無料電子ブック

学習

nativescript

Free unaffiliated eBook created from
Stack Overflow contributors.

#nativescrip

t

1: nativescriptをいめる

Nativescriptは、ウェブJSおよびhtmlをして、iOSとAndroidパイプラインのウィンドウをターゲットにすることができる、パフォーマンスのいクロスプラットフォームのモバイルアプリランタイムです。それはいくつかのなでられました

- にでるものUIのないは、
- すべてのネイティブAPIにアクセスできるため、なクロスプラットフォームプラグインをできます
- にネイティブなUI
- TypescriptとAngular 2とのな
- オープンソース、Telerikからのな

Examples

インストールまたはセットアップ

Nativescriptをまたはインストールするの。

のは、WindowsまたはOSXシステムをセットアップし、がしたにえて、トラブルシューティングガイドににするためになをしています。

また、されるワークフロー、IDE、エミュレータののもあります。

マックOS

1. の Node.js LTSがインストールされていることをしてください。 Homebrewをしている、これは`brew install node4-lts`できます。
2. ターミナルをき、 `npm install -g nativescript`ます。 `EACCES`エラーがしたは、 `sudo npm install -g nativescript`して`sudo npm install -g nativescript`。
3. コマンドプロンプトで`ruby -e "$(curl -fsSL https://www.nativescript.org/setup/mac)"`ます。これにはがかかることがあります。
4. がしていることをするには、ターミナルに`tns doctor`します。
5. エラーがあるは、 トラブルシューティングガイドをフォローアップしてください 。

Windows

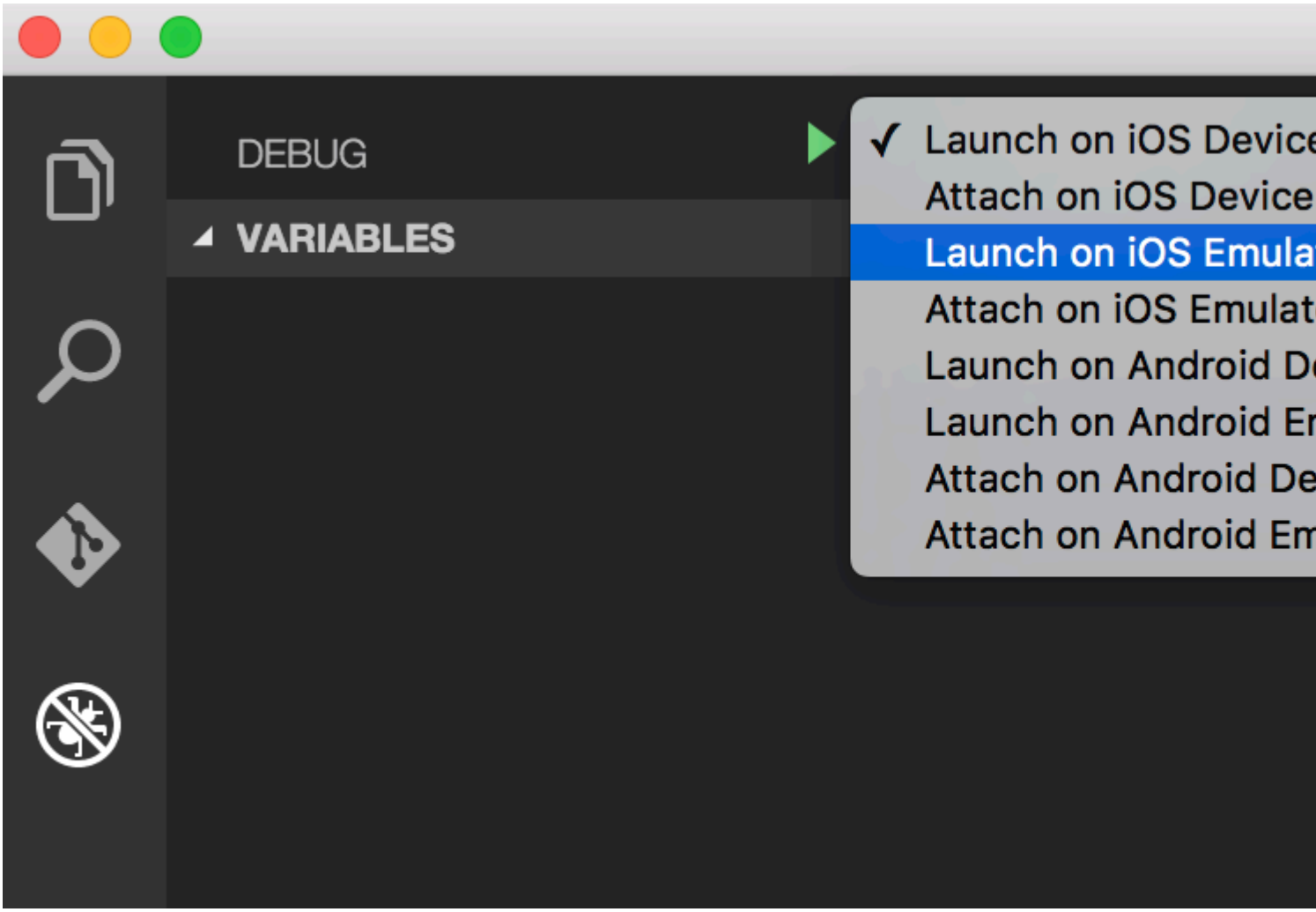
1. のnodeJS LTSがインストールされていることをする
2. コマンドプロンプトをき、 `$ npm install -g nativescript`とします。
3. コマンドプロンプトで、 `$ @powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-object net.webclient).DownloadString('https://www.nativescript.org/setup/win'))"`しばらくがかかります
4. がしていることをするには、コマンドプロンプトで`$ tns doctor`としますあなたのcmd

