.tar commands

and paste the unpacked folder to the new “node” folder which we also create new in the next line

```
sudo mv node-v0.10.8-linux-arm-pi/ node/
```

After that we check the steps with the file manager of the GUI ...

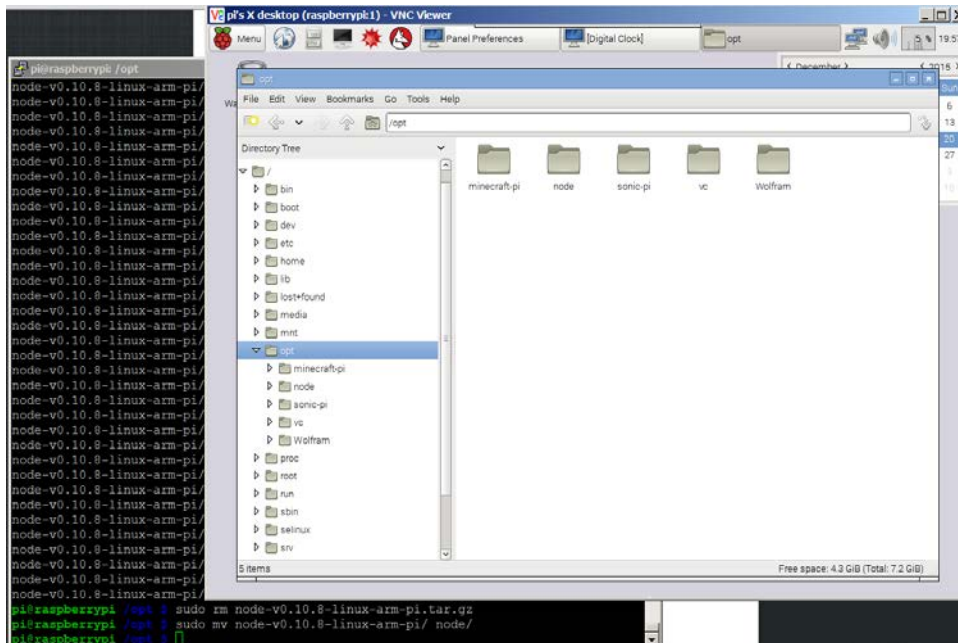


Fig. 4.4.5 Check the unpacking and removing \*.tar commands

... and see that there is the created node folder with the unpacked nodeJS folder in it.

Now have to enter the nodeJS configuration file with the nano editor ...

```
nano ~/.profile
```

... to set the Path variable for running the node application with the “*node*” command from everywhere in the local system without to enter always the whole path to the “*node.bin*”. We can do this by appending the following line to the end of the text file.

```
export PATH=$PATH:/opt/node/bin
```

We save Quit and close the nano editor again (“*Ctrl+X*” - “*Y*” - “*Enter*”) and at least reload the “*.profile*” file if we use a precompiled binary with

```
source ~/.profile
```

Done!!! We have finished installing the nodeJS platform on the Raspberry PI and can now try some nodeJS stuff.

## 4.5 Simple Testing of Node JS

We are now ready to place a \*.txt file to the desktop which we give the filename (“*helloworld*”) and the typical \*.js extension and enter the

```
Console.log("Hello world!");
```

code line into the file.

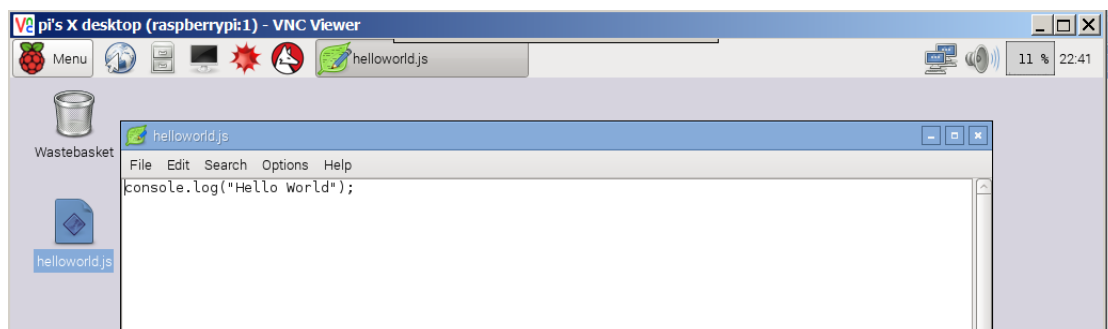
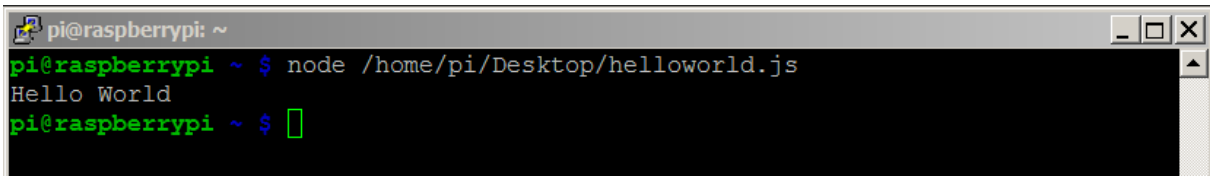


Fig. 4.5.1 Create a “*helloworld.js*” file

After that we can check if the node works with calling the *“helloworld.js”* file.

A terminal window on a Raspberry Pi. The prompt is 'pi@raspberrypi: ~'. The user enters the command 'node /home/pi/Desktop/helloworld.js'. The output is 'Hello World'. The prompt returns to 'pi@raspberrypi: ~ \$' with a cursor.

```
pi@raspberrypi: ~  
pi@raspberrypi ~ $ node /home/pi/Desktop/helloworld.js  
Hello World  
pi@raspberrypi ~ $
```

Fig. 4.5.2 Run the *“helloworld.js”* file with the node command

It works! So we can go for a harder exercise following in assessment 3!

## 4.6 Running XXX with nodeJS on RaspberryPI

## 5. What is Unity?

*by Anita Angerer, Georg Schreiner, Mario Schartner*

Unity is a software environment in which you can combine 2 or 3 dimensional content with human computer interaction and export it to a multiplatform compatible virtual scene file.

Like other graphical game engines like the famous “CryEngine” or the long existing “Unreal Engine” unity combines all elements for creating a virtual world.

For the first approach of learning to use Java Script in Unity there is chosen an intermediate tutorial called “Roll-a-ball” in which the connection between JS and Unity gets described.

### 5.1 Unity from the CGI perspective

Today Unity is the most common engine to use for creating games and gamified applications where human interaction is asked.

The success of this engine already starts with the licensing system, because it is initially free and when earning a certain amount of profit you have to pay it only once and not a monthly fee. This makes sense for beginners and start-ups and consequently there is a big fast growing community from that.

But also the functionality which we basically take a look on will show an easy usability for starting to work with the software

#### 2.1.1 Basic elements in a virtual world

For creating a virtual world which should be equal to our real world (e.g. physics, etc.) there are some common things the virtual world has to use.

Initially one has to know the basic concept of working with a 3d CGI (Computer Generated Imaging). The following things are necessary for implementing 3d scenes:

##### Objects/Models

Every CGI artist starts with a blank scene, so there is nothing than empty 3 dimensional space. The artist commonly starts to create certain objects, also called models, which fill this space and further will be used within this scene or other scenes. The artist creates his detailed model basically out of standard shapes like cubes, spheres or cylinders.

##### Camera/View/Perspective

Every scene has at least one camera to show the content from where the artist wants to see it or show it to the viewer/user. Of course the camera itself has many options to simulate the real ones beginning from the lenses, the angle, the sensors, the depth of field and much more.

##### Textures/Materials/Shaders

When creating an object within a 3d scene it is displayed gray. This can be compared with a potter, which initially forms his object out of colorless clay. To give the object some properties of reflecting light you have to apply shaders onto your object. A shader determines the amount of light to be reflected or gets diffused.

A material is also a shader with some properties we know from certain materials like metal or plastic.

