

 eBook Gratuit

# APPRENEZ nativescript

eBook gratuit non affilié créé à partir des  
**contributeurs de Stack Overflow.**

#nativescrip

t

# Table des matières

À propos.....	1
<b>Chapitre 1: Démarrer avec nativescript.....</b>	<b>2</b>
Remarques.....	2
Exemples.....	2
Installation ou configuration.....	2
macOS.....	2
les fenêtres.....	2
Utilisation du code Visual Studio pour le développement NativeScript.....	3
Votre premier programme Hello World.....	3
Comment déboguer l'application nativescript-Android sur WiFi (sans racine).....	4
<b>Chapitre 2: Accéder à des sites natifs.....</b>	<b>6</b>
Exemples.....	6
Ecrire le code Java dans nativescript et l'utiliser directement en javascript.....	6
utiliser apis natif directement en javascript.....	9
<b>Chapitre 3: Affichage des données sous forme de liste (en utilisant Repeater, ListView ou .....)</b>	<b>11</b>
Remarques.....	11
Exemples.....	11
Utilisation du module Repeater pour afficher les données (NativeScript Core).....	11
Utilisation du module Repeater avec ObservableArray (NativeScript Core).....	11
Utilisation du module ListView avec ObservableArray (NativeScript Core).....	12
Utilisation de ListView pour afficher des données (NativeScript + Angular-2).....	13
Utilisation de * ngFor Structural Directive pour afficher des données (nativeScript + Angu.....	14
Utilisation du répéteur avec des rappels (JavaScript).....	14
<b>Chapitre 4: en utilisant un widget natif.....</b>	<b>16</b>
Exemples.....	16
Utiliser surfaceView dans ng2-TNS-Android: étape par étape.....	16
Utiliser surfaceView dans ng2-TNS-Android: exemple tout prêt.....	17
<b>Chapitre 5: Implémentation d'animations dans Natifscript.....</b>	<b>19</b>
Exemples.....	19
Animation de fond de StackLayout.....	19

Utilisation de la fonction de synchronisation d'animation et des propriétés d'animation.....	19
<b>Opacité.....</b>	<b>20</b>
<b>Traduire.....</b>	<b>20</b>
<b>Échelle.....</b>	<b>20</b>
<b>Tourner.....</b>	<b>21</b>
<b>Chapitre 6: mettre en œuvre l'interface.....</b>	<b>22</b>
Exemples.....	22
implémenter View.OnLayoutChangeListener dans Nativescript.....	22
<b>Chapitre 7: Modèle de style nativescript.....</b>	<b>23</b>
Exemples.....	23
Ajout d'un exemple de disposition dans votre application.....	23
<b>Méthode 1: CSS global.....</b>	<b>23</b>
<b>Méthode 2: CSS spécifique à la plate-forme.....</b>	<b>23</b>
<b>Méthode 3: CSS spécifique au composant.....</b>	<b>24</b>
<b>Chapitre 8: Modèle multithreading.....</b>	<b>25</b>
Remarques.....	25
Exemples.....	25
utiliser les travailleurs dans le service angular2.....	25
<b>Chapitre 9: StatusBar.....</b>	<b>28</b>
Exemples.....	28
Cacher / Montrer - Android.....	28
Faire statusBar Transparent android.....	28
<b>Chapitre 10: Variables globales.....</b>	<b>29</b>
Exemples.....	29
Console.....	29
Timer (JavaScript).....	29
<b>Crédits.....</b>	<b>31</b>

# À propos

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)

---

# Chapitre 1: Démarrer avec nativescript

## Remarques

Nativescript est une application mobile multi-plateforme hautement performante, qui vous permet de cibler iOS et Android (avec Windows en cours de développement) à l'aide de technologies Web (JS et HTML). Il a été créé avec plusieurs objectifs clés:

- Visuellement performant: pas d'interface utilisateur Jank même sur Android
- Extensible: vous avez accès à toutes les API natives pour créer facilement des plug-ins multi-plateformes
- Interface utilisateur entièrement native
- Intégré avec Typographie et Angulaire 2
- Open Source, avec le fort soutien de Telerik

## Exemples

### Installation ou configuration

Instructions détaillées sur la configuration ou l'installation de Nativescript.

