



**Kostenloses eBook**

**LERNEN**

**nativescript**

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

**#nativescript**

**t**

# Inhaltsverzeichnis

Über.....	1
<b>Kapitel 1: Erste Schritte mit Nativescript.....</b>	<b>2</b>
Bemerkungen.....	2
Examples.....	2
Installation oder Setup.....	2
Mac OS.....	2
Windows.....	2
Verwenden von Visual Studio Code für die NativeScript-Entwicklung.....	3
Dein erstes Hello World Programm.....	3
So debuggen Sie die Nativescript-Android-App über WLAN (ohne Root).....	4
<b>Kapitel 2: Animationen in Nativescript implementieren.....</b>	<b>6</b>
Examples.....	6
Hintergrundanimation von StackLayout.....	6
Verwendung der Animationszeitfunktion und der Animationseigenschaften.....	6
<b>Opazität.....</b>	<b>7</b>
<b>Übersetzen.....</b>	<b>7</b>
<b>Rahmen.....</b>	<b>7</b>
<b>Drehen.....</b>	<b>8</b>
<b>Kapitel 3: Anzeigen von Daten als Liste (mit Repeater, ListView oder * ngFor für {N} + Ang.....</b>	<b>9</b>
Bemerkungen.....	9
Examples.....	9
Verwenden des Repeater-Moduls zur Anzeige von Daten (NativeScript Core).....	9
Verwenden des Repeater-Moduls mit ObservableArray (NativeScript Core).....	9
Verwenden des ListView-Moduls mit ObservableArray (NativeScript Core).....	10
Verwenden von ListView zum Anzeigen von Daten (NativeScript + Angular-2).....	11
* NgFor Structural Directive zur Anzeige von Daten verwenden (nativeScript + Angular-2).....	12
Repeater mit Rückrufen verwenden (JavaScript).....	12
<b>Kapitel 4: Globale Variablen.....</b>	<b>14</b>
Examples.....	14
Konsole.....	14

Timer (JavaScript).....	14
<b>Kapitel 5: Multithreading-Modell.....</b>	<b>16</b>
Bemerkungen.....	16
Examples.....	16
Verwenden Sie Worker in angle2 Service.....	16
<b>Kapitel 6: natives Widget verwenden.....</b>	<b>19</b>
Examples.....	19
Mit surfaceView in ng2-TNS-Android: Schritt für Schritt.....	19
Verwenden von surfaceView in ng2-TNS-Android: Vollständiges Beispiel.....	20
<b>Kapitel 7: Schnittstelle implementieren.....</b>	<b>22</b>
Examples.....	22
implementieren Sie View.OnLayoutChangeListener in Nativescript.....	22
<b>Kapitel 8: Statusleiste.....</b>	<b>23</b>
Examples.....	23
Hide / Show - Android.....	23
Statusleiste transparent machen Android.....	23
<b>Kapitel 9: Vorlage für Nativescript-Vorlage.....</b>	<b>24</b>
Examples.....	24
Hinzufügen eines Musterlayouts in Ihrer App.....	24
<b>Methode 1: Globales CSS.....</b>	<b>24</b>
<b>Methode 2: Plattformspezifisches CSS.....</b>	<b>24</b>
<b>Methode 3: Komponentenspezifisches CSS.....</b>	<b>25</b>
<b>Kapitel 10: Zugriff auf native Apis.....</b>	<b>26</b>
Examples.....	26
Schreiben Sie Java-Code in Nativescript und verwenden Sie ihn direkt in Javascript.....	26
Verwenden Sie native Apis direkt in Javascript.....	29
<b>Credits.....</b>	<b>31</b>



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](mailto:info@zzzprojects.com)

---

# Kapitel 1: Erste Schritte mit Nativescript

## Bemerkungen

Nativescript ist eine hochperformante plattformübergreifende mobile App-Laufzeitumgebung, mit der Sie iOS und Android (mit Fenstern in der Pipeline) mithilfe von Webtechnologien (JS und HTML) ansprechen können. Es wurde mit einer Reihe von Hauptzielen erstellt:

- Visuell performant: kein UI-Jank, auch bei Android haben Sie butterweiche fps
- Erweiterbar: Sie haben Zugriff auf alle nativen APIs, um plattformübergreifende Plugins zu erstellen
- Vollständig native Benutzeroberfläche
- Hochgradig integriert mit Typoscript und Angular 2
- Open Source mit starker Unternehmensunterstützung von Telerik

## Examples

### Installation oder Setup

Detaillierte Anweisungen zum Einrichten oder Installieren von Nativescript.

