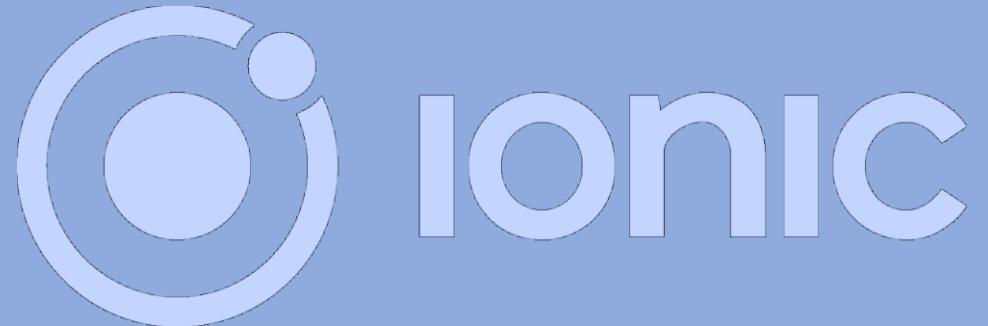


Application Mobiles



Diginamic & Christophe Germain

Bref historique des mobiles



~1990 Norme GSM 2G

1992 1er smartphone : IBM Simon

1999 Nokia 3210, Blackberry OS (C++)

2002 BlackBerry 5810

2006 Norme GSM 3G

2007 iPhone 1, lancement d'iOS (XNU)

2007 Lancement d'Android (Linux)

2008 App store et Android Market

2009 Smartphone Samsung Galaxy i7500

2010 Norme GSM 4G, Windows Phone (C++/C#)

2012 Google Play : Android Market + Google Movies + Google Music



Applications hybrides :

- ▶ Mix entre applications web et applications natives :
Code « web » + librairie accédant aux fonctionnalités de l'OS
⇒ appli native
- ▶ **Apache Cordova** (2009) : code en JavaScript, HTML, CSS
- ▶ **Xamarin** (2011) : code en C#
- ▶ **React Native** (2015) : code en Javascript
+ code natif Java , Objective-C ou Swift 
- ▶ **Ionic Capacitor** (2018) : code en JavaScript, HTML, CSS
- ▶ **Avantages** : Cross-plateforme, fonctionnalités de l'OS
- ▶ **Inconvénients** : 1 production par OS, installations  

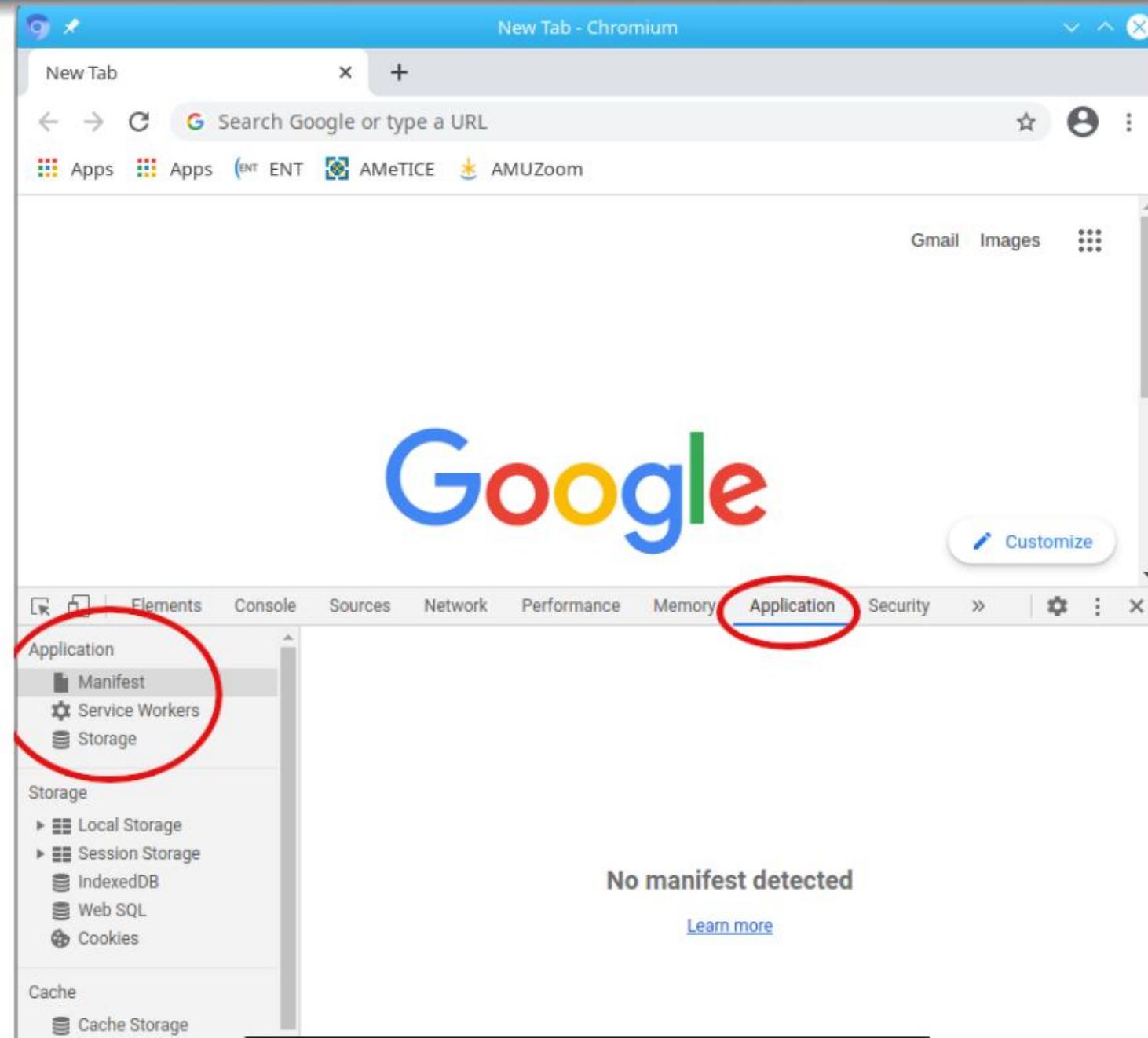
Applications mobiles (3/3)

Progressive Web Applications **PWA** :

- ▶ Amélioration des applications hybrides :
 - ▶ **Fiabilité** : peut fonctionner sans réseau (cache)
 - ▶ **Sécurité** : utilisation exclusive de réseau sécurisé (https)
 - ▶ **Installable** : comme une appli native mais via le web
 - ▶ **Progressif** : Updates automatiques via des notifications

- ▶ 3 caractéristiques pour être une **PWA** :
 - ▶ Utilisation de https
 - ▶ Avoir un manifeste (installations)
 - ▶ Utiliser un **Service Worker** (updates, cache, etc.)

PWA et developer tool



Objectif de la partie « Applications mobiles »

