



EBook Gratis

APRENDIZAJE nativescript

Free unaffiliated eBook created from
Stack Overflow contributors.

#nativescript

t

Tabla de contenido

Acerca de.....	1
Capítulo 1: Empezando con nativescript.....	2
Observaciones.....	2
Examples.....	2
Instalación o configuración.....	2
Mac OS.....	2
Windows.....	2
Usando el código de Visual Studio para el desarrollo de NativeScript.....	3
Tu primer programa Hello World.....	3
Cómo depurar la aplicación nativescript-android a través de WiFi (sin Root).....	4
Capítulo 2: Acceso a apis nativas.....	6
Examples.....	6
Escriba código java en nativescript y utilícelo directamente en javascript.....	6
usar apis nativos directamente en javascript.....	9
Capítulo 3: Barra de estado.....	11
Examples.....	11
Ocultar / mostrar - Android.....	11
Make StatusBar Android transparente.....	11
Capítulo 4: Implementando animaciones en nativos.....	12
Examples.....	12
Animación de fondo de StackLayout.....	12
Uso de la función de temporización de animación y propiedades de animación.....	12
Opacidad.....	13
Traducir.....	13
Escala.....	13
Girar.....	14
Capítulo 5: implementar interfaz.....	15
Examples.....	15
implementar View.OnLayoutChangeListener en Nativescript.....	15
Capítulo 6: Modelo multihilo.....	16

Observaciones.....	16
Examples.....	16
utilizar trabajadores en servicio angular2.....	16
Capítulo 7: Mostrar datos como lista (usando Repeater, ListView o * ngFor para {N} + aplic.....	19
Observaciones.....	19
Examples.....	19
Uso del módulo Repeater para mostrar datos (NativeScript Core).....	19
Usando el módulo Repeater con ObservableArray (NativeScript Core).....	19
Usando el módulo ListView con ObservableArray (NativeScript Core).....	20
Usando ListView para mostrar datos (NativeScript + Angular-2).....	21
Uso de * ngPara la directiva estructural para mostrar datos (nativeScript + Angular-2).....	22
Uso de repetidor con devoluciones de llamada (JavaScript).....	22
Capítulo 8: Plantilla de nativos de estilo.....	24
Examples.....	24
Agregando un diseño de muestra en tu aplicación.....	24
Método 1: CSS global.....	24
Método 2: CSS específico de la plataforma.....	24
Método 3: CSS específico del componente.....	25
Capítulo 9: usando widget nativo.....	26
Examples.....	26
Usando surfaceView en ng2-TNS-Android: paso a paso.....	26
Uso de surfaceView en ng2-TNS-Android: ejemplo completo listo.....	27
Capítulo 10: Variables globales.....	29
Examples.....	29
Consola.....	29
Temporizador (JavaScript).....	29
Creditos.....	31

Acerca de

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

Capítulo 1: Empezando con nativescript

Observaciones

Nativescript es un tiempo de ejecución de aplicaciones móviles multiplataforma de alto rendimiento, que le permite apuntar a iOS y Android (con ventanas en la tubería) utilizando tecnologías web (JS y html). Fue creado con una serie de objetivos clave:

- Visualmente Performant: no hay UI Jank, incluso en Android, tienes fps suaves y mantecosos
- Extensible: tiene acceso a todas las API nativas para crear complementos de plataforma cruzada fáciles
- Interfaz de usuario completamente nativa
- Altamente integrado con mecanografiado y angular 2
- Código abierto, con un fuerte respaldo corporativo de Telerik.

Examples

Instalación o configuración

Instrucciones detalladas sobre cómo configurar o instalar Nativescript.

Los siguientes ejemplos muestran los pasos necesarios para configurar un sistema Windows u OSX y luego firmar las publicaciones de solución de problemas en caso de que tenga algún problema.

Además, hay ejemplos de cómo configurar flujos de trabajo, IDE y emuladores recomendados.

Mac OS

1. Asegúrate de tener instalado el LTS de Node.js [más reciente](#) . Si usa [Homebrew](#), esto se puede hacer con `brew install node4-lts` .
2. Abra Terminal y escriba `npm install -g nativescript` . Si recibe un error `EACCES` , use `sudo npm install -g nativescript` .
3. En el símbolo del sistema, escriba `ruby -e "$(curl -fsSL https://www.nativescript.org/setup/mac)"` . (Esto podría tomar un tiempo.)
4. Para verificar que lo anterior ha funcionado, escriba `tns doctor` en Terminal.
5. Si hay algún error, siga con la [guía de solución de problemas](#) .