Die folgenden Beispiele zeigen die erforderlichen Schritte zum Einrichten eines Windows- oder OSX-Systems und zum Signieren von Postfehlern zu Fehlerbehebungshandbüchern, falls Probleme auftreten.

Darüber hinaus gibt es Beispiele für die Einrichtung empfohlener Workflows, IDEs und Emulatoren.

### Mac OS

1. Stellen Sie sicher, dass Sie das [neueste](#) LTS für Node.js installiert haben. Wenn Sie [Homebrew verwenden](#), können Sie dies mit `brew install node4-lts`.
2. Öffnen Sie das Terminal und geben Sie `npm install -g nativescript`. Wenn Sie einen `EACCES` Fehler erhalten, verwenden Sie `sudo npm install -g nativescript`.
3. `ruby -e "$(curl -fsSL https://www.nativescript.org/setup/mac)"` der Eingabeaufforderung `ruby -e "$(curl -fsSL https://www.nativescript.org/setup/mac)"`. (Dies kann eine Weile dauern.)
4. Um zu überprüfen, ob das oben beschriebene funktioniert hat, geben Sie `tns doctor` in Terminal ein.
5. Wenn Fehler auftreten, folgen Sie der [Anleitung](#) zur [Fehlerbehebung](#).

### Windows

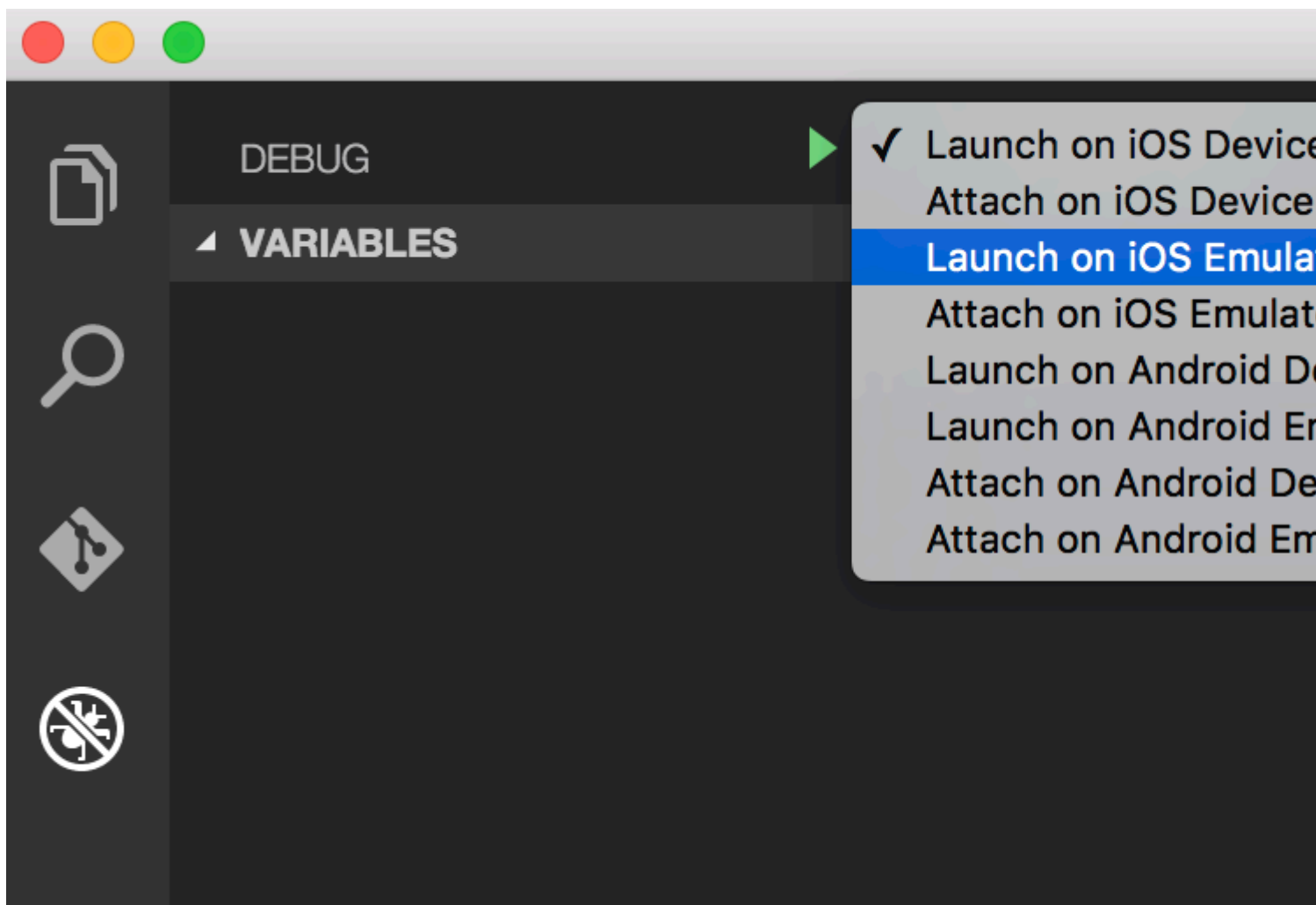
1. Stellen Sie sicher, dass Sie den neuesten [nodeJS LTS](#) installiert haben
2. Öffnen Sie die Eingabeaufforderung, und geben Sie `$ npm install -g nativescript`