A texture is picture with patterns or every content we want to be layed onto our object.

All together we can now take a photo of a pot, take it as a texture on our 3 dimensional pot and choose a material which equals the lightning & reflecting properties of the real pot.

### Light/shadows

There has to be set some light to the scene in order to create e.g. daylight or night-scenes.

Basically there are three types of light to represent the common light sources in our real world.

With directional light you can light up your whole scene from one direction with parallel beams. So this light is chosen to simulate e.g. sun in you scene.

Spot light is, described by its name, to illuminate a certain spot of your scene. This type of light is used for car lamps, stage lamps and many more light applications.

The third big type of light sources are point lights, which shine in a spherical manner and are used for light bulbs and any other light sources of this kind.

### Animation/Motion

If the artist wants to render (export) a series of images (video), it only makes sense if there is some motion within the scene.

That means that in our game/graphic engine software there has to be the option of moving our objects in the 3 dimensional space of our scene.

### Physics/Simulation

To get the viewer or user the experience of objects in the scene behaving like the same ones in real world there has to be implemented physics.

Like already mentioned, the virtual world owns its behavior from our configured physics.

The most important configuration option is the gravity factor, which defines the behavior of objects falling to the ground and of course the collision mathematics which declares the behavior of two or more objects colliding with or touching each other.

### 2.1.2 Starting and working with Unity

When looking at the Unity surface after generating a project with its working path and the other scene preparing configuration, we see the canvas, which is splitted into our working monitor and the in-game monitor, in which we can simulate the actual behavior of the scene.

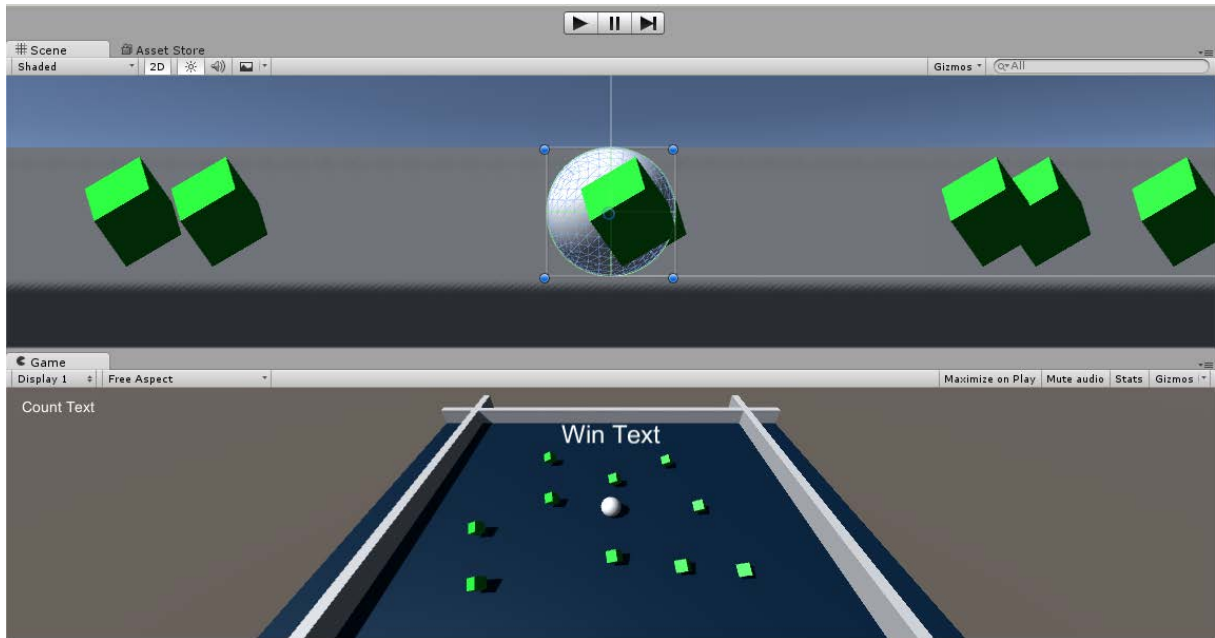


Figure 1: The „canvas“ splits the screen into a working area and an in-game monitor.

#### The “project tab”

There is the “project tab” (Figure 2) which shows all of our project content including our scenes, our defined materials, our programmed scripts and everything else we want to use for our scene.

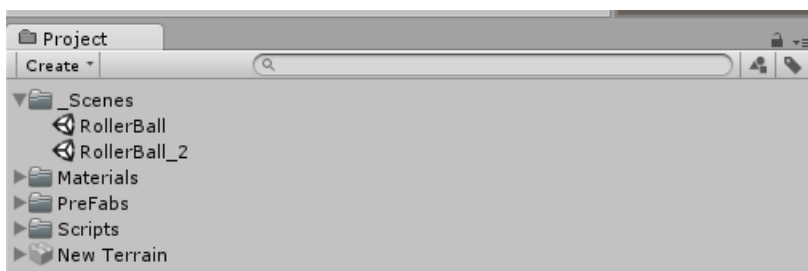


Figure 2: „project tab“

#### The “hierarchy- and inspector tab”:

The “hierarchy tab” (Figure 3) lists every object in our actual open scene and the “inspector tab” (Figure 4) shows all properties of our selected object.



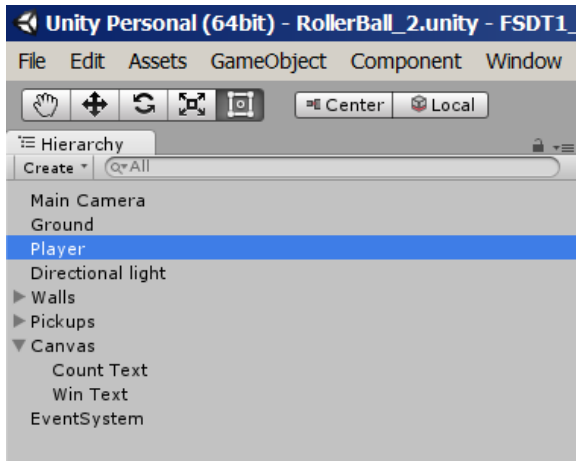


Figure 3: The „Hierarchy Tab“ shows every object in the current scene

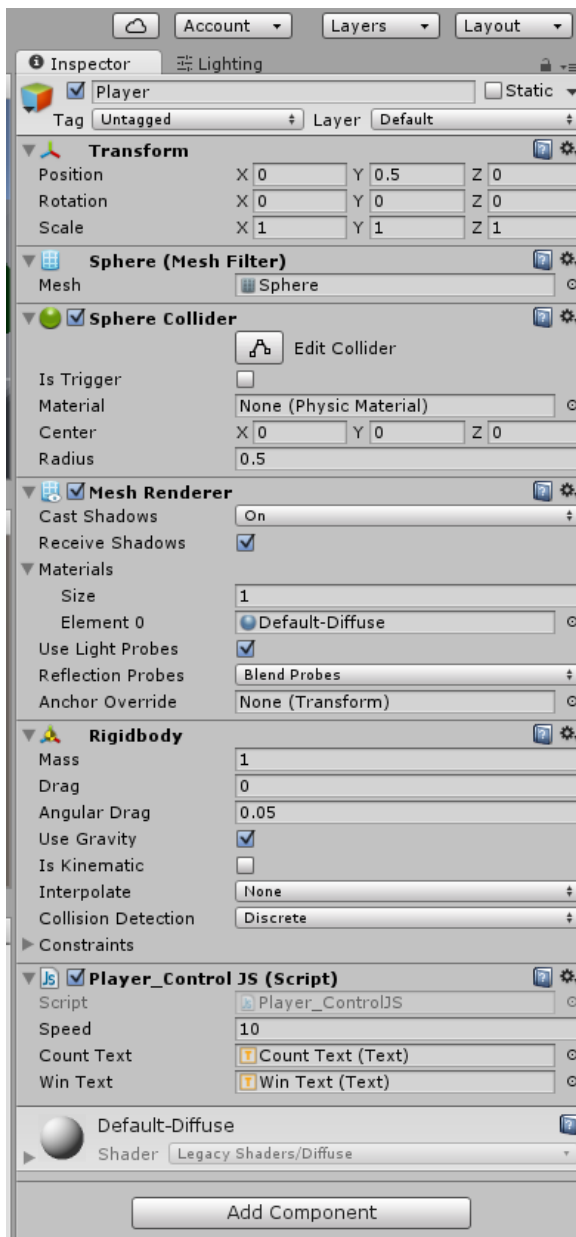


Figure 4: The „inspector“ lists up all properties of the actual object

Above the hierarchy tab there are the modi to move within and edit the scene/objects.