Windows

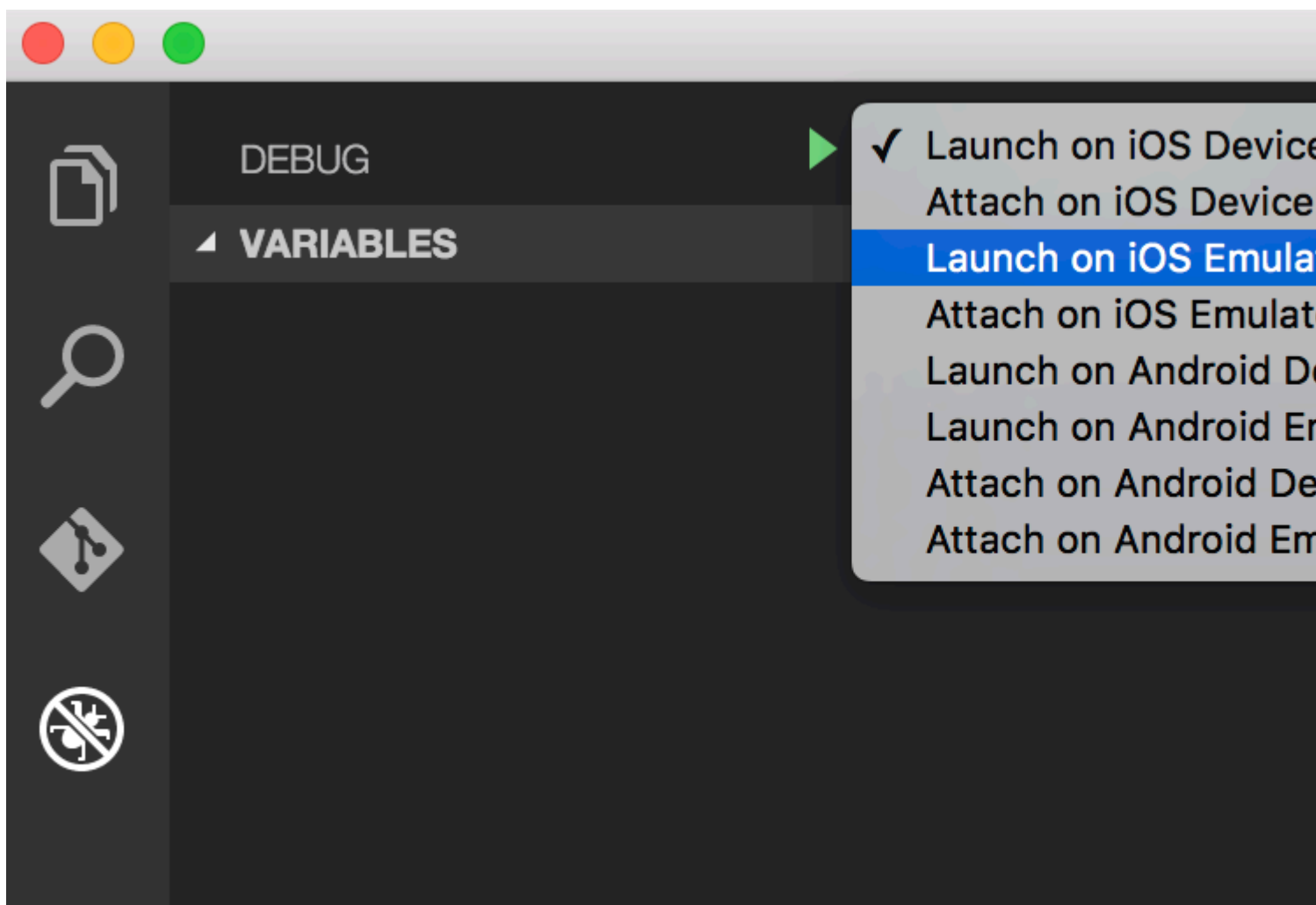
1. Asegúrate de que tienes el último [nodeJS LTS](#) instalado
2. Abra el símbolo del sistema y escriba `$ npm install -g nativescript`
3. En el símbolo del sistema, escriba `$ @powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-object net.webclient).DownloadString('https://www.nativescript.org/setup/win'))"`

- esto podría tomar un tiempo
- 4. Para verificar que lo anterior haya funcionado, escriba `$ tns doctor` en el símbolo del sistema (su cmd)
- 5. Si hay algún error, siga con la [guía de solución de problemas](#).