3. `$ @powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-object net.webclient).DownloadString('https://www.nativescript.org/setup/win'))"` **der Eingabeaufforderung** `$ @powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-object net.webclient).DownloadString('https://www.nativescript.org/setup/win'))"` **Dies könnte der** `$ @powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-object net.webclient).DownloadString('https://www.nativescript.org/setup/win'))"` **sein dauert eine Weile**
4. `$ tns doctor` in der Eingabeaufforderung (Ihre Cmd) ein, um zu überprüfen, ob die obigen `$ tns doctor`
5. Wenn Fehler auftreten, folgen Sie der [Anleitung](#) zur [Fehlerbehebung](#)

## Verwenden von Visual Studio Code für die NativeScript-Entwicklung

[Visual Studio Code](#) ist ein Open-Source-Code-Editor mit vielen Funktionen von Microsoft. Um es für die NativeScript-Entwicklung einzurichten, öffnen Sie die Befehlspalette ( `F1` oder `⌘ + Umschalt + P` ) und geben Sie `ext install NativeScript` .

Sobald die NativeScript-Erweiterung installiert ist, sollten Sie mit dem Debugger Haltepunkte in Ihrem Code festlegen. Wenn ein Gerät angeschlossen ist oder ein Emulator ausgeführt wird, können Sie Ihre App über die Registerkarte Debug starten.



## Dein erstes Hello World Programm

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

Stellen Sie dann sicher, dass ein Gerät angeschlossen ist oder ein Emulator ausgeführt wird (andernfalls wird der Standardemulator gestartet oder ein Fehler wird angezeigt. Ich würde genymotion für Android empfehlen).

