



Angular Advanced short recap – day #1



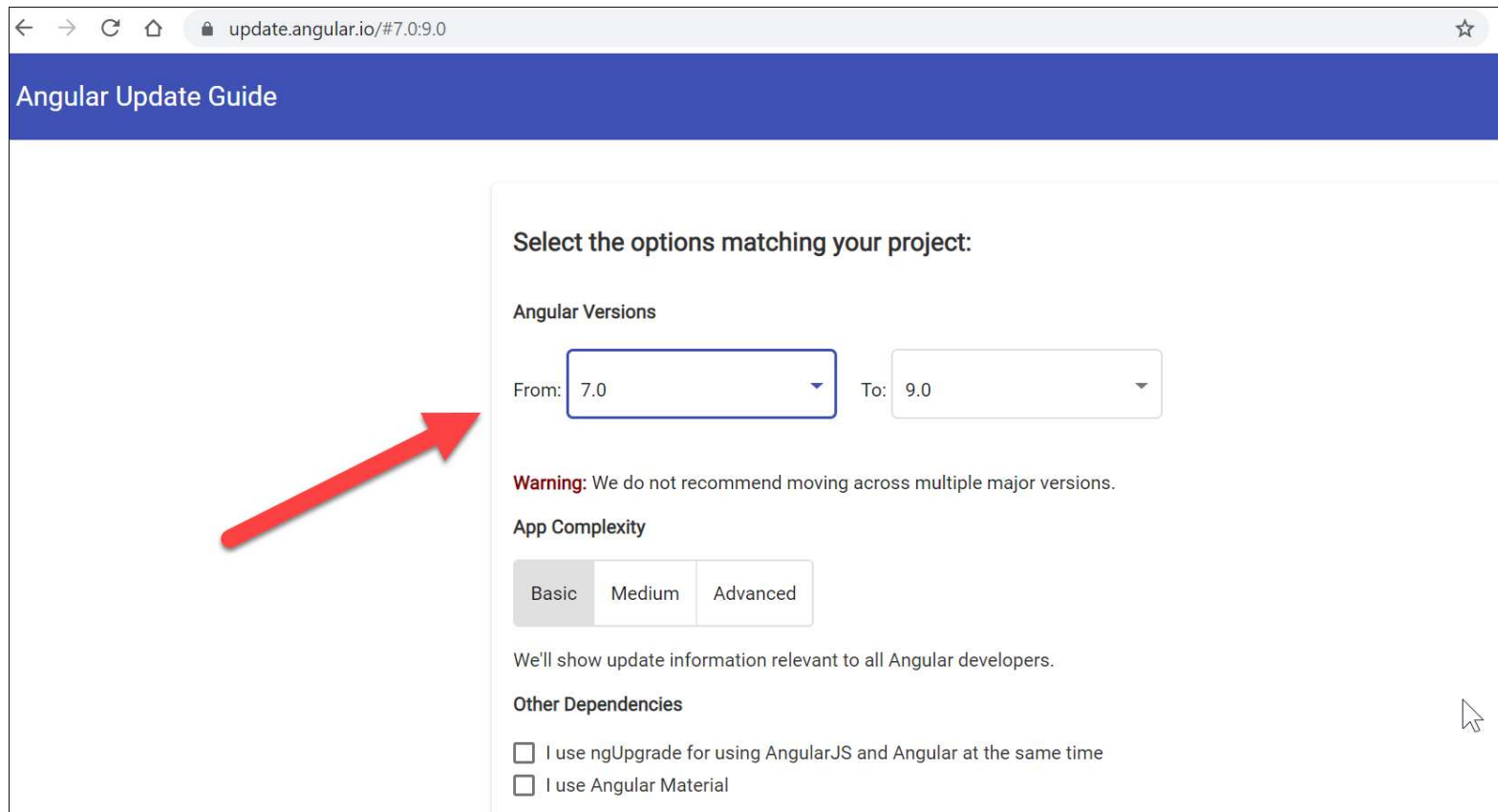
Peter Kassenaar
info@kassenaar.com

Recap day 1: **Architecture**

- Angular CLI – general **Architecture**
 - Flags, Components, Services, Modules.
- NG applications with multiple **modules**
 - Using multiple modules in your app
 - Modules and the router
- **Lazy loading** modules
 - `PreloadAllModules`
 - Custom Loading **strategies**
 - Multiple / Nested router outlets
- **Content projection & Smart/View Components**

On ng update

- DON'T try to update over **major versions** (e.g. from NG 7 → NG 9)

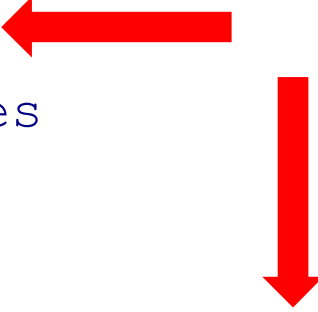


The screenshot shows the 'Angular Update Guide' website. The browser address bar displays 'update.angular.io/#7.0:9.0'. The page title is 'Angular Update Guide'. The main content area contains a form titled 'Select the options matching your project:'. Under the 'Angular Versions' section, there are two dropdown menus: 'From: 7.0' and 'To: 9.0'. A large red arrow points to the 'From: 7.0' dropdown. Below this, a warning message states: 'Warning: We do not recommend moving across multiple major versions.' Under the 'App Complexity' section, there are three buttons: 'Basic', 'Medium', and 'Advanced'. Below this, a note says: 'We'll show update information relevant to all Angular developers.' Under the 'Other Dependencies' section, there are two checkboxes: 'I use ngUpgrade for using AngularJS and Angular at the same time' and 'I use Angular Material'.

