Building a Mobile App

with

Angular 4
Ionic 3

Olivier Overstraete - Johan Coppieters
NOWJOBS

NOWJOBS NOW
Get the job done.

Agenda

1. Angular
2. Ionic
3. Example
4. Install
5. Template app

5. Real App ➡ Conference
6. Service / Providers
7. First version
8. Assignment
9. Running on device

Angular 4

- Two way Data binding / Change detection
- Nice templating
- Model-View-Controller
- Modules
- TypeScript
- Often less code
- Support / Used at Google
- Maturity from AngularJS in 2009
- Ideal for mobile apps & Single Page Web apps

Ionic 3

- Multi platform support
    - iOS / Android / Windows
    - write once - run everywhere
- Native looks
- Good components
- Angular 4
- Very well supported plugins (native access)
- Open source
- Good quality, well maintained
- Enterprise support

Angular Example

- Demo template, style, component, 2-way data binding, pipe
- Install
  - make directory
  - install angular (global)
  - create empty app

```
$ mkdir AngularDemo
$ cd AngularDemo/
```

```
$ npm install -g @angular/cli
```

```
$ ng new demo1
```

NOWJOBS NOW
Get the job done.

Main: index.html

```html
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>Demo1</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>

<body>
  <app-root></app-root>
</body>

</html>
```

new tag: app-root
name can be whatever
but it needs to exist

Other files

main.ts -> bootstrap app.module

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule);
```

app.module -> bootstrap AppComponent

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

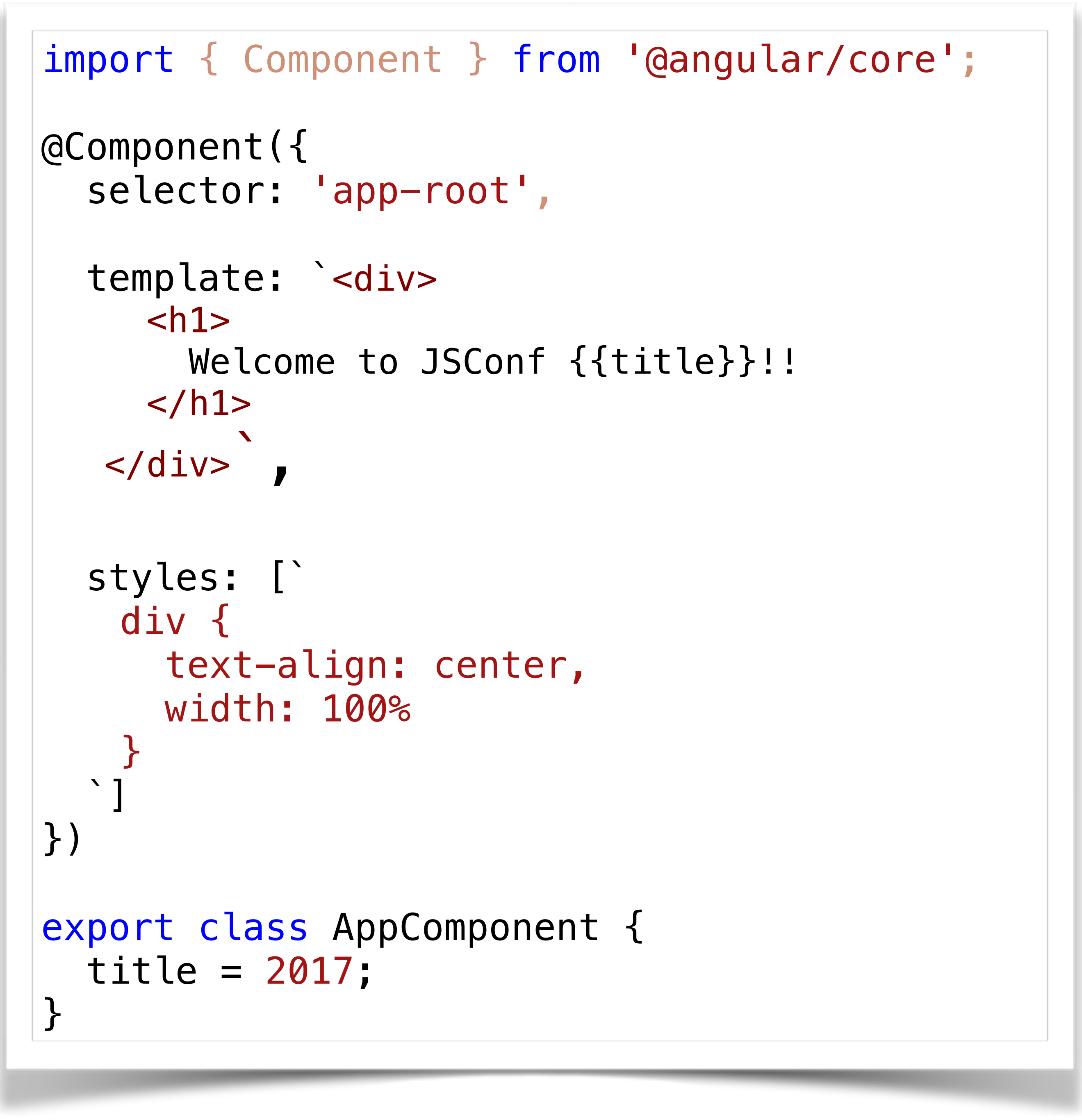
AppComponent (in app.component.ts)

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 2017;
}
```

... or ...

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<div>
     <h1>
       Welcome to JSConf {{title}}!!
     </h1>
   </div>`,


  styles: [`
    div { text-align: center}
   `]
})
export class AppComponent {
  title = 2017;
}
```

Anatomy



```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',

  template: `<div>
     <h1>
       Welcome to JSConf {{title}}!!
     </h1>
   </div>` ,


  styles: [`
   div {
     text-align: center,
     width: 100%
   }
  `]
})

export class AppComponent {
  title = 2017;
}
```

- Import Angular lib / Component
- @Component decorator
- Name of tag (selector)
- Template
- Styles
- TypeScript Class

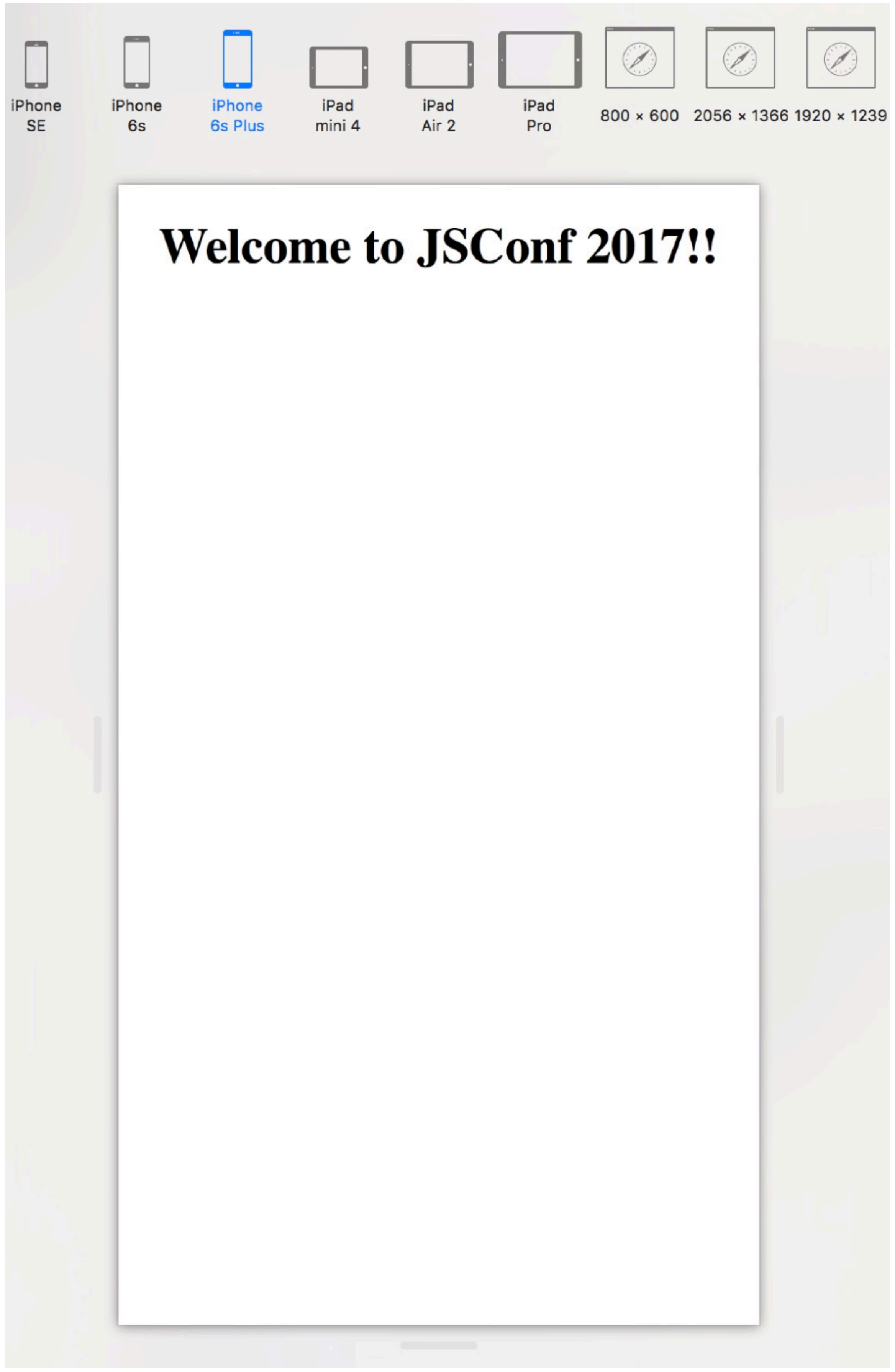
See changes

