



Бесплатная электронная книга

УЧУСЬ

nativescript

Free unaffiliated eBook created from
Stack Overflow contributors.

#nativescrip

t

.....	1
1: nativescript	2
.....	2
Examples.....	2
.....	2
Macos.....	2
Windows.....	2
Visual Studio NativeScript.....	3
Hello World.....	3
nativescript-android Wi-Fi ().....	4
2:	7
Examples.....	7
.....	7
(JavaScript).....	7
3: native apis	9
Examples.....	9
Java nativescript javascript.....	9
apis javascript.....	12
4:	14
Examples.....	14
surfaceView ng2-TNS-Android:	14
surfaceView ng2-TNS-Android:	15
5:	17
.....	17
Examples.....	17
angular2.....	17
6: (Repeater, ListView * ngFor	20
.....	20
Examples.....	20
Repeater (NativeScript Core).....	20
Repeater ObservableArray (NativeScript Core).....	21

ListView ObservableArray (NativeScript Core).....	21
ListView (NativeScript + Angular-2).....	22
* ngFor (nativeScript + Angular.....	23
(JavaScript).....	23
7: Nativescript.....	25
Examples.....	25
StackLayout.....	25
.....	25
.....	26
.....	26
.....	26
.....	27
8:	28
Examples.....	28
View.OnLayoutChangeListener Nativescript.....	28
9:	29
Examples.....	29
/ -	29
Bar	29
10:	30
Examples.....	30
.....	30
1: CSS.....	30
2: CSS	30
3: CSS.....	31
.....	32

Около

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

глава 1: Начало работы с nativescript

замечания

Nativescript - это высокопроизводительное межплатформенное приложение для мобильных приложений, которое позволяет настроить таргетинг на iOS и Android (с окнами в конвейере) с использованием веб-технологий (JS и html). Он был создан с целым рядом ключевых целей:

- Визуальный исполнитель: нет UI Jank даже на android у вас есть маслянистые гладкие fps
- Расширяемость: у вас есть доступ ко всем собственным API-интерфейсам, для создания простых кросс-платформенных плагинов
- Полностью собственный интерфейс
- Высоко интегрированный с машинописным и угловым 2
- Open Source, с сильной корпоративной поддержкой от Telerik

Examples

Установка или настройка

Подробные инструкции по настройке или установке Nativescript.

В следующих примерах приведены необходимые шаги по настройке системы Windows или OSX, а затем подпишите сообщение в руководства по устранению неполадок в случае возникновения каких-либо проблем.

Кроме того, есть примеры того, как настроить рекомендуемые рабочие процессы, IDE и эмуляторы.

Macos

1. Убедитесь, что у вас установлен [самый последний](#) Node.js LTS. Если вы используете [Homebrew](#), это можно сделать с помощью `brew install node4-lts`.
2. Откройте терминал и введите `npm install -g nativescript`. Если вы получаете ошибку `EACCES`, используйте `sudo npm install -g nativescript`.
3. В командной строке введите `ruby -e "$(curl -fsSL https://www.nativescript.org/setup/mac)"`. (Это может занять некоторое время.)
4. Чтобы убедиться, что это работает, введите `tns doctor` в Terminal.
5. Если есть какие-либо ошибки, следуйте инструкциям по [устранению неполадок](#).