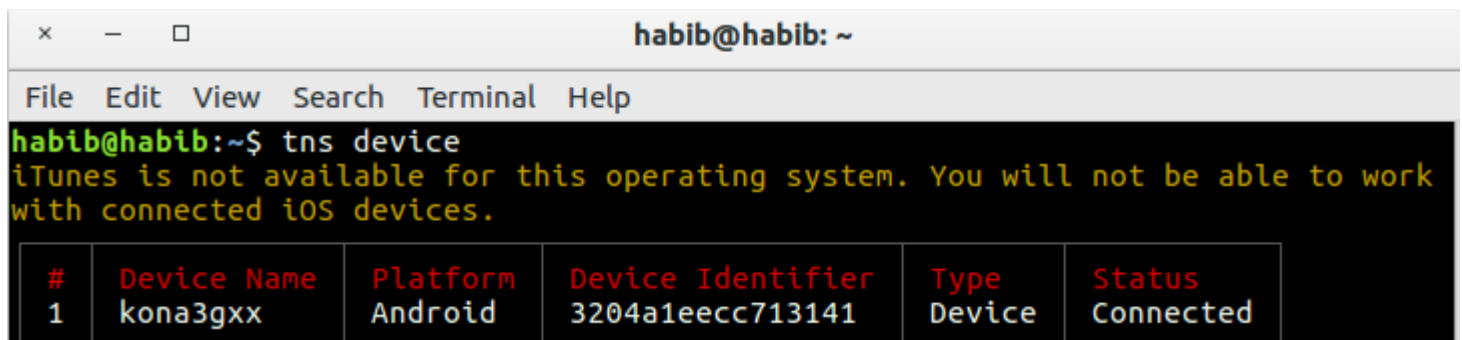
```
$ tns run android
```

Wenn Sie den Standard-Android-Emulator verwenden möchten, fügen Sie das Flag `--emulator`.

Ab tns 2.5 ist Livesync nun die Standardaktion für `tns run <platform>`, die beim Speichern von Dateiänderungen automatisch neu kompiliert wird. Dies kann Ihre Entwicklungszeit drastisch verbessern. Wenn Sie jedoch Änderungen an Ihren Plugins vornehmen, müssen Sie sie erneut kompilieren.

## So debuggen Sie die Nativescript-Android-App über WLAN (ohne Root)

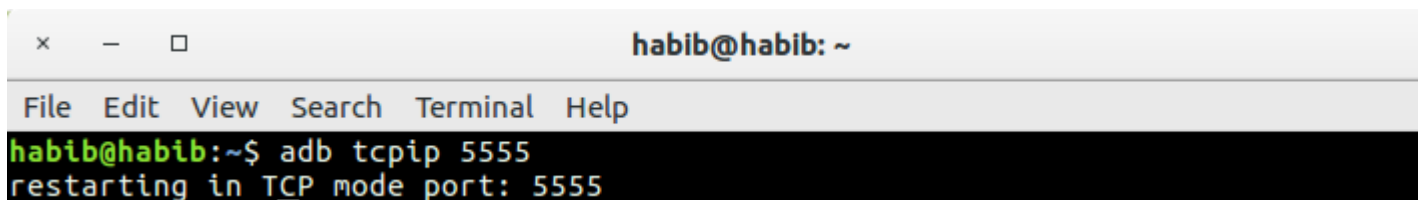
1-Sie müssen Ihr Gerät über ein USB-Kabel an Ihren Computer anschließen. Stellen Sie sicher, dass das USB-Debugging funktioniert. Sie können überprüfen, ob es beim Ausführen von `adb devices` (oder `tns device`) `tns device`.



```
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` - `adb tcpip 5555`



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

3-Trennen Sie Ihr Gerät (entfernen Sie das USB-Kabel).

4-Gehen Sie zu Einstellungen -> Über Telefon -> Status, um die IP-Adresse Ihres Telefons anzuzeigen.

5- `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-Wenn Sie `adb devices` (oder `tns device`) erneut `tns device`, sollte Ihr Gerät angezeigt werden.

```
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- Jetzt können Sie `tns run android` und `tns livesync android` Befehle verwenden.

### ANMERKUNGEN :

1 - Wenn sich das WLAN-Netzwerk ändert, müssen Sie die Schritte 1 bis 3 nicht wiederholen (dadurch wird Ihr Telefon in den WLAN-Debug-Modus versetzt). Sie müssen sich erneut mit Ihrem Telefon verbinden, indem Sie die Schritte 4 bis 6 ausführen.

2-Android-Telefone verlieren beim Neustart den WLAN-Debug-Modus. Wenn Ihre Batterie dann leer ist, müssen Sie von vorne beginnen. Wenn Sie sonst den Akku im Auge behalten und das Telefon nicht neu starten, können Sie wochenlang ohne Kabel leben!

### WARNUNG :

Wenn Sie die Option aktiviert lassen, ist dies gefährlich. Jeder Benutzer in Ihrem Netzwerk kann sich bei einem Debugging mit Ihrem Gerät verbinden, selbst wenn Sie sich im Datennetzwerk befinden. Tun Sie dies nur, wenn Sie mit einem vertrauenswürdigen WLAN verbunden sind, und trennen Sie die Verbindung, wenn Sie fertig sind.