```
$ ng serve --open
```

```
(serves to http://localhost:4100 +
 opens a browser window +
 compiles all TypeScript)
```

More info

https://angular.io/guide/quickstart



Welcome to JSConf 2017!!

NOWJOBS NOW
Get the job done.

*ngIf, *ngFor, ...

- Templating your page
- Close interaction with your controller / model
- Use methods or instance variables

Make a list

```
<ul>
  <li *ngFor="let session of sessions”>
    <span class="room">{{session.room}}</span>
    {{session.name}}
  </li>
</ul>
```

"session" -> local block scoped variable
"sessions" -> this.sessions from current component, should be an array

“session.room" & "session.name" -> both fields of "session"

Make a component

```
export class Session {
  name: "";
  speaker: "";
  room: "";
  info: "";

  constructor(name, speaker, room, info) {
    this.name = name; this.speaker = speaker;
    this.room = room; this.info = info;
  }
}
```

```
export class AppComponent {
  title = 2017;

  sessions = [
    new Session("Build a mobile app", "Johan Coppieters", "D101", "bla bla"),
    new Session("Ionic 3", "Olivier Overstraete", "D102", "blo blo"),
    new Session("nodejs services", "Ruben Callewaert", "D103", "bli bli")
  ];
}
```

Get a click

```
<ul>
  <li *ngFor="let session of sessions" (click)=“select(session)">
    <span class="room">{{session.room}}</span>
    {{session.name}}
  </li>
</ul>
```

"onSelect"

-> function / method of current component

```
  select(session) {
    this.selectedSession = session;
  }
```

Add the function

```
export class AppComponent {
  title = 'JSConf 2017';
  selectedSession = null;
  sessions = [
    new Session("Build a mobile app", "Johan Coppieters", "D101", "bla bla"),
    new Session("Ionic 3", "Olivier Overstraete", "D102", "blo blo"),
    new Session("nodejs services", "Ruben Callewaert", "D103", "bli bli")
  ];


  select(session) {
    if (this.selectedSession == session)
      this.selectedSession = null;
    else
      this.selectedSession = session;
  }
}
```

Reflect choice

```
<ul>
  <li *ngFor="let session of sessions"
   [class.selected]="session == selectedSession”
   (click)=“select(session)">
     <h3>{{session.room}}</h3>
     <h2>{{session.name}}</h2>
  </li>
</ul>
```

```
styles: [`
    ul { list-style-type: none; padding: 0 }
    h3 { color: darkgrey }
    li { border: solid 1px grey;
         padding: 5px; margin: 3px;
         cursor: pointer }
    li.selected { border: solid 1px red }
  `]
```

Reflect choice (be nice)

```
<ul>
  <li *ngFor="let session of sessions"
      [class.selected]=“isSelected(session)”
      (click)=“select(session)”>
     <h3>{{session.room}}</h3>
     <h2>{{session.name}}</h2>
 </li>
</ul>
```

```
  select(session) {
    if (this.selectedSession == session)
      this.selectedSession = null;
    else
      this.selectedSession = session;
  }

  isSelected(session) {
    return (this.selectedSession == session);
  }
```

Display choice

```
<div *ngIf="selectedSession">
  <h2>{{selectedSession.name}}</h2>
  <div><label>Room: </label>{{selectedSession.room}}</div>
  <div><label>Speaker: </label>{{selectedSession.speaker}}</div>
  <div><label>Info: </label>{{selectedSession.info}}</div>
</div>
```

"selectedSession"
-> this.selectedSession from current component,
should be an object here

"selectedSession.room", "selectedSession.name", ...
-> all fields of a session stored in this.selectedSession

You need the *ngIf !! otherwise you get something like:

```
EXCEPTION: TypeError: Cannot read property 'name' of undefined in [null]
```

Apply a change

two-way binding

```
<div *ngIf="selectedSession">
      <div><label>Name: </label>
           <input type=text [(ngModel)]="session.name" />
      </div>
      <div><label>Room: </label>
           <input type=text [(ngModel)]="session.room" />
      </div>
      <div><label>Speaker: </label>
           <input type=text [(ngModel)]="session.speaker" />
      </div>
      <div><label>Info: </label>
           <textarea [(ngModel)]="session.info"></textarea>
      </div>
</div>
```

two-way binding ==
  data changes are reflected in the DOM
  DOM changes (input types) are reflected in the data

format time

- Times are in numbers

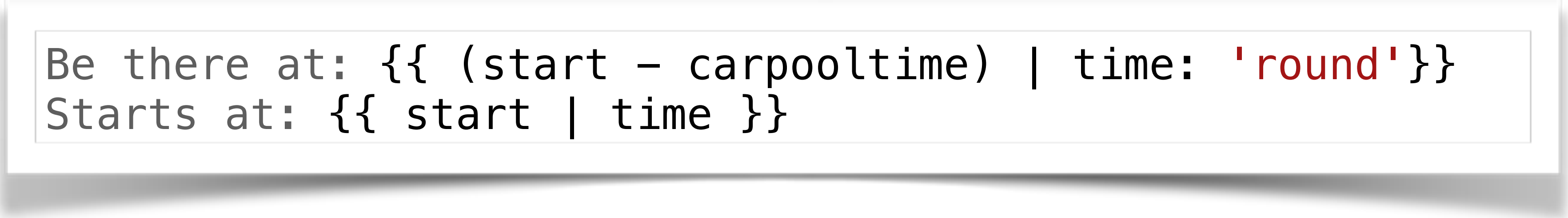
  format: hhmm

- We could add a function to format it,

  but you can feel: this is not a one time use

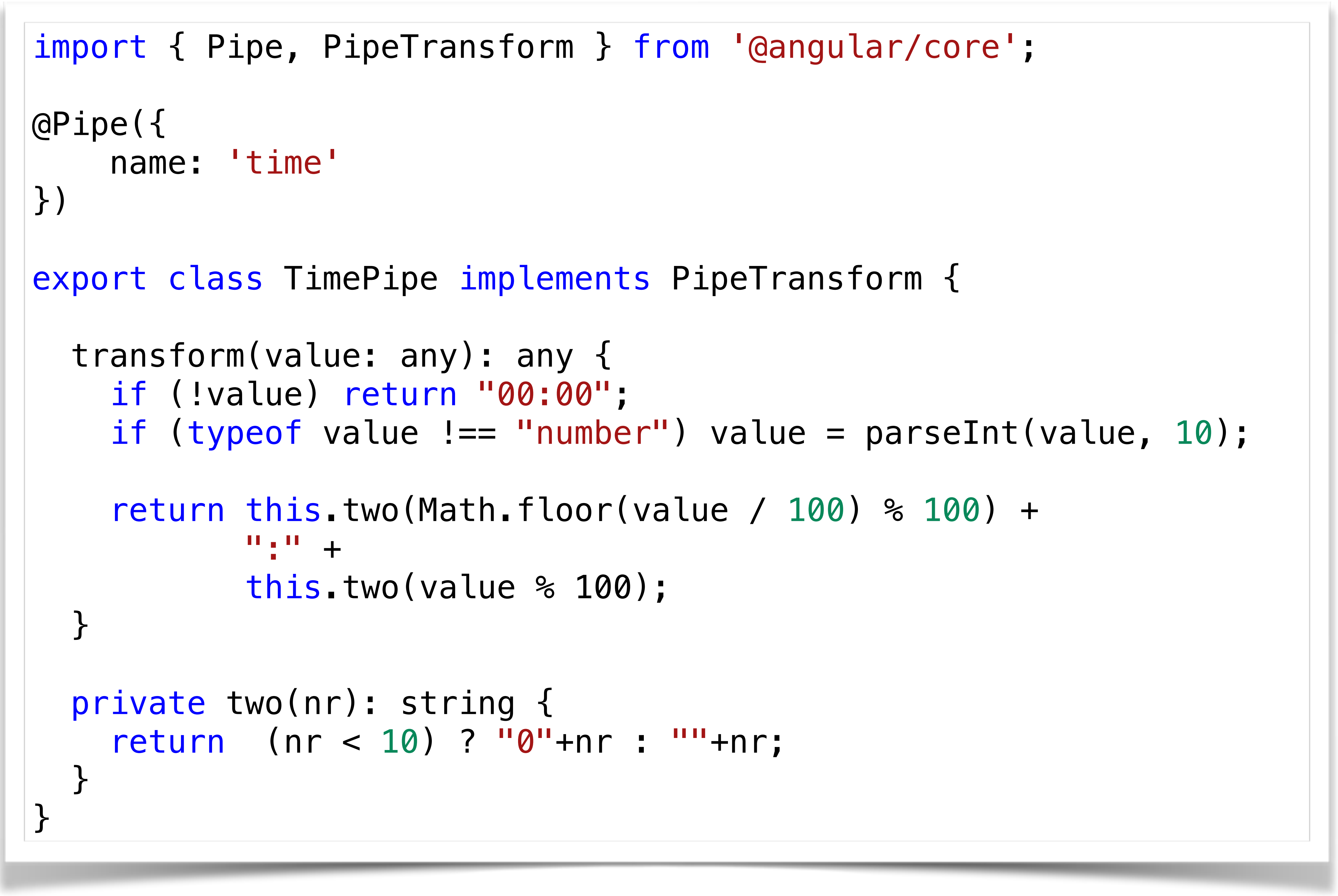
- Angular has pipes for it

- let's design a multi purpose time formatter