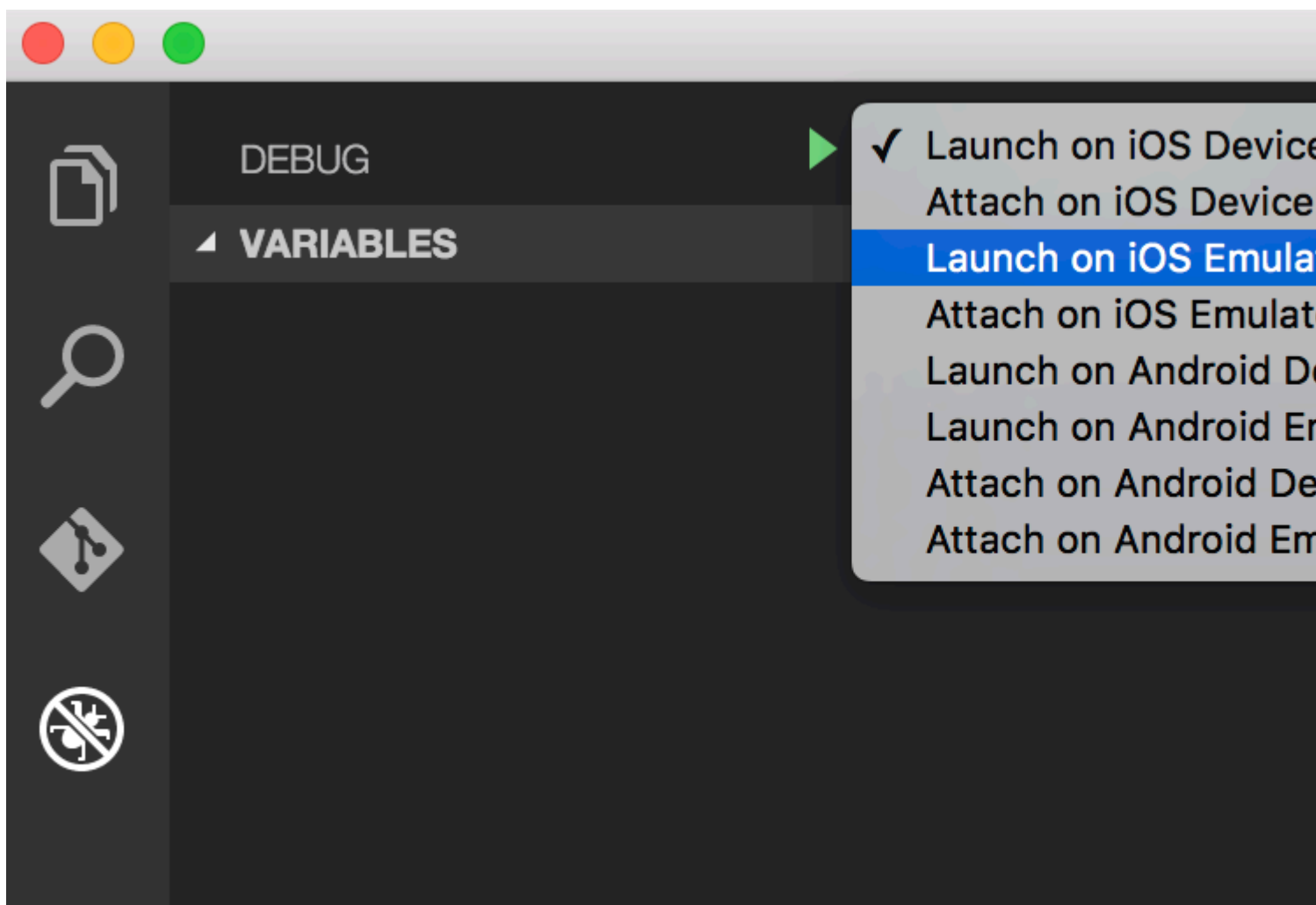
Windows

1. Убедитесь, что у вас установлен самый последний [узел nodeJS LTS](#)
2. Откройте командную строку и введите `$ npm install -g nativescript`
3. В командной строке введите `$ @powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-object net.webclient).DownloadString('https://www.nativescript.org/setup/win'))"`
- это может занять некоторое время
4. Чтобы проверить вышеописанное, введите `$ tns doctor` в командной строке (ваш cmd)
5. Если есть какие-либо ошибки, выполните [руководство по устранению неполадок](#)

Использование кода Visual Studio для разработки NativeScript

[Visual Studio Code](#) - это редактор кода с открытым исходным кодом и многофункциональный код от Microsoft. Чтобы настроить его для разработки NativeScript, откройте командную палитру (`F1` или `⌘ + Shift + P`) и введите `ext install NativeScript` .

Как только расширение NativeScript установлено, отладчик должен позволить вам установить точки останова в вашем коде. Когда устройство подключено или работает эмулятор, вы можете запустить свое приложение со вкладки Debug.



Ваша первая программа Hello World

```
$ mkdir hello-world
```

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

Затем убедитесь, что у вас подключено устройство или работает эмулятор (если вы этого не сделаете, должен запускаться эмулятор по умолчанию или возникает ошибка. Я бы рекомендовал genymotion для android).