### referenz :

1-Norman Peitek. 2014. Debuggen Ihrer Android-App über WLAN (ohne Root!). [ONLINE] Verfügbar unter: <https://futurestud.io/blog/how-to-debug-ihre-android-app-over-wifi-ohne-wurzelfrei> . [Zugriff am 8. August 2016].

2-usethe4ce. Android-Anwendungen über WLAN ausführen / installieren / debuggen ?. [ONLINE] Verfügbar unter: <http://stackoverflow.com/a/10236938/4146943> . [Zugriff am 8. August 2016].

Erste Schritte mit Nativescript online lesen: <https://riptutorial.com/de/nativescript/topic/921/erste-schritte-mit-nativescript>



# Kapitel 2: Animationen in Nativescript implementieren

## Examples

### Hintergrundanimation von StackLayout

Animieren der Hintergrundfarbe des Stacklayouts beim Tippen auf die Schaltfläche

*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
    });
  }
}
```

*seiten / main-common.css*

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

### Verwendung der Animationszeitfunktion und der Animationseigenschaften.

*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 für andere Animationseigenschaften

Mit *cubicBezier* können Sie auch Ihre eigene *Timing-Funktion* schreiben.

### 1. Verwendung von CubicBezier

```

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

```

### 2. Animationseigenschaften

## Opazität

```

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

```

## Übersetzen

```

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

```

# Rahmen

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

---

# Drehen

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

Animationen in Nativescript implementieren online lesen:

<https://riptutorial.com/de/nativescript/topic/5970/animationen-in-nativescript-implementieren>

# Kapitel 3: Anzeigen von Daten als Liste (mit Repeater, ListView oder \* ngFor für {N} + Angular-2-Apps)

## Bemerkungen

Hinweis: Verwenden Sie den Repeater nicht in {N} + Angular-2-Anwendungen! Die \* ngRepeat-Anweisung ist in Angular-2 veraltet. Wenn Sie sich wiederholende Elementmuster anzeigen möchten, verwenden Sie entweder die ListView- oder \* ngFor-Direktive.

## Examples

### Verwenden des Repeater-Moduls zur Anzeige von Daten (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;
}
```

### Verwenden des Repeater-Moduls mit 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;
}
```

## Verwenden des ListView-Moduls mit 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);
    // });
}

```

## Verwenden von ListView zum Anzeigen von Daten (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);
  }
}

```

## \* NgFor Structural Directive zur Anzeige von Daten verwenden (nativeScript + Angular-2)

### *ngfor.component.html*

```

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

```

### *ngfür.Komponente.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;
  }
}

```

## Repeater mit Rückrufen verwenden (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>
```

Anzeigen von Daten als Liste (mit Repeater, ListView oder \* ngFor für {N} + Angular-2-Apps)  
online lesen: <https://riptutorial.com/de/nativescript/topic/5226/anzeigen-von-daten-als-liste-mit-repeater--listview-oder---ngfor-fur--n--plus-angular-2-apps->



# Kapitel 4: Globale Variablen

## Examples

### Konsole

Mit der globalen `console` NativeScript können Sie Werte zum Debuggen an Ihr Terminal drucken. Die einfachste Verwendung besteht darin, einen Wert an die `console.log()` Funktion zu übergeben:

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

Das `console` verfügt über mehrere andere Methoden, einschließlich `dump()`, `trace()`, `assert()` und [mehr](#).

```
// 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)

Mit der globalen `timer` Variablen von NativeScript können Sie Timeouts und Intervalle für asynchrone verzögerte Funktionsaufrufe festlegen.

