FREE eBook

LEARNING nativescript

Free unaffiliated eBook created from
Stack Overflow contributors.

#nativescrip

t

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: nativescript

It is an unofficial and free nativescript ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official nativescript.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with nativescript

Remarks

Nativescript is a highly performant cross-platform mobile app runtime, which allows you to target iOS and android (with windows in the pipeline) using web technologies (JS and html). It was created with a number of key aims:

- Visually Performant: no UI Jank even on android you have buttery smooth fps
- Extensible: you have access to all native APIs, to create easy cross platform plugins
- Completely native UI
- Highly Integrated with Typescript and Angular 2
- Open Source, with strong corporate backing from Telerik

Examples

Installation or Setup

Detailed instructions on getting Nativescript set up or installed.

The following examples show the required steps to set up a Windows or OSX system and then sign post to troubleshooting guides in case you have any trouble.

In addition, there are examples of how to set up recommended workflows, IDEs and emulators.

macOS

1. Ensure you have the most recent Node.js LTS installed. If you use Homebrew this can be done with `brew install node4-lts`.
2. Open Terminal and type `npm install -g nativescript`. If you get an `EACCES` error, use `sudo npm install -g nativescript`.
3. In the command prompt type `ruby -e "$(curl -fsSL https://www.nativescript.org/setup/mac)"` . (This might take a while.)
4. To verify that the above has worked, type `tns doctor` in Terminal.
5. If there are any errors, follow up with the troubleshooting guide.

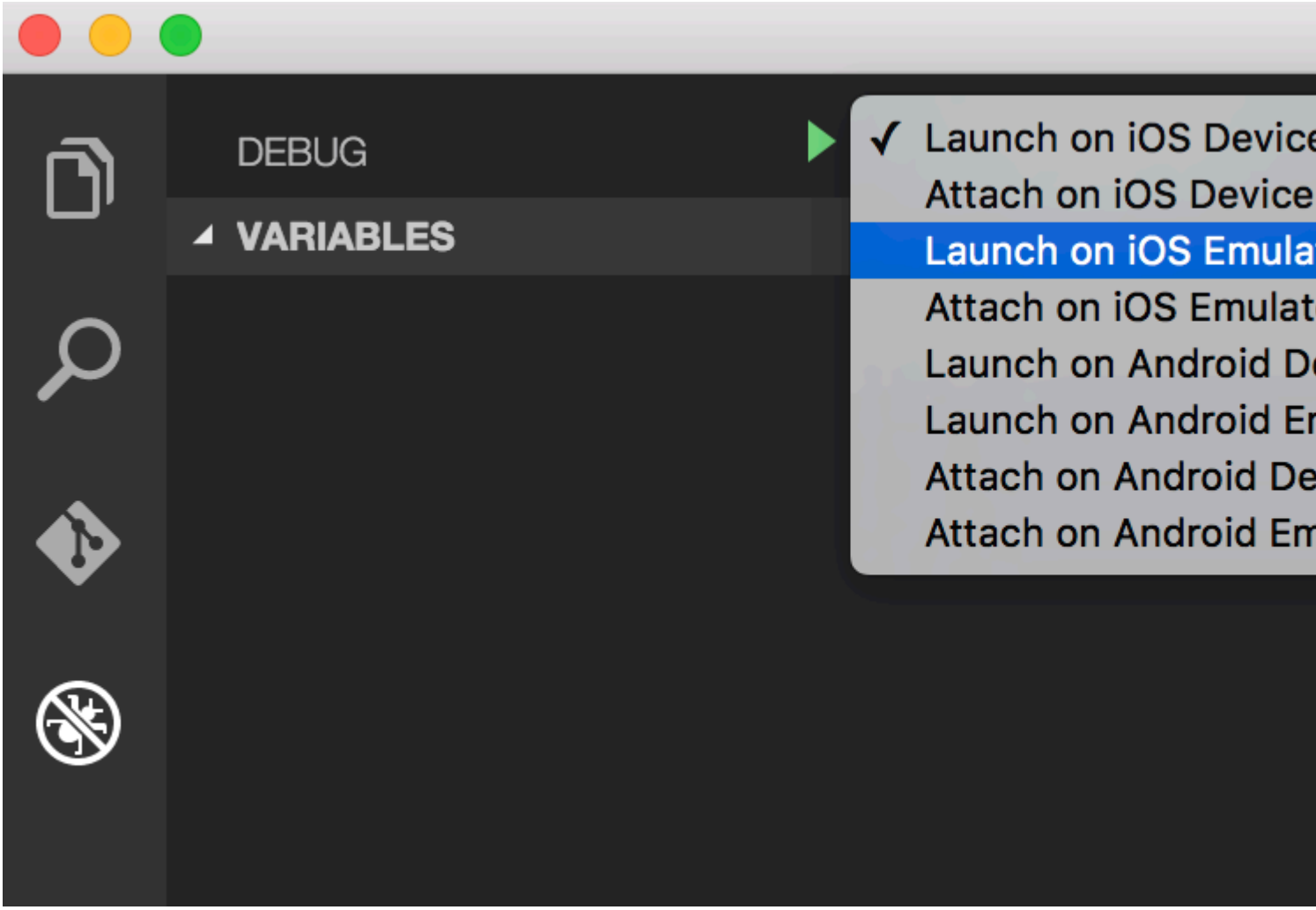
Windows

1. Ensure you have the latest nodeJS LTS installed
2. Open command prompt and type `$ npm install -g nativescript`
3. In the command prompt type `$ @powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-object net.webclient).DownloadString('https://www.nativescript.org/setup/win'))"` - this might take a while
4. To verify the above has worked, type `$ tns doctor` in command prompt (your cmd)
5. If there are any errors, follow up with the troubleshooting guide

Using Visual Studio Code for NativeScript development

Visual Studio Code is an open-source and feature-rich code editor from Microsoft. To set it up it for NativeScript development, open the Command Palette (`F1` or ⌘+`Shift`+`P`) and type `ext install NativeScript`.