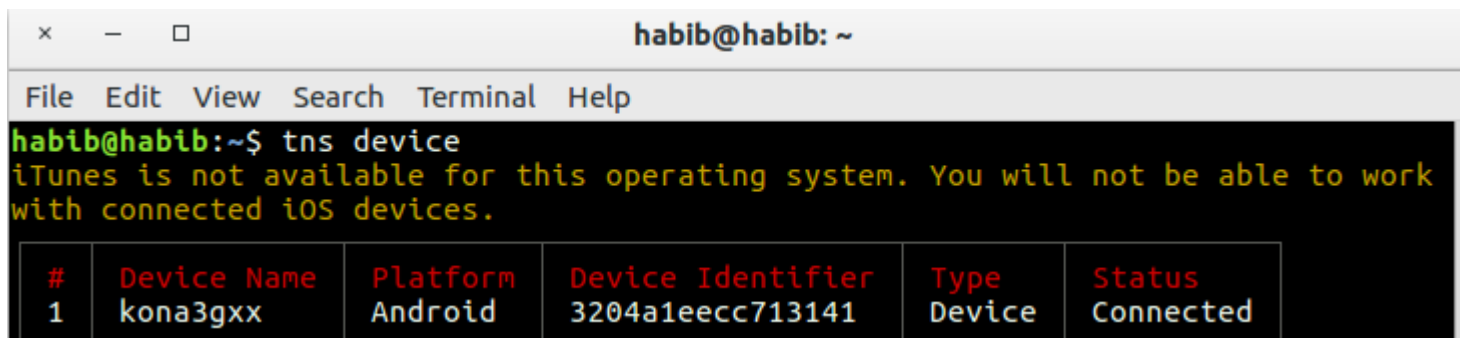
```
$ tns run android
```

Если вы хотите использовать эмулятор Android по умолчанию, добавьте флаг `--emulator`.

Начиная с `tns 2.5 livesync` теперь является действием по умолчанию для `tns run <platform>`, которое будет автоматически перекомпилироваться при сохранении изменений файла. Это может значительно улучшить время разработки, однако, если вы вносите изменения в свои плагины, вам нужно будет перекомпилировать их правильно.

Как отлаживать приложение nativescript-android через Wi-Fi (без корня)

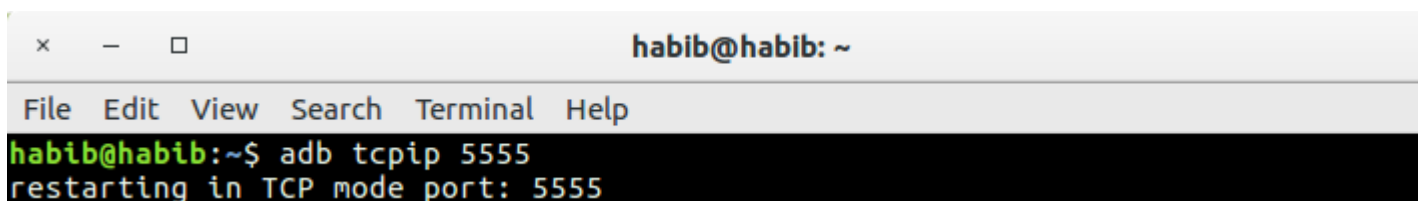
1-Вам необходимо подключить устройство к компьютеру через USB-кабель. Убедитесь, что отладка USB работает. Вы можете проверить, отображается ли оно при запуске `adb devices` (или `tns device`).



```
habib@habib: ~
File Edit View Search Terminal Help
habib@habib:~$ tns device
iTunes is not available for this operating system. You will not be able to work
with connected iOS devices.
```

#	Device Name	Platform	Device Identifier	Type	Status
1	kona3gxx	Android	3204a1eccc713141	Device	Connected

2-Run `adb tcpip 5555`



```
habib@habib:~$ adb tcpip 5555
restarting in TCP mode port: 5555
```

3-Отключите устройство (удалите USB-кабель).

4-Перейдите в Настройки -> О телефоне -> Состояние, чтобы просмотреть IP-адрес вашего телефона.