Les exemples suivants illustrent les étapes requises pour configurer un système Windows ou OSX, puis vous connectez à des guides de dépannage en cas de problème.

En outre, il existe des exemples de configuration des flux de travail, IDE et émulateurs recommandés.

### macOS

1. Assurez-vous d'avoir installé le [plus récent](#) Node.js LTS. Si vous utilisez [Homebrew](#), cela peut être fait avec `brew install node4-lts`.
2. Ouvrez Terminal et tapez `npm install -g nativescript`. Si vous obtenez une erreur `EACCES`, utilisez `sudo npm install -g nativescript`.
3. Dans l'invite de commande, tapez `ruby -e "$(curl -fsSL https://www.nativescript.org/setup/mac)"`. (Cela pourrait prendre un certain temps.)
4. Pour vérifier que ce qui précède a bien fonctionné, tapez `tns doctor` in Terminal.
5. S'il y a des erreurs, suivez le [guide de dépannage](#).

### les fenêtres

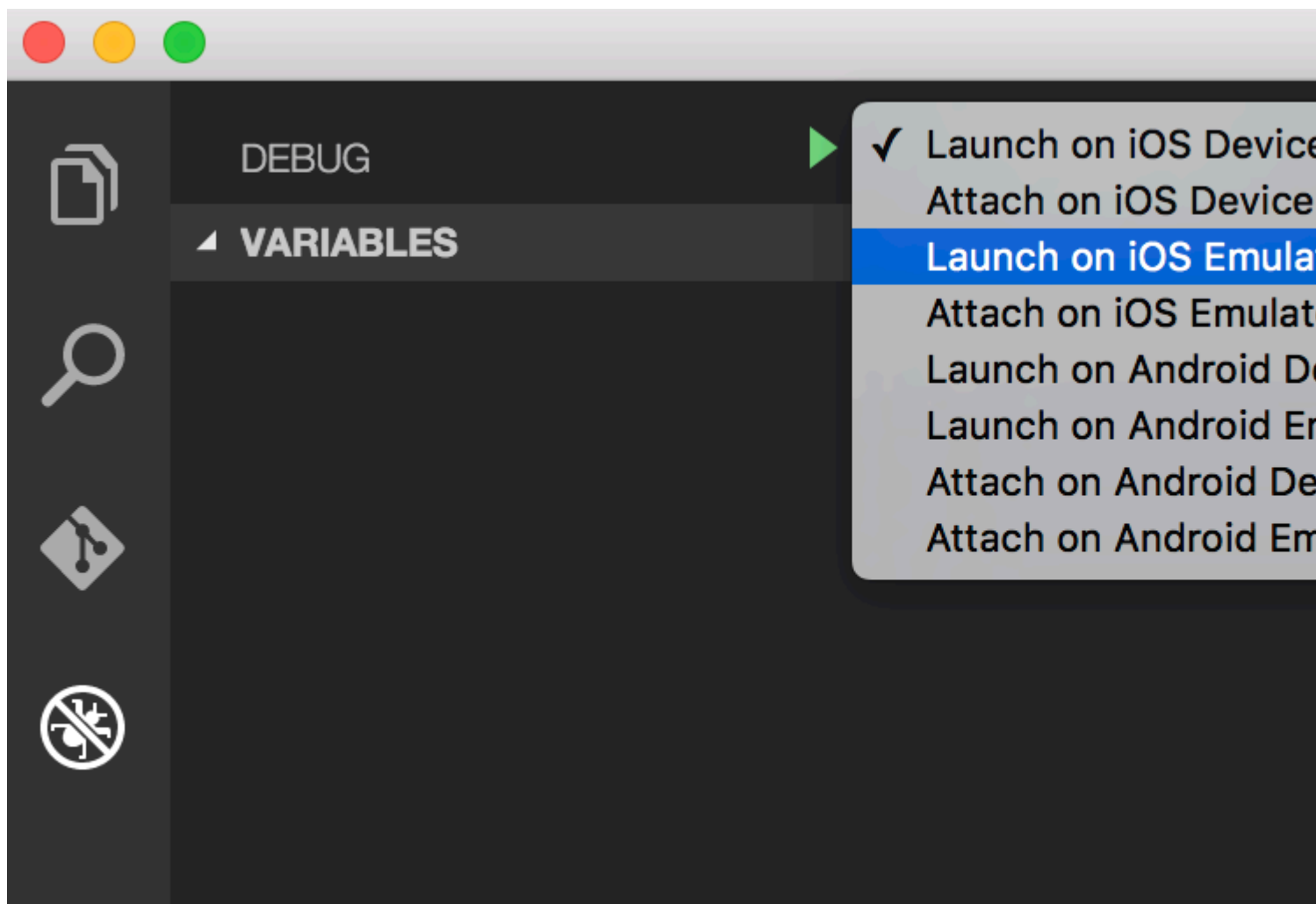
1. Assurez-vous que le dernier [nœudJS LTS est installé](#)
2. Ouvrez l'invite de commande et tapez `$ npm install -g nativescript`
3. Dans l'invite de commande, tapez `$ @powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-object net.webclient).DownloadString('https://www.nativescript.org/setup/win'))"`  
- cela pourrait prendre un certain temps