Usando el código de Visual Studio para el desarrollo de NativeScript

[Visual Studio Code](#) es un editor de código de código abierto y rico en funciones de Microsoft. Para configurarlo para el desarrollo de NativeScript, abra la paleta de comandos (`F1` o `⌘ + Shift + P`) y escriba `ext install NativeScript`.

Una vez que se instala la extensión NativeScript, el depurador debería permitirle establecer puntos de interrupción en su código. Cuando se conecta un dispositivo o se ejecuta un emulador, puede iniciar su aplicación desde la pestaña Depurar.



Tu primer programa 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
```

Luego, asegúrese de tener un dispositivo conectado o un emulador en ejecución (si no lo hace, el

emulador predeterminado debería iniciarse o se generará un error. Recomendaría Genymotion para Android).

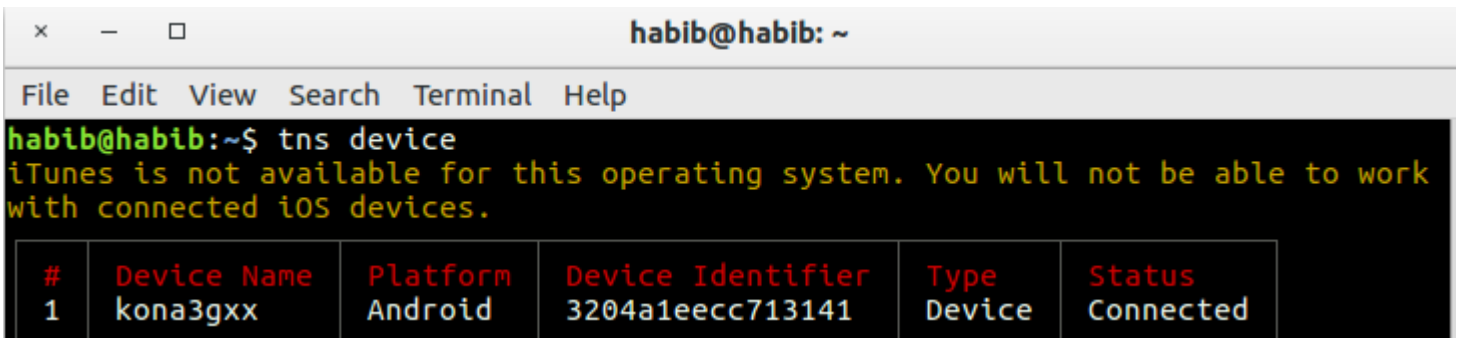
```
$ tns run android
```

Si desea utilizar el emulador de Android predeterminado, agregue la `--emulator`.

A partir de tns 2.5 `livesync` es ahora la acción predeterminada para `tns run <platform>`, que se volverá a compilar automáticamente cuando guarde los cambios de archivo. Esto puede mejorar dramáticamente su tiempo de desarrollo, sin embargo, si realiza cambios en sus complementos, deberá recompilarlos correctamente.

Cómo depurar la aplicación nativescript-android a través de WiFi (sin Root)

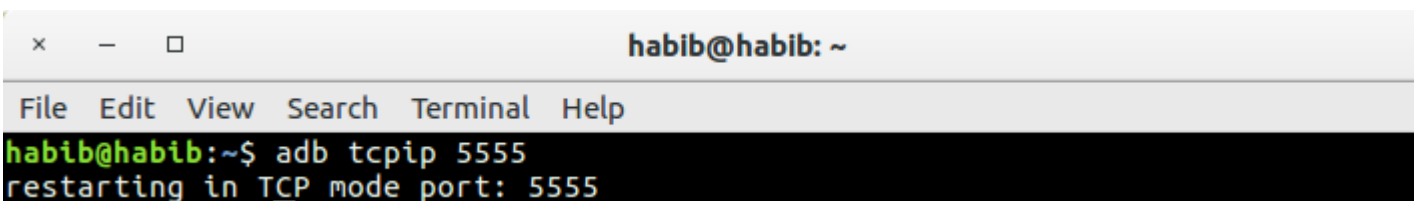
1-Necesita conectar su dispositivo a su computadora mediante un cable USB. Asegúrese de que la depuración USB está funcionando. Puede verificar si aparece cuando se ejecutan `adb devices` (o `tns device`).