The lightning tab offers global scene settings like applying a so called “skybox”, which simulates the sky and its horizon, and other illuminating properties.

At least the console shows scene or scripting errors which have to be first maintained to play back the scene.

### 2.1.3 Introduction to scripting with the Roll-a-Ball Tutorial

All above mentioned basic parameters can be found in every common 3d platform.

In most cases there is also the option of scripting within the software for simulation purpose.

The difference between them and a game engine is the forced real-time interaction - the gaming purpose - where else the pure CGI applications concentrate on realistic and detailed implementation of the scene objects into a real world film/video clip.

The purpose of scripting in game engines is therefore much higher.

#### How to help yourself in Unity

Because of its big community the Unity team is obviously focusing on keeping the software entry simple.

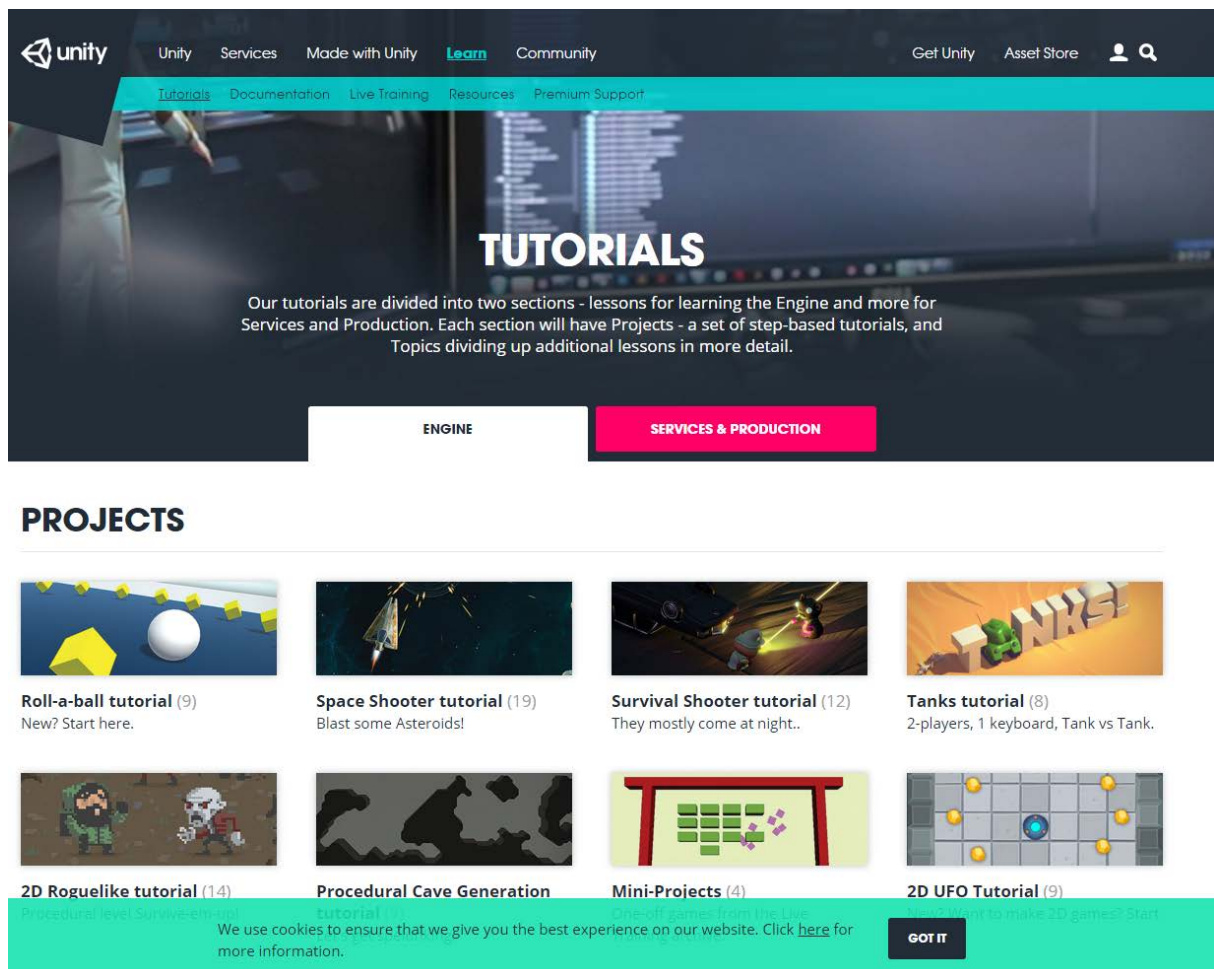


Figure 5: The Unity Homepage offers tutorials to get started

Therefore they have some tutorials for beginner, intermediates and pros on their website. But no matter you are working on- or offline you can use the software reference.

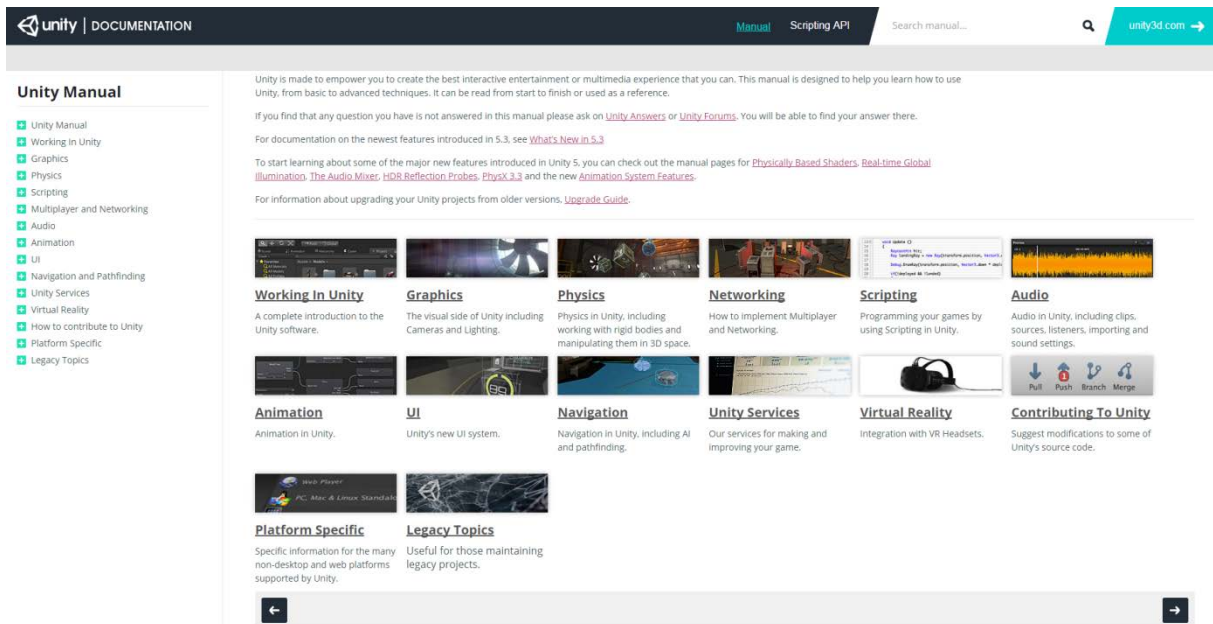


Figure 6: Unity also offers well-documented software references

Because scripting is such an essential part in game engines Unity differentiates between the “Manual” for the CG driven part and a detailed “Scripting API Reference” for helping the users to manipulate interaction behavior of their game.