Once the NativeScript extension is installed, the debugger should allow you to set breakpoints in your code. When a device is connected or an emulator is running, you can start your app from the Debug tab.



Your first Hello World program

```
$ mkdir hello-world
$ cd hello-world
$ tns create hello-world --ng
$ tns platform add android #You can only add ios on an OSX machine
```

Then ensure you have a device connected or an emulator running (if you don't, the default emulator should start or an error will be raised. I would recommend genymotion for android).

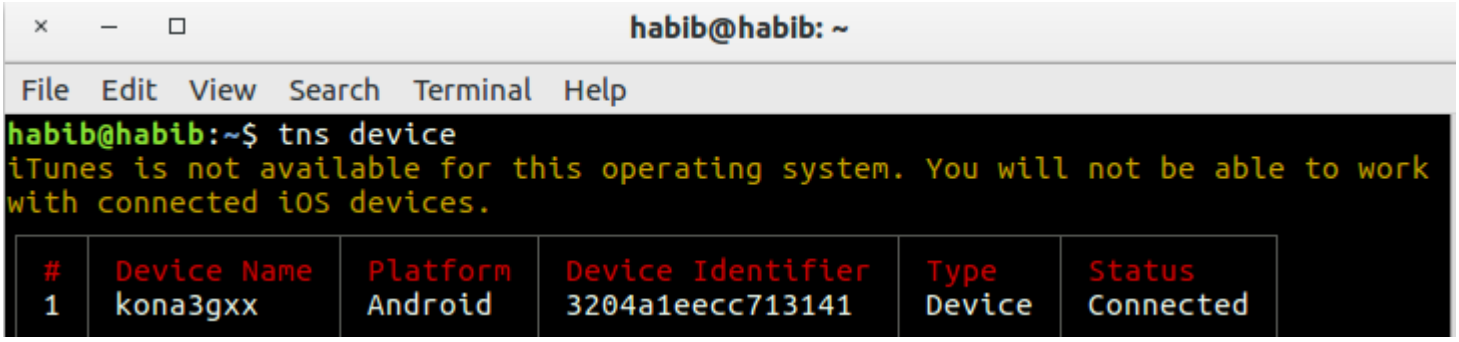
```
$ tns run android
```

If you want to use the default android emulator, add the `--emulator` flag.

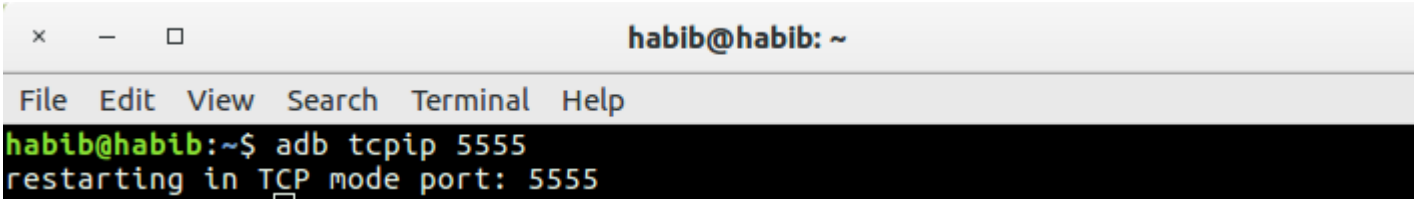
As of tns 2.5 livesync is now the default action for `tns run <platform>`, which will automatically re-compile when you save file changes. This can dramatically improve your development time, however, if you make changes to your plugins, you will need to recompile properly.

How to Debug nativescript-android App over WiFi (without Root)

1-You need to connect your device to your computer via USB cable. Make sure USB debugging is working. You can check if it shows up when running `adb devices`(or `tns device`).