```
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`

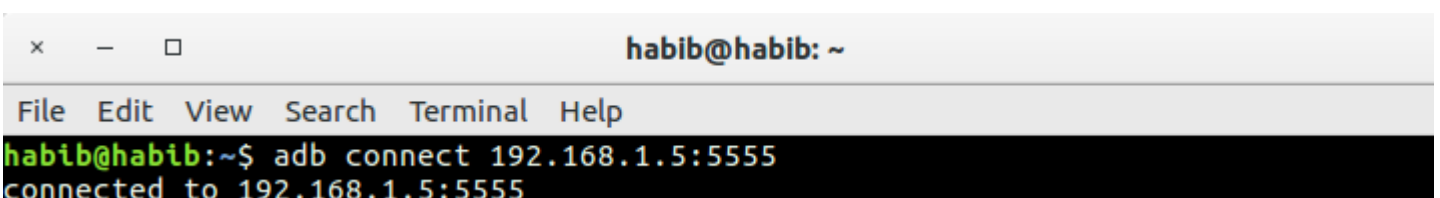


```
habib@habib:~$ adb tcpip 5555
restarting in TCP mode port: 5555
```

3-Desconecte su dispositivo (retire el cable USB).

4-Vaya a Configuración -> Acerca del teléfono -> Estado para ver la dirección IP de su teléfono.

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



```
habib@habib:~$ adb connect 192.168.1.5:5555
connected to 192.168.1.5:5555
```