4. Pour vérifier que ce qui précède a fonctionné, tapez `$ tns doctor` dans l'invite de commande (votre cmd)
5. S'il y a des erreurs, suivez le [guide de dépannage](#)

## Utilisation du code Visual Studio pour le développement NativeScript

[Visual Studio Code](#) est un éditeur de code open source et riche en fonctionnalités de Microsoft. Pour le configurer pour le développement NativeScript, ouvrez la palette de commandes ( `F1` ou `⌘ + Maj + P` ) et tapez `ext install NativeScript` .

Une fois l'extension NativeScript installée, le débogueur doit vous permettre de définir des points d'arrêt dans votre code. Lorsqu'un périphérique est connecté ou qu'un émulateur est en cours d'exécution, vous pouvez démarrer votre application à partir de l'onglet Débogueur.



## Votre premier programme 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
```

Ensuite, assurez-vous d'avoir un périphérique connecté ou un émulateur en cours d'exécution (si vous ne le faites pas, l'émulateur par défaut devrait démarrer ou une erreur sera générée. Je

recommande genymotion pour Android).

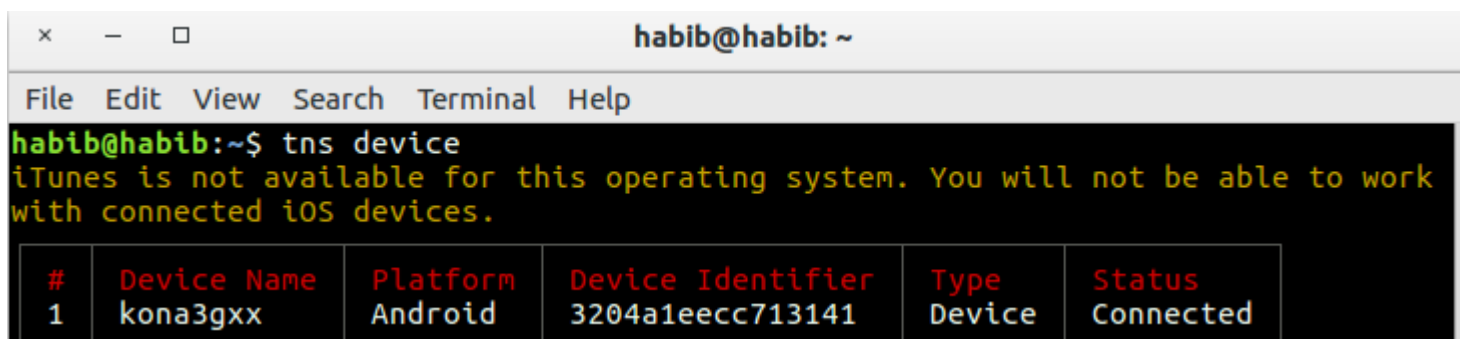
```
$ tns run android
```

Si vous souhaitez utiliser l'émulateur Android par défaut, ajoutez l'indicateur `--emulator`.

À partir de tns 2.5, `livesync` est désormais l'action par défaut pour `tns run <platform>`, qui sera automatiquement recompilé lorsque vous enregistrez des modifications de fichiers. Cela peut considérablement améliorer votre temps de développement, cependant, si vous modifiez vos plugins, vous devrez compiler correctement.

## Comment déboguer l'application nativescript-Android sur WiFi (sans racine)

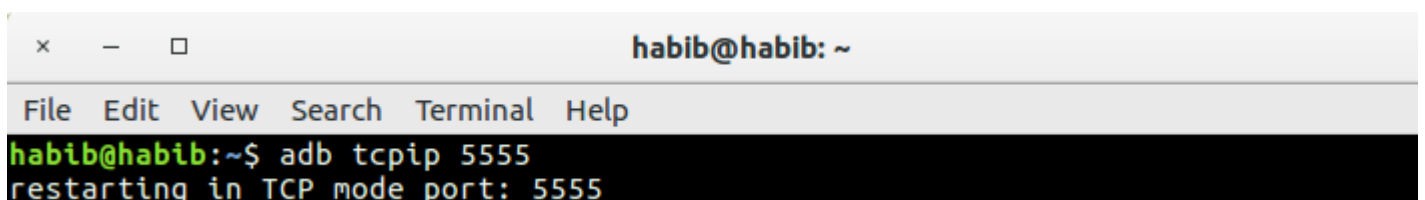
1-Vous devez connecter votre appareil à votre ordinateur via un câble USB. Assurez-vous que le débogage USB fonctionne. Vous pouvez vérifier s'il apparaît lors de l'exécution d' `adb devices` (ou d' `tns device` ).



```
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`

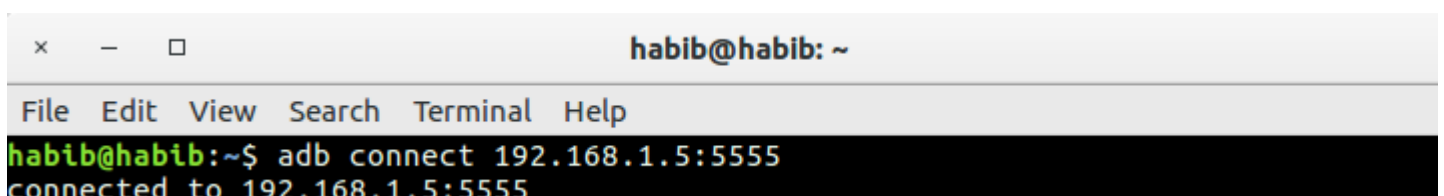


```
habib@habib:~$ adb tcpip 5555
restarting in TCP mode port: 5555
```

3-Déconnectez votre appareil (retirez le câble USB).

4-Allez dans Paramètres -> À propos du téléphone -> État pour afficher l'adresse IP de votre téléphone.