5. エラーがあるは、 トラブルシューティングガイドをフォローアップしてください

NativeScriptのVisual Studioコードの

Visual Studioコードは、Microsoftのオープンソースでなコードエディタです。 NativeScriptにするには、コマンドパレット `F1`または `Ctrl` + `Shift` + `P` をき、 `ext install NativeScript`とします。

NativeScriptエクステンションがインストールされると、デバッガでコードにブレークポイントをできるようになります。デバイスがされている、またはエミュレータがしているは、[デバッグ]タブからアプリをできます。



あなたのHello Worldプログラム

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

に、デバイスがされているかエミュレータがしていることをしますそうでない、デフォルトのエミュレータがするか、エラーがします。

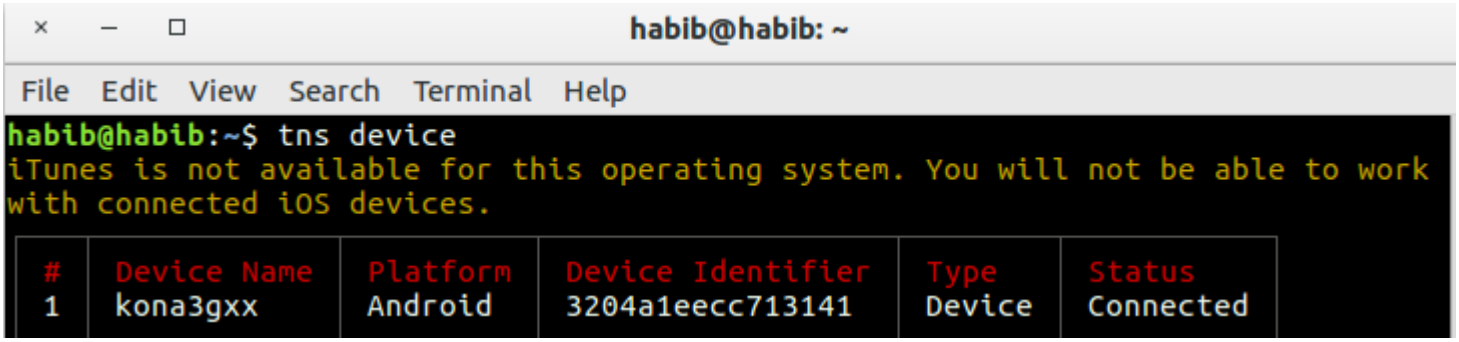
```
$ tns run android
```

デフォルトのアンドロイドエミュレータをするは、-- `--emulator`フラグをします。

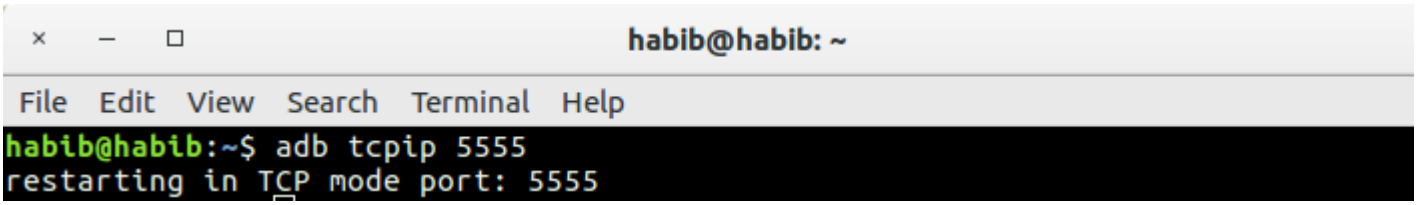
tns 2.5ので、livesnsyncは`tns run <platform>`のデフォルトアクションになりました。これはファイルのをするとにコンパイルされます。これによりがにされますが、プラグインにをえたは、にコンパイルするがあります。

WiFiでnativescript-androidアプリケーションをデバッグするルートなし

1 - USBケーブルでデバイスをコンピュータにするがあります。 USBデバッグがしていることをしてください。 `adb devices` または`tns device` のにされるかどうかをできます。