```
Be there at: {{ (start - carpooltime) | time: 'round'}}
Starts at: {{ start | time }}
```

Pipe Example



```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
    name: 'time'
})

export class TimePipe implements PipeTransform {

  transform(value: any): any {
    if (!value) return "00:00";
    if (typeof value !== "number") value = parseInt(value, 10);

    return this.two(Math.floor(value / 100) % 100) +
           ":" +
           this.two(value % 100);
  }

  private two(nr): string {
    return  (nr < 10) ? "0"+nr : ""+nr;
  }
}
```

Pipe Example, with extra parameter

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
    name: 'time'
})

export class TimePipe implements PipeTransform {

  transform(value: any, format: string): any {
    if (!value) return "00:00";
    if (typeof value !== "number") value = parseInt(value, 10);

    return this.two(Math.floor(value / 100) % 100) + ":" + this.quarters(value, format);
  }

  private quarters(nr, format): string {
    if (format && format === "round") {
      nr = Math.floor((nr % 100) / 15);
      return (nr % 4 === 0) ?
          "00" : ((nr % 4 === 1) ? "15" : ((nr % 4 === 2) ? "30" : "45"));
    } else {
      return this.two(nr % 100);
    }
  }

  private two(nr): string {
    return  (nr < 10) ? "0"+nr : ""+nr;
  }
}
```

Make a separate component

```
template: `
  <h1>Welcome to {{title}}!!</h1>

  <ul>
    <li *ngFor="let s of sessions"
        (click)="select(s)"
        [class.selected]="isSelected(s)">
      <h3>{{s.from | time }}-{{s.till | time: round }}
          in {{s.room}}</h3>
      <h2>{{s.name}}</h2>
    </li>
  </ul>

  <div *ngIf="selectedSession">
    <session-detail [session]="selectedSession">
    </session-detail>
  </div>
`
```

[session] == one-way input binding

! but still live binding of the input param

The separate component

```
@Component({
  selector: 'session-detail',
  inputs: ['session'],
  template: `
      <div><label>Name: </label>
           <input type=text [(ngModel)]="session.name" />
      </div>
      <div><label>Room: </label>
           <input type=text [(ngModel)]="session.room" />
      </div>
      <div><label>Speaker: </label>
           <input type=text [(ngModel)]="session.speaker" />
      </div>
      <div><label>Info: </label>
           <textarea [(ngModel)]="session.info"></textarea>
      </div>
  `
})
export class SessionComponent {
  session: Session;
}
```

or

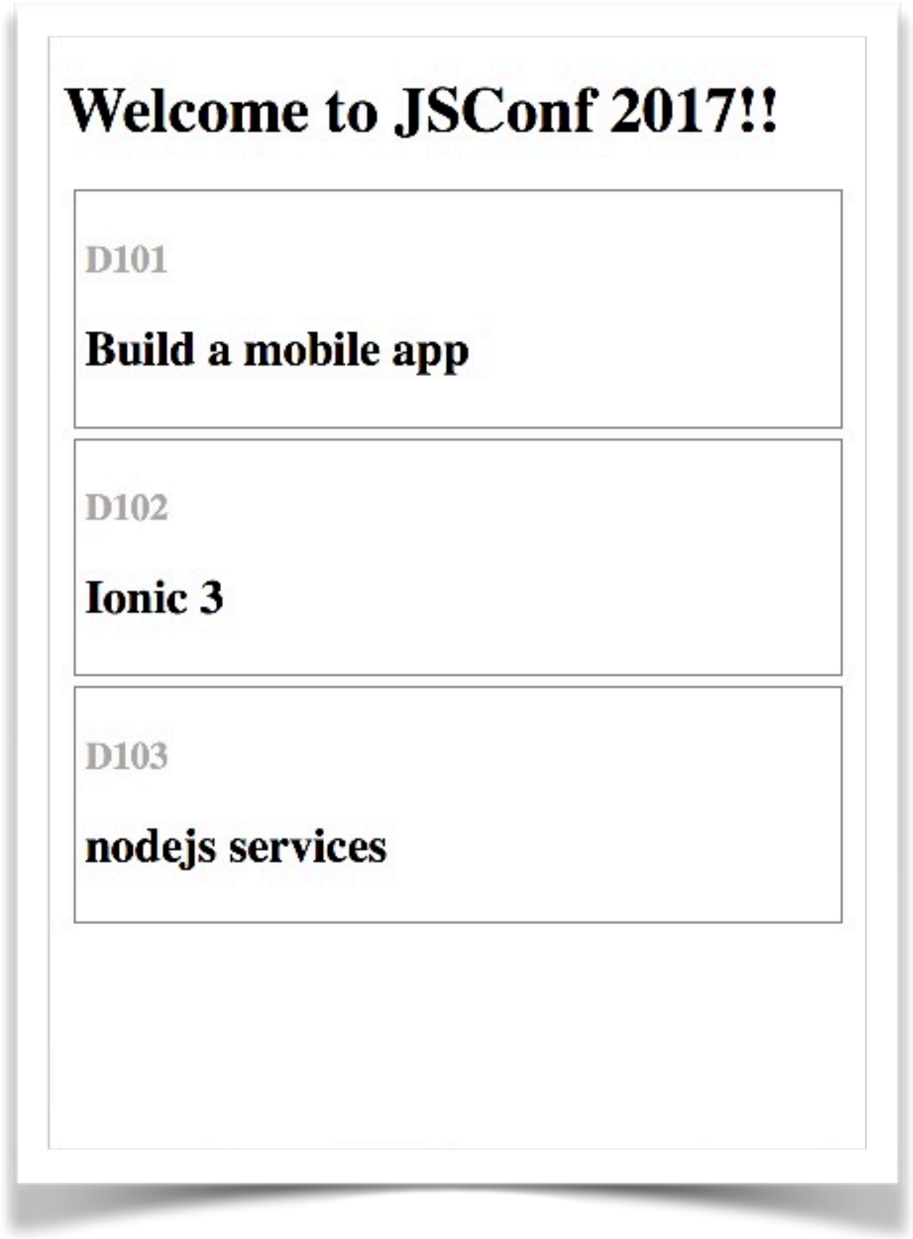
```
export class SessionComponent {
  @Input(session): Session;
}
```

Don't forget

to import/export in your module definition (app.module.ts)
1) import FormsModule
2) declarations: SessionComponent

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { SessionComponent } from './session.component';

@NgModule({
  declarations: [
    AppComponent, SessionComponent
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

It works !!



Welcome to JSConf 2017!!