5-Exécuter `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 vous utilisez à `adb devices` (ou `tns device` ), vous devriez voir votre appareil.

```
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- Maintenant, vous pouvez utiliser `tns run android`, `tns livesync android` commandes.

## REMARQUES :

1-Lorsque le réseau WiFi change, vous n'avez pas à répéter les étapes 1 à 3 (celles-ci définissent votre téléphone en mode de débogage wifi). Vous devez vous connecter à nouveau à votre téléphone en exécutant les étapes 4 à 6.

Les téléphones 2-Android perdent le mode wifi-debug lors du redémarrage. Ainsi, si votre batterie est morte, vous devez recommencer. Sinon, si vous surveillez votre batterie et ne redémarrez pas votre téléphone, vous pouvez vivre sans câble pendant des semaines!

## ATTENTION :

laisser l'option activée est dangereux, toute personne de votre réseau peut se connecter à votre appareil en mode débogage, même si vous êtes dans un réseau de données. Ne le faites que lorsque vous êtes connecté à un réseau Wi-Fi fiable et n'oubliez pas de le déconnecter une fois terminé!

## référence :

1-Norman Peitek. 2014. Comment déboguer votre application Android sur WiFi (sans racine!). [EN LIGNE] Disponible sur: <https://futurestud.io/blog/how-to-debug-your-android-app-over-wifi-without-root> . [Consulté le 8 août 2016].