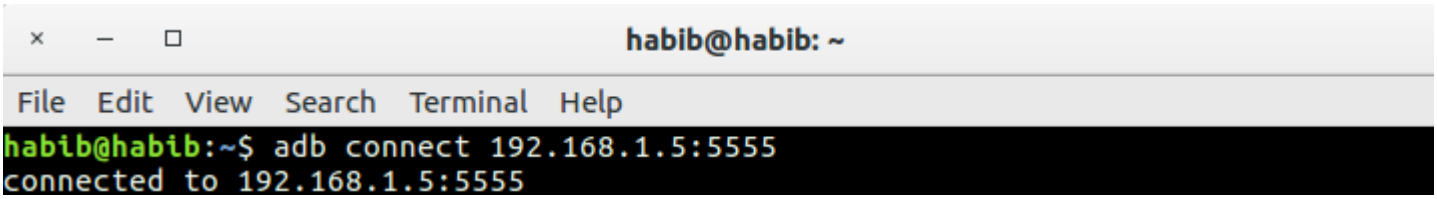
2ラン`adb tcpip 5555`



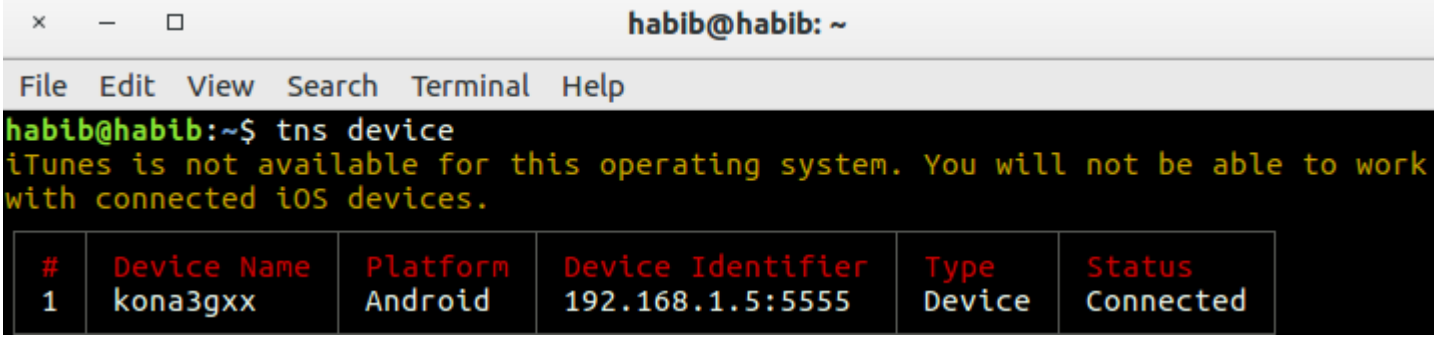
3 - デバイスをりしますUSBケーブルをします。

4 - [] - > [について] - > [ステータス]にして、おいののIPアドレスをします。

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



6 - `adb devices` または`tns device` をすると、デバイスがされます。



7-、あなたは、 `tns run android` 、 `tns livesync android`コマンドを`tns run android`する`tns run android`をすることができます。

ノート

1 - WiFiネットワークがされた、13これらはWi-Fiデバッグモードにされていますをりすはありません。46をして、にするがあります。

2-Androidは、にWi-Fiデバッグモードをいます。したがって、がしたは、からやりすがあります。それのは、あなたのバッテリーをって、あなたのをしない、あなたはケーブルなしでもきることができます

オプションをにしたままにしておくとです。データネットワークにいても、ネットワークのでもデバッグでデバイスにできます。できるWi-Fiにしているときにのみし、したらをすることをれないでください

リファレンス

1-Norman Peitek。 WiFiであなたのAndroid Appをデバッグするルートなし。 [オンライン] https://futurestud.io/blog/how-to-debug-your-android-app-over-wifi-without-rootからできます 。 [201688アクセス]

2-usethece。 Wi-FiでAndroidアプリケーションを/インストール/デバッグしますか [オンライン] http : //stackoverflow.com/a/10236938/4146943 。 [201688アクセス]

オンラインでnativescriptをいめるをむ https://riptutorial.com/ja/nativescript/topic/921/nativescriptをいめる

2: インターフェースをする

Examples

NativescriptにView.OnLayoutChangeListenerをする

```
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();
          }
      }
    });
```

surfaceViewをするhttp://stackoverflow.com/documentation/proposed/changes/79536

リスナーを

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

リスナーを

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

オンラインでインターフェースをするをむ https://riptutorial.com/ja/nativescript/topic/5560/インターフェースをする

3: グローバル

Examples

コンソール

NativeScriptのグローバル`console`をすると、にをしてデバッグすることができます。もないは、`console.log()`にをすことです。

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

`console`オブジェクトには、 `dump()` 、 `trace()` 、 `assert()`などのいくつかのメソッドがありま す 。

```
// 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");
```