D101

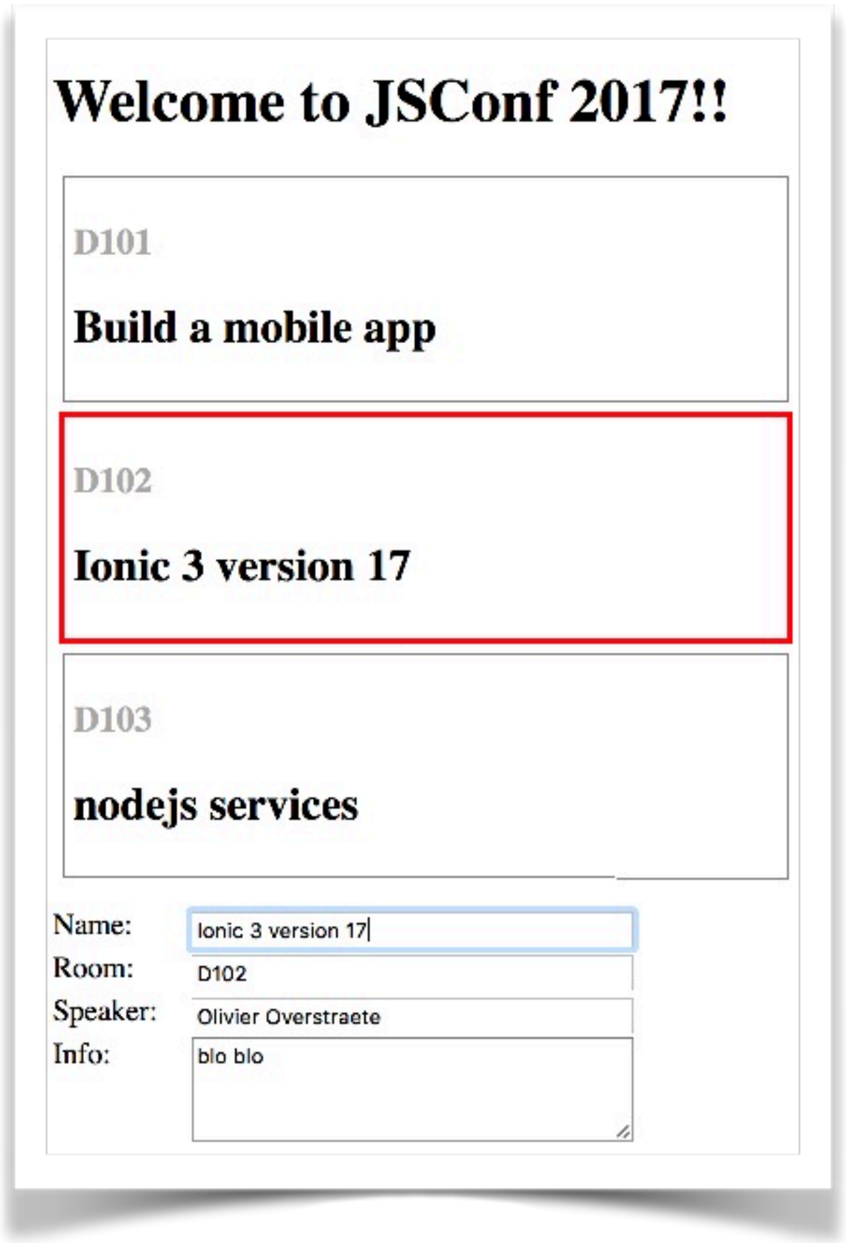
Build a mobile app

D102

Ionic 3

D103

nodejs services



Welcome to JSConf 2017!!

D101

Build a mobile app

D102

Ionic 3 version 17

D103

nodejs services

Name:	Ionic 3 version 17
Room:	D102
Speaker:	Olivier Overstraete
Info:	blo blo

but it looks bad,
doesn't have native mobile looks, ...
Help!   Ionic?

NOWJOBS NOW
Get the job done.

JS

Angular basic stuff

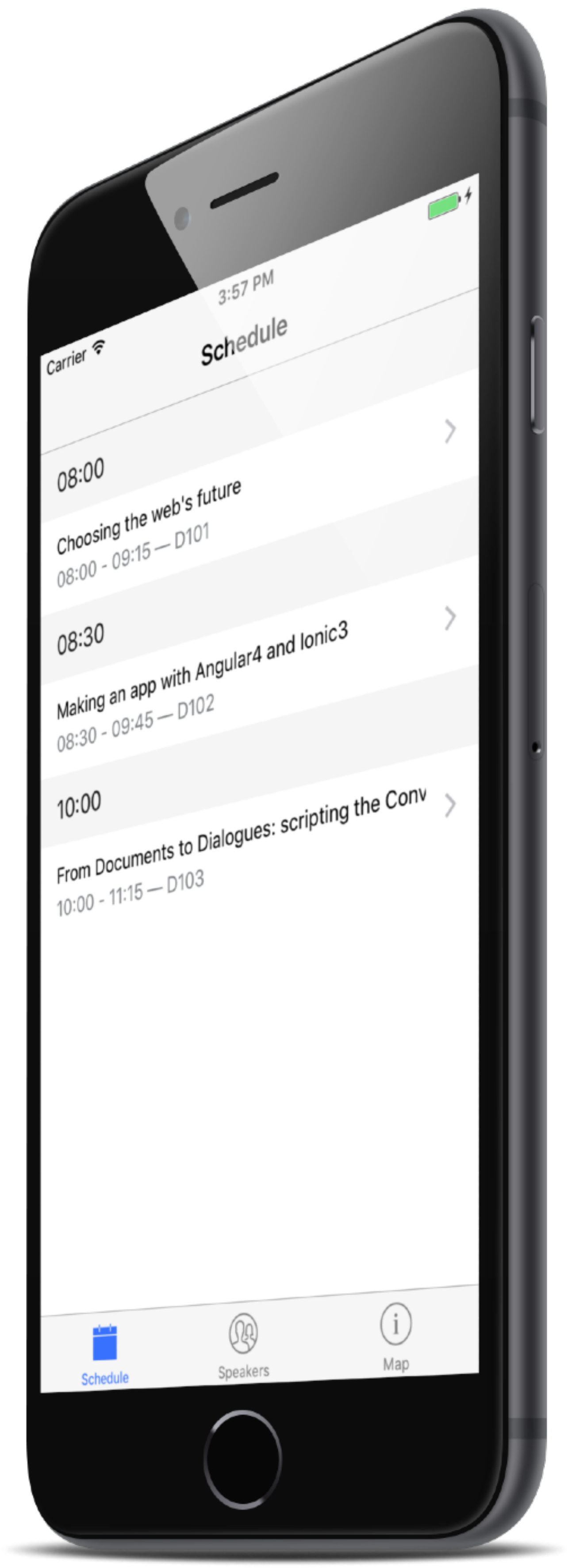
- Classes
- Components
- Templates
- Services (will do in between ionic)
- Pipes
- Routing (but not if we use Ionic)
- Testing

Ionic 3

- Multi platform support
    - iOS / Android / Windows
    - write once - run everywhere
- Native looks
- Good components
- Angular 4
- Very well supported plugins (native access)
- Open source
- Good quality, well maintained
- Enterprise support