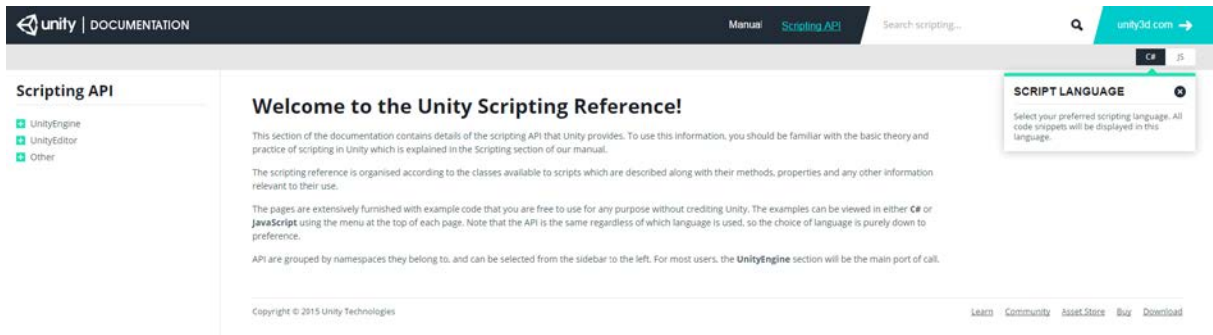


Figure 7: References for C# and JS are available

Because there are more than one scripting languages which can be used in Unity of course every example in the documentation is at least described for “C#” and “JS” sometimes also for “Boo”.

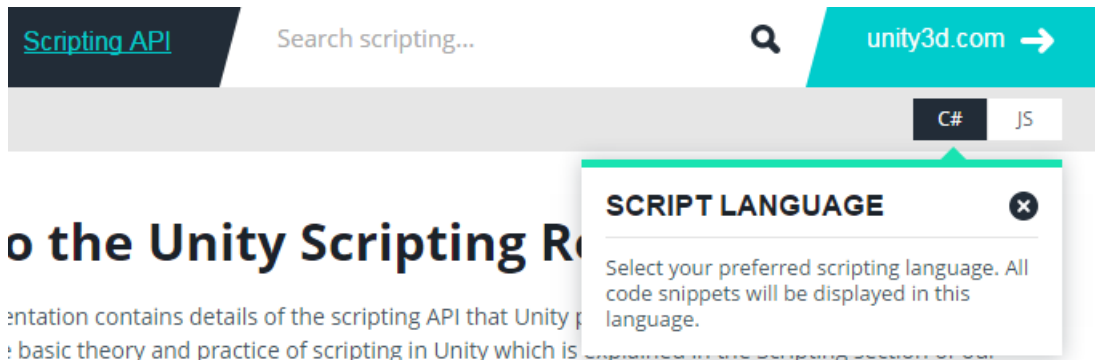


Figure 8: You can change the script language to C#, JS, or sometimes to BOO.

For the purpose of using I/O hardware for VR applications in the medical area for treating chronic/persistent pain the Unity manual is devoting a whole chapter on integrating hardware e.g. HMDs (head mounted displays) like the Oculus rift, which our VR group definitely has to go through soon.

### 2.1.4 The first initial steps to create content for the scene

If a new scene is opened a skybox already appears. If not, by opening the lightning tab via the window menu the skybox can be used.

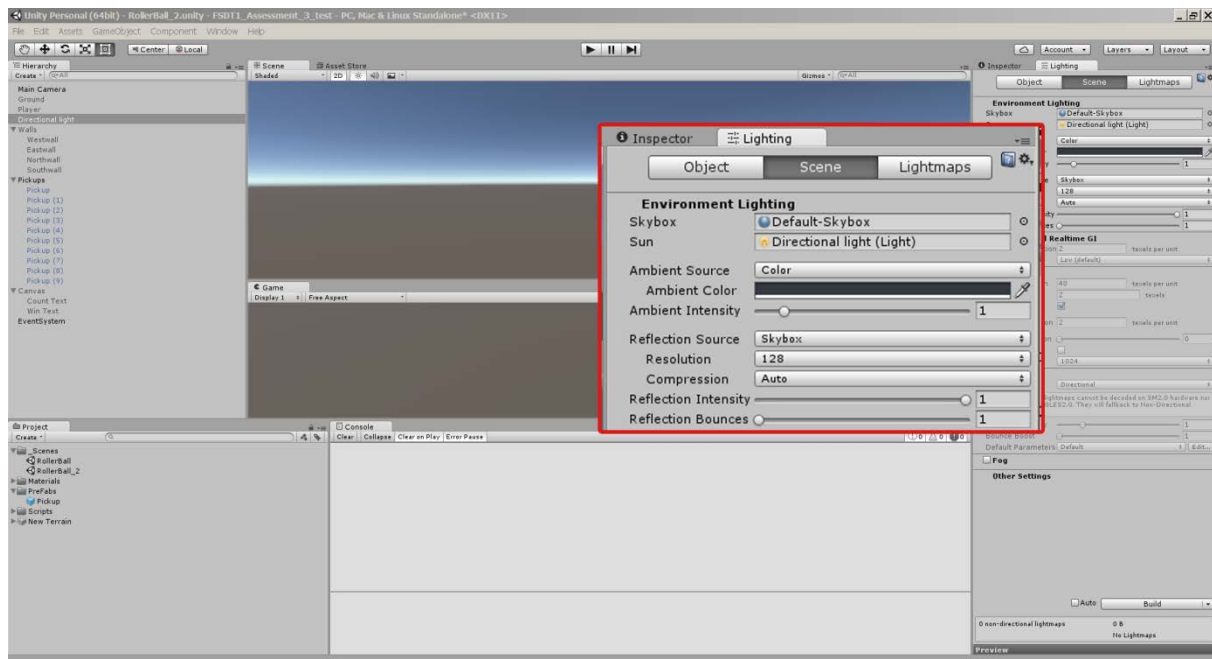


Figure 9: The lightning box

This report only focuses in detail on the scripting part of the Roll-a-ball tutorial, so for a detailed description of creating the virtual scene environment you have to go through the video tutorial itself (<https://unity3d.com/learn/tutorials/projects/roll-ball-tutorial>).

The report enters the tutorial while we already have created all elements and like in the tutorial shown the game is already scripted in C#.

So it will be described what to do to rewrite the scripts for JS with the correct grammar and of course what each code snippet is standing for and manipulating.

Like for all topics also for scripting there is a whole “code academy” – like topic spending lessons on basic-, intermediate- and pro- knowledge beginning with variables & functions up to high skilled lessons for special purposes – of course in both, C# and JS, languages.

<https://unity3d.com/learn/tutorials/topics/scripting>

### 2.1.5 The Roll-a-ball scene

In the following picture we can see that the prepared scene contains a “playground” which is called “plane” in Unity. It has been renamed to “Ground”.

There also is a sphere called “Player”, four scaled cubes grouped called “Walls”, some cubes with an assigned green material grouped to “Pickups” and two text screens called “wintext” and “counttext” both grouped into the “Canvas” group with it needed Event System object.

Of course the scene also has a camera to be able to see the scene and a directional light for lighting the scene.

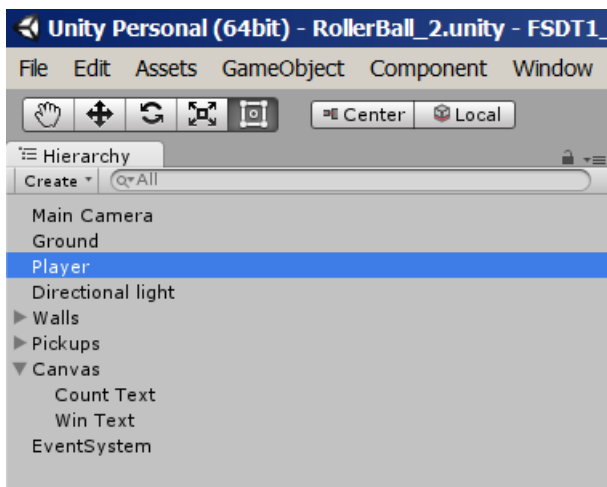


Figure 10: The hierarchy tab shows all objects within a scene

It is now possible to manipulate every single object by writing a script for it and add it as a component like this can be done for materials, rigid bodies and other components the various object types need initially to exist.