#### Importieren

```
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);
```

#### Intervalle

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

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

Globale Variablen online lesen: <https://riptutorial.com/de/nativescript/topic/3133/globale-variablen>

# Kapitel 5: Multithreading-Modell

## Bemerkungen

Der neue Chrome-V8-Motor ist teilweise ES7-konform. Wenn wir also "use strict"; hinzufügen "use strict"; Zum Anfang unserer Datei (Typoscript tun dies, wenn Typoscript transpiles ist) müssen wir sicherstellen, dass alle Funktionen, die sich im globalen Gültigkeitsbereich befinden, tatsächlich dem globalen Gültigkeitsbereich zugewiesen werden. Wir sollten also

```
self.functionName oder global.functionName .
```

## Examples

### Verwenden Sie Worker in angle2 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>
```

Multithreading-Modell online lesen: <https://riptutorial.com/de/nativescript/topic/7878/multithreading-modell>

# Kapitel 6: natives Widget verwenden

## Examples

### Mit surfaceView in ng2-TNS-Android: Schritt für Schritt

Zum Beispiel möchten Sie surfaceView in ng2-nativescript verwenden. Da surfaceView in nativescript vorhanden ist, sollten wir placeholder .

Zuerst sollten wir die Anforderungen importieren:

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

Fügen Sie dann den Platzhalter zu Ihrer HTML-Datei hinzu:

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

Fügen Sie diese Methode Ihrer Klasse hinzu:

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

Typoskript weiß nicht , was ist android und wir sollten Plattform Deklarationsdateien folgen Sie diesem hinzufügen [Antwort](#) , um sie hinzuzufügen.

wegen eines [Problems](#) in der aktuellen Version von ng2-nativescript sollten wir einige zusätzliche Arbeit erledigen:

Ändern Sie den Platzhalter in:

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

OnInit importieren:

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

Ihre Klasse sollte OnInit implementieren

```
export class AppComponent implements OnInit
```

und füge diese Zeilen zu deiner Klasse hinzu:

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

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

Jetzt haben Sie eine Oberflächenansicht in Ihrer Nativescript-App :)

## Rufen Sie Methoden von SurfaceView auf

Zum Beispiel möchten Sie `getHolder()` aufrufen:

Fügen Sie Ihrem Platzhalter eine Variable und ein geladenes Ereignis folgendermaßen hinzu:

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

und fügen Sie der Klasse die `onLoaded`-Methode hinzu:

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

## ACHTUNG :

Es ist nicht garantiert, dass die `android` Eigenschaft ( `element.android` ) in `ngAfterViewInit` verfügbar ist. `ngAfterViewInit` haben wir stattdessen ein `loaded` Ereignis verwendet.

## Verwenden von surfaceView in ng2-TNS-Android: Vollständiges Beispiel

### 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>
```

natives Widget verwenden online lesen: <https://riptutorial.com/de/nativescript/topic/5834/natives-widget-verwenden>



# Kapitel 7: Schnittstelle implementieren

## Examples

### implementieren Sie 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();
        }
    }
});
```

Erstellen Sie eine Oberflächenansicht

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

Listener hinzufügen:

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

Listener entfernen:

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

Schnittstelle implementieren online lesen:

<https://riptutorial.com/de/nativescript/topic/5560/schnittstelle-implementieren>

# Kapitel 8: Statusleiste

## Examples

### Hide / Show - Android

Dies ist eine Statusleiste, die oben auf dem Bildschirm mit Symbolen für die Uhr, Uhr usw. angezeigt wird.



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

### Verbergen:

```
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 );
```

## Statusleiste transparent machen Android

Öffnen Sie `APP_Resources/values/styles.xml` und fügen Sie die hinzu

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

in dem

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

Sektion.

Statusleiste online lesen: <https://riptutorial.com/de/nativescript/topic/6007/statusleiste>

---

# Kapitel 9: Vorlage für Nativescript-Vorlage

## Examples

### Hinzufügen eines Musterlayouts in Ihrer App

#### *Hauptkomponente.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 { }
```

---

## Methode 1: Globales CSS

*app.css* - Gilt global für alle Layouts.

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

---

## Methode 2: Plattformspezifisches CSS

*platform.android.css* - Gilt global für alle Layouts im Android-Gerät.

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

*platform.ios.css* - Gilt global für alle Layouts in iOS-Geräten.

```
.btn{
```

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

*app.css*

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

---

## Methode 3: Komponentenspezifisches CSS

*pages / main / main.android.css* - Gilt für bestimmte Komponenten in Android-Geräten.

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

*pages / main / main.ios.css* - Gilt für bestimmte Komponenten in iOS-Geräten.

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

*pages / main / main-common.css* - Gilt für bestimmte Komponenten in allen Geräten.