ionic

Conference app



08:00

Choosing the web's future
08:00 - 09:15 — D101

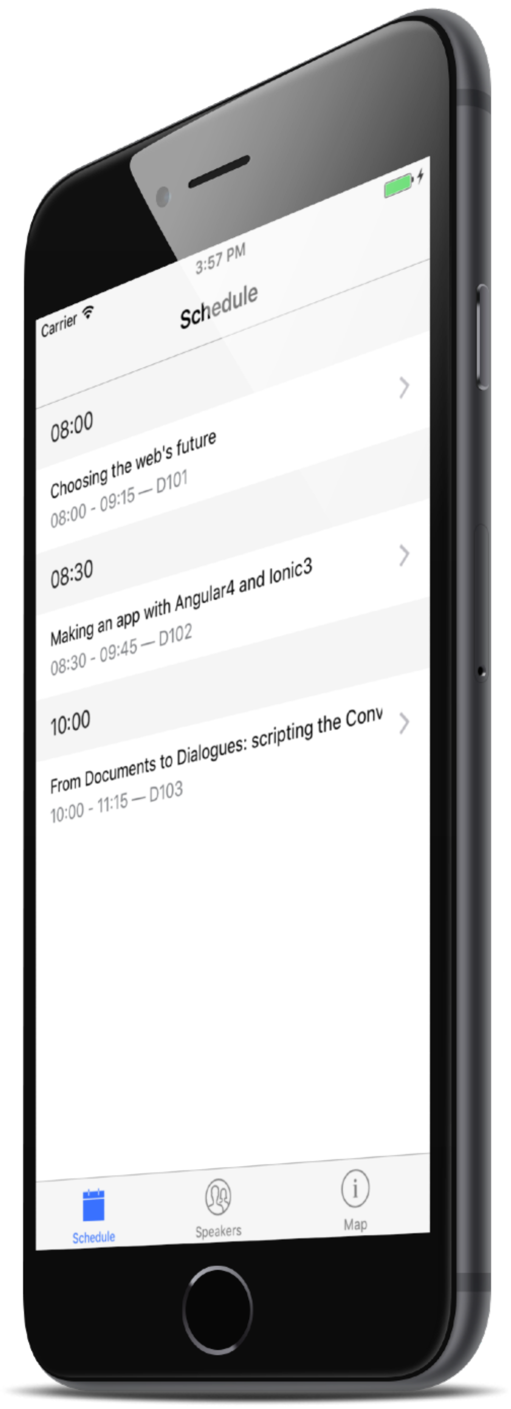
08:30

Making an app with Angular4 and Ionic3
08:30 - 09:45 — D102

10:00

From Documents to Dialogues: scripting the Conv
10:00 - 11:15 — D103

Best one can be used in the 2018 edition with your name under it


Carrier 3:57 PM

Schedule

08:00

Choosing the web's future
08:00 - 09:15 — D101

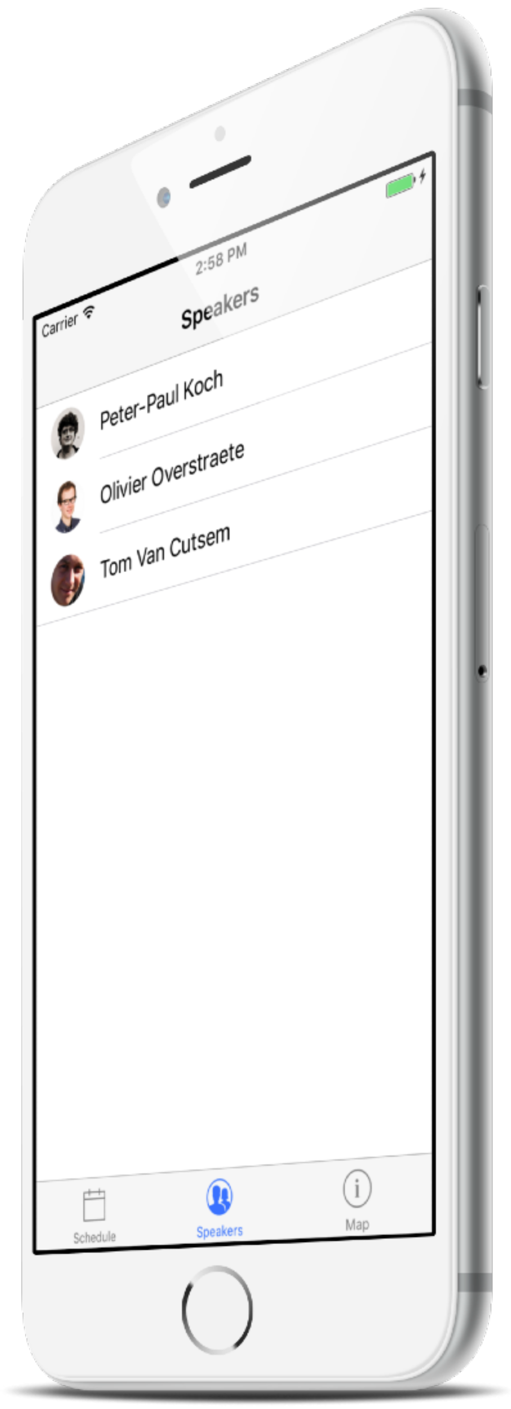
08:30

Making an app with Angular4 and Ionic3
08:30 - 09:45 — D102

10:00

From Documents to Dialogues: scripting the Conv
10:00 - 11:15 — D103

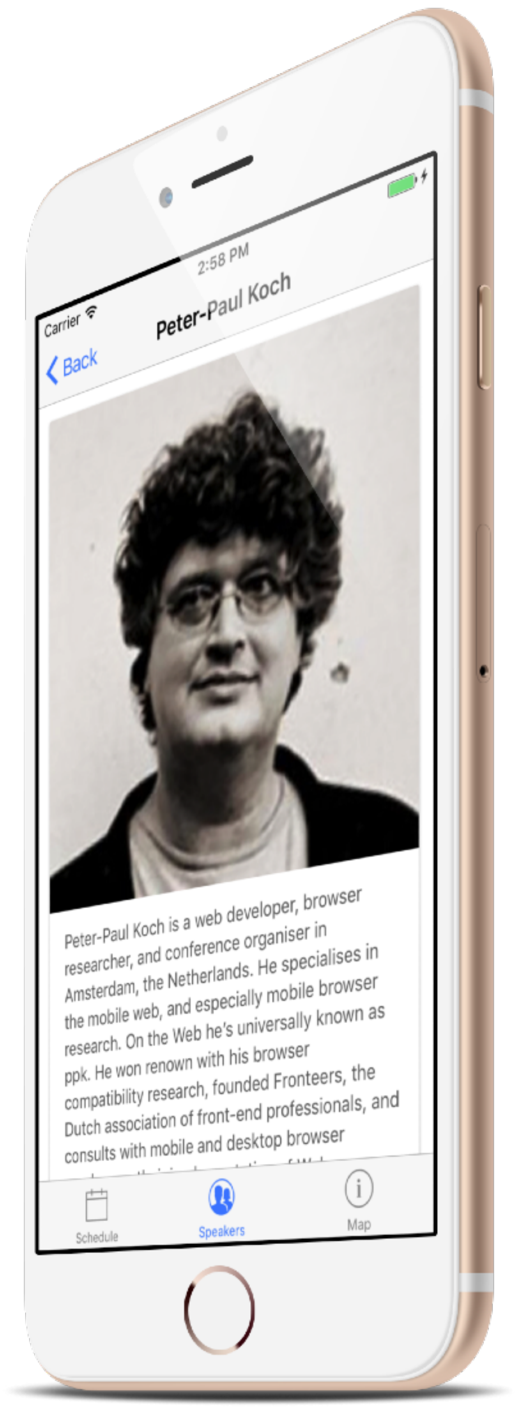
Schedule Speakers Map


Carrier 2:58 PM

Speakers

- Peter-Paul Koch
- Olivier Overstraete
- Tom Van Cutsem

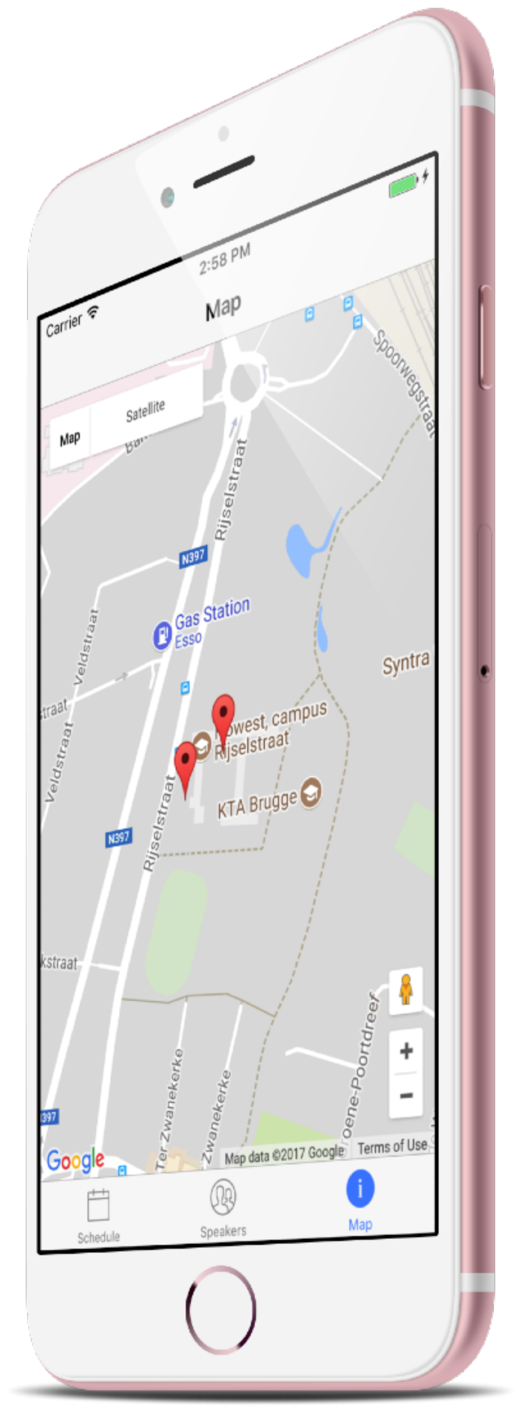
Schedule Speakers Map


Carrier 2:58 PM

Back Peter-Paul Koch

Peter-Paul Koch is a web developer, browser researcher, and conference organiser in Amsterdam, the Netherlands. He specialises in the mobile web, and especially mobile browser research. On the Web he's universally known as ppk. He won renown with his browser compatibility research, founded Fronteers, the Dutch association of front-end professionals, and consults with mobile and desktop browser

Schedule Speakers Map



NOWJOBS NOW
Get the job done.