All types of components added to an object can be found in the “inspector” tab – in our case - on the right side of the screen.

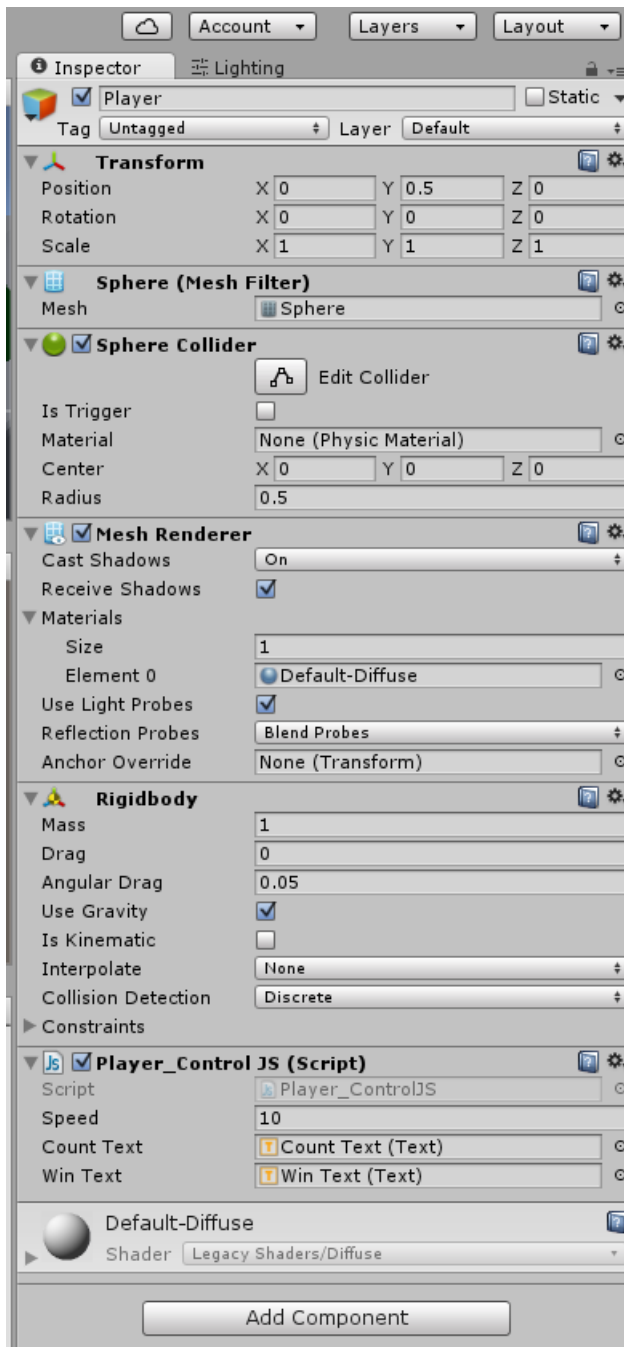


Figure 11: The inspector tab shows all the properties of the “Player” object.

To move our sphere game object called “Player” on the grid/plane there additionally has to be added a “rigid body” component after creating a script as a component for “player”.

A rigidbody makes the object recognizing other objects called “collision” and let the objects behavior after the declared rules configured in the scene physics.

So in the case of “player” the sphere will fall onto the “ground” plane and remain there because of the gravity as the only force configured in the physics at the moment. If the rigid body component of “Player” wouldn’t exist, the sphere wouldn’t collide with the plane, fall through it to an infinite negative z-coordinate.

### 2.1.6 Opening/Editing the script

When adding the “player” script component you can whether choose between a C# or a JS interpreted script. In the scripting topic on the Unity website there is a video chapter about the main differences.

For editing, the script is e.g. opening up in the Visual Studio Editor showing the initial runtime functions “Start” and “Update” and “#pragma strict”.

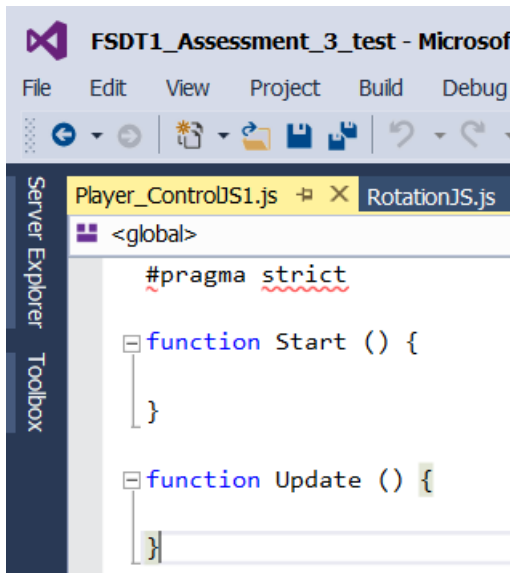


Figure 12: This is an empty JS file in Unity with the constructors Start and Update.

#### #pragma strict:

The usage of pragma is to give the precompiler certain directives how to treat the compiling code. “strict” in this case means that there needs to be strict data types (e.g. integer or float)

#### Functions - “Start ()”, “Update()”, “LateUpdate()” and “FixedUpdate()”:

This functions are set by Unity. While all the initializing code like assigning values to variables have to be coded into the “Start” block, all of the “runtime” code is in the “Update” code block. It gets worked off cyclic on every frame.

When using code in “Fixed Update()” the cycle isn’t every frame but the time you define to be used instead. That means if the script is used for physics it is better to use the “FixedUpdate” not to be dependent to the current used frame rate.

“LateUpdate()” is processed if the whole code in “Update()” and “FixedUpdate()” is already done for the actual cycle. Because of this, the possibility arises to keep the correct processing order. An example of using the “LateUpdate()” will be shown in the “Camera\_ControlJS” component script where the camera position always should get calculated after the every other scene object has finished their tasks per frame.



For more details: <https://unity3d.com/learn/tutorials/modules/beginner/scripting/update-and-fixedupdate>

### 2.1.7 The “Player\_Control JS” script

The initial “FixedUpdate()” block for “player”:

```
#pragma strict

var speed: float;
private var rb: Rigidbody;

function FixedUpdate () {

    var moveHorizontal: float = Input.GetAxis("Horizontal");
    var moveVertical: float = Input.GetAxis("Vertical");

    var movement: Vector3 = new Vector3 (moveHorizontal, 0.0f,
moveVertical);

    rb.AddForce(movement * speed);
}
```

Initially there are two variables generated to catch the direction of the user input. This two direction variables are carrying the information to move the “player” into the positive or negative x – or z – direction. After that the two values get assigned to a declared variable “movement” with the Unity predefined data type Vector3 representing the 3d axes “x”, “y” and “z”.

Moreover we use Rigidbody component class, which is part of the “player” object to declare a variable “rb” and get able to manipulate the physical behavior of our sphere.

The “rigidbody” component is loaded at the beginning of the runtime.

```
function Start() {
    rb = GetComponent.<Rigidbody>();
}
```

Instead of leaving the variable “rb” public like the “speed” which we want to make changeable over the inspector tab, we keep that private and invisible outside the script.

The last code line takes the User Input multiplied with a preconfigured speed value, turns it into a physical property and physics – sensitive rigidbody component which makes the “player” object moving.

Because the script is applied to the object via the component module, there is no need to write the object name before their methods and properties.

Therefore a

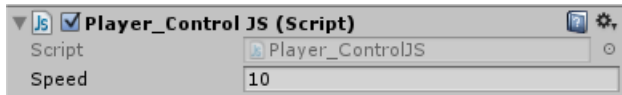
```
Player.rb.AddForce (movement * speed);
```

is not necessary.



Because of the “strict” parameter the compiler needs to have the data types given. Therefore every time a variable gets defined “: datatype” has to be added.