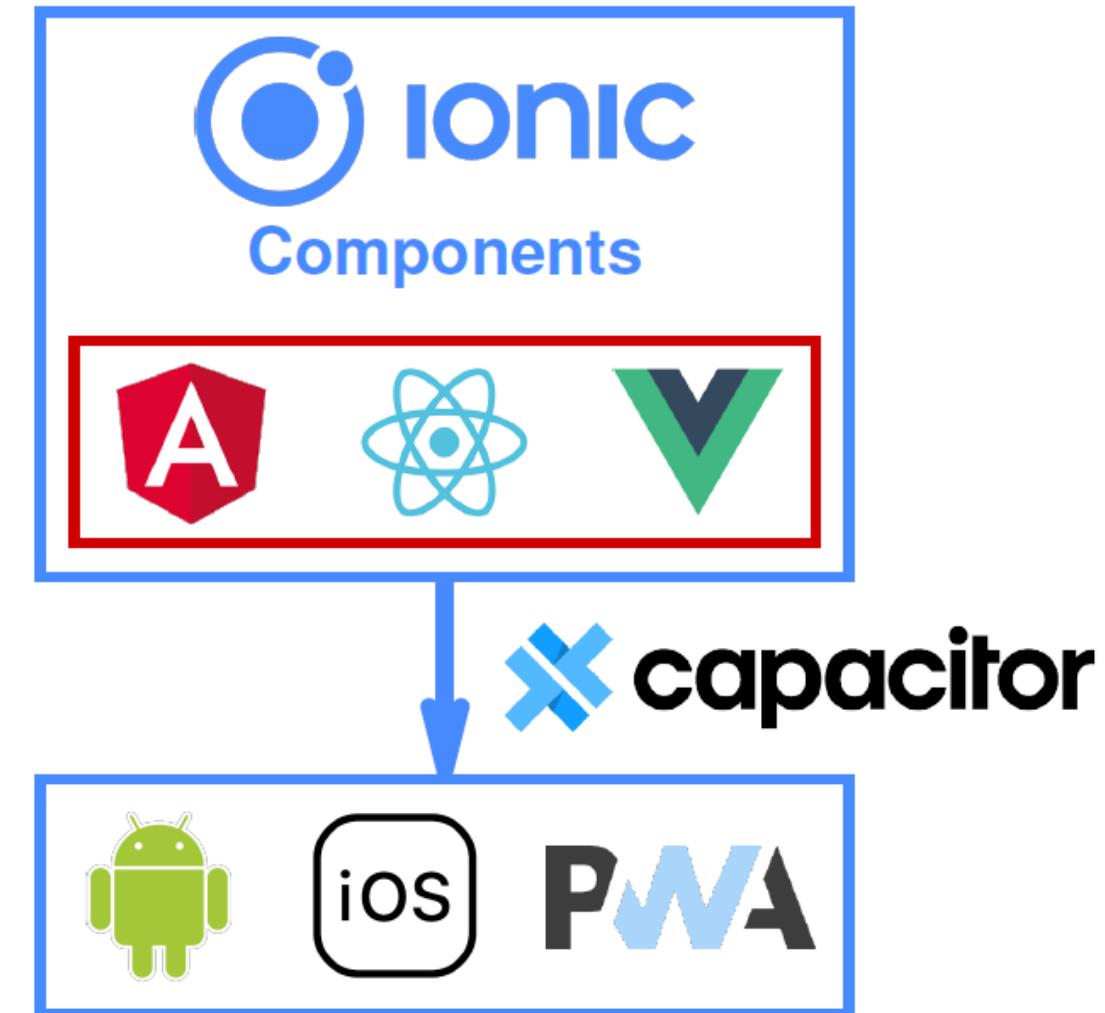
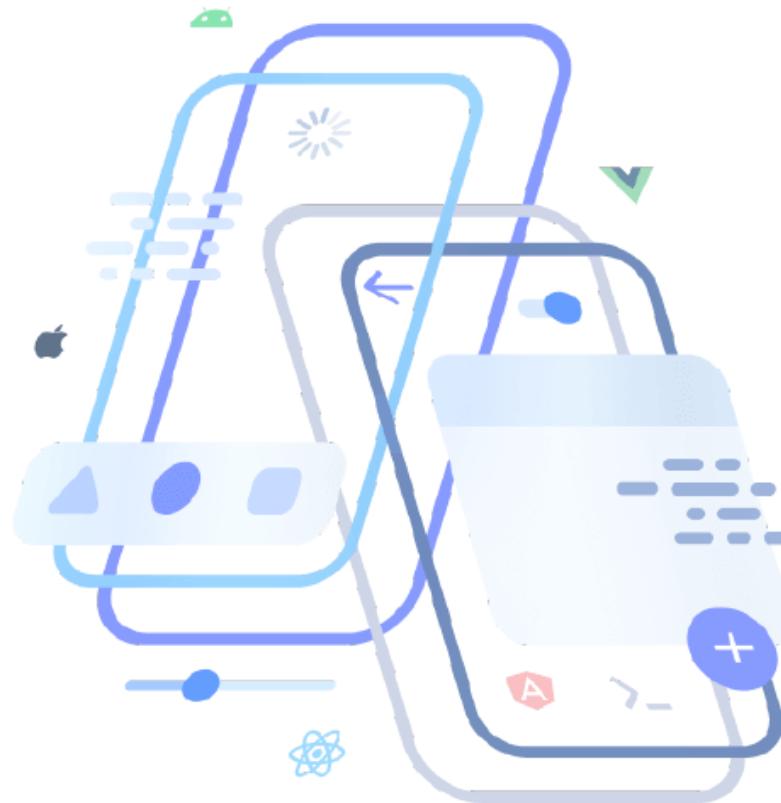
Objectif : développement d'une application hybride



1 Utilisation de Ionic

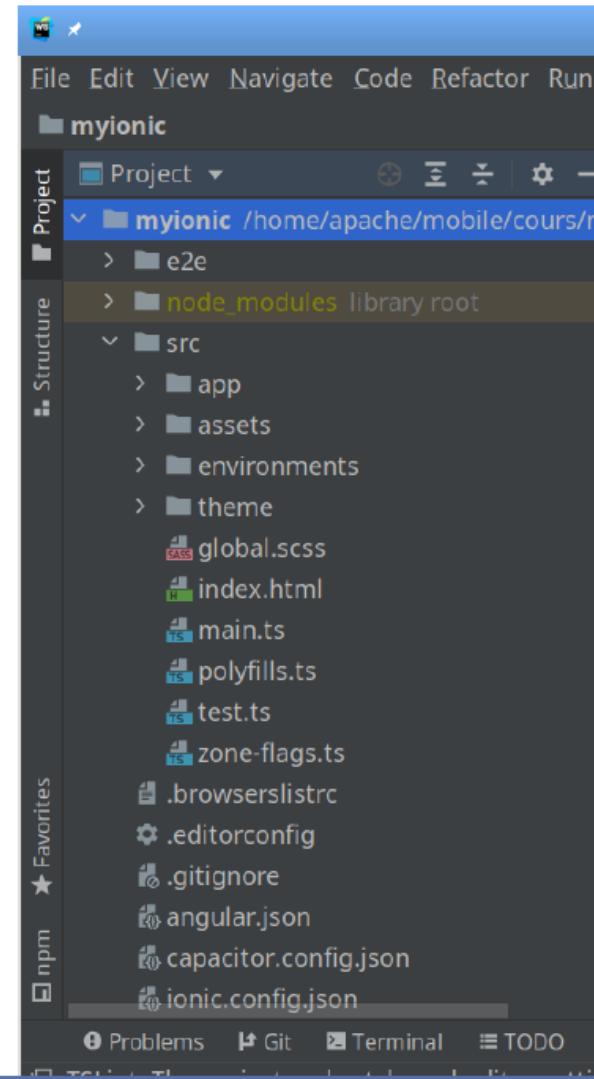
2 Utilisation de Ionic Capacitor

3 TPs : portage du forum sur et/ou



Premiers pas en Ionic

- ▶ Installation : `npm install -g @ionic/cli`
- ▶ Création projet : `ionic start monprojet`
 - ▶ Framework :   
 - ▶ Template : blank
 - ▶ Compte ionic : au choix
- ▶ Test : `ionic serve`
⇒ `http://127.0.0.1:8100`
- ▶ Capacitor : inclus par défaut



Anatomie d'une application / page

The screenshot shows a code editor interface with two files open:

- app.component.html**:

```
1  <!--  
2   Le composant racine Ionic débute  
3   et termine par une balise ion-app,  
4   qui contient un ion-router-outlet  
5   qui permet d'accéder aux  
6   différentes pages (vues).  
7  -->  
8  <ion-app>  
9    <ion-router-outlet></ion-router-outlet>  
10 </ion-app>
```
- home.page.html**:

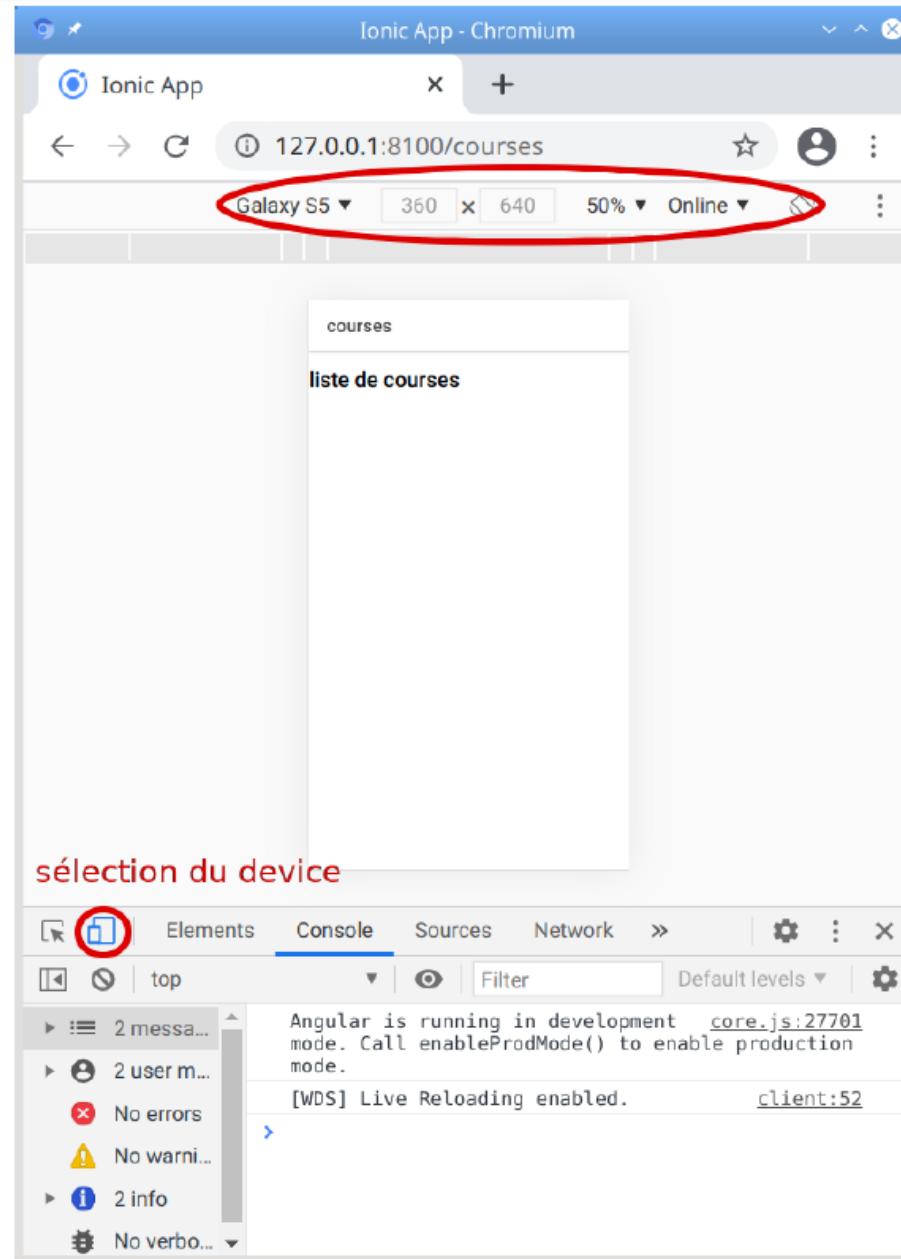
```
1  <!-- haut de la page -->  
2  <ion-header>  
3    <ion-toolbar>  
4      <ion-title>header de la page</ion-title>  
5    </ion-toolbar>  
6  </ion-header>  
7  <!-- le contenu de la page -->  
8  <ion-content>  
9    <div>  
10       <strong>Le contenu de la page</strong>  
11    </div>  
12  </ion-content>  
13  <!-- bas de la page -->  
14  <ion-footer>  
15    <ion-toolbar>  
16      <ion-title>footer de la page</ion-title>  
17    </ion-toolbar>  
18  </ion-footer>
```

A browser preview window shows the rendered output:

- Header**: header de la page
- Content**: Le contenu de la page
- Footer**: footer de la page

The code editor includes a sidebar with project navigation, a status bar at the bottom, and a toolbar with various icons.