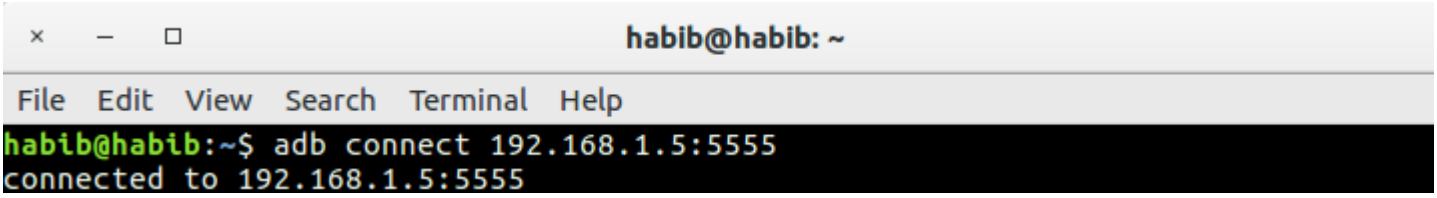
2-Run `adb tcpip 5555`



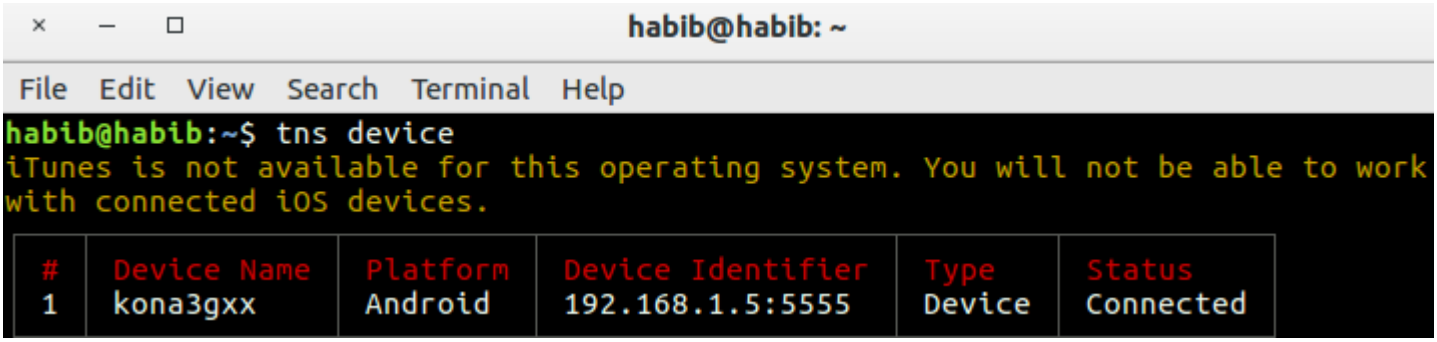
3-Disconnect your device (remove the USB cable).

4-Go to the Settings -> About phone -> Status to view the IP address of your phone.

5-Run `adb connect <IP address of your device>:5555`



6-If you run `adb devices` (or `tns device`) again, you should see your device.



7- Now you can use `tns run android` , `tns livesync android` commands.

NOTES :

1-when WiFi network changes you do not have to repeat steps 1 to 3 (these set your phone into wifi-debug mode). You do have to connect to your phone again by executing steps 4 to 6.

2-Android phones lose the wifi-debug mode when restarting. Thus, if your battery died, you have to start over. Otherwise, if you keep an eye on your battery and do not restart your phone, you can live without a cable for weeks!

WARNING :

leaving the option enabled is dangerous, anyone in your network can connect to your device in debug, even if you are in data network. Do it only when connected to a trusted Wi-Fi and remember to disconnect it when done!

reference:

1-Norman Peitek. 2014. How to Debug Your Android App over WiFi (without Root!). [ONLINE] Available at: https://futurestud.io/blog/how-to-debug-your-android-app-over-wifi-without-root. [Accessed 8 August 2016].

2-usethe4ce. 2012. Run/install/debug Android applications over Wi-Fi?. [ONLINE] Available at: http://stackoverflow.com/a/10236938/4146943. [Accessed 8 August 2016].

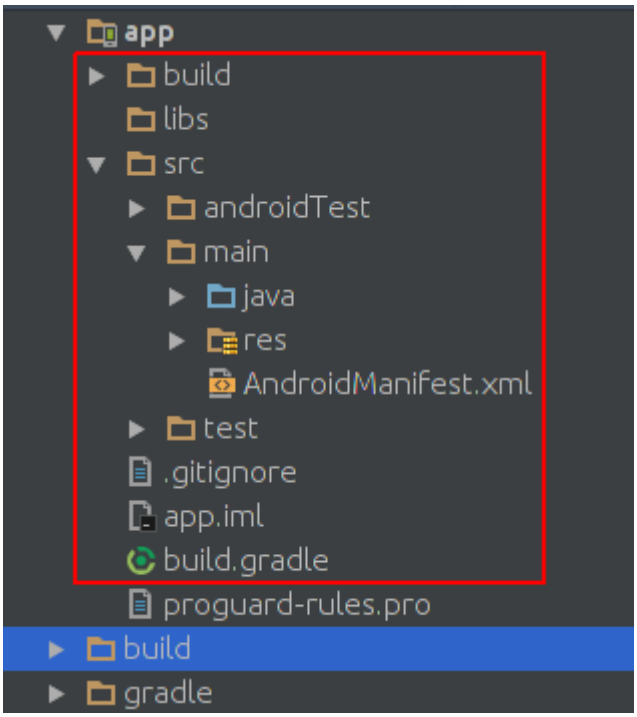
Read Getting started with nativescript online: https://riptutorial.com/nativescript/topic/921/getting-started-with-nativescript