The inspector tab of “player” is now showing the speed property for configuring sensibility of our user input.



### Writing the “colliding” function called OnTriggerEnter:

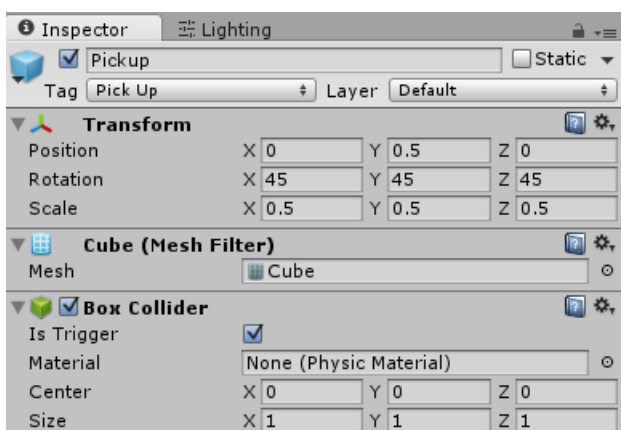
```
function OnTriggerEnter(other: Collider)
{
    if (other.gameObject.CompareTag("Pick Up"))
    {
        other.gameObject.SetActive(false); // Vorsicht! Hier
        vergisst man leicht other.ga.... ! Dann löst sich die Kugel auf
    }
    //Destroy(other.gameObject);
}
```

The function asks the compiler if every other object which “player” is actually colliding with is set as trigger (“TriggerEnter” event) and moreover includes the string “Pick Up”.

If both is correct the other game Object gets hidden. It is very important to write the “other” term, otherwise our ball (sphere) itself gets hidden.

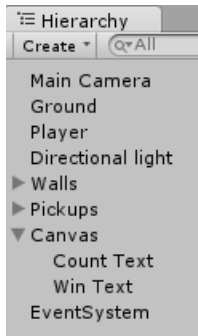
For declaring if an object is equipped with a trigger Event we have to set the “IsTrigger” – flag in the Unity GUI in the inspector tab of the correct Object or we set it directly in our script with the command from the API.

```
public var isTrigger: bool;
```



### Adding a “counting” - function and a winning message

Initially two text objects called “count\_Text” and “Win\_Text” have to be created in Unity.



Both are grouped in a “Canvas” Object with a belonging Event System.

For both a global variable at the beginning in our “Player\_ControlJS” script has to be defined as data type “Text”. Additionally we also need a count variable. Integer is completely sufficient because there is no need for floating point to count to 10.

```
var countText: Text;  
var WinText: Text;  
  
private var count: int;
```

Further, the variables get their initial values in the “Start()” codeblock, where already the “rb” variable exists.

```
function Start() {  
    rb = GetComponent.<Rigidbody>();  
    count = 0;  
    SetCountText();  
    WinText.text = "";  
}
```

The whole counting process depending text screening is controlled by the “SetCountText()” function which is declared as follows in the coming code block.

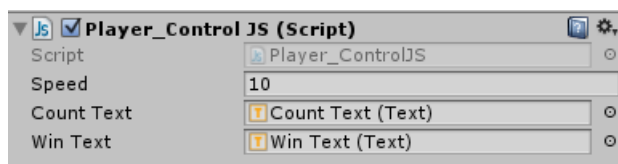
```
function SetCountText ()  
{  
    countText.text = "Count: " + count.ToString();  
    if (count >= 10)  
    {  
        WinText.text = "You Win!!!!";  
    }  
}
```

The “.text” property screens the Text with the additionally string – converted count variable. Furthermore if the “count” variable reaches a value of ten, the “WinText” object shows the “You Win!!!!” string.

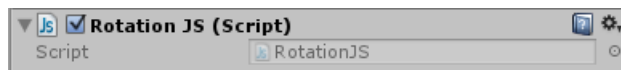
Of course the scoring has to happen in the “OnTriggerEnter()” function as following.

```
function OnTriggerEnter(other: Collider)
{
    if (other.gameObject.CompareTag("Pick Up"))
    {
        other.gameObject.SetActive(false); // Vorsicht! Hier
vergisst man leicht other.ga.... ! Dann löst sich die Kugel auf
        count++;
        //countText.text = "Count: " + count.ToString();
        SetCountText();
    }
    //Destroy(other.gameObject);
}
```

At least the generated Text objects have to get linked via the Unity GUI with our public variables.



### 2.1.8 Rotating the “Pick Up” object cubes



The RotationJS script has to be added as a component whether to all cubes which should get rotating or you can work with a “Prefab” which is nothing more than an editable group object which inherits its properties to the Objects in this group.

Because in this example all cubes exactly should act the same way we can simply use the “Pick Up” Prefab and apply our “RotationJS” script to it.

```
// Use this for initialization
void Start () {
}

// Update is called once per frame
void Update () {
    transform.Rotate(new Vector3(15, 30, 45) * Time.deltaTime);
}
```

For this rotation simply the “transform” component with its “Rotate” property can be three – dimensional manipulated via a Vector3 object controlling all 3 rotational axis and change them over a certain period of time.

### 2.1.9 Following the “Player” object with the camera keeping the same distance overall

At the beginning there was a short comparison of the various “Update()” functions and their differences. If the camera has to get the exact movement, then this knowledge is absolutely necessary for doing that.

Like mentioned earlier the “LateUpdate()” function should be used for processes which have to wait until the scene is fully processed from frame to frame before to do their calculations. Camera movement in most cases is included into that.



When creating a new script as a component for the “Main Camera” object the “Update()” function should get changed to the “LateUpdate()” function to work properly in the future.

```
public GameObject player;
private Vector3 offset;

// Use this for initialization
void Start ()
{
    offset = transform.position - player.transform.position;
    //Offset als Fixwert zw. Kamera und Ball berechnen
}

// LateUpdate is called once per frame after all Update procedures
have been done

void LateUpdate () {
    transform.position = player.transform.position + offset;
    //Kamerawert jedes Frame neu berechnen zusammengesetzt aus v
    ariierendem Ballwert und Offset Fixwert
}
```

Initially again there has to be generated a public variable for linking the “Player” object and movement values to our “Camera\_ControlJS” script.

For the internal calculations we need a private “offset” variable which initially in the “Start()” function gets calculated and at the end of the frame in the “LateUpdate()” function is used to keep the same distance from frame to frame.

## 5.2 Conclusion

Because of the big community, the plenty of tutorials and both great Unity manual as well as scripting API documentation entering Unity and learning fast seems to be really simple.

It is really catchy to use JS with the possibility of combining it with computer – graphical creativity and animation. The effect of “learning by doing” or “guerilla scripting” after this first contact seems to be well given is followed by a big outcome within a short time.

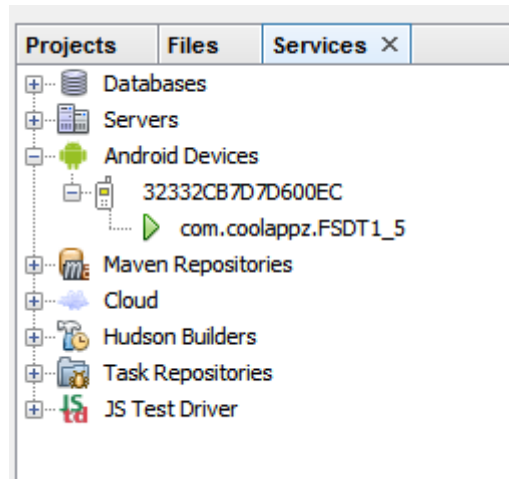
## 6. Create a most simple Mobile Web App with the cross-platform tool Cordova.