Ionic Install

- Ionic CLI & Cordova

```
$ npm install -g ionic cordova
```

Getting started

- Make Ionic project

```
$ ionic start <name>
```

- Start from template

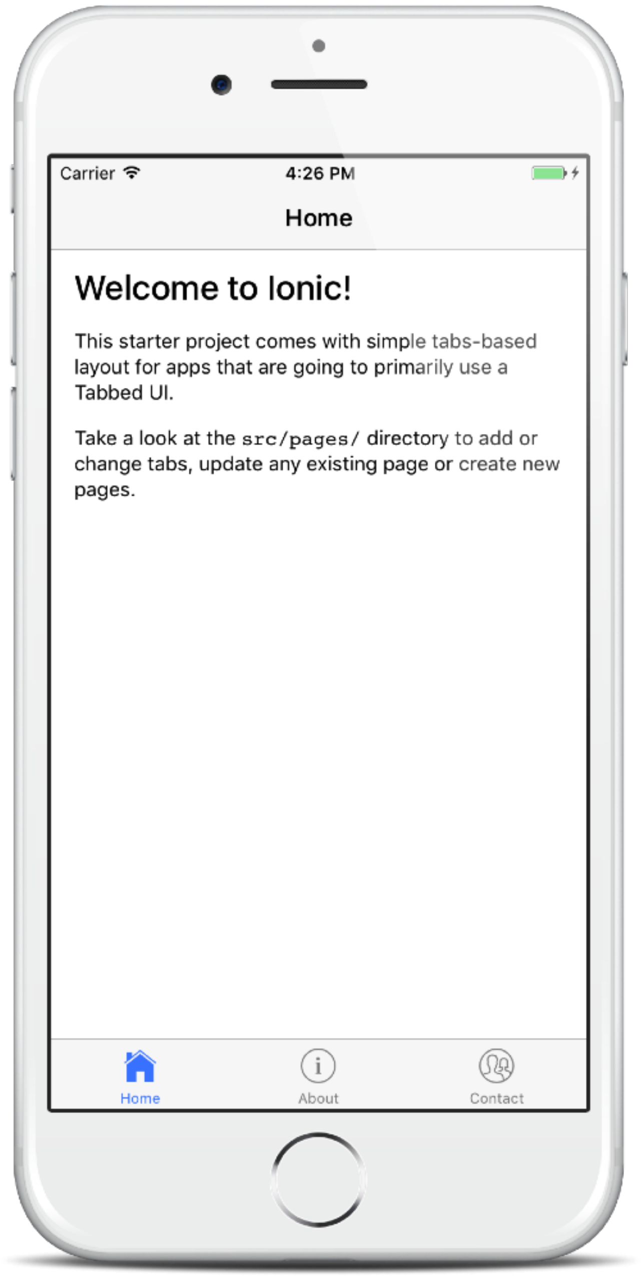
```
$ ionic start <name> <template>
```

  Tabs, sidemenu, maps, …

- See template list

```
$ ionic start --list
```

Ionic templates


Carrier 4:26 PM

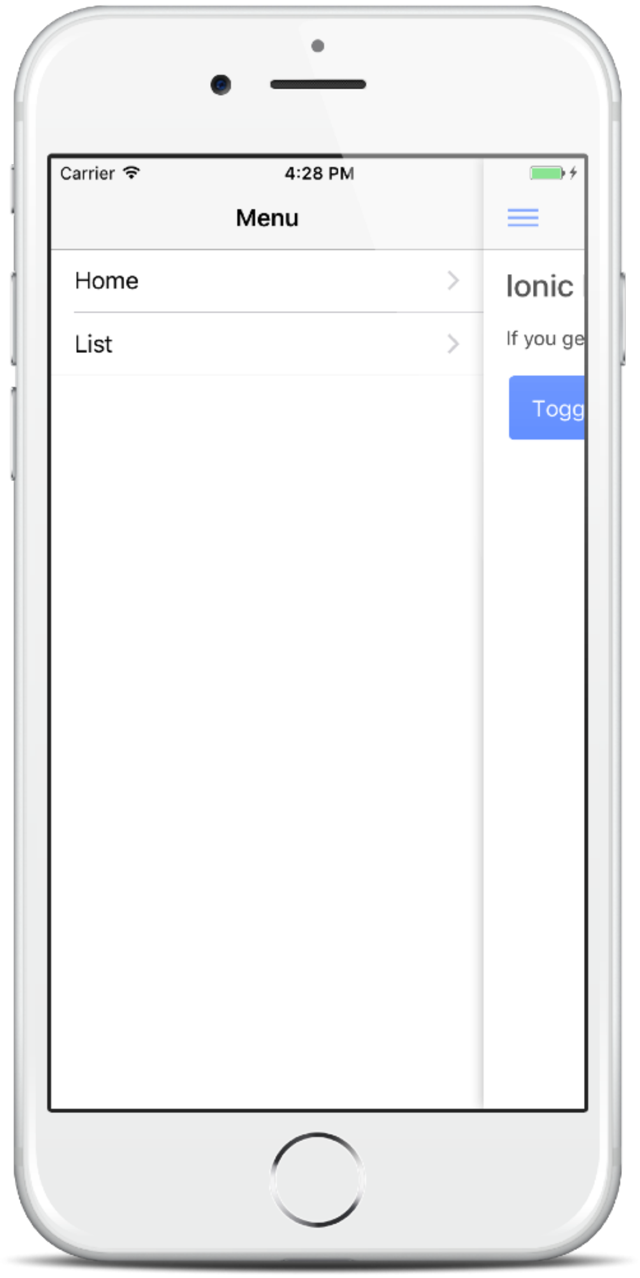
Home

Welcome to Ionic!

This starter project comes with simple tabs-based layout for apps that are going to primarily use a Tabbed UI.

Take a look at the `src/pages/` directory to add or change tabs, update any existing page or create new pages.

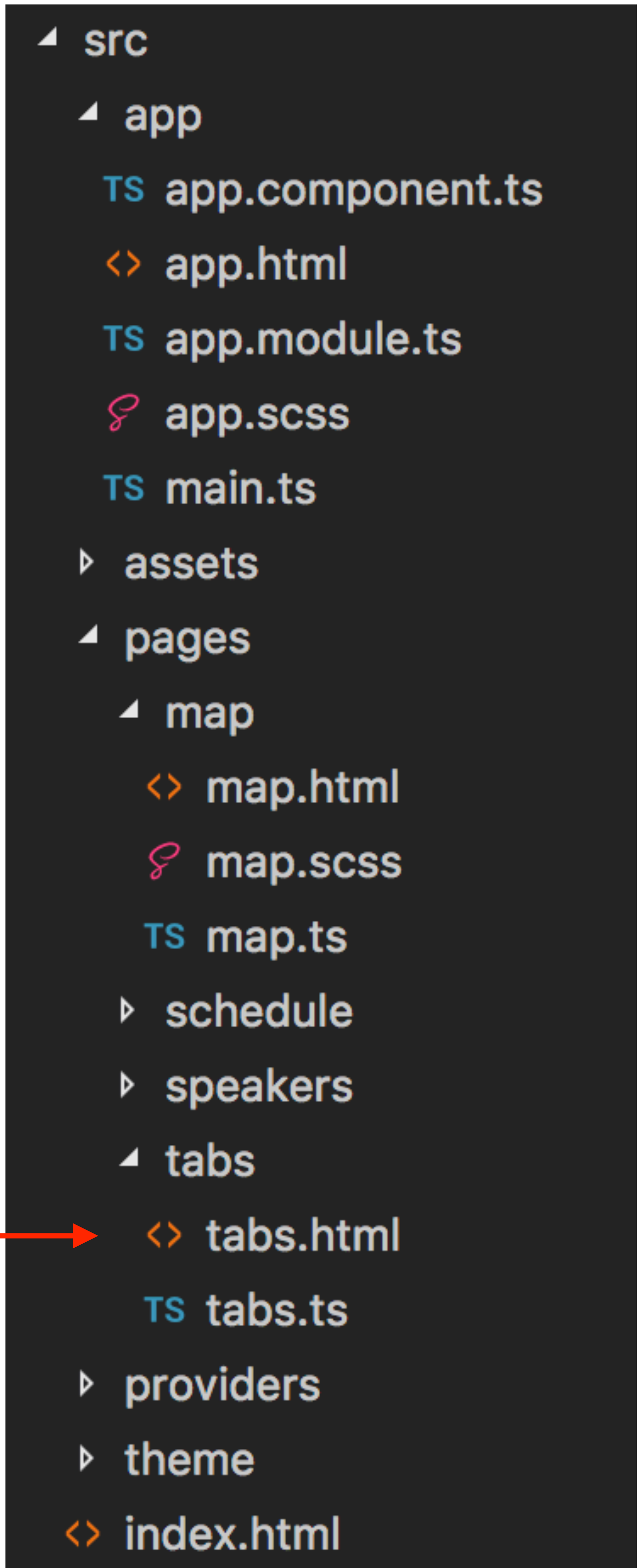
Tabs


Carrier 4:28 PM

Sidemenu

NOWJOBS NOW
Get the job done.