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

Vorlage für Nativescript-Vorlage online lesen:

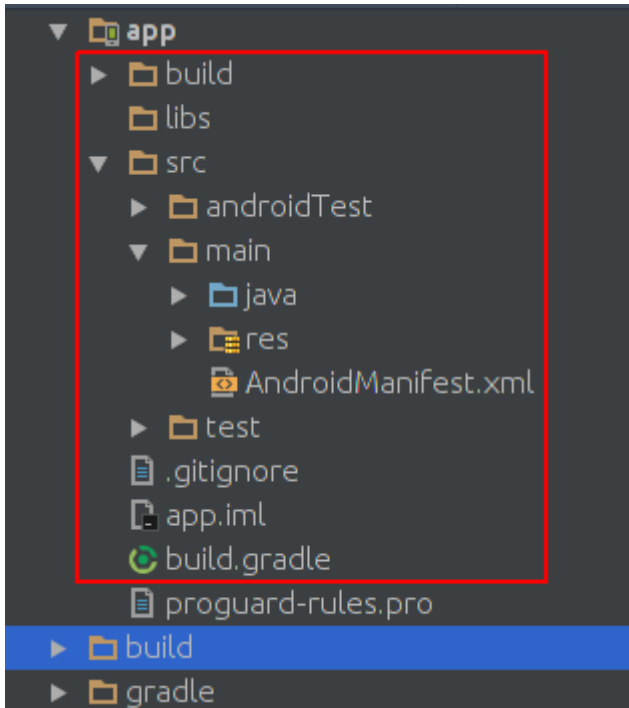
<https://riptutorial.com/de/nativescript/topic/3872/vorlage-fur-nativescript-vorlage>