6-Si vuelve a `adb devices` (o `tns device`), debería ver su dispositivo.

```
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	192.168.1.5:5555	Device	Connected

7- Ahora puede usar `tns run android`, `tns livesync android` comandos `tns livesync android`.

NOTAS:

1: cuando la red WiFi cambia, no tiene que repetir los pasos 1 a 3 (estos configuran su teléfono en el modo de depuración de wifi). Debe volver a conectarse a su teléfono ejecutando los pasos 4 a 6.

Los teléfonos 2-Android pierden el modo wifi-debug al reiniciar. Por lo tanto, si tu batería se agotó, tienes que volver a empezar. De lo contrario, si vigila la batería y no reinicia el teléfono, ¡puede vivir sin cable durante semanas!

ADVERTENCIA :

dejar la opción habilitada es peligroso, cualquier persona en su red puede conectarse a su dispositivo en depuración, incluso si está en una red de datos. ¡Hágalo solo cuando esté conectado a una red Wi-Fi de confianza y recuerde desconectarlo cuando haya terminado!

referencia :

1-Norman Peitek. 2014. Cómo depurar tu aplicación de Android a través de WiFi (sin Root!). [ONLINE] Disponible en: <https://futurestud.io/blog/how-to-debug-your-android-app-over-wifi-without-root> . [Accedido el 8 de agosto de 2016].

2-usethe4ce. 2012. Ejecutar / instalar / depurar aplicaciones de Android a través de Wi-Fi ?. [ONLINE] Disponible en: <http://stackoverflow.com/a/10236938/4146943> . [Accedido el 8 de agosto de 2016].

Lea Empezando con nativescript en línea:

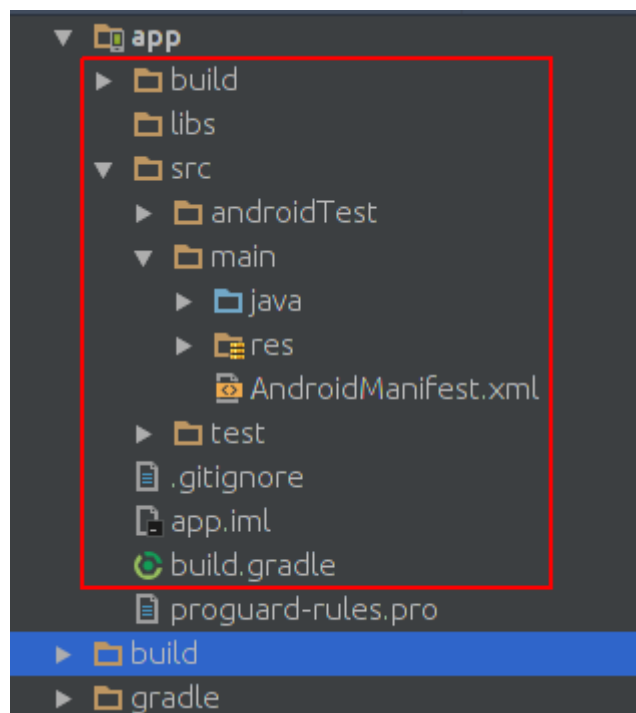
<https://riptutorial.com/es/nativescript/topic/921/empezando-con-nativescript>

Capítulo 2: Acceso a apis nativas

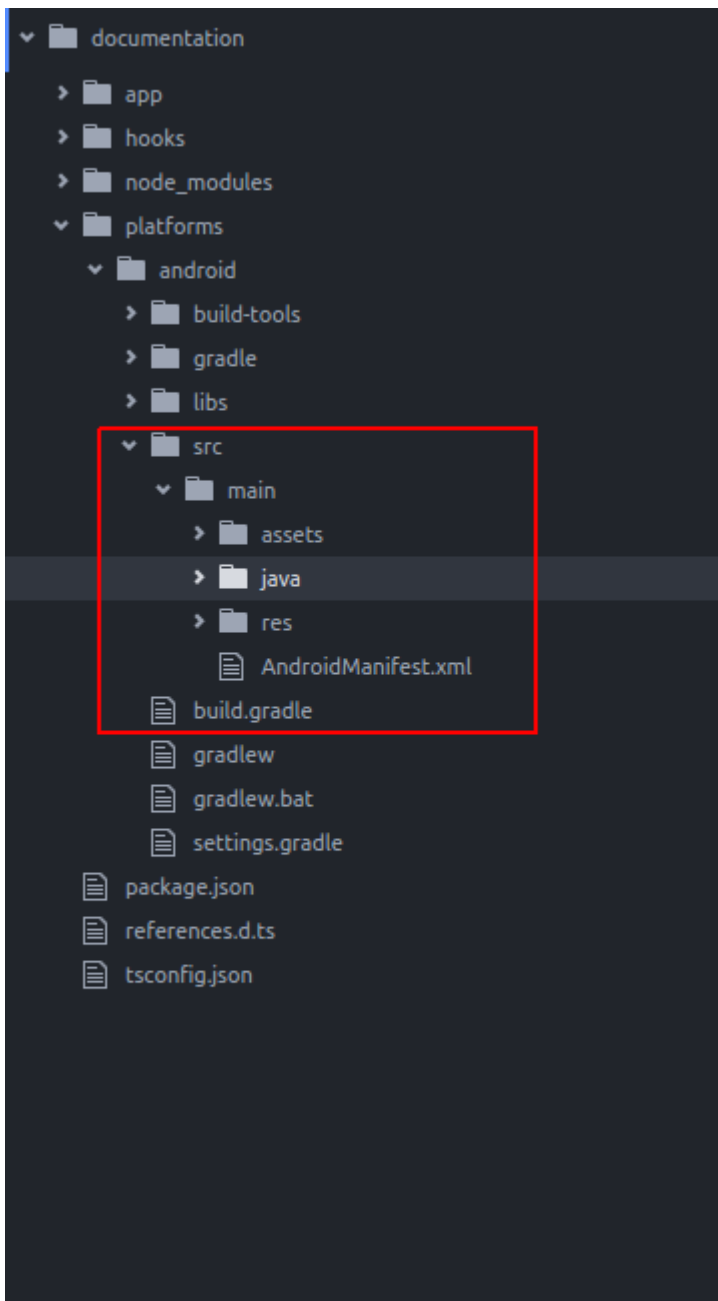
Examples

Escriba código java en nativescript y utilícelo directamente en javascript

Esta es la imagen de la estructura del proyecto en el estudio de Android:

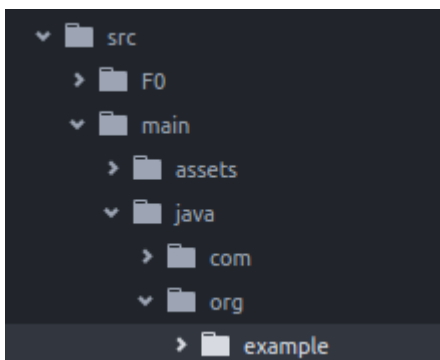


Esta es la imagen de la estructura del proyecto del proyecto nativescript:



Como veis son iguales. así que podemos escribir código java en nativescript como escribimos en Android Studio.

Queremos agregar Toast a la aplicación predeterminada de nativescript. después de crear un nuevo proyecto nativescript, cree un directorio en el directorio `java/org/example` como este:



Cree un nuevo archivo `MyToast.java` en el directorio de `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();
    }
}
```