Setting up tabs


```
src
  app
    app.component.ts
    app.html
    app.module.ts
    app.scss
    main.ts
  assets
  pages
    map
      map.html
      map.scss
      map.ts
    schedule
    speakers
    tabs
      tabs.html
      tabs.ts
  providers
  theme
  index.html
```

```
<ion-tabs>

  <ion-tab [root]="tab1Root"
           tabTitle="Schedule"
           tabIcon="calendar">
  </ion-tab>

  <ion-tab … ></ion-tab>
  <ion-tab … ></ion-tab>

</ion-tabs>
```

All icons:
https://ionicframework.com/docs/ionicons/

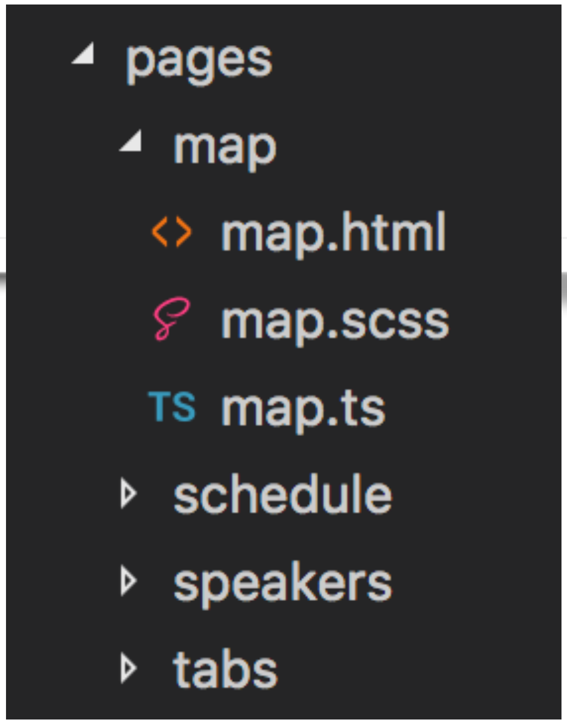
NOWJOBS
Get the job done.

Setting up tabs

```
import { SchedulePage } from '../schedule/schedule';
…

@Component({
  templateUrl: 'tabs.html'
})

export class TabsPage {

  tab1Root = SchedulePage;
  tab2Root = SpeakersPage;
  tab3Root = MapPage;

  constructor() {}
}
```

Creating a page

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';

@Component({
  selector: 'page-map',
  templateUrl: 'map'
})

export class MapPage {
  …
}
```



```
pages
  map
    map.html
    map.scss
    map.ts
  schedule
  speakers
  tabs
```

NOWJOBS NOW
Get the job done.

Custom Ionic elements

Top title bar

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Page title
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content>
</ion-content>
```

Icons

```
<ion-icon
  name=“heart”>
</ion-icon>
```

https://ionicframework.com/docs/components/

Custom Ionic elements

List of input fields

```
<ion-list>
  <ion-item>
    <ion-label fixed>Username</ion-label>
    <ion-input type="text" value=""></ion-input>
  </ion-item>
  …
</ion-list>
```

https://ionicframework.com/docs/components/

Run your code

- Run in the browser `$ ionic serve`
- Compare all platforms `$ ionic lab`

Demo app on GitHub

https://github.com/iOlivier/JSconf-Belgium-Ionic-app.git

In deze demo werken tot aan eerste pagina met statische data.

Na deze demo dan uitleg over service/provider om deze dan toe te passen in de volgende demo.

Demo

1. *Start the project*
2. *Setup the tabs*
3. *Setup the first page*

Make a service

- Singleton object
- fetching content
- inherit from Angular Service object
- Good practice: Convert incoming data to object
- If wanted: do caching, provide other functions

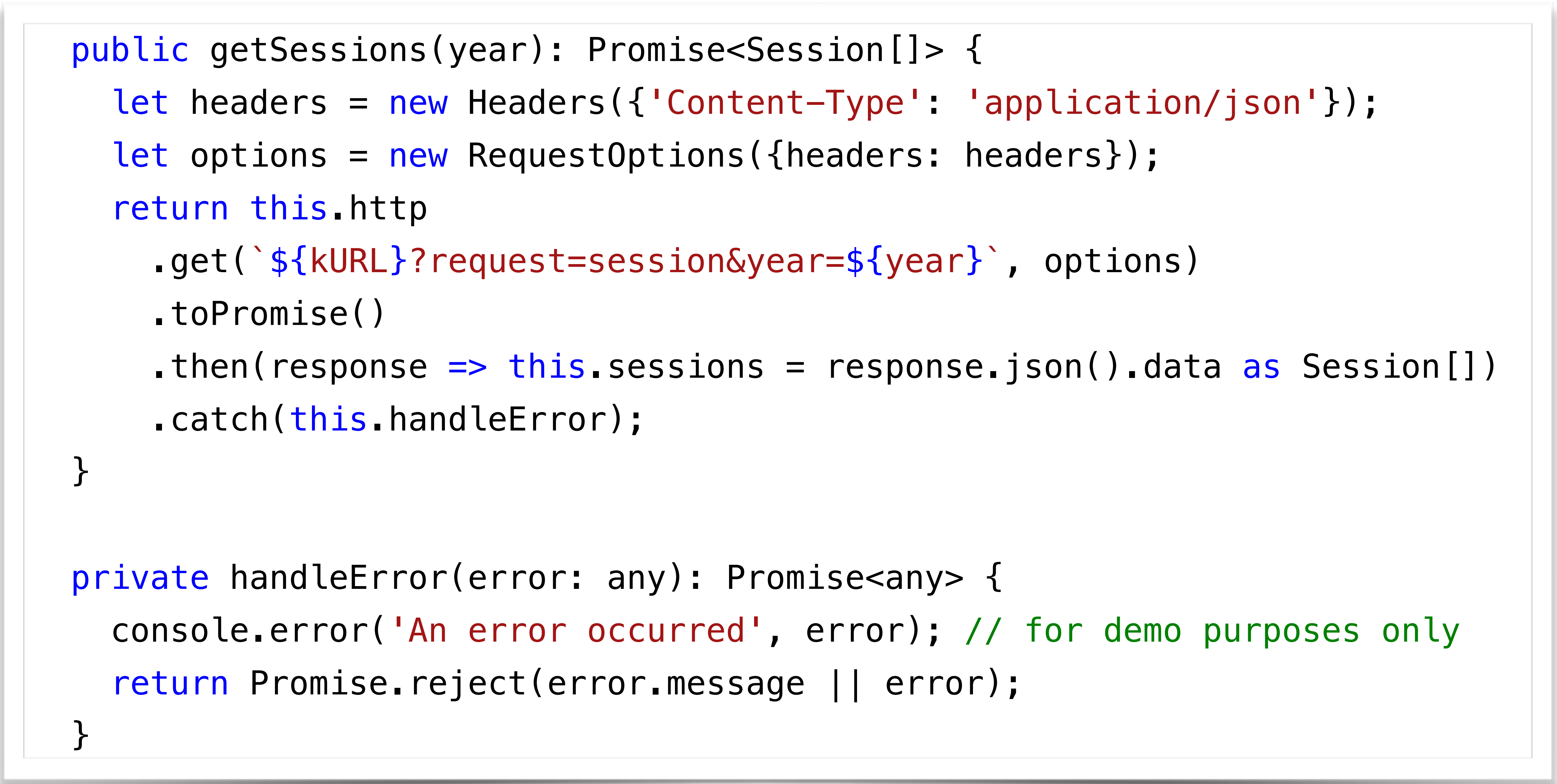
SessionsService

Minimal service
Calling http web service
Convert Observable to Promise

```
@Injectable()
export class SessionService {
  private url: '...';

  constructor(private http: Http) {
  }

  public getSessions(year): Promise<Session[]> {
   return this.http
      .get(this.url)
      .toPromise()
      .then(response => response.json().data as Session[])
  }
}
```

getSessions

parse and return sessions



```
public getSessions(year): Promise<Session[]> {
  let headers = new Headers({'Content-Type': 'application/json'});
  let options = new RequestOptions({headers: headers});
  return this.http
    .get(`${kURL}?request=session&year=${year}`, options)
    .toPromise()
    .then(response => this.sessions = response.json().data as Session[])
    .catch(this.handleError);
}

private handleError(error: any): Promise<any> {
  console.error('An error occurred', error); // for demo purposes only
  return Promise.reject(error.message || error);
}
```

getSpeakers

Could be server call

But Array functions are so nice

Return a resolved promise

```
getSpeakers(year): Promise<any> {
  // fake service call, cheat: return a resolved promise

  return Promise
    .resolve(this.sessions
                 .map(session => new Speaker(session.speaker, session.bio));
}
```

```
const url = "https://jsconf.be/static/images/speakers/";

export class Speaker {
  name = "";
  bio =  "";
  url = "";

  constructor(name, bio) {
    this.name = name;
    this.bio = bio;
    this.url = url + name.replace(" ", "-").toLowerCase() + ".jpg";
  }
}
```

using Services

Services are Injectable

Don't use "new"

```
export class AppComponent {
  sessionService: SessionService;

  constructor(sService: SessionService) {
    this.sessionService = sService;
  }
}
```

So let it inject with
  Angular dependency injection
  in the constructor

Add them to your module

```
@NgModule({
  declarations: [
    AppComponent, SessionComponent
  ],
  imports: [
    BrowserModule, FormsModule, HttpModule
  ],
  providers: [SessionService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

fetchSessions

get them when the component is **ready** for them