# Kapitel 10: Zugriff auf native Apis

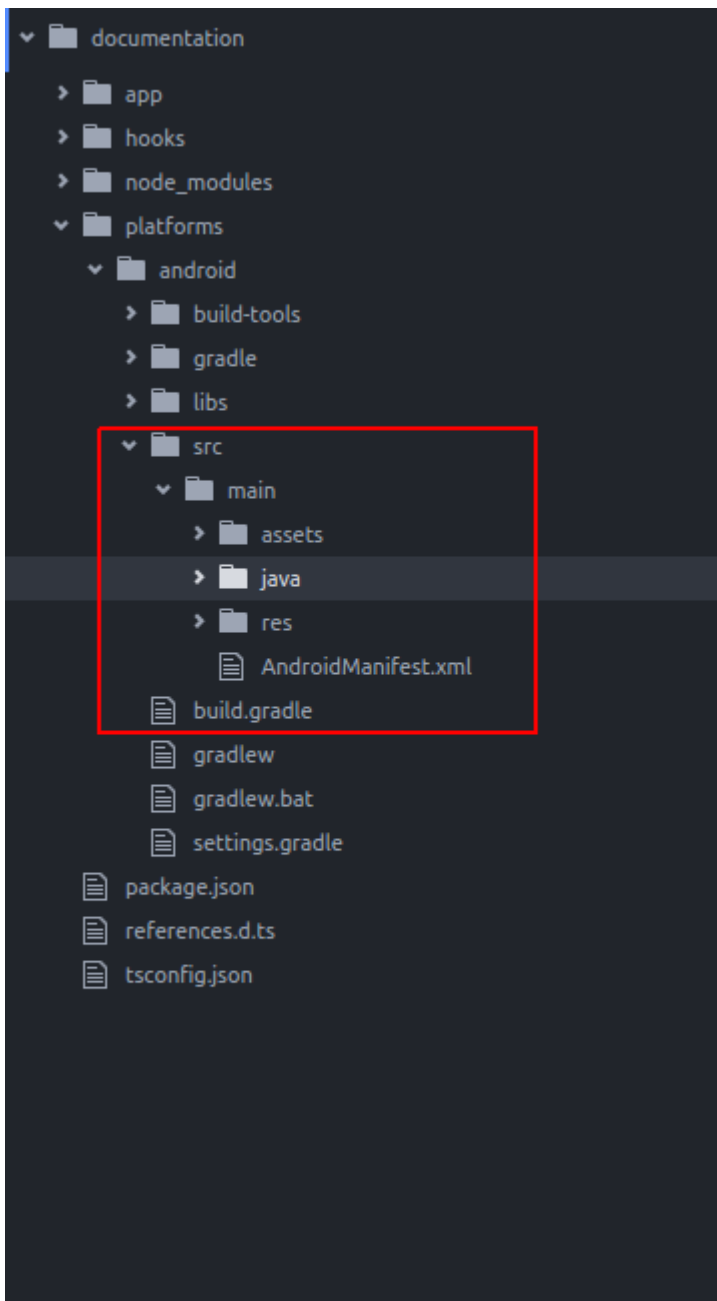
## Examples

Schreiben Sie Java-Code in Nativescript und verwenden Sie ihn direkt in Javascript

Dies ist das Bild der Projektstruktur in Android Studio:

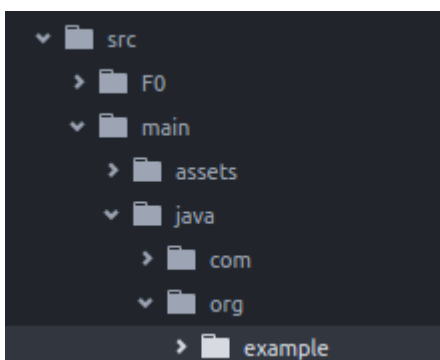


Dies ist das Bild der Projektstruktur des Nativescript-Projekts:



Wie Sie sehen, sind sie gleich. So können wir Java-Code in Nativescript schreiben, während wir in Android Studio schreiben.

Wir möchten Toast zur Standard-App von nativescript hinzufügen. Nachdem Sie ein neues Nativescript-Projekt erstellt haben, erstellen Sie ein Verzeichnis wie das `java/org/example` Verzeichnis:



Erstellen Sie eine neue Datei `MyToast.java` im `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();
    }
}
```

**Hinweise :** Den Paketnamen nicht vergessen.

### ***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");
    }
}
```

Wenn Sie jetzt die Taste drücken, wird ein Toast angezeigt.

## Anmerkungen :

1. Die showToast-Funktion akzeptiert den Kontext, um ihn an `Toast.makeText` **ZU** `Toast.makeText` wir übergeben ihm einen Kontext auf folgende Weise: `application.android.context`
2. typescript weiß nicht, was `org` ist, also haben wir es deklariert: `declare var org:any;`

## Verwenden Sie native Apis direkt in Javascript

Wir möchten Toast zur Standard-App für Nativescript hinzufügen.

```
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();
  }
}
```

`Toast.makeText` **Toast** zu erstellen, sollten wir `Toast.makeText` und es ist im `android.widget.Toast` Paket enthalten. `Toast.makeText` akzeptiert Kontext als erstes Argument und wir können den Kontext in Nativescript auf folgende Weise abrufen: `application.android.context`

Zugriff auf native Apis online lesen: <https://riptutorial.com/de/nativescript/topic/5188/zugriff-auf->





# Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Nativescript	<a href="#">Adam Diament</a> , <a href="#">Community</a> , <a href="#">George Edwards</a> , <a href="#">HabibKazemi</a> , <a href="#">Hardik Vaghani</a> , <a href="#">Housseem Yahiaoui</a> , <a href="#">Richard Hubley</a> , <a href="#">user6939352</a>
2	Animationen in Nativescript implementieren	<a href="#">Madhav Poudel</a>
3	Anzeigen von Daten als Liste (mit Repeater, ListView oder * ngFor für {N} + Angular-2-Apps)	<a href="#">Nick Iliev</a> , <a href="#">Tim Hallyburton</a> , <a href="#">William KLEIN</a>
4	Globale Variablen	<a href="#">Tim Hallyburton</a> , <a href="#">TJ VanToll</a>
5	Multithreading-Modell	<a href="#">HabibKazemi</a>
6	natives Widget verwenden	<a href="#">HabibKazemi</a>
7	Schnittstelle implementieren	<a href="#">HabibKazemi</a>
8	Statusleiste	<a href="#">HabibKazemi</a>
9	Vorlage für Nativescript-Vorlage	<a href="#">George Edwards</a> , <a href="#">Madhav Poudel</a> , <a href="#">Nick Iliev</a>
10	Zugriff auf native Apis	<a href="#">HabibKazemi</a>