Chapter 2: Accessing native apis

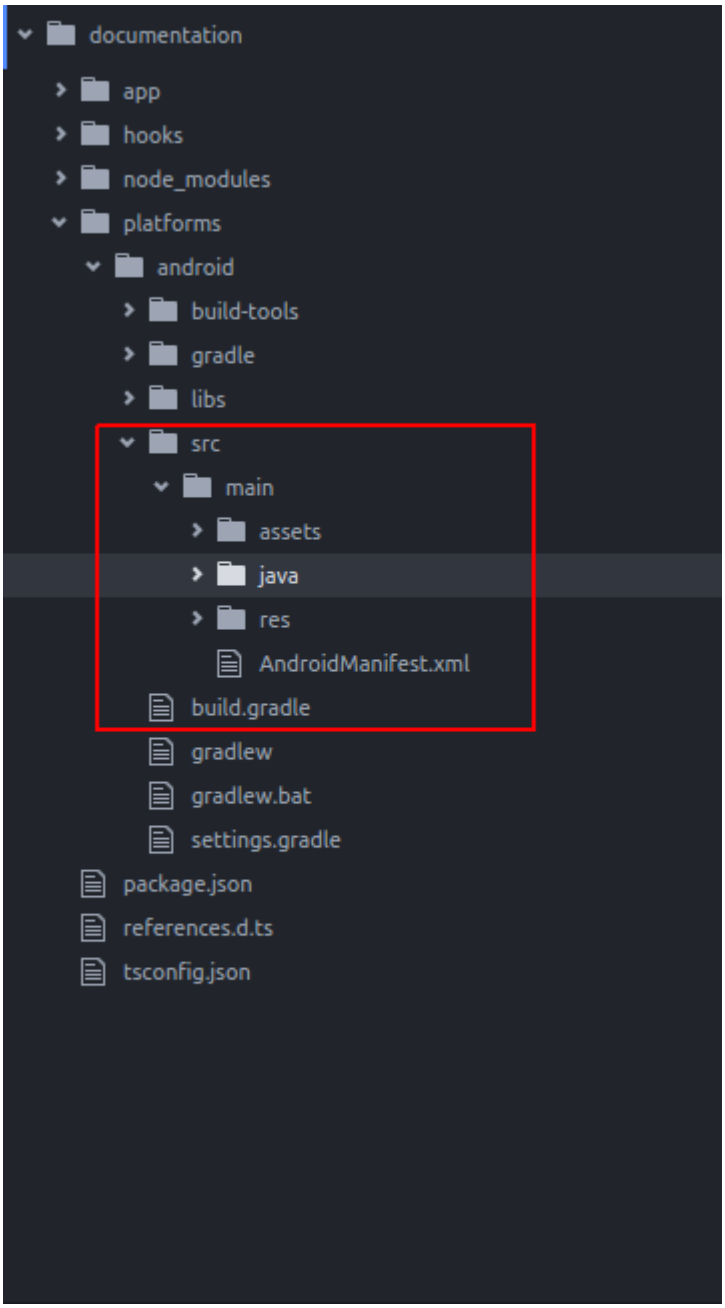
Examples

Write java code in nativescript and use it directly in javascript

This is the image of project structure in Android studio:

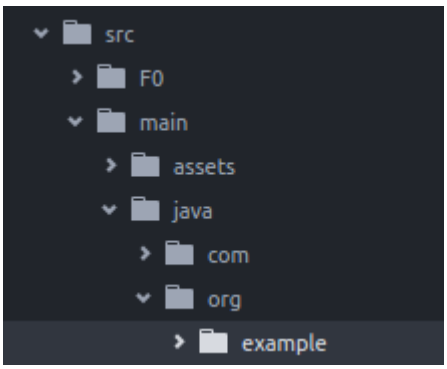


This is the image of project structure of nativescript project:



As you see they are same. so we can write java code in nativescript as we write in android studio.

We want to Add Toast to the default app of nativescript. after creating a new nativescript project create a directory the `java/org/example` directory like this:



create a new `MyToast.java` file in `example` directory;

MyToast.java:

```
package org.example;

import android.widget.Toast;
import android.content.Context;


public class MyToast{

    public static void showToast(Context context,String text ,String StrDuration ){
      int duration;
      switch (StrDuration){
          case "short":
              duration = Toast.LENGTH_SHORT;
              break;
          case "long":
              duration = Toast.LENGTH_LONG;
              break;
      }
        Toast.makeText(context,text, Toast.LENGTH_SHORT).show();
    }
}
```

Notes: don't forget the package name;

app.component.ts:

```
import {Component} from "@angular/core";
let application = require("application");

declare var org:any;
@Component({
    selector: "my-app",
    templateUrl: "app.component.html",
})
export class AppComponent {
    public counter: number = 16;

    public get message(): string {
        if (this.counter > 0) {
            return this.counter + " taps left";
        } else {
            return "Hoorraaay! \nYou are ready to start building!";
        }
    }

    public onTap() {
        this.counter--;
        org.example.MyToast.showToast(application.android.context,"You pressed the
button","short");
    }
}
```

now when you press the button it will show a toast;

Notes:

1. showToast function accepts context to pass it to `Toast.makeText` an we passed a context to it in this way :`application.android.context`
2. typescript doesn't know what `org`is, so we declared it: `declare var org:any;`

use native apis directly in javascript

We want to add Toast to nativescript default app.

```
import {Component} from "@angular/core";
let application = require("application");

declare var android:any;

@Component({
    selector: "my-app",
    templateUrl: "app.component.html",
})
export class AppComponent {
    public counter: number = 16;

    public get message(): string {
        if (this.counter > 0) {
            return this.counter + " taps left";
        } else {
            return "Hoorraaay! \nYou are ready to start building!";
        }
    }

    public onTap() {
        this.counter--;
        this.showToast("You pressed the button","short");
    }



    public showToast(text:string ,StrDuration:string ):void{
      let duration:number;
      switch (StrDuration){
          case "short":
              duration = android.widget.Toast.LENGTH_SHORT;
              break;
          case "long":
              duration = android.widget.Toast.LENGTH_LONG;
              break;
      }
        android.widget.Toast.makeText(application.android.context,text,
android.widget.Toast.LENGTH_SHORT).show();
   }
}
```

for creating toast we should call `Toast.makeText` and it's in the `android.widget.Toast` package. `Toast.makeText` accepts context as first argument and we can get the context in nativescript in this way:`application.android.context`