```
export class AppComponent {
  title = 2017;
  selectedSession = null;
  sessions = [];
  sessionService: SessionService;

  constructor(sService: SessionService) {
    this.sessionService = sService;
  }

  ngOnInit() {
    this.fetchSessions();
  }

  fetchSessions(): void {
    this.sessionService.getSessions(2017)
                     .then(sessions => this.sessions = sessions);
  }
}
```

Store them in an instance variable
Angular's binding will automatically update the DOM tree

Angular Lifecycles

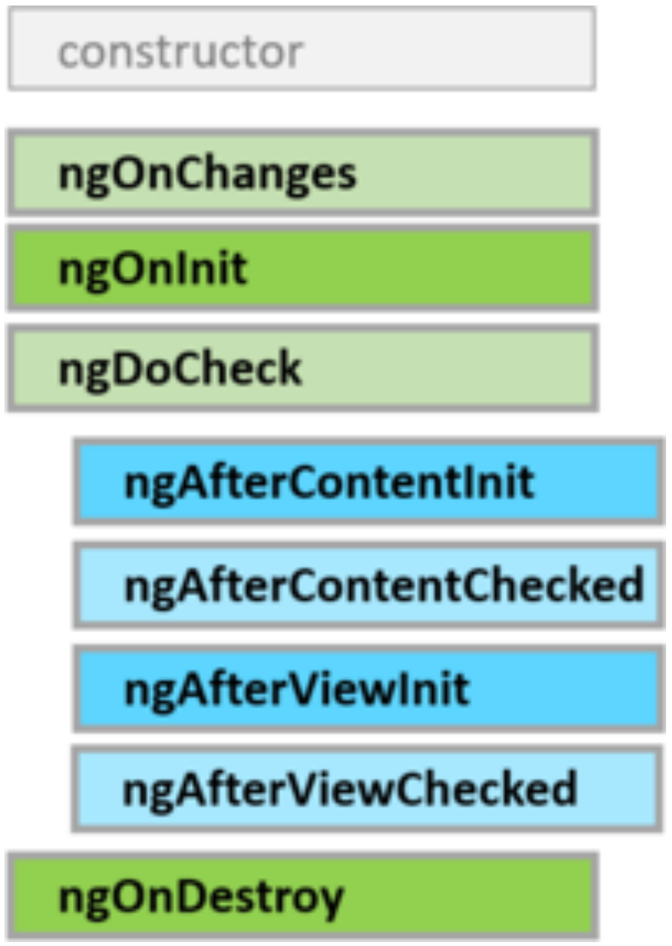
Some important / obvious:

ngOnChanges()
- when data-bound input properties are changed
- receive old / new values

ngOnInit()
- when input properties are assigned, first display of data bound values, after first ngOnChanges

ngAfterViewInit()
- after view is completely set up, including children

ngOnDestroy()
- when component goes away



Misschien kunnen we ook hier eens de vergelijking maken met de ionic lifecycles

https://angular.io/guide/lifecycle-hooks

Demo

Setup schedule page with data provider

NOWJOBS NOW
Get the job done.

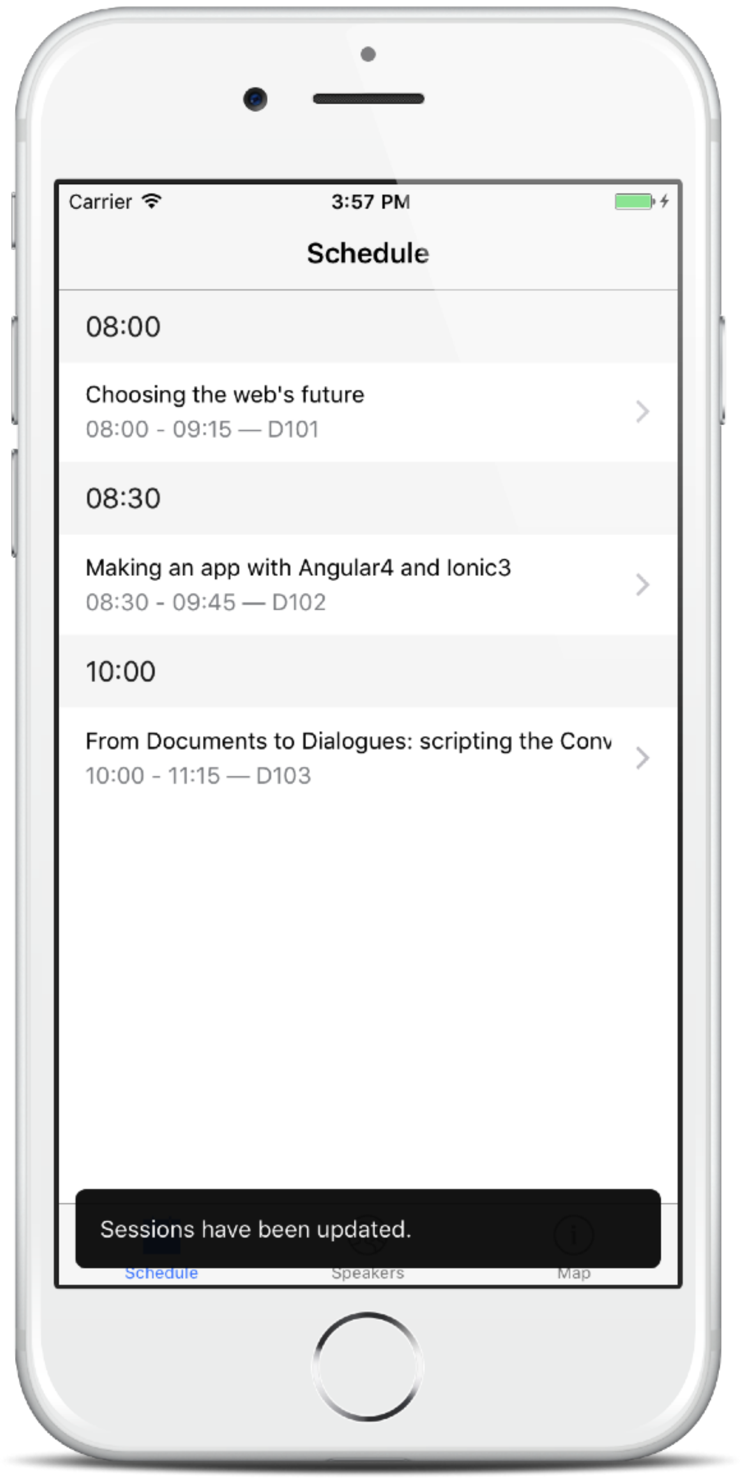
JS

Refresh list

```
<ion-refresher (ionRefresh)="doRefresh($event)">
  <ion-refresher-content></ion-refresher-content>
</ion-refresher>
```

```
doRefresh(refresher: Refresher) {

  this.confData.getSessions(new Date().getFullYear())
  .then(sessions => {
    this.sessions = sessions;
    refresher.complete();
  });
}
```

ToastController



Carrier 3:57 PM

Schedule

08:00

Choosing the web's future
08:00 - 09:15 — D101

08:30

Making an app with Angular4 and Ionic3
08:30 - 09:45 — D102

10:00

From Documents to Dialogues: scripting the Conv
10:00 - 11:15 — D103

Sessions have been updated.

ToastController

```
import { ToastController } from 'ionic-angular';

constructor(private toastCtrl: ToastController, …) { … }

doRefresh(refresher: Refresher) {
    this.confData.getSessions(new Date().getFullYear()).then(sessions => {
      this.sessions = sessions;
      refresher.complete();

      const toast = this.toastCtrl.create({
        message: 'Sessions have been updated.',
        duration: 3000
      });
      toast.present();
    });
 }
```

Demo

Add refresher to list

NavController

```
import { NavController } from 'ionic-angular';
```

```
export class SpecificPage {

  constructor(public navCtrl: NavController) { … }

  goToDetailScreen() {
    this.navCtrl.push(SpecificDetailPage, { nr: 100 });
  }

  goBack() {
    this.navCtrl.pop();
  }
}
```

NavParams

```
import { NavParams } from 'ionic-angular';
```

```
export class SpecificPage {

  private number = 0;

  constructor(public navParams: NavParams) {
    this.number = navParams.get("nr");
  }
}
```

Demo

Creating a detail view

Passing data to that view

Nu we de basis hebben gezien kunnen ze het zelf eens proberen voor de tweede tab. Deze tab volgt dezelfde werkwijze.

Hands-on

Try to make the speakers page

NOWJOBS NOW
Get the job done.

JS

ViewChild

```
import { ElementRef } from '@angular/core';
…
export class MapPage {

  @ViewChild('mapCanvas') mapElement: ElementRef;

  createMap() {
    …
    let mapEle = this.mapElement.nativeElement;
    …
  }
}
```

```
<ion-content>
  <div style="height: 100%; width: 100%" #mapCanvas></div>
</ion-content>
```

Zet je die stijl niet beter in Styles: ['']

NOWJOBS NOW
Get the job done.

JS

Demo van de derde tab (Map). Dit is tegelijk een demo van ViewChild

Demo

ViewChild example

NOWJOBS NOW
Get the job done.

Deploy to device

- Add platform
- Build app for platform
- Run app on device

```
$ ionic platform add <platform>

$ ionic cordova build <platform>

$ ionic cordova run <platform>
```

Thank you

Oliver Overstraete

Johan Coppieters