2 usethe4ce. 2012. Exécuter / installer / déboguer des applications Android sur Wi-Fi?. [EN LIGNE] Disponible à l' adresse : <http://stackoverflow.com/a/10236938/4146943> . [Consulté le 8 août 2016].

Lire Démarrer avec nativescript en ligne: <https://riptutorial.com/fr/nativescript/topic/921/demarrer-avec-nativescript>

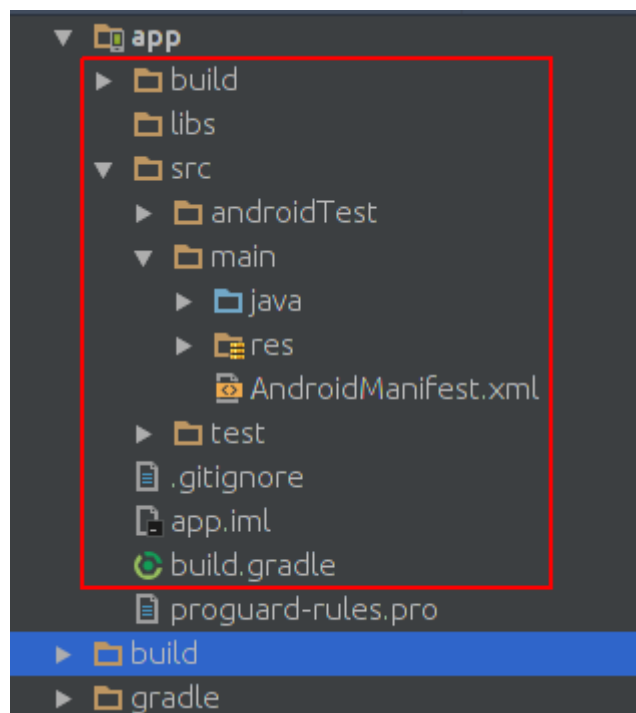


# Chapitre 2: Accéder à des sites natifs

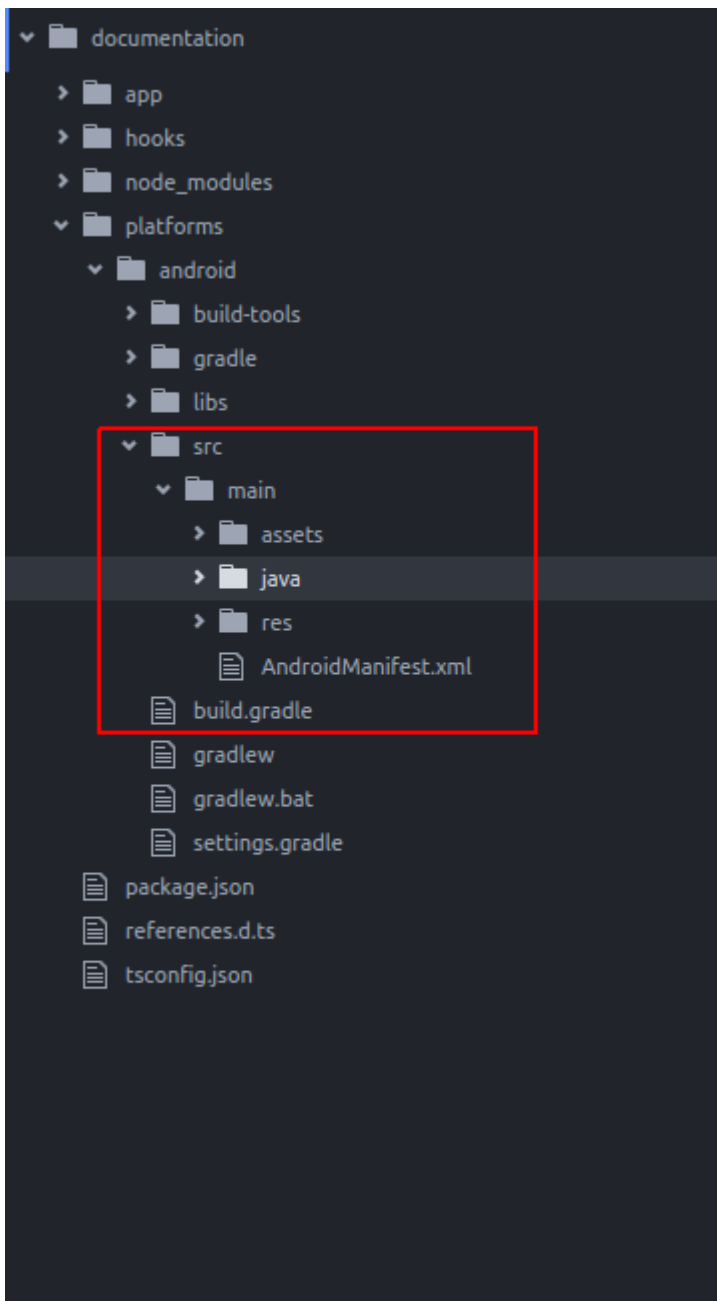
## Exemples

Ecrire le code Java dans nativescript et l'utiliser directement en javascript

Voici l'image de la structure du projet en studio Android:

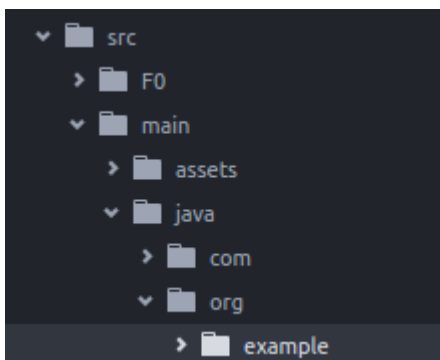


Voici l'image de la structure du projet nativescript:



Comme vous voyez, ils sont identiques. Nous pouvons donc écrire du code Java en nativescript lorsque nous écrivons dans Android Studio.

Nous voulons ajouter Toast à l'application par défaut de nativescript. Après avoir créé un nouveau projet nativescript, créez un répertoire comme celui-ci dans le répertoire `java/org/example` :



créer un nouveau fichier `MyToast.java` dans le répertoire `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();
    }
}
```