タイマーJavaScript

NativeScriptのグローバル`timer`をすると、のびしのタイムアウトとをできます。

インポート

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

タイムアウト

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

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

インターバル

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

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

オンラインでグローバルをむ https://riptutorial.com/ja/nativescript/topic/3133/グローバル

4: ステータスバー

Examples

I - アンドロイド

これはステータスバーで、あなたののに、のアイコンがされます。


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

す

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

ショー

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

statusBarをなアンドロイドにする

`APP_Resources/values/styles.xml`をき、

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

のに

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

セクション。

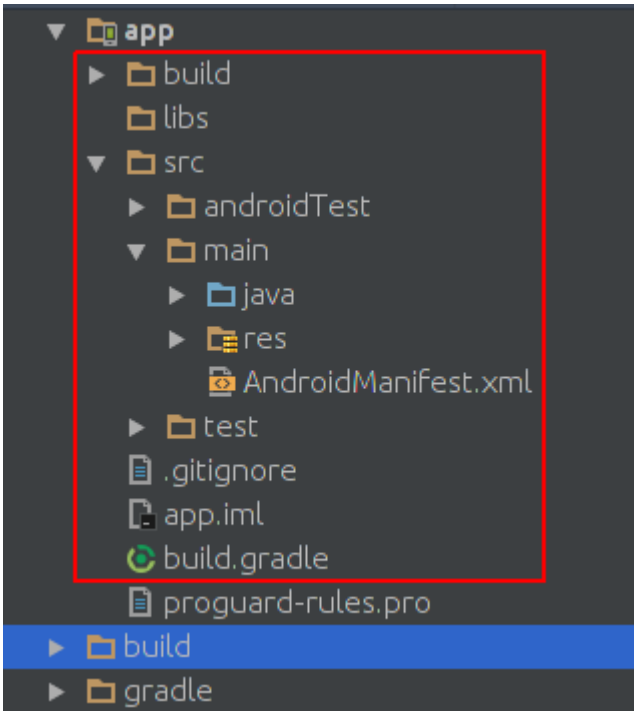
オンラインでステータスバーをむ https://riptutorial.com/ja/nativescript/topic/6007/ステータスバー