Read Accessing native apis online: https://riptutorial.com/nativescript/topic/5188/accessing-native-apis

Chapter 3: Displaying data as list (using Repeater, ListView or *ngFor for {N}+Angular-2 apps)

Remarks

> Note: Don't use Repeater in {N}+Angular-2 applications! The *ngRepeat is obsolete directive in Angular-2. When you need to display repeating item patterns use either ListView or *ngFor structural directive.

Examples

Using Repeater module to display data (NativeScript Core)

page.xml

```
<Page xmlns="http://schemas.nativescript.org/tns.xsd" navigatingTo="navigatingTo">
  <Repeater items="{{ myItems }}">
      <Repeater.itemTemplate>
        <Label text="{{ title || 'Downloading...' }}" textWrap="true" />
      </Repeater.itemTemplate>
  </Repeater>
</Page>
```

page.ts

```
import { EventData, Observable } from "data/observable";
import { Page } from "ui/page";

let viewModel = new Observable();
var myItems = [ {title: "Core Concepts"},
                {title: "User Interface"},
                {title: "Plugins"},
                {title: "Cookbook"},
                {title: "Tutorials"} ];

export function navigatingTo(args: EventData) {
    var page = <Page>args.object;

    viewModel.set("myItems", myItems);

    page.bindingContext = viewModel;
}
```

Using Repeater module with ObservableArray (NativeScript Core)

page.xml

```
<Page xmlns="http://schemas.nativescript.org/tns.xsd" navigatingTo="navigatingTo">
  <Repeater items="{{ myItems }}">
      <Repeater.itemTemplate>
        <Label text="{{ title || 'Downloading...' }}" textWrap="true" class="title" />
      </Repeater.itemTemplate>
  </Repeater>
</Page>
```

page.ts

```
import { EventData, Observable } from "data/observable";
import { ObservableArray } from "data/observable-array";
import { Page } from "ui/page";

let viewModel = new Observable();
let myItems = new ObservableArray({title: "Core Concepts"}, {title: "User Interface"}, {title:
"Plugins"}, {title: "Cookbook"}, {title: "Tutorials"});

export function navigatingTo(args: EventData) {

    var page = <Page>args.object;
    viewModel.set("myItems", myItems);

    // The Repeater will be updated automatically when new item is pushed.
    myItems.push({title:"Publishing"});

    page.bindingContext = viewModel;
}
```

Using ListView module with ObservableArray (NativeScript Core)

page.xml

```
<Page xmlns="http://schemas.nativescript.org/tns.xsd" navigatingTo="navigatingTo">
  <ListView items="{{ myItems }}" itemTap="listViewItemTap">
      <ListView.itemTemplate>
        <Label text="{{ title || 'Downloading...' }}" textWrap="true" class="title" />
      </ListView.itemTemplate>
  </ListView>
</Page>
```

page.ts

```
import { EventData, Observable } from "data/observable";
import { ObservableArray } from "data/observable-array";
import { Page } from "ui/page";
import { ItemEventData } from "ui/list-view";

import frameModule = require("ui/frame");

let viewModel = new Observable();
let myItems = new ObservableArray(  {title: "Core Concepts"},
                                    {title: "User Interface"},
                                    {title: "Plugins"},
                                    {title: "Cookbook"},
                                    {title: "Tutorials"}  );
```

```
export function navigatingTo(args: EventData) {

    var page = <Page>args.object;
    viewModel.set("myItems", myItems);

    // ListView will be updated automatically when new item is pushed.
    myItems.push({title:"Publishing"});

    page.bindingContext = viewModel;
}

export function listViewItemTap(args:ItemEventData) {
    var itemIndex = args.index;

    // example how to navigate details-page & pass the tapped item context
    // frameModule.topmost().navigate({
    //     moduleName: "./details-page",
    //     context: myItems.getItem(itemIndex);
    // });
}
```

Using ListView to display data (NativeScript + Angular-2)

creating-listview.component.html

```
<ListView [items]="countries" (itemTap)="onItemTap($event)">
    <template let-country="item" let-i="index">
        <StackLayout orientation="horizontal">
            <Label [text]='(i + 1) +".) "' ></Label>
            <Label [text]='country.name'></Label>
        </StackLayout>
    </template>
</ListView>
```

creating-listview.component.ts

```
import { Component, ChangeDetectionStrategy, Input }  from "@angular/core";

class Country {
    constructor(public name: string) { }
}

var europianCountries = ["Austria", "Belgium", "Bulgaria", "Croatia", "Cyprus", "Czech
Republic",
"Denmark", "Estonia", "Finland", "France","Germany", "Greece", "Hungary", "Ireland", "Italy",
"Latvia", "Lithuania", "Luxembourg", "Malta", "Netherlands","Poland", "Portugal", "Romania",
"Slovakia",
"Slovenia","Spain", "Sweden", "United Kingdom"];

@Component({
    selector: "creating-listview",
    styleUrls:["./creating-listview.component.css"],
    templateUrl: "./creating-listview.component.html",
    changeDetection: ChangeDetectionStrategy.OnPush
})

export class CreatingListViewComponent {
```

```
    public countries: Array<Country>;

    constructor() {
        this.countries = [];

        for (var i = 0; i < europianCountries.length; i++) {
            this.countries.push(new Country(europianCountries[i]));
            }
    }

    public onItemTap(args) {
        console.log("Item Tapped at cell index: " + args.index);
    }
}
```