*by Christian Nasel*

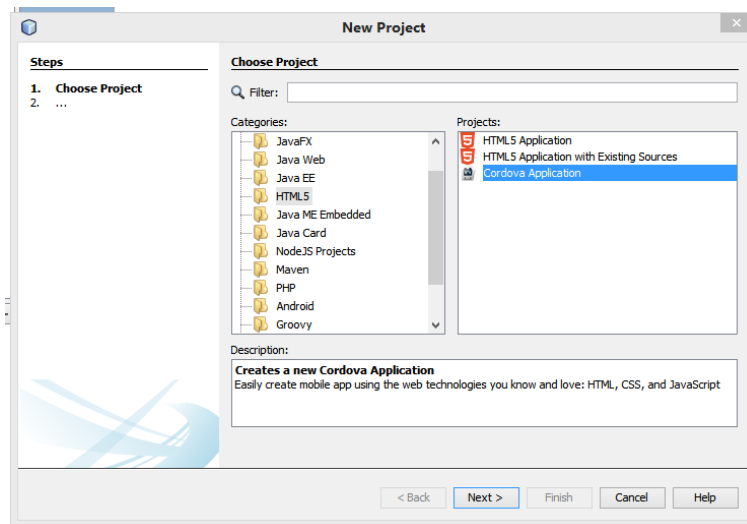
### 6.1 Setup of the IDE

This report describes the implementation of a Cordova-application in NetBeans, compiling it and load it to an Android mobile. Following the various steps listed below:

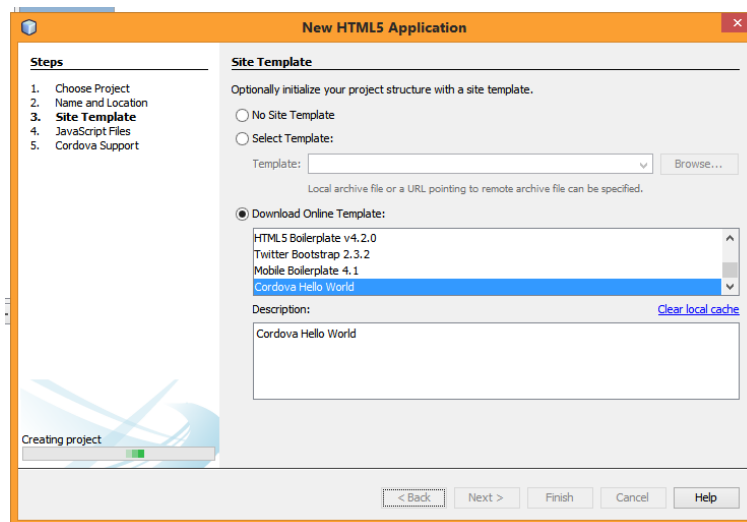
- Install NetBeans IDE version greater 8.0.0 with HTML5 package since some of the used plugins require these versions. It is assumed that you are already familiar with the NetBeans IDE.
- Next install node.js from: '<https://nodejs.org/en/>'. You may test if node.js is already available on your system typing in: '`node --version`' at the command prompt of the system console.
- Install git for Your platform from: '<https://git-scm.com/downloads>'. Again, you can test for existing git-installations by entering: '`git --version`' at the command prompt of the system console.
- Download android-studio (or at least android sdk and avd) for your platform from: '<http://developer.android.com/sdk/index.html>' and install the package on your computer.
- Open the android sdk in android studio and make sure that you possess the right API for Your mobile. In most cases there will be no conflict, but in the presented project API 10 that corresponds to Android 2.3.3 was needed. Simply check the box left to the appropriate API and then click '*finish*'. The requested downloads will start automatically.
- Run: '`npm install -g cordova`' at the command prompt of the system console. This installs the cordova package
- Start the NetBeans IDE. Installing the '*NodeJS*', the '*Nbandroid*', the '*Android Gradle Support*' and the '*RunMyScript*'-plugins, which will make Your life easier. To get these navigate to '`main-menu\Tools\Plugins`' and install them. Especially, '*Nbandroid*' is necessary since it adds a service to NetBeans IDE that allows to check whether the android debug bridge (adb) is functional or not.



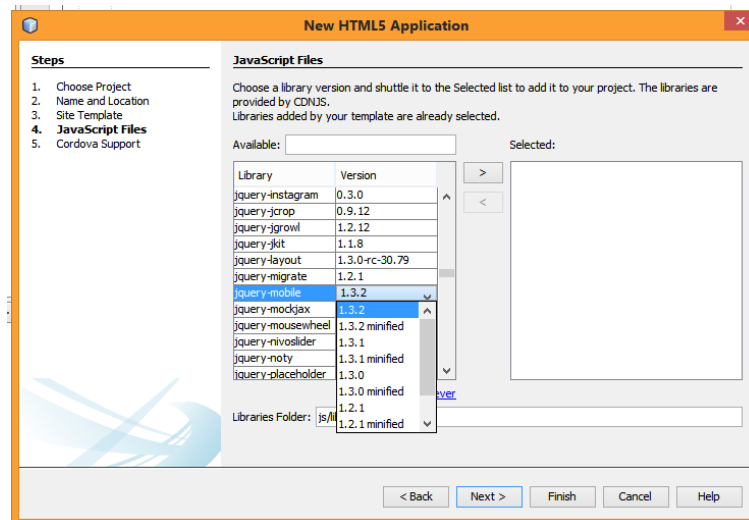
- Select: 'main-menu\File\New Project ...'. Create a new HTML5 Cordova Application-project.:



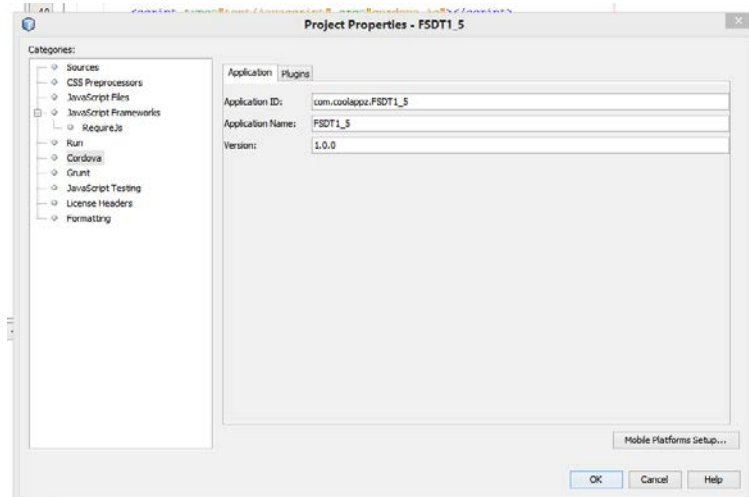
- Next select the 'Cordova Hello World'-template for the current project.:



- Afterwards select which libraries you want to integrate in your application. The jQuery and jQuery mobile (minified releases) were chosen for this application. These libraries will be local in your project.

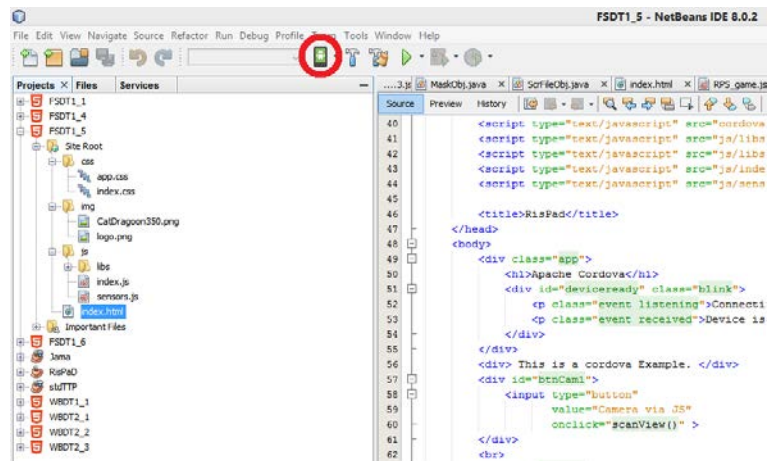


- Finish the template creation and right click the project in the project navigator of Your NetBeans IDE, select 'properties' and click on Cordova. Here you can set the android path and app-name. Also you can select and deselect the cordova plugins for the management of the mobile functionalities. Selecting: Mobile Platforms Setup enables to set the path to the android sdk on your hard disk.