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

```
habib@habib: ~
File Edit View Search Terminal Help
habib@habib:~$ adb connect 192.168.1.5:5555
connected to 192.168.1.5:5555
```

6 - Если вы снова запустите `adb devices` (или `tns device`), вы увидите свое устройство.

```
habib@habib:~$ tns device
iTunes is not available for this operating system. You will not be able to work
with connected iOS devices.
```

#	Device Name	Platform	Device Identifier	Type	Status
1	kona3gxx	Android	192.168.1.5:5555	Device	Connected

7- Теперь вы можете использовать команды `tns run android`, `tns livesync android`.

ЗАМЕТКИ :

1 - при изменении сети Wi-Fi вам не нужно повторять шаги с 1 по 3 (они устанавливают ваш телефон в режим отладки Wi-Fi). Вам нужно снова подключиться к телефону, выполнив шаги с 4 по 6.

2-Android-телефоны теряют режим отладки Wi-Fi при перезапуске. Таким образом, если ваша батарея умерла, вам нужно начать все сначала. В противном случае, если вы следите за своей батареей и не перезагружаете телефон, вы можете жить без кабеля в течение нескольких недель!

ПРЕДУПРЕЖДЕНИЕ:

оставляя включенную опцию опасной, любой в вашей сети может подключаться к вашему устройству при отладке, даже если вы находитесь в сети передачи данных. Делайте это только при подключении к доверенному Wi-Fi и не забудьте отключить его, когда это будет сделано!

ссылка :

1-Норман Пейтек. 2014. Как отладить Android-приложение через Wi-Fi (без Root!). [ONLINE] Доступно по адресу: <https://futurestud.io/blog/how-to-debug-your-android-app-over-wifi-without-root> . [Доступ к 8 августа 2016 года].

2-usethe4ce. 2012. Запуск / установка / отладка приложений Android через Wi-Fi ?. [ONLINE] Доступно по адресу: <http://stackoverflow.com/a/10236938/4146943> . [Доступ к 8 августа 2016 года].

Прочитайте [Начало работы с nativescript онлайн](#):

<https://riptutorial.com/ru/nativescript/topic/921/начало-работы-с-nativescript>

глава 2: Глобальные переменные

Examples

Приставка

Глобальная `console` переменная NativeScript позволяет печатать значения для вашего терминала для отладки. Простейшее использование передаёт значение функции

`console.log()` :

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

`console` объект имеет несколько других методов, включая `dump()` , `trace()` , `assert()` и [другие](#) .

```
// 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");
```

Таймер (JavaScript)

Глобальная переменная `timer` NativeScript позволяет устанавливать таймауты и интервалы для асинхронных отложенных вызовов функций.

Импорт

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

Таймауты

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

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

Интервалы

```
var callback = function(){
  console.log("I will be executed every 500 ms")
```

```
}  
var intervalId = timer.setInterval(callback, 500);  
  
// clearing the interval  
timer.clearInterval(intervalId);
```

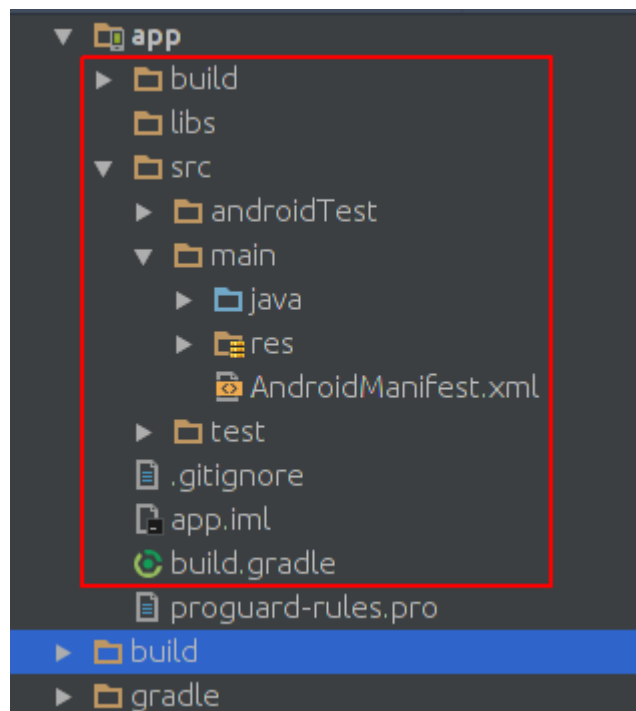
Прочитайте Глобальные переменные онлайн: <https://riptutorial.com/ru/nativescript/topic/3133/глобальные-переменные>