Using *ngFor Structural Directive to display data (nativeScript + Angular-2)

ngfor.component.html

```
<StackLayout>
    <Label *ngFor="let item of items" [text]="item"></Label>
</StackLayout>
```

ngfor.component.ts

```
import { Component } from "@angular/core";

var dataItems = ["data-item 1", "data-item 2", "data-item 3"]

@Component({
    selector: 'ngfor-component',
    styleUrls:["./ngfor.component.css"],
    templateUrl: "./ngfor.component.html",
})

export class NgForComponent {
    public items:Array<string> = [];

    constructor(){
        this.items = dataItems;
    }
}
```

Using Repeater with Callbacks (JavaScript)

page.js

```
var context = {
items: [
        {id: 1, name: "Foo"},
        {id: 2, name: "Bar"},
        {id: 3, name: "Joe"}
    ]
}
```

```
exports.loaded = function(args){
    var page = args.object;
    page.bindingContext = context;
}

exports.showEntry = function(args){
    // select the tapped entry without passing an index or anything like that
    var selectedEntry = args.view.bindingContext;
    console.log(selectedEntry.id + " " + selectedEntry.name);
}
```

page.xml

```
<Repeater items="{{ items }}" >

    <Repeater.itemTemplate>
        <Label text="{{ name }}" tap="showEntry" />
    </Repeater.itemTemplate>

</Repeater>
```

Read Displaying data as list (using Repeater, ListView or *ngFor for {N}+Angular-2 apps) online: https://riptutorial.com/nativescript/topic/5226/displaying-data-as-list--using-repeater--listview-or--ngfor-for--n-plusangular-2-apps-

Chapter 4: Global Variables

Examples

Console

NativeScript's global `console` variable lets you print values to your terminal for debugging. The simplest usage is passing a value to the `console.log()` function:

```
console.log("hello world");
```

The `console` object has several other methods, including `dump()`, `trace()`, `assert()` and more.

```
// Prints the state of a full object.
console.dump({ firstName: "Native", lastName: "Script"});

// Prints the current stack trace
console.trace();

// Asserts a boolean condition, and prints to the console if the assertion fails.
console.assert(1 === 1, "This won't print as the condition is true");
console.assert(1 === 2, "This will print as the condition is false");
```

Timer (JavaScript)

NativeScript's global `timer` variable lets you set timeouts and intervals for asynchronous delayed function calls.

Importing

```
var timer = require("timer")
```

Timeouts

```
var callback = function(){
    console.log("I will be executed once after 500ms");
}
var timeoutId = timer.setTimeout(callback, 500);

// clearing the timeout
timer.clearTimeout(timeoutId);
```

Intervals

```
var callback = function(){
    console.log("I will be executed every 500 ms")
}
var intervalId = timer.setInterval(callback, 500);
```

```
// clearing the interval
timer.clearInterval(intervalId);
```

Read Global Variables online: https://riptutorial.com/nativescript/topic/3133/global-variables

Chapter 5: implement Interface

Examples

implement View.OnLayoutChangeListener in Nativescript

```
let playerLayoutChangeListener = new android.view.View.OnLayoutChangeListener( {
      onLayoutChange : function ( v:View, left:number, top:number, right:number,
bottom:number, oldLeft:number, oldTop:number, oldRight:number, oldBottom:number):any {
          if (left != oldLeft || top != oldTop || right != oldRight || bottom != oldBottom) {
              console.log("OnLayoutChangeListener");
              __this.changeSurfaceLayout();
          }
      }
    });
```

create a surfaceView http://stackoverflow.com/documentation/proposed/changes/79536

Add Listener:

```
    surfaceView.addOnLayoutChangeListener(playerLayoutChangeListener);
```

remove Listener:

```
    surfaceView.removeOnLayoutChangeListener(playerLayoutChangeListener);
```

Read implement Interface online: https://riptutorial.com/nativescript/topic/5560/implement-interface

Chapter 6: Implementing Animations in Nativescript

Examples

Background Animation of StackLayout

Animating Background color of stacklayout on tapping button

pages/main.component.ts

```
import {Component, ElementRef, ViewChild} from "@angular/core";
import {Color} from "color";
import {View} from "ui/core/view";

    @Component({
        selector: "main",
        template: `
            <StackLayout #el>
              <Button text="Apply Changes" (tap)="changeBgColor()"></Button>
            </StackLayout>
        `,
        styleUrls: ["pages/main/main-common.css"],
    })

    export class MainComponent {
        @ViewChild("el") el: ElementRef;
        changeBgColor() {
            let el = <View>this.el.nativeElement;
            el.animate({
                backgroundColor: new Color("#222"),
                duration: 300
            });
        }
    }
```

pages/main-common.css

```
StackLayout{
    background-color: #333;
}
```

Use of animation timing function and animation properties.

pages/main.component.ts

```
import {Component, ElementRef, ViewChild} from "@angular/core";
import {View} from "ui/core/view";
import {AnimationCurve} from "ui/enums";

@Component({
```

```
    selector: "main",
    template: `
        <StackLayout>
          <Image #img src="~/assets/images/user-shape.png"></Image>
          <Button text="Apply Changes" (tap)="animateImage()"></Button>
        </StackLayout>
    `,
    styleUrls: ["pages/main/main-common.css"],
})

export class MainComponent {
    @ViewChild("img") img: ElementRef;
    animateImage() {
        let img = <View>this.img.nativeElement;
        img.animate({
            translate: { x: 0, y: 120 },
            duration: 2000,
            curve: AnimationCurve.easeIn
        });
    }
}
```

#snippet for other animation properties

You can also write your own timing function using cubicBezier.