- Connect Your mobile to your computer via USB, but before this enable USB-debug mode on the phone. Otherwise the adb will not work and the direct upload of your app to the phone is not possible.
- Set the target device to: Cordova (Android device) in the main tool-bar in the IDE (red circle).





- Well, you are ready to start with the app. The figure above nicely displays the tidy site map structure of our project.

## 6.2 Building the Application

The aim of this project was to trigger a camera-call by the proposed app, which after taking a photo returns the URI of the image to the app, where further processing would be possible. There are few things, which should be considered before you try to launch any app at your mobile. Apache Cordova follows a straight forward policy in updating the cordova plug-ins. During the build the IDE will always try to connect to the update repository and in case of any failure the respective plugin will be dismantled from your application. Nevertheless, your app would work, but the plugin will no longer exist in the application. Therefore, one has to get rid of the auto-updater manually before the build. Simply comment the 'update-plugins' tag in build.xml out and do not forget to clean up the dependencies in the residual build. However, most of our readers will not encounter any update-problem, but in our case an incompatibility problem due to our relatively old API-version occurred. For an example for the manual clean up look to the graph below.:

```

<!--
<target name="update-plugins">
  <plugintask/>
</target>
-->

<target name="update-android">
  <echo level="info" message="${cordova.command} prepare android"/>
  <exec executable="${cordova.command}" resolveexecutable="true" searchpath="true" failonerror="true">
    <env key="${cordova.path.key}" path="${cordova.path.value}:${android.sdk.home}/tools:${android.sdk.home}/platform-tools">
    <env key="JAVA_HOME" path="${jdk.home}"/>
    <arg value="prepare"/>
    <arg value="android"/>
  </exec>
</target>

<target name="update-ios">
  <echo level="info" message="${cordova.command} prepare ios"/>
  <exec executable="${cordova.command}" resolveexecutable="true" searchpath="true" failonerror="true">
    <env key="${cordova.path.key}" path="${cordova.path.value}:${android.sdk.home}/tools:${android.sdk.home}/platform-tools">
    <env key="JAVA_HOME" path="${jdk.home}"/>
    <arg value="prepare"/>
    <arg value="ios"/>
  </exec>
</target>

<target name="rebuild-android" depends="clean-android,build-android"/>
<!--target name="build-android" depends="create-android,update-plugins"-->
<target name="build-android" depends="create-android">
  <echo level="info" message="${cordova.command} -d build android"/>
  <exec executable="${cordova.command}" resolveexecutable="true" searchpath="true" failonerror="true">
    <env key="${cordova.path.key}" path="${cordova.path.value}:${android.sdk.home}/tools:${android.sdk.home}/platform-tools">
    <arg value="-d"/>
    <arg value="build"/>
    <arg value="android"/>
  </exec>
</target>

<!--target name="sim-android" depends="create-android,update-plugins"-->
<target name="sim-android" depends="create-android">
  <echo level="info" message="${cordova.command} -d ${android.target.device} android"/>
  java -jar "${cordova.bin}" --help

```

Once the project is cleaned, the appropriate minimal – android SDK version correctly matching our mobile android version has to be chosen. Normally, no problem will occur here, but, as said, the old android API caused some problems. To fix this problem the minSdkVersion and targetSdkVersion were set to 10 (our old API) and 21 (API for avd-simulations and newer mobiles) in the androidManifest.xml documents (run-time and debug scripts), respectively. Besides, here you can also check for your required permissions.:

```

<?xml version='1.0' encoding='utf-8'?>
<manifest android:hardwareAccelerated="true" android:versionCode="10000" android:versionName="1.0"
  <supports-screens android:anyDensity="true" android:largeScreens="true" android:smallScreens="true" android:xlargeScreens="true" />
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
  <application android:hardwareAccelerated="true" android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:configChanges="orientation|keyboardHidden|keyboard|screenSize" android:label="@string/launcher_name">
      <intent-filter android:label="@string/launcher_name">
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
  <uses-sdk android:minSdkVersion="10" android:targetSdkVersion="21" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.VIBRATE" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-permission android:name="android.permission.RECORD_AUDIO" />
  <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
  <uses-permission android:name="android.permission.READ_PHONE_STATE" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.RECORD_VIDEO" />
</manifest>

```

Next, the layout of our app is created in the index.html files of our app. Two buttons for the two different program solutions were defined, some text, a new background image and, of course, a mobile log-console. Latter object is useful to get any clue about the current state of the app.:

```

<body>
  <div class="app">
    <h1>Apache Cordova</h1>
    <div id="deviceready" class="blink">
      <p class="event listening">Connecting to Device</p>
      <p class="event received">Device is Ready</p>
    </div>
    <div> This is a cordova Example. </div>
    <div id="btnCam1">
      <input type="button"
        value="Camera via JS"
        onclick="scanView()" >
    </div>
    <br>
    <div id="btnCam2">
      <input type='button'
        value='Camera via jQuery'>
    </div>

    <textarea id='txtLog' >app.console: &#013;&#010;</textarea>
  </body>

```

The styles of buttons: 'btnCam1' and 'btnCam2' as well as the log-textarea were defined in a separate css-file named app.css . Finally, the same camera functionality was implemented in two different ways, just to test stability of jQuery library functions. Firstly, the new HTML5-event attribute: 'onclick' was used for 'btnCam1', which triggers a call to a JS function previously defined in the sensors.js file.:

```

function scanView() {
  nCnt++;
  navigator.camera.getPicture(
    function(imageURI) {
      $('#txtLog').append('URI: ').append(imageURI).append('\r');
      $('#txtLog').scrollTop($('#txtLog')[0].scrollHeight - $('#txtLog').height());
    },
    function(err) {
      $('#txtLog').append('no URI ').append(err).append('\r');
      $('#txtLog').scrollTop($('#txtLog')[0].scrollHeight - $('#txtLog').height());
    },
    { quality: 50, destinationType: Camera.DestinationType.FILE_URI }
  );
  $('#txtLog').append('JS ').append(nCnt).append(' ');
}

```

Secondly, a jQuery event.listener after setup of the DOM tree was set for 'btnCam2', just to train this type of code as well.:

```

$(function(){
  $('#txtLog').textinput("option", "autogrow", false);
  // .textarea($('#txtLog').append('set autogrow: ' + $('#input').textinput("option", "autogrow") );
  $('#txtLog').append('set listener').append('\r');
  $('#txtLog').scrollTop($('#txtLog')[0].scrollHeight - $('#txtLog').height());

  $('#btnCam2').click(function() {
    navigator.camera.getPicture(
      function(imageURI) {
        $('#txtLog').append('URI: ').append(imageURI).append('\r');
        $('#txtLog').scrollTop($('#txtLog')[0].scrollHeight - $('#txtLog').height());
      },
      function(err) {
        $('#txtLog').append('no URI ').append(err).append('\r');
        $('#txtLog').scrollTop($('#txtLog')[0].scrollHeight - $('#txtLog').height());
      },
      { quality: 50, destinationType: Camera.DestinationType.FILE_URI }
    );
    $('#txtLog').append('jQuery ').append(nCnt).append(' ');
  });
});

```

As can be seen from the code examples the navigator.camera.getPicture –function takes three parameters, where the first and the second define methods to handle the success or fail of the function call, while the third parameter sets the response type. In this context, one should

take care for the proper order of script-file calls, because functions called at run time not yet loaded will fail, anyway. Example of a rational call order.:

```
<link rel="stylesheet" type="text/css" href="css/app.css">
<link rel="stylesheet" href="js/libs/jquery-mobile/jquery.mobile.min.css"/>
<link rel="stylesheet" type="text/css" href="css/index.css">

<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="js/libs/jquery/jquery.min.js"></script>
<script type="text/javascript" src="js/libs/jquery-mobile/jquery.mobile.min.js"></script>
<script type="text/javascript" src="js/index.js"></script>
<script type="text/javascript" src="js/sensors.js"></script>
```

Ok. Build the application with the 'run' button in the main-toolbar of the IDE and see how it automatically launches at the mobile. Done. Good fun.