глава 3: Доступ к native apis

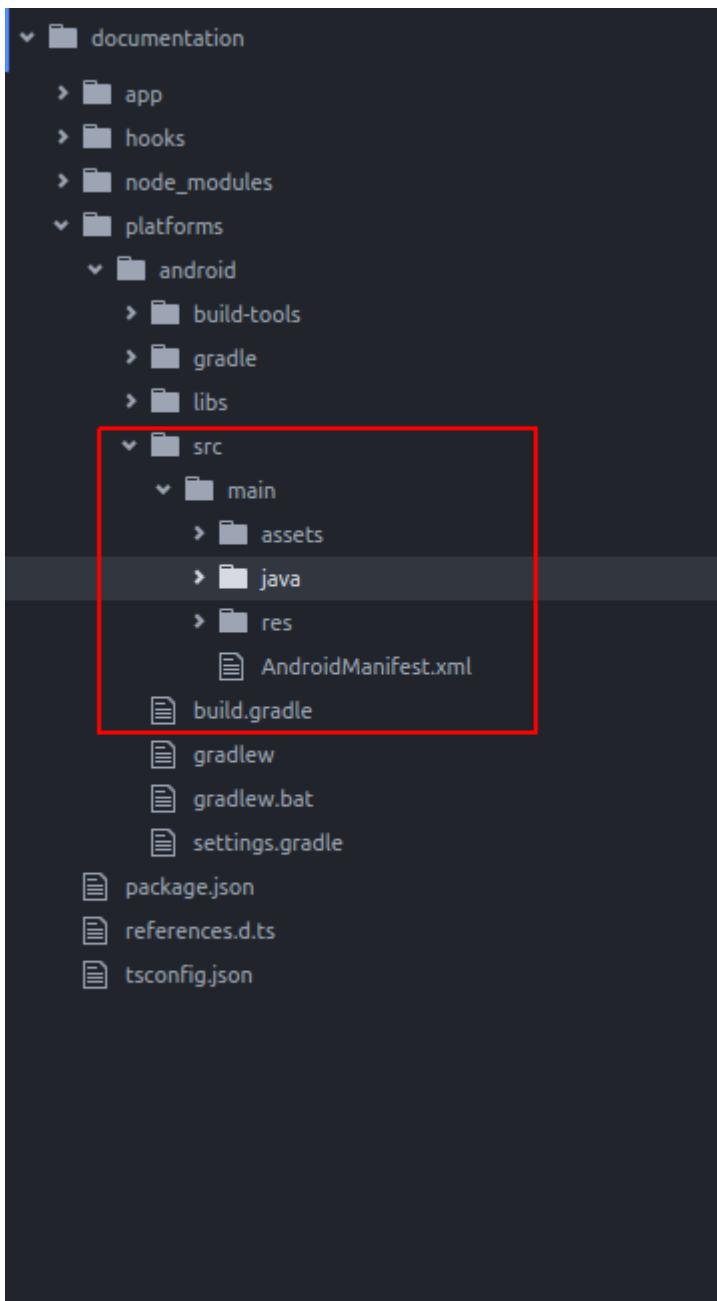
Examples

Напишите код Java в nativescript и используйте его непосредственно в javascript

Это образ структуры проекта в студии Android:

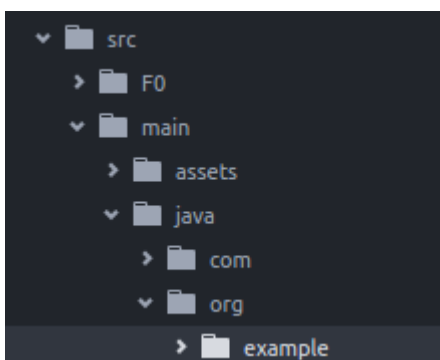


Это образ проектной структуры проекта nativescript:



Как видите, они такие же. поэтому мы можем написать код Java в nativescript, как мы пишем в android studio.