Notas : no olvide el nombre del paquete;

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");
    }
}
```

ahora cuando presionas el botón se mostrará un brindis;

Notas :

1. La función `showToast` acepta el contexto para pasarlo a `Toast.makeText` y le pasamos un contexto de esta manera: `application.android.context`
2. Typescript no sabe qué es `org` , por lo que lo declare `var org:any; : declare var org:any;`

usar apis nativos directamente en javascript

Queremos añadir Toast a la aplicación predeterminada nativescript.

```
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 "Hoorraay! \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();
  }
}
```

para crear brindis debemos llamar a `Toast.makeText` y está en el paquete `android.widget.Toast` . `Toast.makeText` acepta el contexto como primer argumento y podemos obtener el contexto en nativescript de esta manera: `application.android.context`


Lea Acceso a apis nativas en línea: <https://riptutorial.com/es/nativescript/topic/5188/acceso-a->

Capítulo 3: Barra de estado

Examples

Ocultar / mostrar - Android

Esta es una barra de estado que ves en la parte superior de tu pantalla con íconos de batería, reloj



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

Esconder:

```
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 Android transparente

APP_Resources/values/styles.xml y agregue el

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

en el

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

sección.

Lea Barra de estado en línea: <https://riptutorial.com/es/nativescript/topic/6007/barra-de-estado>

Capítulo 4: Implementando animaciones en nativos

Examples

Animación de fondo de StackLayout

Color de fondo animado de stacklayout al tocar el botón

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
    });
  }
}
```

páginas / main-common.css

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

Uso de la función de temporización de animación y propiedades de animación.

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 para otras propiedades de animación

También puede escribir su propia función de temporización utilizando cubicBezier.

1. Uso de cubicBezier

```

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

```

2. Propiedades de animación

Opacidad

```

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

```

Traducir

```

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

```


Escala

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

Girar

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

Lea [Implementando animaciones en nativos en línea](https://riptutorial.com/es/nativescript/topic/5970/implementando-animaciones-en-nativos):

<https://riptutorial.com/es/nativescript/topic/5970/implementando-animaciones-en-nativos>

Capítulo 5: implementar interfaz

Examples

implementar View.OnLayoutChangeListener en 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();
        }
    }
});
```

crear una vista de superficie <http://stackoverflow.com/documentation/proposed/changes/79536>

Añadir escucha:

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

eliminar Oyente:

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

Lea implementar interfaz en línea: <https://riptutorial.com/es/nativescript/topic/5560/implementar-interfaz>

Capítulo 6: Modelo multihilo

Observaciones

El nuevo motor chrome v8 es parcialmente compatible con ES7. Así que si añadimos `"use strict"`; En la parte superior de nuestro archivo (los caracteres tipográficos hacen eso cuando se procesa con transcripción) tenemos que asegurarnos de que cualquier función que esté en el ámbito global se asigne realmente al ámbito global. así que deberíamos usar `self.functionName` o `global.functionName`.

Examples

utilizar trabajadores en servicio 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>
```

Lea Modelo multihilo en línea: <https://riptutorial.com/es/nativescript/topic/7878/modelo-multihilo>

Capítulo 7: Mostrar datos como lista (usando Repeater, ListView o * ngFor para {N} + aplicaciones Angular-2)

Observaciones

Nota: ¡No use Repeater en {N} + aplicaciones Angular-2! El * ngRepeat es una directiva obsoleta en Angular-2. Cuando necesite mostrar patrones de elementos repetidos, use ListView o * ngPara una directiva estructural.

Examples

Uso del módulo Repeater para mostrar datos (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;
}
```

Usando el módulo Repeater con 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;
}

```

Usando el módulo ListView con 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);
    // });
}

```

Usando ListView para mostrar datos (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 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);
}
}

```

Uso de * ngPara la directiva estructural para mostrar datos (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;
  }
}

```

Uso de repetidor con devoluciones de llamada (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>
```

Lea Mostrar datos como lista (usando Repeater, ListView o * ngFor para {N} + aplicaciones Angular-2) en línea: <https://riptutorial.com/es/nativescript/topic/5226/mostrar-datos-como-lista-usando-repeater--listview-o---ngfor-para--n--plus-aplicaciones-angular-2->

Capítulo 8: Plantilla de nativos de estilo

Examples

Agregando un diseño de muestra en tu aplicación

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 { }
```

Método 1: CSS global

app.css : se aplica globalmente a todos los diseños.