**Notes** : n'oubliez pas le nom du paquet;

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

maintenant, lorsque vous appuyez sur le bouton, il affiche un toast;

## Notes :

1. La fonction `showToast` accepte le contexte pour le transmettre à `Toast.makeText` et nous lui avons passé un contexte de cette manière: `application.android.context`
2. typescript ne sait pas quelle `org` est, nous l'avons donc déclaré: `declare var org:any;`

## utiliser apis natif directement en javascript

Nous voulons ajouter Toast à l'application par défaut 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();
  }
}
```

pour créer des toasts, nous devons appeler `Toast.makeText` et il est dans le package

`android.widget.Toast . Toast.makeText` accepte le contexte comme premier argument et nous pouvons obtenir le contexte dans nativescript de cette manière: `application.android.context`

Lire Accéder à des sites natifs en ligne: <https://riptutorial.com/fr/nativescript/topic/5188/accéder-a->

[des-sites-natifs](#)

---

# Chapitre 3: Affichage des données sous forme de liste (en utilisant Repeater, ListView ou \* ngFor pour {N} + applications Angular-2)

## Remarques

Remarque: n'utilisez pas Repeater dans les applications {N} + Angular-2! La directive \* ngRepeat est obsolète dans Angular-2. Lorsque vous avez besoin d'afficher des motifs d'éléments répétés, utilisez la directive structurelle ListView ou \* ngFor.

## Exemples

### Utilisation du module Repeater pour afficher les données (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;
}
```