Мы хотим добавить Toast к стандартным приложениям nativescript. после создания нового проекта nativescript создайте каталог `java/org/example` например:



создать новый файл `MyToast.java` в каталоге `example` ;

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();
    }
}
```

Примечания : не забудьте имя пакета;

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 "Hoorraay! \nYou are ready to start building!";
        }
    }

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

теперь, когда вы нажимаете кнопку, вы увидите тост;

Примечания :

1. Функция `showToast` принимает контекст для передачи его `Toast.makeText` и мы передали ему контекст таким образом: `application.android.context`
2. `typescript` не знает, что такое `org` , поэтому мы объявили его: `declare var org:any;`

использовать собственный `apis` прямо в `javascript`

Мы хотим добавить `Toast` в приложение `nativescript` по умолчанию.

```
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();
  }
}
```

для создания тоста мы должны назвать `Toast.makeText` и он находится в пакете `android.widget.Toast` . `Toast.makeText` принимает контекст как первый аргумент, и мы можем получить контекст в `nativescript` таким образом: `application.android.context`

Прочитайте [Доступ к native apis онлайн: https://riptutorial.com/ru/nativescript/topic/5188/доступ-к-native-apis](https://riptutorial.com/ru/nativescript/topic/5188/доступ-к-native-apis)

глава 4: использование собственного виджета

Examples

Использование surfaceView в ng2-TNS-Android: шаг за шагом

Например, вы хотите использовать surfaceView в ng2-nativescript. Поскольку у нас нет surfaceView в nativescript, мы должны использовать placeholder .

сначала мы должны импортировать требования:

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

затем добавьте местозаполнитель в ваш html-файл:

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

Добавьте этот метод в свой класс:

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

typescript не знает, что такое android и мы должны добавить файлы декларации платформы, следуйте этому [ответу](#), чтобы добавить их.

из-за [проблемы](#) в текущей версии ng2-nativescript мы должны сделать дополнительную работу:

замените местозаполнитель на:

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

Импорт OnInit:

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

ваш класс должен реализовать OnInit

```
export class AppComponent implements OnInit
```

и добавьте эти строки в свой класс:

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

теперь у вас есть `surfaceView` в вашем приложении nativescript :)

Способы вызова `SurfaceView`

Например, вы хотите вызвать `getHolder()` :

добавьте переменную и загруженное событие в ваш заполнитель следующим образом:

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

и добавьте метод `onLoaded` в ваш класс:

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

ВНИМАНИЕ :

Не гарантируется, что свойство `android (element.android)` будет доступно в `ngAfterViewInit` ПОЭТОМУ `ngAfterViewInit` ЭТОГО МЫ ИСПОЛЬЗОВАЛИ `loaded` СОБЫТИЕ.

Использование `surfaceView` в `ng2-TNS-Android`: пример готовый

`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>
```

Прочитайте использование собственного виджета онлайн:

<https://riptutorial.com/ru/nativescript/topic/5834/использование-собственного-виджета>

глава 5: Многопоточная модель

замечания

Новый движатель chrome v8 частично совместим с ES7. Поэтому, если мы добавим "use strict"; В начало нашего файла (машинописные тексты делают это, когда transpiles typescript), мы должны убедиться, что любые функции, которые находятся в глобальной области, фактически назначены глобальной области. поэтому мы должны использовать self.functionName ИЛИ global.functionName .

Examples

использовать Работников в службе angular2

/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>
```

Прочитайте Многопоточная модель онлайн: <https://riptutorial.com/ru/nativescript/topic/7878/многopоточная-модель>

глава 6: Отображение данных в виде списка (с использованием Repeater, ListView или * ngFor для приложений {N} + Angular-2)

замечания

Примечание. Не используйте ретранслятор в приложениях {N} + Angular-2! * NgRepeat устаревшая директива в Angular-2. Когда вам нужно отображать повторяющиеся шаблоны элементов, используйте либо конструкцию ListView, либо * ngFor.

Examples

Использование модуля Repeater для отображения данных (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;
}
```

```
}
```

Использование модуля Repeater с 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;
}
```

Использование модуля ListView с 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);
    // });
}

```

Использование ListView для отображения данных (NativeScript + Angular-2)

Создание-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>

```

СОЗДАЕМ-listview.component.ts