Visualisation d'une page dans un browser



Création d'une nouvelle page

The screenshot shows a development environment with the following components:

- Code Editor:** An open file named `courses.page.html` containing the following code:

```
<ion-header>
  <ion-toolbar>
    <ion-title>courses</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <h2>liste de courses</h2>
</ion-content>
```
- Terminal:** A terminal window showing the command used to generate the page:

```
gonzales@shalmaneser:/home/apache/mobile/cours/myionic$ ionic generate page courses
```
- Browser Preview:** An Ionic App - Chromium browser window displaying the generated page at `127.0.0.1:8100/courses`. The page title is "courses" and the content is "liste de courses".
- IDE UI:** The browser preview and terminal are part of a larger IDE interface with tabs for `home.page.html` and `courses.page.html`, and toolbars for Angular CLI Server and Git.

Two specific elements are highlighted with red circles:

- The URL in the browser's address bar: `127.0.0.1:8100/courses`.
- The command in the terminal: `ionic generate page courses`.

Chargement des pages (1/3)

The screenshot shows a code editor with the following details:

- Title Bar:** myionic - app-routing.module.ts
- File Menu:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Project Bar:** myionic > src > app > app-routing.module.ts
- Toolbar:** Angular CLI Server, Git status indicators.
- Left Sidebar:** Project, Commit, Structure, Bookmarks, npm.
- Code Area:** The file content is as follows:

```
// Chaque page est associée à un module du même nom. Quand on souhaite accéder à la
// page, on charge le module qui la contient (on appelle cela un "lazy-loaded module").
// Cela permet de ne pas charger toute l'application en une seule fois => cela paraît
// plus rapide quand l'utilisateur accède à l'application.

const routes: Routes = [
  {
    path: 'home',
    loadChildren: () => import('./home/home.module').then( m => m.HomePageModule)
  },
  {
    path: 'courses',
    loadChildren: () => import('./courses/courses.module').then( m => m.CoursesPageModule)
  }
];

@NgModule({
  imports: [
    // Le RouterModule.forRoot crée et initialise le module de "routage" utilisé dans
    // toute l'application. Dans sa config, on spécifie une stratégie pour "préloader"
    // les modules de l'application :
    // QuicklinkStrategy: précharge seules les routes référencées par la page actuelle.
    // PreloadAllModules: précharge tous les modules
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ]
})
```

The code defines a routes array containing two entries for 'home' and 'courses' paths, each loading a specific module. It then uses the @NgModule decorator to import RouterModule and specify the routes array. The RouterModule.forRoot method is used with a preloading strategy set to PreloadAllModules.

Chargement des pages (2/3)

The screenshot shows a code editor interface with the following details:

- Title Bar:** myionic - home.module.ts
- File Menu:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Toolbar:** Angular CLI Server, Git status indicators, search, and settings.
- Project Explorer:** Shows the project structure: myionic ~tmp, .angular, e2e, node_modules, src, app, courses, and home. The home folder is expanded, showing files like home-routing.module.ts, home.page.ts, and home.module.ts.
- Code Editor:** The current file is home.module.ts. The code defines a module named HomePageModule. It imports CommonModule, FormsModule, IonicModule, and HomePageRoutingModule from '@angular' and '@ionic/angular'. It also imports HomePage from './home.page'. The @NgModule decorator specifies imports (CommonModule, FormsModule, IonicModule, HomePageRoutingModule), declarations ([HomePage]), and exports (HomePageModule). A note in the code indicates that imports specify modules used by home.module, and declarations specify the page and components used.
- Bottom Status Bar:** Git, TODO, Problems, Terminal, Event Log, and file system icons.

```
// Chaque page a un module, similaire au app.module.ts d'Angular
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { IonicModule } from '@ionic/angular';
import { FormsModule } from '@angular/forms';
import { HomePage } from './home.page';

import { HomePageRoutingModule } from './home-routing.module';

// dans les imports, on indique les modules qui sont utilisés par home.module.
// dans les déclarations, on spécifie la page et les composants utilisés.

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    HomePageRoutingModule
  ],
  declarations: [HomePage]
})
export class HomePageModule {}
```

Chargement des pages (3/3)

```
// Chaque module a son propre routage. Cela permet, notamment, d'inclure plusieurs
// pages dans le même module.
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomePage } from './home.page';

const routes: Routes = [
  {
    // dans app-routing.module.ts, le module HomePageModule est chargé quand
    // on accède à la route '/home'. Ce que l'on indique ci-dessous, c'est
    // que, pour accéder à la page HomePage, l'utilisateur doit accéder dans
    // son browser à la route '/home/myHome'.
    path: 'myHome',
    component: HomePage,
  },
];

@NgModule({
  // RouterModule.forChild indique que l'on rajoute des routes au RouterModule
  // de l'application. MAIS on ne crée pas de nouveau RouterModule. Une appli
  // a donc 1 seul RouterModule.forRoot et 0 ou plusieurs RouterModule.forChild.
  imports: [RouterModule.forChild(routes)],
})
```

- ▶ **Création** : ionic generate page mapage
 - ⇒ **Extension** : .page
 - ⇒ crée le template HTML, le TypeScript, le CSS/SCSS et le module et son routage
- ▶ **Rapidité** : Modules des pages chargés dynamiquement
- ▶ **Composition** : Les pages contiennent des **composants**
 - ⇒ ionic generate component moncomposant
 - ⇒ Composant ≠ Page

Quelques composants ionic

Création d'une liste

The screenshot shows a code editor interface for an Ionic project named "myionic". The current file is "courses.page.html". The code displays a simple list component:

```
1 <ion-content>
2   <h2 class="ion-margin">liste de courses</h2>
3
4   <!-- ion-list : crée une liste avec le look and feel du smartphone (iphone, etc) -->
5   <ion-list inset="true">
6     <!-- chaque élément de la liste est un ion-item -->
7     <ion-item *ngFor="let elt of cours">
8       <!-- pour bien positionner le contenu de l'item, il est pratique d'encapsuler
9         celui-ci dans un ion-label -->
10      <ion-label>
11        <h4>{{elt.nom}}</h4>
12        <p>nb étudiants : {{elt.nb_etud}}</p>
13      </ion-label>
14    </ion-item>
15  </ion-list>
16
17 </ion-content>
18
```

A preview window on the right shows the resulting mobile-style list:

liste de courses

cours1
nb étudiants : 3

cours2
nb étudiants : 6

At the bottom of the IDE, there are tabs for Problems, Git, Terminal, TODO, and Event Log.

Création d'une ion-card

The screenshot shows a code editor interface with the following details:

- Title Bar:** myionic - courses.page.html
- File Path:** myionic > src > app > courses > courses.page.html
- Toolbar:** File Edit View Navigate Code Refactor Run Tools Git Window Help; Angular CLI Server dropdown; various icons for file operations.
- Project Explorer:** Shows files: courses.page.html (selected), courses.page.ts.
- Code Editor:** Displays the HTML template for a card component.

```
1  <ion-content>
2      <h2 class="ion-margin">Liste de courses</h2>
3      <ion-list inset="true">
4          <ion-item *ngFor="let elt of cours">
5              <ion-card>
6                  <!-- une ion-card possède un header et un content -->
7                  <ion-card-header>
8                      <!-- un header a un titre, voire un sous-titre -->
9                      <ion-card-title>{{elt.nom}}</ion-card-title>
10                     <ion-card-subtitle>sous-titre</ion-card-subtitle>
11                 </ion-card-header>
12                 <ion-card-content>
13                     <!-- un contenu peut être composé de plusieurs
14                         ion-item -->
15                     nb étudiants : {{elt.nb_etud}}
16                 </ion-card-content>
17             </ion-card>
18         </ion-item>
19     </ion-list>
20 </ion-content>
```
- Preview Area:** Shows the rendered user interface with two cards. Card 1 has title "cours1" and subtitle "SOUS-TITRE", with text "nb étudiants : 3". Card 2 has title "cours2" and subtitle "SOUS-TITRE", with text "nb étudiants : 6".
- Bottom Bar:** Problems, Git, Terminal, TODO.
- Status Bar:** 21:1 LF UTF-8 2 spaces TypeScript 4.0.5 master.

Les ancrés, moussaillon !