### Utilisation du module Repeater avec 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;
}
```

## Utilisation du module ListView avec 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);
    // });
}

```

## Utilisation de ListView pour afficher des données (NativeScript + Angular-2)

### *creation-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>

```

### *créer-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);
  }
}

```

## Utilisation de \* ngFor Structural Directive pour afficher des données (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;
  }
}

```

## Utilisation du répéteur avec des rappels (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>
```

Lire Affichage des données sous forme de liste (en utilisant Repeater, ListView ou \* ngFor pour {N} + applications Angular-2) en ligne: <https://riptutorial.com/fr/nativescript/topic/5226/affichage-des-donnees-sous-forme-de-liste--en-utilisant-repeater--listview-ou---ngfor-pour--n--plus-applications-angular-2->

# Chapitre 4: en utilisant un widget natif

## Exemples

### Utiliser surfaceView dans ng2-TNS-Android: étape par étape

Par exemple, vous voulez utiliser surfaceView dans ng2-nativescript. Comme nous n'avons pas surfaceView dans nativescript, nous devons utiliser un placeholder .

Nous devons d'abord importer les exigences:

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

puis ajoutez l'espace réservé à votre fichier html:

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

Ajoutez cette méthode à votre classe:

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

TypeScript ne sait pas ce qui est android et nous devrions ajouter des fichiers de déclaration de plate-forme suivre cette [réponse](#) pour les ajouter.

En raison d'un [problème](#) dans la version actuelle de ng2-nativescript, nous devrions faire un travail supplémentaire:

changez l'espace réservé pour:

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

Importer OnInit:

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

vosre classe devrait implémenter OnInit

```
export class AppComponent implements OnInit
```

et ajoutez ces lignes à votre classe:

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

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

maintenant vous avez un SurfaceView dans votre application nativescript :)

## Méthodes d'appel de SurfaceView

Par exemple, vous voulez appeler `getHolder()` :

Ajoutez une variable et un événement chargé à votre espace réservé comme ceci:

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

et ajoutez la méthode `onLoaded` à votre classe:

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

## ATTENTION :

Il n'est pas garanti que la propriété `android` (`element.android`) soit disponible dans `ngAfterViewInit`, nous avons donc utilisé l'événement `loaded` au lieu de cela.

## Utiliser surfaceView dans ng2-TNS-Android: exemple tout prêt

### 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>
```

Lire en utilisant un widget natif en ligne: <https://riptutorial.com/fr/nativescript/topic/5834/en-utilisant-un-widget-natif>

# Chapitre 5: Implémentation d'animations dans Natifscript

## Exemples

### Animation de fond de StackLayout

Animation de la couleur d'arrière-plan du stacklayout sur le bouton de frappe

*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;
}
```

Utilisation de la fonction de synchronisation d'animation et des propriétés d'animation.

*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 pour d'autres propriétés d'animation

*Vous pouvez également écrire votre propre fonction de synchronisation en utilisant cubicBezier.*

### 1. Utilisation de cubicBezier

```

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

```

### 2. Propriétés d'animation

## Opacité

```

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

```

## Traduire

```

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

```

# Échelle

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

---

# Tourner

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

Lire Implémentation d'animations dans Natifscript en ligne:

<https://riptutorial.com/fr/nativescript/topic/5970/implementation-d-animations-dans-nativescript>



---

# Chapitre 6: mettre en œuvre l'interface

## Exemples

### implémenter `View.OnLayoutChangeListener` dans 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();
        }
    }
});
```

créer une `surfaceView` <http://stackoverflow.com/documentation/proposed/changes/79536>

Ajouter un auditeur:

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

supprimer l'auditeur:

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

Lire mettre en œuvre l'interface en ligne: <https://riptutorial.com/fr/nativescript/topic/5560/mettre-en-oeuvre-l-interface>

---

# Chapitre 7: Modèle de style nativescript

## Exemples

### Ajout d'un exemple de disposition dans votre application

*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 { }
```

---

## Méthode 1: CSS global

*app.css* - S'applique globalement à toutes les mises en page.