1. Use of cubicBezier

```
img.animate({
        translate: { x: 0, y: 120 },
        duration: 2000,
        curve: AnimationCurve.cubicBezier(0.1, 0.2, 0.1, 1)
  });
```

2. Animation Properties

Opacity

```
img.animate({
    opacity: 0,
    duration: 2000
});
```

Translate

```
img.animate({
    translate: { x: 120, y: 0},
    duration: 2000
});
```

Scale

```
img.animate({
    scale: { x: 1.5, y: 1.5},
    duration: 2000
});
```

Rotate

```
img.animate({
    rotate: 270,
    duration: 2000
});
```

Read Implementing Animations in Nativescript online:
https://riptutorial.com/nativescript/topic/5970/implementing-animations-in-nativescript

Chapter 7: Multithreading Model

Remarks

The new chrome v8 engine is partially ES7 compliant. So if we add `"use strict";` to top of our file (typescript do that when transpiles typescript) we have to make sure that any functions that are on the global scope are actually assigned to the global scope. so we should use `self.functionName` or `global.functionName`.

Examples

use Workers in angular2 service

`/app/services/greeting.service.ts` :

```
import { Injectable } from '@angular/core';
import {greetingTypes,request,response}
            from  './greeting.interface'

@Injectable()
export class Greeting{

    private worker;
    constructor(){
       this.worker = new Worker('../workers    /greeting.worker');
    }

    sayHello(message:string, answerCallback:Function){
        let requestData:request =
            {'type':greetingTypes.HELLO ,'message':message} ;

        this.worker.postMessage(requestData);
        this.worker.onmessage = (msg)=>{
            let response:response = msg.data;

            if(response.type == greetingTypes.HELLO){
                answerCallback(response.answer)
            }
        }
    }

    sayBye(message:string, answerCallback:Function){
        let requestData:request =     {'type':greetingTypes.BYE ,'message':message};

        this.worker.postMessage(requestData);
        this.worker.onmessage = (msg)=>{
            let response:response = msg.data;

            if(response.type == greetingTypes.BYE)
                answerCallback(response.answer)
        }
    }
}
```

app/services/greeting.interface.ts :

```
export enum greetingTypes{
    BYE,
    HELLO
}

export interface request{
   type:greetingTypes,
   message:string
}

export interface response{
    type:greetingTypes,
    answer:string
}
```

app/workers/greeting.worker.ts :

```
require("globals");
import {greetingTypes,request,response} from
            '../services/greeting.interface';

self.onmessage = (msg)=> {
   let request:request = msg.data;
   let responseData:response;
   if(request.type ==  greetingTypes.HELLO)
        console.log('worker got the message: ' +
                        request.message);
        responseData = {'type':greetingTypes.HELLO,
                            'answer': 'HELLO!'};
        global.postMessage(responseData);

   if(request.type == greetingTypes.BYE )
        console.log('worker got the message: ' +request.message);
        responseData = {'type':greetingTypes.BYE ,
                           'answer':'goodBye!'};
            global.postMessage(responseData);

    };
```

app/app.component.ts :

```
import {Component} from "@angular/core";
import {Greeting} from './services/greeting.service';
@Component({
    selector: "my-app",
    templateUrl: "app.component.html",
    providers:[Greeting]
})
export class AppComponent {

    constructor(private greeting:Greeting){}


public tapHello() {

    this.greeting.sayHello('hi',
```

```
        (answer)=>{console.log('answer from worker : '+ answer)});
}


public tapBye() {
    this.greeting.sayBye('bye',
        (answer) => {console.log('answer from worker : ' + answer)});
}
```

}

`app/app.component.html` :

```
<StackLayout>
    <Button text="sayBye" (tap)="tapBye()"></Button>
    <Button text="sayHello" (tap) = "tapHello()"></Button>
</StackLayout>
```

Read Multithreading Model online: https://riptutorial.com/nativescript/topic/7878/multithreading-model

Chapter 8: StatusBar

Examples

Hide/show - android

This is a statusbar that you see on top of your screen with icons of battry,clock ... .


```
let frame = require("ui/frame");
```

Hide:

```
frame.topmost().android.activity.getWindow().
getDecorView().setSystemUiVisibility(android.view.View.SYSTEM_UI_FLAG_FULLSCREEN);
```

Show:

```
frame.topmost().android.activity.getWindow().
getDecorView().setSystemUiVisibility(android.view.View.SYSTEM_UI_FLAG_VISIBLE );
```

Make statusBar Transparent android

open `APP_Resources/values/styles.xml` and add the

```
<item name="android:windowTranslucentStatus">true</item>
```

in the

```
<style name="AppThemeBase" parent="Theme.AppCompat.Light.NoActionBar"> </style>
```

section.