The screenshot shows an IDE interface with the following details:

- Title Bar:** myionic - courses.page.html
- File Menu:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Project Path:** myionic > src > app > courses > courses.page.html
- Toolbars:** Angular CLI Server, Git status indicators.
- Code Editor:** The file content is as follows:

```
1 <ion-content>
2   <h2 class="ion-margin">liste de courses</h2>
3   <ion-list inset="true">
4     <ion-item *ngFor="let elt of cours">
5       <ion-label>
6         <h4>{{elt.nom}}</h4>
7         <p>nb étudiants : {{elt.nb_etud}}</p>
8         <p>
9           <!-- en ionic, avec Angular, on peut la balise <a> ou la balise <ion-anchor>
10          (obsoète en ionic 5) mais il faut éviter <ion-router-link>. Les ancrés
11          permettent de passer d'une page à l'autre avec une animation. -->
12          <a routerLink='/home'>vers Home</a>
13        </p>
14       </ion-label>
15     </ion-item>
16   </ion-list>
17 </ion-content>
```

The code demonstrates the use of `<ion-anchor>` or `<a>` tags instead of `<ion-router-link>` to create anchors for navigating between pages in Ionic.

Sidebar: Project, Structure, Favorites, npm.

Bottom Bar: Problems, Git, Terminal, TODO, Event Log.

Status Bar: 18:1 LF UTF-8 2 spaces TypeScript 4.0.5 master

Les back buttons

The screenshot shows a code editor interface with the following details:

- Title Bar:** myionic - home.page.html
- File Path:** myionic > src > app > home > home.page.html
- Toolbar:** File Edit View Navigate Code Refactor Run Tools Git Window Help; Angular CLI Server dropdown, navigation icons, Git status.
- Left Sidebar:** Project, Structure, Favorites (with a star icon).
- Code Area:** home.page.html content (lines 1-19). The code includes an ion-header with an ion-toolbar containing an ion-title and an ion-back-button, and an ion-content section with a strong element.
- Preview Area:** A browser-like window showing the rendered page. It has a header with "header de la page" and a back button labeled "Back". The main content area displays "Le contenu de la page".
- Bottom Bar:** Problems, Git, Terminal, TODO, Event Log.
- Status Bar:** 19:1 LF UTF-8 2 spaces TypeScript 4.0.5 master.

2 manières de construire les menus :

① Créer des « side »-menus :

⇒ Utiliser la balise `<ion-menu>`

② Créer des menus déroulants :

⇒ Utiliser un popover

Un side-menu pour toute l'application (1/2)

```
File Edit View Navigate Code Refactor Run Tools Git Window Help
myionic > src > app > home > home.page.html
Project app.component.html : home.page.html
1  <ion-app>
2    <!-- Attention: tous les side-menus doivent
3      être déclarés à la racine de l'appli,
4      donc dans app.component.html, et en
5      dehors de ion-header, ion-content et
6      ion-footer -->
7    <ion-menu side="end"
8      menuId="first" contentId="main">
9      <ion-header>
10        <ion-toolbar color="primary">
11          <ion-title>Start Menu</ion-title>
12        </ion-toolbar>
13      </ion-header>
14      <ion-content>
15        <ion-list>
16          <ion-item>Item 1</ion-item>
17          <ion-item>Item 2</ion-item>
18        </ion-list>
19      </ion-content>
20    </ion-menu>
21    <!-- On doit passer le contentId des menus
22      au ion-router-outlet -->
23    <ion-router-outlet id="main"></ion-router-out
24  </ion-app>
```

```
File Edit View Navigate Code Refactor Run Tools Git Window Help
myionic - home.page.html
Project app.component.html : home.page.html
1  <ion-header [translucent]="true">
2    <ion-toolbar>
3      <ion-title>header de la page</ion-title>
4      <ion-buttons slot="start">
5        <ion-back-button></ion-back-button>
6      </ion-buttons>
7
8      <ion-buttons slot="end">
9        <!-- création d'un bouton pour pouvoir
10          ouvrir le menu de l'application -->
11        <ion-menu-button></ion-menu-button>
12      </ion-buttons>
13    </ion-toolbar>
14  </ion-header>
15
16  <ion-content>
17    <div>
18      <strong>Le contenu de la page</strong>
19    </div>
20  </ion-content>
21
```

Un side-menu pour toute l'application (2/2)

The image consists of two side-by-side screenshots of a mobile browser window displaying an Ionic application. Both screenshots show the same interface with a header and a menu icon.

Screenshot 1: The header contains "header de la page" and a "Back" button. Below the header is the text "Le contenu de la page". To the right of the header is a red circle highlighting a horizontal menu icon (three horizontal lines). Below this icon is the text "icone du menu".

Screenshot 2: The header contains "Mon menu" and a "Back" button. Below the header is the text "Le contenu". A vertical list of menu items is visible, each labeled "Menu Item".

Both screenshots include a developer toolbar at the bottom with tabs for Elements, Console, Sources, Network, and a Filter field. The console tab is active in both.

Plusieurs side-menus

myionic - app.component.html

```
File Edit View Navigate Code Refactor Run Tools Git Window Help
myionic > src > app > app.component.html
```

Project Commit Structure Bookmarks npm

```
app.component.html x
1 <ion-app>
2   <!-- Tous les side-menus ont le même
3       contentId. En revanche, ils ont des
4       menuId différents -->
5   <ion-menu menuId="first" contentId="main">
6     <ion-content>
7       <ion-list>
8         <ion-item>Item 1</ion-item>
9         <ion-item>Item 2</ion-item>
10        </ion-list>
11    </ion-content>
12  </ion-menu>
13
14  <ion-menu menuId="second" contentId="main">
15    <ion-content>
16      <ion-list>
17        <ion-item>XXX</ion-item>
18      </ion-list>
19    </ion-content>
20  </ion-menu>
21
22  <!-- On doit passer le contentId des menus
23      au ion-router-outlet -->
24  <ion-router-outlet id="main"></ion-router-outlet>
```

Angular CLI Server Git: ✓ ✓ ✓ ○ ↻ 🔍

```
home.page.ts x
1 import {Component, OnInit} from '@angular/core';
2 import {MenuController} from '@ionic/angular';
3
4 @Component({
5   selector: 'app-home',
6   templateUrl: 'home.page.html',
7   styleUrls: ['home.page.scss'],
8 })
9 export class HomePage implements OnInit {
10   // pour sélectionner le menu que HomePage
11   // affichera, on inclut par dependency
12   // injection une instance de MenuController.
13   constructor(private menu: MenuController) {}
14
15   ngOnInit() {
16     // dans le ngOnInit, on indique via son
17     // menuId quel menu sera accessible de la
18     // page (tous les autres seront rendus
19     // inaccessible)
20     this.menu.enable(
21       shouldEnable: true,
22       menuId: 'first');
23   }
24 }
```

Git Event Log

File Git TODO Problems Terminal 13:1 LF UTF-8 2 spaces* TypeScript 4.4.4 master

Organisation possible du code par composants

The screenshot shows a code editor with several files open, illustrating the organization of an Ionic Angular application:

- Project Structure:** The left sidebar shows the project structure with folders like `courses`, `home`, `menu-first`, `menu-second`, and `app`. Files include `home.module.ts`, `home.page.html`, `home.page.scss`, `home.page.spec.ts`, `home.page.ts`, `home-routing.module.ts`, `menu-first.component.ts`, `menu-first.component.html`, `menu-first.component.spec.ts`, `menu-second.component.ts`, `menu-second.component.html`, `menu-second.component.spec.ts`, `app.component.html`, `app.component.scss`, `app.component.spec.ts`, `app.component.ts`, `app.module.ts`, `app-routing.module.ts`, `assets`, `environments`, and `theme`.
- File `app.module.ts`:** This file contains the main application module configuration. It imports `BrowserModule`, `RouteReuseStrategy`, `IonicModule`, `IonicRouteStrategy`, `AppComponent`, `AppRoutingModule`, and `MenuFirstComponent`, `MenuSecondComponent`. It defines the `@NgModule` block with declarations, entry components, imports, providers, and bootstrap.
- File `menu-first.component.html`:** This template defines a `<ion-menu>` component with `menuId="first"` and `contentId="main"`. It contains an `<ion-content>` section with an `<ion-list>` containing two items: `Item 1` and `Item 2`.
- File `menu-second.component.html`:** This template defines a `<ion-menu>` component with `menuId="second"` and `contentId="main"`. It contains an `<ion-content>` section with an `<ion-list>`.
- File `app.component.html`:** This template includes the `<ion-app>` component. It includes the `<app-menu-first>` and `<app-menu-second>` components defined in the `menu-first` and `menu-second` modules. It also includes an `<ion-router-outlet id="main">` component.

The code editor interface includes tabs for `File`, `Edit`, `View`, `Navigate`, `Code`, `Refactor`, `Run`, `Tools`, `Git`, `Window`, and `Help`. A status bar at the bottom shows the time (16:14), encoding (UTF-8), spaces (2), TypeScript version (4.4.4), and branch (master).