```

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

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

var europeanCountries = ["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 < europeanCountries.length; i++) {
      this.countries.push(new Country(europeanCountries[i]));
    }
  }

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

```

Использование * ngFor Структурная директива для отображения данных (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;
  }
}

```

Использование ретранслятора с обратными вызовами (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>
```

Прочитайте [Отображение данных в виде списка \(с использованием Repeater, ListView или *ngFor для приложений {N} + Angular-2\) онлайн: <https://riptutorial.com/ru/nativescript/topic/5226/отображение-данных-в-виде-списка-с-использованием-repeater-listview-или-ngfor-для-приложений-n-plus-angular-2>](https://riptutorial.com/ru/nativescript/topic/5226/отображение-данных-в-виде-списка-с-использованием-repeater-listview-или-ngfor-для-приложений-n-plus-angular-2)

глава 7: Реализация анимаций в Nativescript

Examples

Фоновая анимация StackLayout

Анимация Цвет фона stacklayout при нажатии кнопки

страниц / 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
    });
  }
}
```

страницы / главный common.css

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

Использование функции синхронизации анимации и свойств анимации.

страниц / 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 для других свойств анимации

Вы также можете написать свою собственную функцию синхронизации с помощью кубика.

1. Использование кубика

```

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

```

2. Анимационные свойства

Помутнение

```

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

```

Переведите

```

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

```

Масштаб

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

Поворот

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

Прочитайте Реализация анимаций в Nativescript онлайн:

<https://riptutorial.com/ru/nativescript/topic/5970/реализация-анимаций-в-nativescript>

глава 8: реализовать интерфейс

Examples

реализовать View.OnLayoutChangeListener в 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 ();
    }
  }
});
```

создать поверхностьView <http://stackoverflow.com/documentation/proposed/changes/79536>

Добавить прослушиватель:

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

удалить прослушиватель:

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

Прочитайте реализовать интерфейс онлайн: <https://riptutorial.com/ru/nativescript/topic/5560/реализовать-интерфейс>

глава 9: Статус бар

Examples

Скрыть / показать - андроид

Это панель состояния, которую вы видите по верх экрана с иконками battery, clock



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

Спрятать:

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

Шоу:

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

Сделать statusBar Прозрачный андроид

откройте APP_Resources/values/styles.xml и добавьте

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

В

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

раздел.

Прочитайте Статус бар онлайн: <https://riptutorial.com/ru/nativescript/topic/6007/статус-бар>

глава 10: Стиль шаблона стиля

Examples

Добавление образца в приложение

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 { }
```

Метод 1: Глобальный CSS

app.css - Применяет глобально ко всем макетам.

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

Метод 2: CSS для платформы

platform.android.css - применяет глобально все макеты в устройстве Android.

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

platform.ios.css - Применяет глобально ко всем макетам в устройстве ios.

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

app.css

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

Метод 3: Компонентный CSS

pages / main / main.android.css - применяется к определенному компоненту в устройстве Android.

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

pages / main / main.ios.css - применяется к определенному компоненту в устройстве ios.

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

pages / main / main-common.css - применяется к определенному компоненту на всех устройствах.

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

Прочитайте [Стиль шаблона стиля онлайн](https://riptutorial.com/ru/nativescript/topic/3872/стиль-шаблона-стиля): <https://riptutorial.com/ru/nativescript/topic/3872/стиль-шаблона-стиля>

кредиты

S. No	Главы	Contributors
1	Начало работы с nativescript	Adam Diament , Community , George Edwards , HabibKazemi , Hardik Vaghani , Housseem Yahiaoui , Richard Hubley , user6939352
2	Глобальные переменные	Tim Hallyburton , TJ VanToll
3	Доступ к native apis	HabibKazemi
4	использование собственного виджета	HabibKazemi
5	Многопоточная модель	HabibKazemi
6	Отображение данных в виде списка (с использованием Repeater, ListView или * ngFor для приложений {N} + Angular-2)	Nick Iliev , Tim Hallyburton , William KLEIN
7	Реализация анимаций в Nativescript	Madhav Poudel
8	реализовать интерфейс	HabibKazemi
9	Статус бар	HabibKazemi
10	Стиль шаблона стиля	George Edwards , Madhav Poudel , Nick Iliev