Read StatusBar online: https://riptutorial.com/nativescript/topic/6007/statusbar

Chapter 9: Styling nativescript template

Examples

Adding a sample layout in your app

main.component.ts

```
import {Component} from "@angular/core";

@Component({
    selector: "main",
    template: `
    <StackLayout>
      <TextField hint="some text"></TextField>
      <Button text="Click me" class="btn"></Button>
    </StackLayout>
    `,
    styleUrls: ["pages/main/main-common.css", "pages/main/main.css"]
})
export class MainComponent { }
```

Method 1 : Global CSS

app.css -- Applies globally to all layouts.

```
StackLayout {
  margin: 10;
  background-color: white;
}
.btn, TextField {
  margin-left: 16;
  margin-right: 16;
}
```

Method 2 : Platform specific CSS

platform.android.css -- Applies globally to all layouts in android device.

```
.btn{
    background-color: #191919;
    color: #fff;
}
```

platform.ios.css -- Applies globally to all layouts in ios device.

```
.btn{
```

```
    background-color: #fff;
    color: #191919;
}
```

app.css

```
@import url("~/platform.css");
```

Method 3 : Component-specific CSS

pages/main/main.android.css -- Applies to specific component in android device.

```
TextField {
  color: #e1e1e1;
  font-size: 12;
}
```

pages/main/main.ios.css -- Applies to specific component in ios device.

```
TextField {
  color: #e3e3e3;
  font-size: 15;
}
```

pages/main/main-common.css -- Applies to specific component in all devices.

```
TextField {
  padding: 4;
}
```

Read Styling nativescript template online: https://riptutorial.com/nativescript/topic/3872/styling-nativescript-template

Chapter 10: using native widget

Examples

Using surfaceView in ng2-TNS-Android : step by step

For example you want to use surfaceView in ng2-nativescript. As we don't have `surfaceView` in nativescript we should use `placeholder`.

first we should import the requirements:

```
import {Component} from "@angular/core";
import placeholder = require("ui/placeholder");
let application= require("application");
```

then add the placeholder to your html file:

```
<Placeholder (creatingView)="creatingView($event)"></Placeholder>
```

Add this method to your class:

```
public creatingView(args: any) {
  var nativeView = new android.view.SurfaceView(application.android.currentContext);
  args.view = nativeView;
}
```

typescript doesn't know what is `android` and we should add platform declaration files follow this Answer to add them.

because of a problem in current version of ng2-nativescript we should do some extra work:

change the placeholder to :

```
<Placeholder  *ngIf="init" (creatingView)="creatingView($event)"></Placeholder>
```

Import OnInit:

```
import {Component,OnInit} from "@angular/core";
```

your class should implement OnInit

```
export class AppComponent implements OnInit
```

and add these lines to your class:

```
public init: boolean = false;
ngOnInit() {
    this.init = true;
```

```
}
```

now you have a surfaceView in your nativescript app :)

Call methods of SurfaceView

For example you want to call `getHolder()`:

add a variable and loaded event to your placeholder like this:

```
  <Placeholder  #surface *ngIf="init" (creatingView)="creatingView($event)"
(loaded)="onLoaded(surface)"></Placeholder>
```

and add the onLoaded method to your class:

```
 onLoaded(element){
  let mSurface = element.android;
  let holder =  mSurface.getHolder();
}
```

ATTENTION:

It's not guaranteed that `android` property (`element.android`) will be available in `ngAfterViewInit` so we used `loaded` event instead of that.

Using surfaceView in ng2-TNS-Android : whole ready example

app.component.ts:

```
import {Component,OnInit} from "@angular/core";
import placeholder = require("ui/placeholder");
let application= require("application");

@Component({
    selector: "my-app",
    templateUrl: "app.component.html",
})
export class AppComponent implements OnInit{

  public creatingView(args: any) {
    var nativeView = new android.view.SurfaceView(application.android.currentContext);
    args.view = nativeView;
  }

  onLoaded(element){
    let mSurface = element.android;
    let holder =  mSurface.getHolder();
  }

  public init: boolean = false;
    ngOnInit() {
        this.init = true;
    }
}
```

app.component.html :

```
<StackLayout>
   <Placeholder  #surface *ngIf="init" (creatingView)="creatingView($event)"
(loaded)="onLoaded(surface)"></Placeholder>
</StackLayout>
```

Read using native widget online: https://riptutorial.com/nativescript/topic/5834/using-native-widget

Credits

S. No	Chapters	Contributors
1	Getting started with nativescript	Adam Diament, Community, George Edwards, HabibKazemi, Hardik Vaghani, Houssem Yahiaoui, Richard Hubley, user6939352
2	Accessing native apis	HabibKazemi
3	Displaying data as list (using Repeater, ListView or *ngFor for {N}+Angular-2 apps)	Nick Iliev, Tim Hallyburton, William KLEIN
4	Global Variables	Tim Hallyburton, TJ VanToll
5	implement Interface	HabibKazemi
6	Implementing Animations in Nativescript	Madhav Poudel
7	Multithreading Model	HabibKazemi
8	StatusBar	HabibKazemi
9	Styling nativescript template	George Edwards, Madhav Poudel, Nick Iliev
10	using native widget	HabibKazemi