Menu déroulant (1/5)

4 étapes :

- ① Créer un composant contenant le code du menu
- ② Rajouter le composant au module de la page
- ③ Dans le html de la page, créer un bouton pour ouvrir le menu
- ④ Dans le TypeScript de la page, ajouter une méthode créant un PopoverController

Menu déroulant (2/5)

The screenshot shows an IDE interface with the following details:

- File Path:** myionic > src > app > menu > menu.component.ts > MenuComponent
- File Type:** menu.component.ts
- Code Content (menu.component.ts):**

```
9  export class MenuComponent implements OnInit {  
10  // penser à rajouter MenuComponent aux déclarations de home.module.ts.  
11  // Le champ ci-dessous montre comment le MenuComponent peut transmettre  
12  // des informations au menu quand celui-ci s'ouvrira  
13  myprop!: string;  
14  nb = [1, 2, 3]; // les items du menu déroulant  
15  
16  constructor(private popover: PopoverController) {}  
17  
18  ngOnInit(): void {}  
19  
20  // la méthode dismiss renverra la valeur de event au HomeComponent  
21  closePopover(event): void { this.popover.dismiss(event); }  
22 }
```

- File Type:** menu.component.html
- Code Content (menu.component.html):**

```
1  <ion-content>  
2    <ion-label>Myprop: {{myprop}}</ion-label>  
3    <ion-list>  
4      <!-- quand on clique, on appelle la fonction qui renvoie l'item sur lequel on clique --&gt;<br/>5      <ion-item *ngFor="let elt of nb" (click)="closePopover(elt)">Menu Item {{elt}}</ion-item>  
6    </ion-list>  
7  </ion-content>
```

- Project Explorer:** Shows the project structure with 'menu' selected.
- Git Status:** Shows 02 changes, 1 addition, 3 deletions.
- Bottom Bar:** Includes Git, TODO, Problems, Terminal, and Event Log tabs.

Menu déroulant (3/5)

myionic – home.page.html

File Edit View Navigate Code Refactor Run Tools Git Window Help

myionic › src › app › home › home.page.html

Angular CLI Server

home.page.html

```
<ion-toolbar>
  <ion-title>header de la page</ion-title>
  <ion-buttons slot="start">
    <ion-back-button></ion-back-button>
  </ion-buttons>
  <!-- Nouveau bouton pour pouvoir ouvrir le menu. Ce bouton a la forme d'une
       icone "...", soit verticale soit horizontale, en fonction du smartphone -->
  <ion-buttons slot="end">
    <ion-button (click)="openMenu($event)">
      <ion-icon slot="icon-only"
                ios="ellipsis-horizontal"
                md="ellipsis-vertical"></ion-icon>
    </ion-button>
  </ion-buttons>
</ion-toolbar>
</ion-header>

<ion-content>
  <div>
    <strong>Le contenu de la page</strong>
  </div>

```

Project Structure Favorites npm

Problems Git Terminal TODO Event Log

Menu déroulant (4/5)

The screenshot shows the code editor interface of Visual Studio Code (VS Code) with the following details:

- Title Bar:** myionic - home.page.ts
- File Path:** myionic > src > app > home > home.page.ts
- Toolbar:** File Edit View Navigate Code Refactor Run Tools Git Window Help; Angular CLI Server dropdown; Git status icons.
- Code Editor:** The file content is as follows:

```
10 export class HomePage {
11     // quand on clique sur le bouton de menu, on veut que cela ouvre un popup
12     constructor(private popover: PopoverController) {}
13
14     openMenu(myevent: MouseEvent): void {
15         // on crée le popup contenant le code du menu
16         this.popover.create({
17             component: MenuComponent, // on précise quoi inclure dans le popup.
18             showBackdrop: true,
19             cssClass: 'my-menu-class', // on peut préciser un css.
20             event: myevent, // l'événement clic souris
21             componentProps: { // ici, on indique les propriétés que l'on veut initialiser
22                 myprop: 'xxxx' // dans le composant MenuComponent
23             }
24         }).then((popoverElement: HTMLIonPopoverElement) => {
25             popoverElement.present(); // affiche le menu
26             popoverElement.onDidDismiss().then((res: OverlayEventDetail<any>) => {
27                 console.log(res);
28             });
29         });
30     }
}
```

The code implements a dropdown menu using the Ionic Popover API. It defines a constructor that takes a PopoverController dependency. The openMenu method creates a new popover instance, specifying the component to be displayed (MenuComponent), setting the backdrop to be shown, applying a CSS class ('my-menu-class'), and providing the mouse event as the trigger. It then presents the popover and handles its dismissal by logging the result to the console.

Menu déroulant (5/5)

The image consists of two side-by-side screenshots of an Ionic application running in a Chromium browser window. Both screenshots show the same page structure:

- Header:** "header de la page" and a three-dot menu icon.
- Content Area:** "Le contenu de la page" and "Myprop : xxxx".
- Side Menu:** A vertical list with three items: "Menu Item 1", "Menu Item 2", and "Menu Item 3".

In the left screenshot, the side menu is open, displaying these three items. In the right screenshot, the side menu is closed, and the content area has expanded to fill the space.

At the bottom of each screenshot, a developer tools console is visible. The right-hand console shows the following log entry, which is circled in red:

```
[{"data": 2, "role": undefined} home.page.ts:27  
  data: 2  
  role: undefined  
  > __proto__: Object
```

A red annotation at the bottom of the right screenshot reads: "après avoir cliqué sur l'item 2" (after clicking on item 2).

Les alertes (1/3)

The screenshot shows the WebStorm IDE interface with the following details:

- Title Bar:** myionic - home.page.html
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Toolbar:** Angular CLI Server, Git status icons, and navigation icons.
- Project Structure:** Shows the file tree: myionic > src > app > home > home.page.html.
- Code Editor:** The main window displays the content of home.page.html. The code includes an ion-header section with an ion-buttons component containing an ion-button that triggers openMenu. It also features an ion-content section with a strong tag and a button that triggers openAlert(). A note in the code explains that while this template uses click events, TPs will use sendMessage().
- Side Panels:** Project, Structure, Favorites, and npm are visible on the left.
- Bottom Navigation:** Problems, Git, Terminal, TODO, Event Log.
- Status Bar:** 28:1 LF UTF-8 2 spaces TypeScript 4.0.5 master

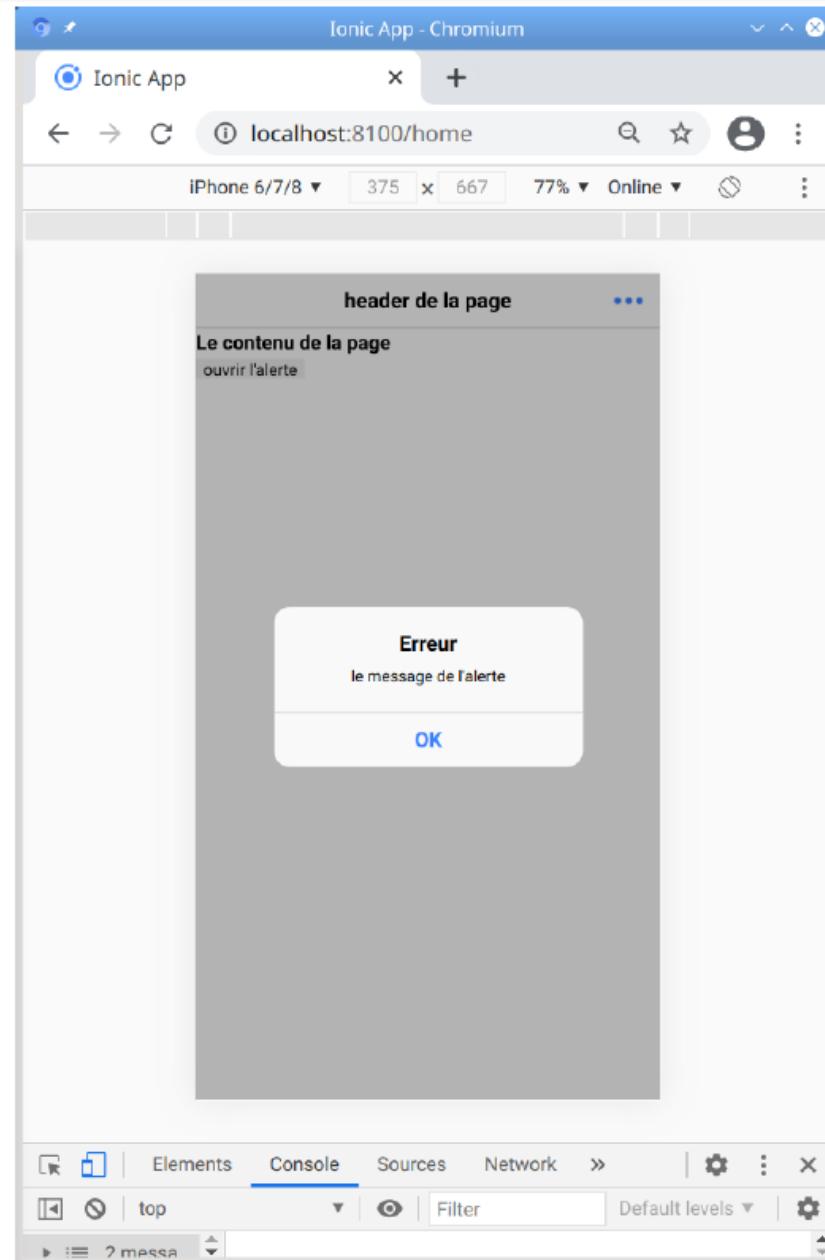
Les alertes (2/3)

The screenshot shows the Visual Studio Code interface with the following details:

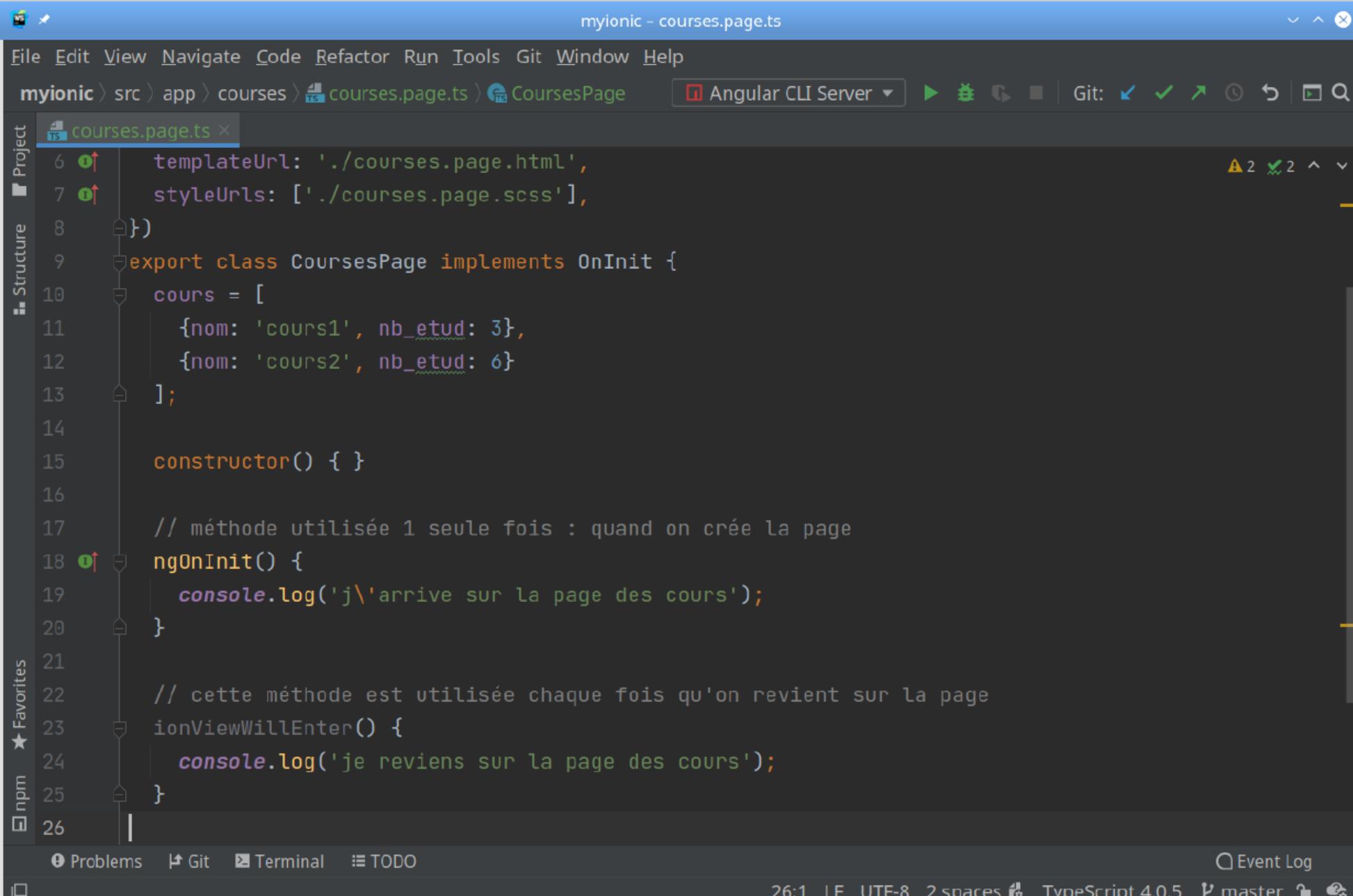
- Title Bar:** myionic - home.page.ts
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- breadcrumbs:** myionic > src > app > home > home.page.ts > HomePage
- Toolbar:** Angular CLI Server, navigation icons, Git status (blue arrow, green checkmark, green arrow), and search icon.
- Project Explorer:** Shows the file structure with 'home.page.ts' selected.
- Editor:** The code for 'HomePage' component:

```
8 styleUrls: ['home.page.scss'],
9 })
10 export class HomePage {
11     // pour ouvrir une alerte, il faut inclure par dependency injection un AlertController
12     constructor(private popover: PopoverController,
13                 private alertController: AlertController) {}
14
15     openMenu(myevent: MouseEvent): void {...}
16
17     async openAlert() {
18         const monMessage = 'le message de l\'alerte';
19         const alert = await this.alertController.create({
20             header: 'Erreur',
21             message: monMessage,
22             buttons: ['OK'],
23         });
24
25         await alert.present(); // afficher le popup de l'alerte
26         await alert.onDidDismiss(); // éventuellement faire une action si on ferme l'alerte
27     }
28 }
```
- Sidebar:** Favorites and npm tabs.
- Bottom Bar:** Problems, Git, Terminal, TODO, Event Log, and status bar showing 32:1 LF UTF-8 2 spaces, TypeScript 4.0.5, master branch, and a few icons.

Les alertes (3/3)



Faire une action quand on arrive/revient sur une page



```
myionic - courses.page.ts
File Edit View Navigate Code Refactor Run Tools Git Window Help
myionic> src > app > courses > courses.page.ts > CoursesPage Angular CLI Server ▶ Git: ✘ ✓ ↗ ↙ ⏪ ⏫ 🔍
courses.page.ts
6 ↑ templateUrl: './courses.page.html',
7 ↑ styleUrls: ['./courses.page.scss'],
8 })
9 export class CoursesPage implements OnInit {
10   cours = [
11     {nom: 'cours1', nb_etud: 3},
12     {nom: 'cours2', nb_etud: 6}
13   ];
14
15   constructor() { }
16
17   // méthode utilisée 1 seule fois : quand on crée la page
18 ↑ ngOnInit() {
19     console.log('je\arrive sur la page des cours');
20   }
21
22   // cette méthode est utilisée chaque fois qu'on revient sur la page
23   ionViewWillEnter() {
24     console.log('je reviens sur la page des cours');
25   }
26
```

The screenshot shows the code for the `courses.page.ts` file in an IDE. The code defines a `CoursesPage` component that implements the `OnInit` interface. It contains two methods: `ngOnInit()` and `ionViewWillEnter()`. Both methods log a message to the console using `console.log`.

```
myionic - courses.page.ts
File Edit View Navigate Code Refactor Run Tools Git Window Help
myionic> src > app > courses > courses.page.ts > CoursesPage Angular CLI Server ▶ Git: ✘ ✓ ↗ ↙ ⏪ ⏫ 🔍
courses.page.ts
6 ↑ templateUrl: './courses.page.html',
7 ↑ styleUrls: ['./courses.page.scss'],
8 })
9 export class CoursesPage implements OnInit {
10   cours = [
11     {nom: 'cours1', nb_etud: 3},
12     {nom: 'cours2', nb_etud: 6}
13   ];
14
15   constructor() { }
16
17   // méthode utilisée 1 seule fois : quand on crée la page
18 ↑ ngOnInit() {
19     console.log('je\arrive sur la page des cours');
20   }
21
22   // cette méthode est utilisée chaque fois qu'on revient sur la page
23   ionViewWillEnter() {
24     console.log('je reviens sur la page des cours');
25   }
26
```

Produire votre application native : le *setup*

2 méthodes pour autoriser Ionic à utiliser capacitor :

- ➊ Créer un projet intégrant directement capacitor :

```
ionic start monprojet blank
```

- ➋ Ajouter capacitor *a posteriori* :

```
ionic integrations enable capacitor
```

Spécifier les plateformes :

- ▶ Faire un `ionic build` auparavant (⇒ crée répertoire `www`)
- ▶ Application native  : `npm install @capacitor/android`
`npx cap add android`
- ▶ Application native  : `npm install @capacitor/ios`
`npx cap add ios`

Produire votre application native : la production

3 étapes :

- ① Compiler l'appli web : `ionic build`
- ② Synchroniser avec l'appli native : `npx cap copy`
- ③ Ouvrir Android studio ou XCode :
 -  : `npx cap open android`
 -  Préciser à la ligne de commande le « path » d'android-studio :
`export CAPACITOR_ANDROID_STUDIO_PATH=path`
 -  : `npx cap open ios`

Production de l'application native : Android studio

The screenshot shows the Android Studio interface with the following details:

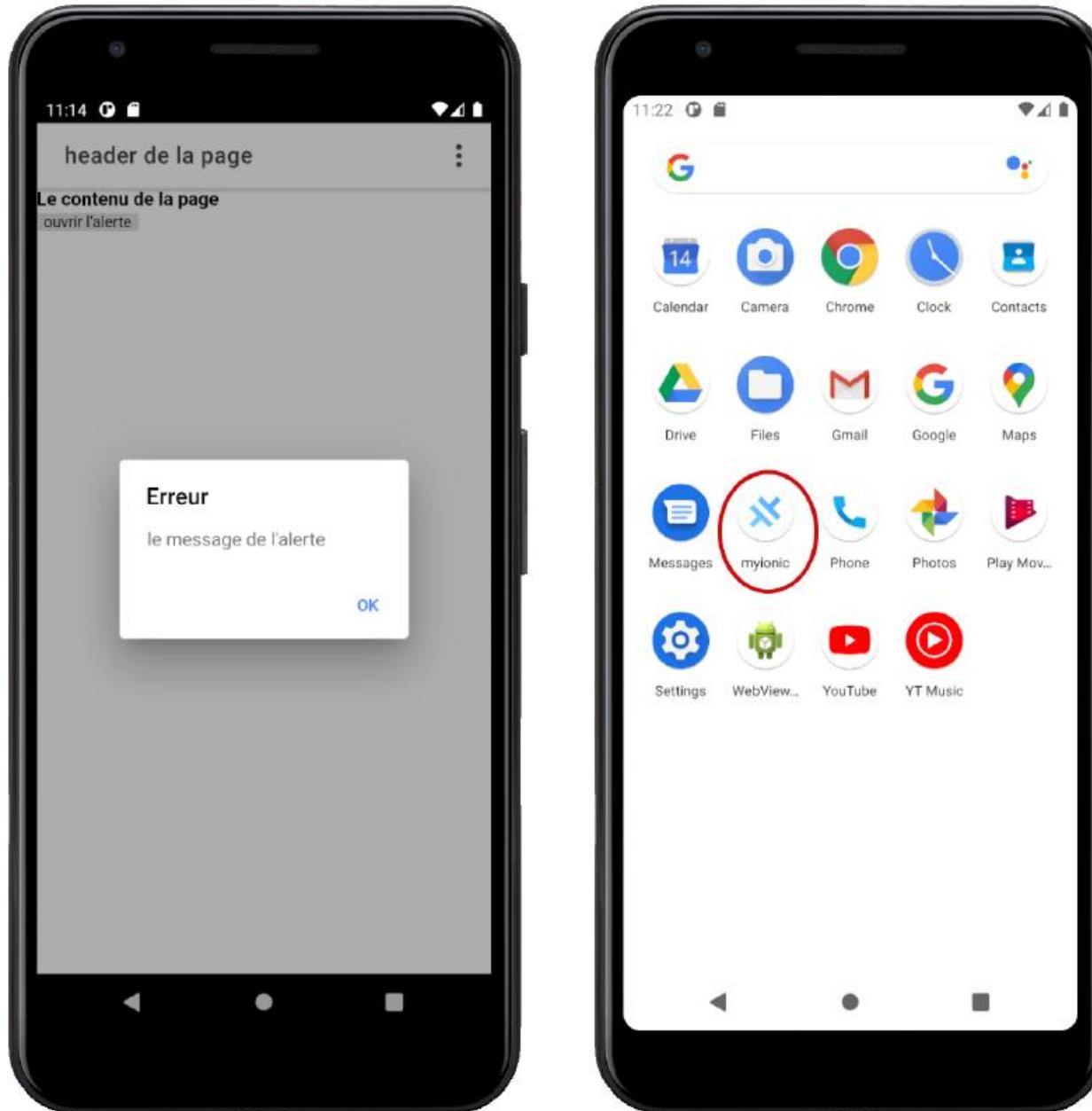
- Toolbar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Device Selection:** The dropdown menu "Pixel_3a_API_30_x86" is highlighted with a red circle.
- Project Structure:** The "Project" view shows the following structure:
 - android
 - app
 - capacitor [capacitor-android]
 - capacitor-cordova-android-plugins
 - android /home/apache/mobile/cours/myio
 - .gradle
 - .idea
 - app
 - build
 - capacitor-cordova-android-plugins
 - gradle
 - .gitignore
 - build.gradle
 - capacitor.settings.gradle
 - gradle.properties
- Build Output:** Shows the build log:

```
> Task :app:stripDebugDebugSymbols UP-TO-DATE
> Task :app:validateSigningDebug UP-TO-DATE
> Task :app:packageDebug UP-TO-DATE
> Task :app:assembleDebug UP-TO-DATE

BUILD SUCCESSFUL in 6s
65 actionable tasks: 65 up-to-date
```
- Bottom Navigation:** TODO, Git, Terminal, Database Inspector, Profiler, Run, Build, Logcat, Event Log, Layout Inspector, master.
- Message:** Failed to start monitoring emulator-5554 (a minute ago)

A tooltip message is displayed: "Sélectionner un device et cliquer sur le bouton run (triangle vert)".

L'application native



Création d'une progressive web application (1/3)

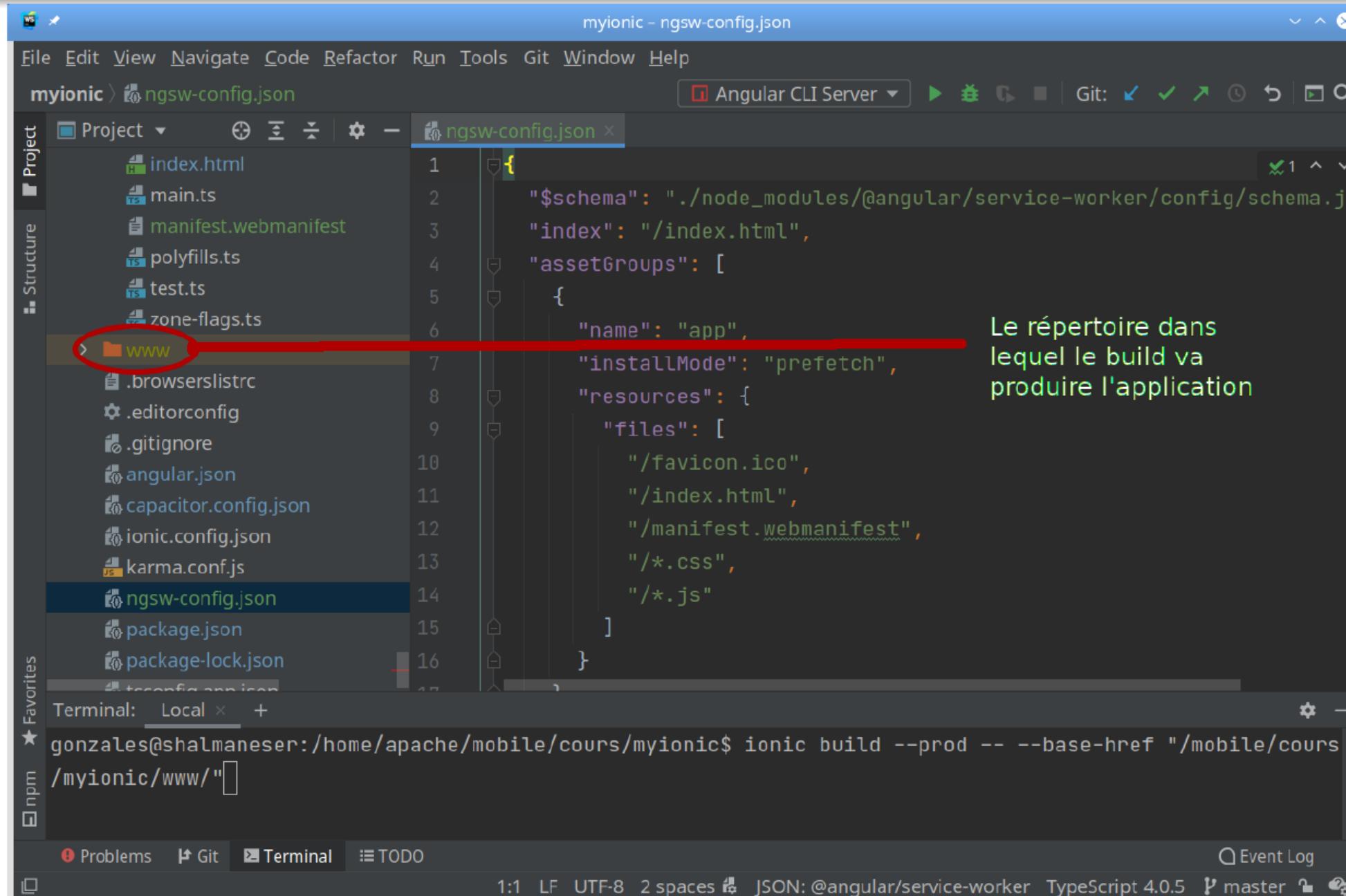
The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- Title Bar:** myionic
- File Menu:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Toolbar:** Angular CLI Server, Git status indicators.
- Project Explorer:** Shows files and folders:
 - Project: global.scss, index.html, main.ts, manifest.webmanifest (selected), polyfills.ts, test.ts, zone-flags.ts.
 - www folder: .browserslistrc, .editorconfig, .gitignore, angular.json, capacitor.config.json, ionic.config.json, karma.conf.js.
 - ngsw-config.json (selected), package.json, package-lock.json.
- Terminal:** Local
 - Message: ✓ Packages installed successfully.
 - Command: gonzales@shalmaneser:/home/apache/mobile/cours/myionic\$ ng add @angular/pwa
- Status Bar:** TypeScript 4.0.5, master, Event Log, TODO.

Annotations in green text are overlaid on the interface:

- Le manifeste du PWA (The PWA manifest) points to the manifest.webmanifest file in the Project Explorer.
- Le fichier qui indique comment gérer le cache (The file that indicates how to manage the cache) points to the ngsw-config.json file in the Project Explorer.
- Commande pour rendre votre appli en PWA (Command to make your app a PWA) points to the terminal command "ng add @angular/pwa".

Création d'une progressive web application (2/3)



The screenshot shows a code editor with the file `ngrx-config.json` open. The project structure on the left includes files like `index.html`, `main.ts`, `manifest.webmanifest`, `polyfills.ts`, `test.ts`, `zone-flags.ts`, and a folder `www`. The `www` folder is highlighted with a red circle. The code in `ngrx-config.json` defines asset groups:

```
$schema": "./node_modules/@angular/service-worker/config/schema.json"
"index": "/index.html",
"assetGroups": [
  {
    "name": "app",
    "installMode": "prefetch",
    "resources": {
      "files": [
        "/favicon.ico",
        "/index.html",
        "/manifest.webmanifest",
        "/*.css",
        "/*.js"
      ]
    }
}
```

A callout text on the right side of the editor states: "Le répertoire dans lequel le build va produire l'application".

```
gonzales@shalmaneser:/home/apache/mobile/cours/myionic$ ionic build --prod -- --base-href "/mobile/cours/myionic/www/"
```

Bottom navigation bar: Problems, Git, Terminal, TODO, Event Log.