5: ネイティブAPIへのアクセス

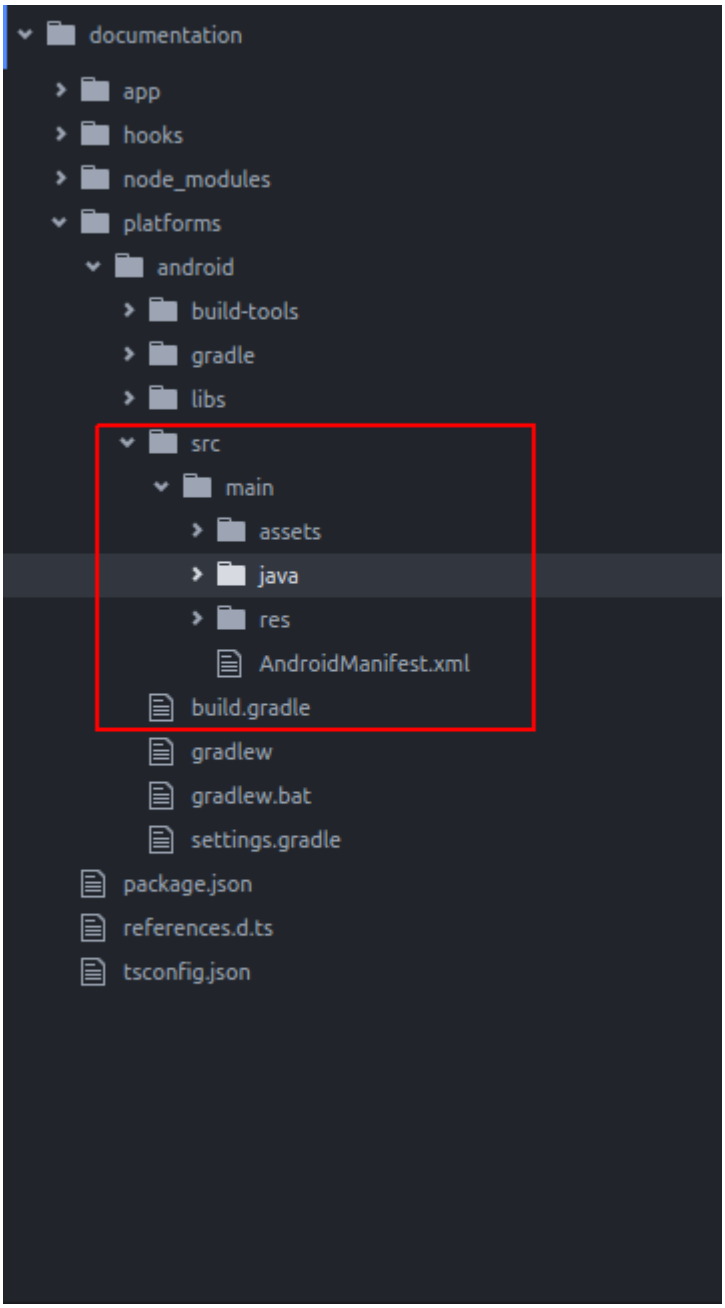
Examples

nativescriptにJavaコードをし、javascriptでする

これはAndroidスタジオのプロジェクトのイメージです

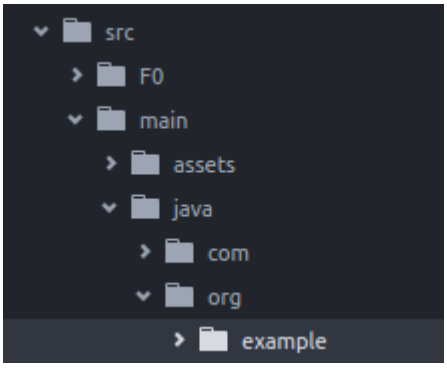


これは、nativescriptプロジェクトのプロジェクトのイメージです。



あなたがるり、らはじです。たちはアンドロイドスタジオでくので、たちはnativescriptでJavaコードをくことができます。

nativescriptののアプリケーションにToastをします。しいnativescriptプロジェクトをした、のように`java/org/example`ディレクトリにディレクトリをします。



`example`ディレクトリにしい`MyToast.java`ファイルをします。

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();
    }
}
```

パッケージはれないでください。

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");
    }
}
```

ボタンをすとトーストがされます。

1. showToastはコンテキストをけれて`Toast.makeText`にします。このようにコンテキストをします `application.android.context`
2. typescriptはどのような`org`かからないので、それをしました `declare var org:any;`

ネイティブapisをjavascriptでする

はToastをnativescriptのデフォルトアプリにしたいとえています。

```
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();
   }
}
```

`Toast.makeText`をするために`Toast.makeText`をびす`Toast.makeText`あり、それは`android.widget.Toast`パッケージにあります。 `Toast.makeText`はのとしてコンテキストをけり、このようにしてnativescriptでコンテキストをできます `application.android.context`

オンラインでネイティブAPIへのアクセスをむ https://riptutorial.com/ja/nativescript/topic/5188/ネイティブapiへのアクセス

6: ネイティブウィジェットをして

Examples

ng2-TNS-AndroidでのsurfaceViewのステップバイステップ

たとえば、ng2-nativescriptでsurfaceViewをしたいとします。 `surfaceView`でsurfaceViewをしていないため、 `placeholder`をするがあります。

にをインポートするがあります。

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

htmlファイルにプレースホルダをします

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

このメソッドをクラスにします。

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

typescriptですがであるかをらない`android` 、 々はプラットフォームファイルは、このわするがをそれらをします。

のバージョンのng2-nativescriptにがあるため、をうがあります。

プレースホルダをのようにします。

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

インポートOnInit

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

クラスでOnInitをするがあります

```
export class AppComponent implements OnInit
```

あなたのクラスにのをしてください

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

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

あなたはあなたのnativescriptアプリでsurfaceViewをっています:)

SurfaceViewのびしメソッド

たとえば、 `getHolder()` をびすとします。

のように、とロードされたイベントをプレースホルダにします。

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

クラスにonLoadedメソッドをします。

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

`android`プロパティ `element.android` が`ngAfterViewInit`でできることはされていないので、わりに`loaded`イベントをしました。

ng2-TNS-AndroidでのsurfaceViewのな

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>
```

オンラインでネイティブウィジェットをしてをむ https://riptutorial.com/ja/nativescript/topic/5834/ネイティブウィジェットをして

7: ネイティブスクリプトテンプレートのスタイル

Examples

あなたのアプリにサンプルレイアウトをする

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 { }
```