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

Método 2: CSS específico de la plataforma

platform.android.css : se aplica globalmente a todos los diseños en dispositivos Android.

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

platform.ios.css : se aplica globalmente a todos los diseños en dispositivos iOS.

```
.btn{
```

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

app.css

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

Método 3: CSS específico del componente

pages / main / main.android.css - Se aplica a componentes específicos en dispositivos Android.

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

pages / main / main.ios.css - Se aplica a componentes específicos en dispositivos ios.

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

pages / main / main-common.css - Se aplica a componentes específicos en todos los dispositivos.

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

Lea [Plantilla de nativos de estilo en línea](https://riptutorial.com/es/nativescript/topic/3872/plantilla-de-nativos-de-estilo): <https://riptutorial.com/es/nativescript/topic/3872/plantilla-de-nativos-de-estilo>

Capítulo 9: usando widget nativo

Examples

Usando surfaceView en ng2-TNS-Android: paso a paso

Por ejemplo, desea utilizar surfaceView en ng2-nativescript. Como no tenemos `surfaceView` en nativescript, debemos usar el `placeholder`.

Primero debemos importar los requisitos:

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

a continuación, agregue el marcador de posición a su archivo html:

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

Agrega este método a tu clase:

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

typescript no sabe qué es `android` y debemos agregar los archivos de declaración de plataforma siguiendo esta [Respuesta](#) para agregarlos.

debido a un [problema](#) en la versión actual de ng2-nativescript, deberíamos hacer un trabajo adicional:

cambiar el marcador de posición a:

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

Importar OnInit:

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

su clase debe implementar OnInit

```
export class AppComponent implements OnInit
```

y agrega estas líneas a tu clase:

```
public init: boolean = false;
```

```
ngOnInit() {
  this.init = true;
}
```

ahora tienes un surfaceView en tu aplicación nativescript :)

Métodos de llamada de SurfaceView

Por ejemplo, desea llamar a `getHolder()` :

agrega una variable y un evento cargado a tu marcador de posición como este:

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

y agrega el método `onLoaded` a tu clase:

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

ATENCIÓN :

No se garantiza que la propiedad de `android` (`element.android`) esté disponible en `ngAfterViewInit` así que usamos un evento `loaded` lugar de eso.

Uso de surfaceView en ng2-TNS-Android: ejemplo completo listo

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() {
    ngOnInit() {
      this.init = true;
    }
  }
}
```

```
}  
}
```

app.component.html:

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

Lea usando widget nativo en línea: <https://riptutorial.com/es/nativescript/topic/5834/usando-widget-nativo>

Capítulo 10: Variables globales

Examples

Consola

La variable de `console` global de NativeScript le permite imprimir valores en su terminal para la depuración. El uso más simple es pasar un valor a la función `console.log()` :

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

El objeto de `console` tiene varios otros métodos, incluidos `dump()` , `trace()` , `assert()` y [más](#) .

```
// 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");
```

Temporizador (JavaScript)

La variable de `timer` global de NativeScript le permite establecer tiempos de espera e intervalos para llamadas de función retardadas asíncronas.

Importador

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

Tiempos de espera

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

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

Intervalos

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



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

Lea Variables globales en línea: <https://riptutorial.com/es/nativescript/topic/3133/variables-globales>

Creditos

S. No	Capítulos	Contributors
1	Empezando con nativescript	Adam Diament , Community , George Edwards , HabibKazemi , Hardik Vaghani , Housseem Yahiaoui , Richard Hubley , user6939352
2	Acceso a apis nativas	HabibKazemi
3	Barra de estado	HabibKazemi
4	Implementando animaciones en nativos	Madhav Poudel
5	implementar interfaz	HabibKazemi
6	Modelo multihilo	HabibKazemi
7	Mostrar datos como lista (usando Repeater, ListView o * ngFor para {N} + aplicaciones Angular-2)	Nick Iliev , Tim Hallyburton , William KLEIN
8	Plantilla de nativos de estilo	George Edwards , Madhav Poudel , Nick Iliev
9	usando widget nativo	HabibKazemi
10	Variables globales	Tim Hallyburton , TJ VanToll