Création d'une progressive web application (3/3)

The screenshot shows a browser window titled "Ionic App - Chromium" displaying a local development URL: "127.0.0.1/mobile/cours/myionic/www/home". The page content includes a header "header de la page" and a button labeled "ouvrir l'alerte". Below the page content, the browser's developer tools are open, specifically the "Application" tab. In the left sidebar under "Application", the "Service Workers" option is selected, showing a list of registered workers. One worker is listed for the URL "http://127.0.0.1/mobile/cours/myionic/www/", with its source file being "ngsw-worker.js". The worker was received on "2/14/2021, 4:17:13 PM" and is currently active and running, indicated by a green status dot and the message "#38 activated and is running". A "stop" link is also present. The "Clients" section shows a single client entry for the same URL, with the word "focus" next to it. At the bottom, there are "Push" and "Sync" sections, each with a text input field and a corresponding "Push" or "Sync" button.

Ionic App - Chromium

Ionic App

127.0.0.1/mobile/cours/myionic/www/home

header de la page

Le contenu de la page

ouvrir l'alerte

Elements Console Sources Network Performance Memory Application Security Lighthouse ⚙️ ⋮ X

Application

- Manifest
- Service Workers
- Clear storage

Service Workers

Offline Update on reload Bypass for network

http://127.0.0.1/mobile/cours/myionic/www/ [Update](#) [Unregister](#)

Source [ngsw-worker.js](#)

Received 2/14/2021, 4:17:13 PM

Status ● #38 activated and is running [stop](#)

Clients http://127.0.0.1/mobile/cours/myionic/www/home [focus](#)

Push Test push message from DevTools. [Push](#)

Sync test-tag-from-devtools [Sync](#)

Un bouton d'installation (1/4)

```
myionic - home.page.ts
File Edit View Navigate Code Refactor Run Tools Git Window Help
myionic > src > app > home > home.page.ts > HomePage
Angular CLI Server ▾ ▶ ⚙ ⏪ ⏴ Git: ↺ ↻ ⏴ ⏵ 🔍
Project home.page.ts x ▾ 1 6 ▾
Structure
10 export class HomePage {
11     // un événement qui sera initialisé dans le constructeur si l'appli n'a pas été installée
12     promptEvent: any = null;
13
14     constructor(private popover: PopoverController,
15                 private alertController: AlertController) {
16         // l'événement "beforeinstallprompt" est émis chaque fois que l'on charge la page et que
17         // l'application n'a pas encore été installée. S'il est émis, on met à jour promptEvent
18         window.addEventListener(
19             type: 'beforeinstallprompt', listener: event => {
20                 this.promptEvent = event;
21             });
22         }
23
24         // quand on clique sur le bouton d'installation, on exécute cette méthode, qui appelle
25         // la méthode "prompt" du promptEvent : c'est ce qui crée le popup vous proposant
26         // d'installer l'application
27         onInstall() {
28             this.promptEvent.prompt();
29         }
30
```

Un bouton d'installation (2/4)

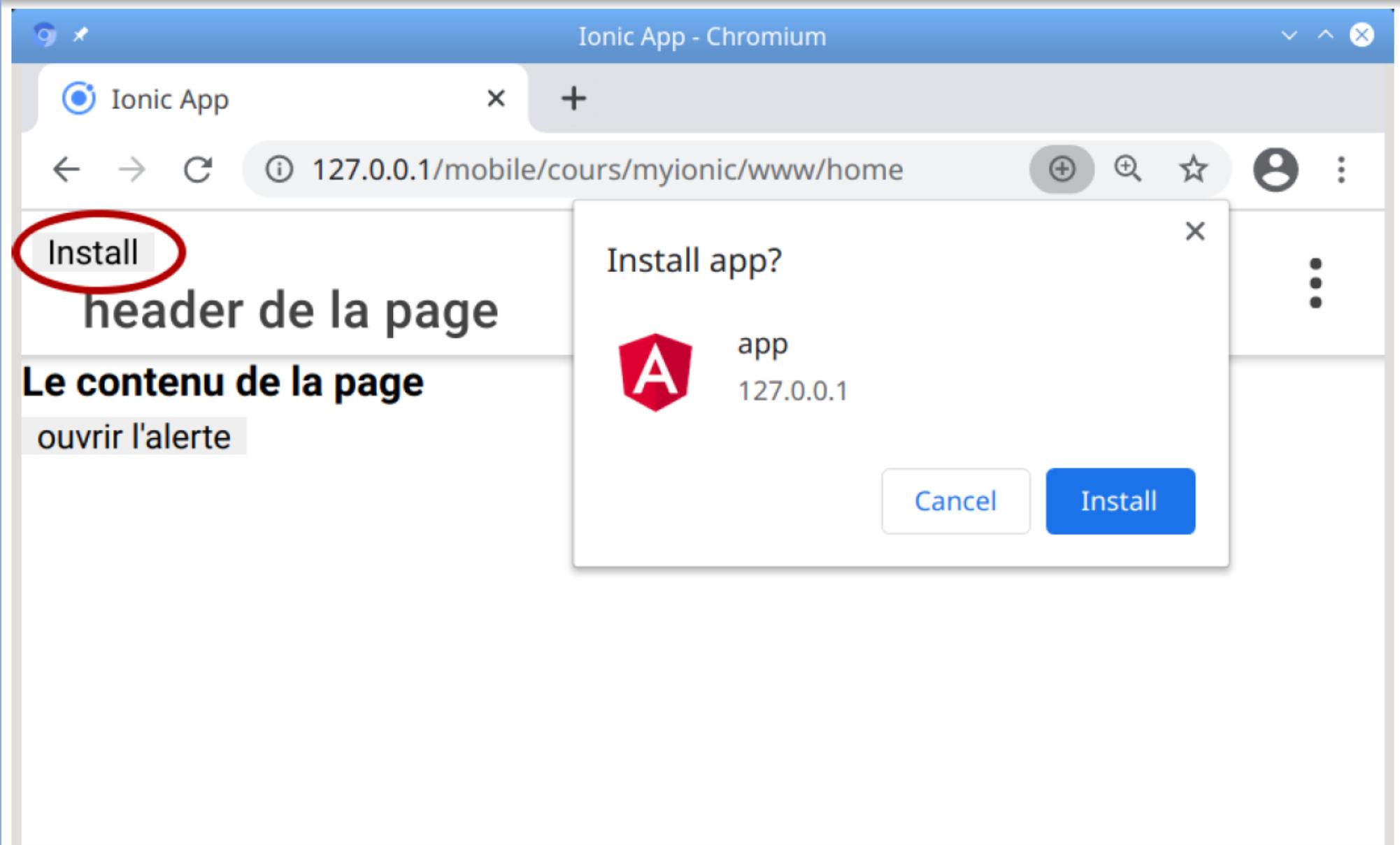
The screenshot shows a code editor interface with the following details:

- Title Bar:** myionic - home.page.html
- File Path:** myionic > src > app > home > home.page.html
- Toolbar:** Includes Angular CLI Server, Git, and other development tools.
- Project Explorer:** Shows the file structure: home.page.html (selected), Project, Structure, Favorites, and npm.
- Code Editor:** Displays the HTML code for the home page, including an ion-header, ion-toolbar, ion-title, and ion-buttons sections. A specific button is highlighted with a blue selection bar.
- Bottom Bar:** Problems, Git, Terminal, TODO, Event Log, and status indicators (5:90, LF, UTF-8, 2 spaces, TypeScript 4.0.5, master).

```
<ion-header>
<ion-toolbar>
  <!-- on n'affiche le bouton d'installation que si l'on sait que l'application
       n'a pas déjà été installée. Si l'on clique sur le bouton, on va appeler la
       méthode onInstall de notre TypeScript, qui va ouvrir le popup d'installation -->
  <button *ngIf="promptEvent" (click)="onInstall()">Install</button>

  <ion-title>header de la page</ion-title>
  <ion-buttons slot="start">
    <ion-back-button></ion-back-button>
  </ion-buttons>
  <!-- Nouveau bouton pour pouvoir ouvrir le menu. Ce bouton a la forme d'une
       icone "...", soit verticale soit horizontale, en fonction du smartphone -->
  <ion-buttons slot="end">
    <ion-button (click)="openMenu($event)">
      <ion-icon slot="icon-only"
                ios="ellipsis-horizontal"
                md="ellipsis-vertical"></ion-icon>
    </ion-button>
  </ion-buttons>
</ion-toolbar>
```

Un bouton d'installation (3/4)



Un bouton d'installation (4/4)

The screenshot shows a Chromium browser window titled "Apps - Chromium". The address bar indicates the URL is "Chromium | chrome://apps". A message on the right side says "Not signed in to Chromium (You're missing out—sign in)". Below this, there are three icons: "Web Store" (blue icon), "Postman" (orange icon with a pen), and an Ionic app icon (red circle with a white "A" and the word "app"). The Ionic app icon is circled in red. Below the icons, the text "L'application est installée" is displayed in red. At the bottom left is a small circular logo with a blue "O". At the bottom right, there is a "Web Store" link and a "Web Store" icon.

Apps - Chromium

Apps

← → C Chromium | chrome://apps

Not signed in to Chromium
(You're missing out—sign in)

Web Store Postman app

L'application est installée

Web Store