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

---

## Méthode 2: CSS spécifique à la plate-forme

*platform.android.css* - S'applique globalement à toutes les mises en page dans les appareils Android.

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

*platform.ios.css* - S'applique globalement à toutes les mises en forme sur le périphérique ios.

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

*app.css*

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

---

## Méthode 3: CSS spécifique au composant

*pages / main / main.android.css* - S'applique à un composant spécifique de l'appareil Android.

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

*pages / main / main.ios.css* - S'applique à un composant spécifique du périphérique ios.

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

*pages / main / main-common.css* - S'applique à un composant spécifique de tous les périphériques.

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

Lire Modèle de style nativescript en ligne: <https://riptutorial.com/fr/nativescript/topic/3872/modele-de-style-nativescript>

# Chapitre 8: Modèle multithreading

## Remarques

Le nouveau moteur chrome v8 est partiellement conforme à la norme ES7. Donc, si on ajoute "use strict"; en haut de notre fichier (le typecript fait cela quand transpiles tape), nous devons nous assurer que toutes les fonctions qui sont sur la portée globale sont réellement affectées à la portée globale. nous devons donc utiliser `self.functionName` OU `global.functionName` .

## Exemples

### utiliser les travailleurs dans le service 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>
```

Lire **Modèle multithreading en ligne**: <https://riptutorial.com/fr/nativescript/topic/7878/modele-multithreading>

# Chapitre 9: StatusBar

## Exemples

### Cacher / Montrer - Android

Ceci est une barre d'état que vous voyez en haut de votre écran avec des icônes de battery, clock



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

#### Cacher:

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

#### Montrer:

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

### Faire statusBar Transparent android

ouvrez `APP_Resources/values/styles.xml` et ajoutez le

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

dans le

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

section.

Lire StatusBar en ligne: <https://riptutorial.com/fr/nativescript/topic/6007/statusbar>

---

# Chapitre 10: Variables globales

## Exemples

### Console

La variable de la `console` globale de NativeScript vous permet d'imprimer des valeurs sur votre terminal pour le débogage. L'utilisation la plus simple consiste à transmettre une valeur à la fonction `console.log()` :

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

L'objet `console` a plusieurs autres méthodes, y compris `dump()`, `trace()`, `assert()` et [plus](#).

```
// 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)

La variable de `timer` globale de NativeScript vous permet de définir des délais et des intervalles pour les appels de fonction retardés asynchrones.

#### Importer

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

#### Des délais d'attente

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

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

#### Les intervalles

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



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

Lire Variables globales en ligne: <https://riptutorial.com/fr/nativescript/topic/3133/variables-globales>

# Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec 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	Accéder à des sites natifs	<a href="#">HabibKazemi</a>
3	Affichage des données sous forme de liste (en utilisant Repeater, ListView ou * ngFor pour {N} + applications Angular-2)	<a href="#">Nick Iliev</a> , <a href="#">Tim Hallyburton</a> , <a href="#">William KLEIN</a>
4	en utilisant un widget natif	<a href="#">HabibKazemi</a>
5	Implémentation d'animations dans Nativescript	<a href="#">Madhav Poudel</a>
6	mettre en œuvre l'interface	<a href="#">HabibKazemi</a>
7	Modèle de style nativescript	<a href="#">George Edwards</a> , <a href="#">Madhav Poudel</a> , <a href="#">Nick Iliev</a>
8	Modèle multithreading	<a href="#">HabibKazemi</a>
9	StatusBar	<a href="#">HabibKazemi</a>
10	Variables globales	<a href="#">Tim Hallyburton</a> , <a href="#">TJ VanToll</a>