1 グローバルCSS

app.css - すべてのレイアウトにグローバルにされます。

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

2 プラットフォームのCSS

platform.android.css - Androidデバイスのすべてのレイアウトにグローバルにされます。

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

platform.ios.css - iosデバイスのすべてのレイアウトにグローバルにされます。

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

app.css

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

3コンポーネントのCSS

pages / main / main.android.css - アンドロイドデバイスののコンポーネントにされます。

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

pages / main / main.ios.css - iosデバイスののコンポーネントにされます。

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

pages / main / main-common.css - すべてのデバイスののコンポーネントにされます。

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

オンラインでネイティブスクリプトテンプレートのスタイルをむ
https://riptutorial.com/ja/nativescript/topic/3872/ネイティブスクリプトテンプレートのスタイル

8: ネイティブスクリプトによるアニメーションの

Examples

StackLayoutのアニメーション

タッピングボタンでのスタックレイアウトののアニメーション

pages / main.component.ts

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

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

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

pages / main-common.css

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

アニメーションタイミングとアニメーションプロパティの。

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
        });
    }
}
```

のアニメーションプロパティの**#snippet**

*cubicBezier*をってのタイミングをくこともできます。

1. cubicベジエの

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

2. アニメーションのプロパティ

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

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

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

```
});
```

する

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

オンラインでネイティブスクリプトによるアニメーションのをむ
https://riptutorial.com/ja/nativescript/topic/5970/ネイティブスクリプトによるアニメーションの

9: マルチスレッドモデル

しいクロムv8エンジンはにES7にしています。したがって、 `"use strict";`をすると`"use strict";`たちは、グローバルスコープにあるがにグローバルスコープにりてられていることをするがあります。 `self.functionName`または`global.functionName`をするがあります。

Examples

angular2サービスでをする

`/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>
```

オンラインでマルチスレッドモデルをむ https://riptutorial.com/ja/nativescript/topic/7878/マルチスレッドモデル

10: リストとしてデーータをするRepeater、ListViewまたは* NgFor + {N} + Angular-2アプリ

{N} +2のアプリケーションでリピータをしないでください * ngRepeatは、Angular-2のれのです。りしパターンをするがあるは、ListViewまたは* ngForディレクティブをします。

Examples

リピータモジュールをしてデータをするNativeScriptコア

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;
}
```

ObservableArrayNativeScript Coreでリピータモジュールをする

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;
}
```

ObservableArrayNativeScript CoreでListViewモジュールをする

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);
    // });
}
```

ListViewをしてデ―タをするNativeScript + Angular-2

creating-listview.component.html

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

creating-listview.component.ts

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

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

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

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

export class CreatingListViewComponent {
    public countries: Array<Country>;
```

```
    constructor() {
        this.countries = [];

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

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

* ngForディレクティブをしてデ―タをする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;
    }
}
```

コ―ルバックきリピ―タの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>
```

オンラインでリストとしてデータをするRepeater、ListViewまたは* NgFor + {N} + Angular-2アプリをむ https://riptutorial.com/ja/nativescript/topic/5226/リストとしてデータをする-repeater-listviewまたは--ngfor-plus--n--plus-angular-2アプリ-

クレジット

S. No		Contributors
1	nativescriptをいめる	Adam Diament, Community, George Edwards, HabibKazemi, Hardik Vaghani, Houssem Yahiaoui, Richard Hubley, user6939352
2	インターフェースをする	HabibKazemi
3	グローバル	Tim Hallyburton, TJ VanToll
4	ステータスバー	HabibKazemi
5	ネイティブAPIへのアクセス	HabibKazemi
6	ネイティブウィジェットをして	HabibKazemi
7	ネイティブスクリプトテンプレートのスタイル	George Edwards, Madhav Poudel, Nick Iliev
8	ネイティブスクリプトによるアニメーションの	Madhav Poudel
9	マルチスレッドモデル	HabibKazemi
10	リストとしてデータをするRepeater、ListViewまたは* NgFor + {N} + Angular-2アプリ	Nick Iliev, Tim Hallyburton, William KLEIN