<https://update.angular.io/>

Configuring the router

```
let routerConfig: ExtraOptions = {  
  enableTracing: true  
  preloadingStrategy: PreloadAllModules  
};  
  
@NgModule({  
  imports: [RouterModule.forRoot(routes, routerConfig)],  
  exports: [RouterModule]  
})
```

Two red arrows originate from the right side of the code. One arrow points horizontally to the left, ending at the 'routerConfig' argument in the 'RouterModule.forRoot' call. The other arrow points vertically downwards, ending at the 'RouterModule' export in the 'NgModule' decorator.

Routing events are actually **Observables**.

Which means we can subscribe! And do something like:

```
this.router.events
    .subscribe(event =>{
        console.log(' router event: ', event);
    })
```

(Don't forget to inject router:

```
constructor(private router: Router) {  })
```

Or, in a more reactive way of programming:

```
this.router.events.pipe(  
    filter(event => event instanceof NavigationEnd),  
    map(...),  
    ...  
) .subscribe(event => {  
    console.log(' router event: ', event)  
});
```


Handle events as desired - logging, setting variables, updating state, ...




Angular Component Libraries


Options and popular choices


Extra – UI Libraries **Popular Choices**

 Getting Started Components

 Star

6,131

 Tweet #ngbootstrap



Bootstrap widgets

The angular way

Angular widgets built from the ground up using only Bootstrap 4 CSS with APIs designed for the Angular ecosystem.

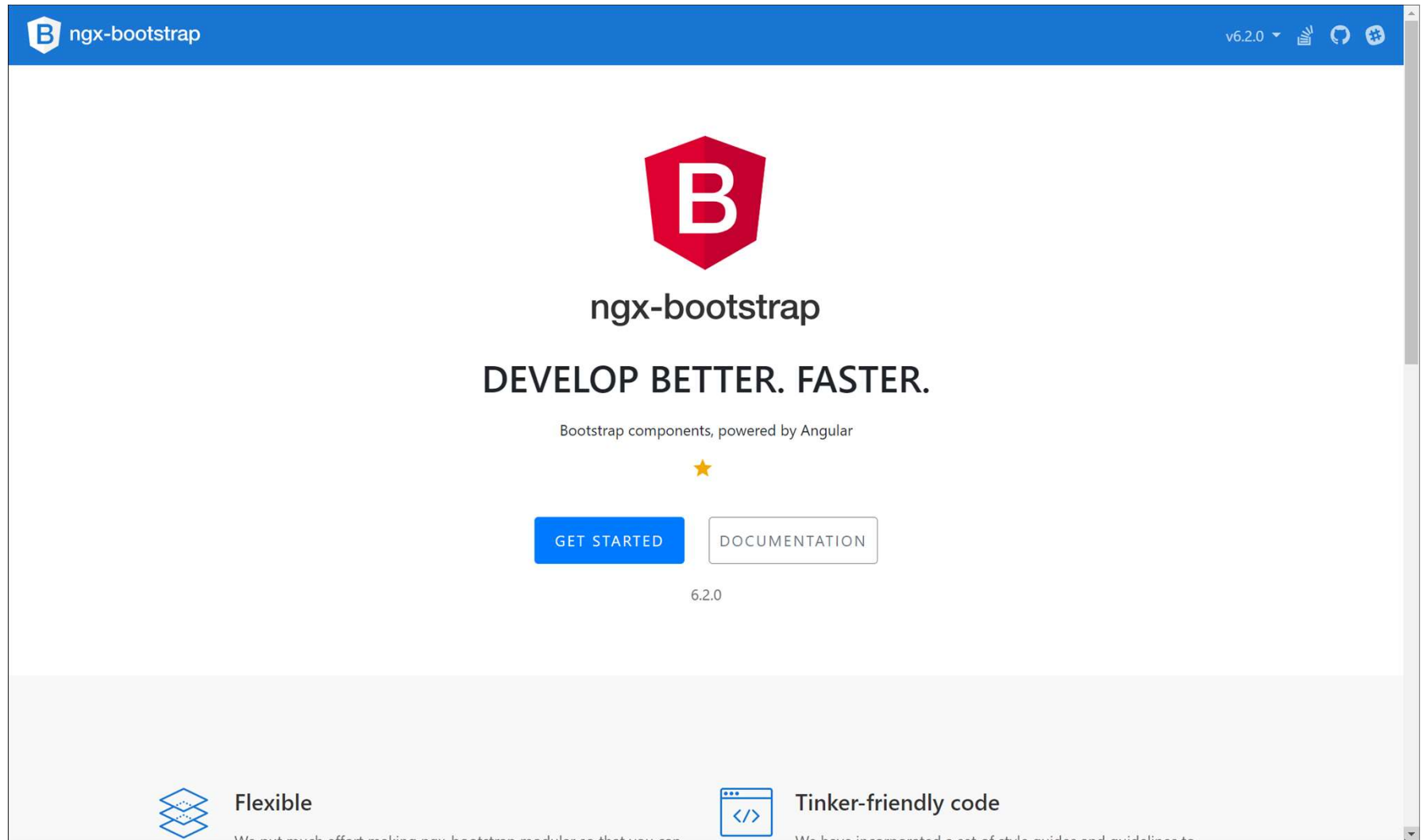
No dependencies on 3rd party JavaScript.

[Demo](#)[Installation](#)

Currently at v3.3.0

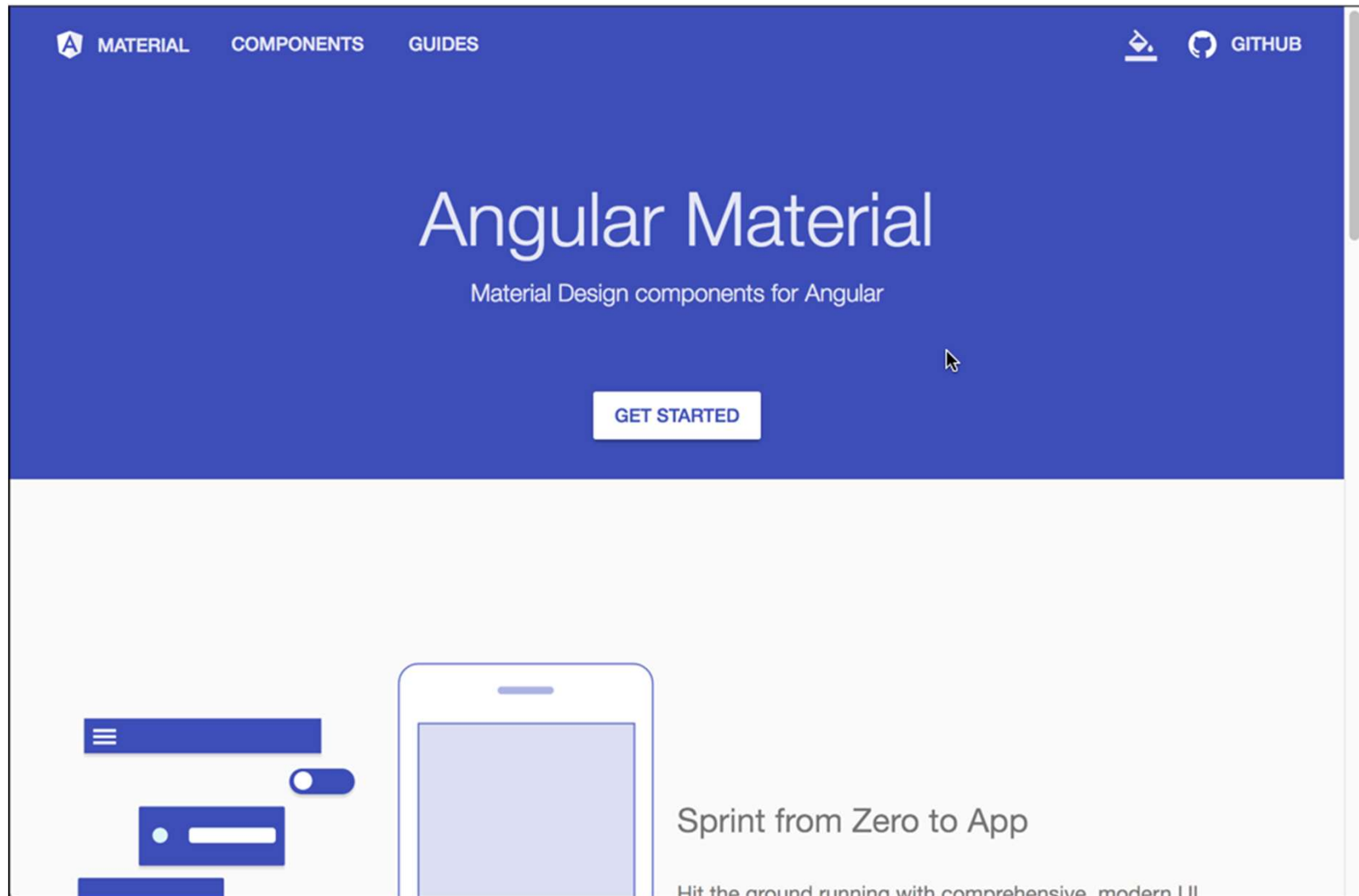
<https://ng-bootstrap.github.io/#/home>

Ngx-bootstrap – same goals, same approach



<https://valor-software.com/ngx-bootstrap/#/>

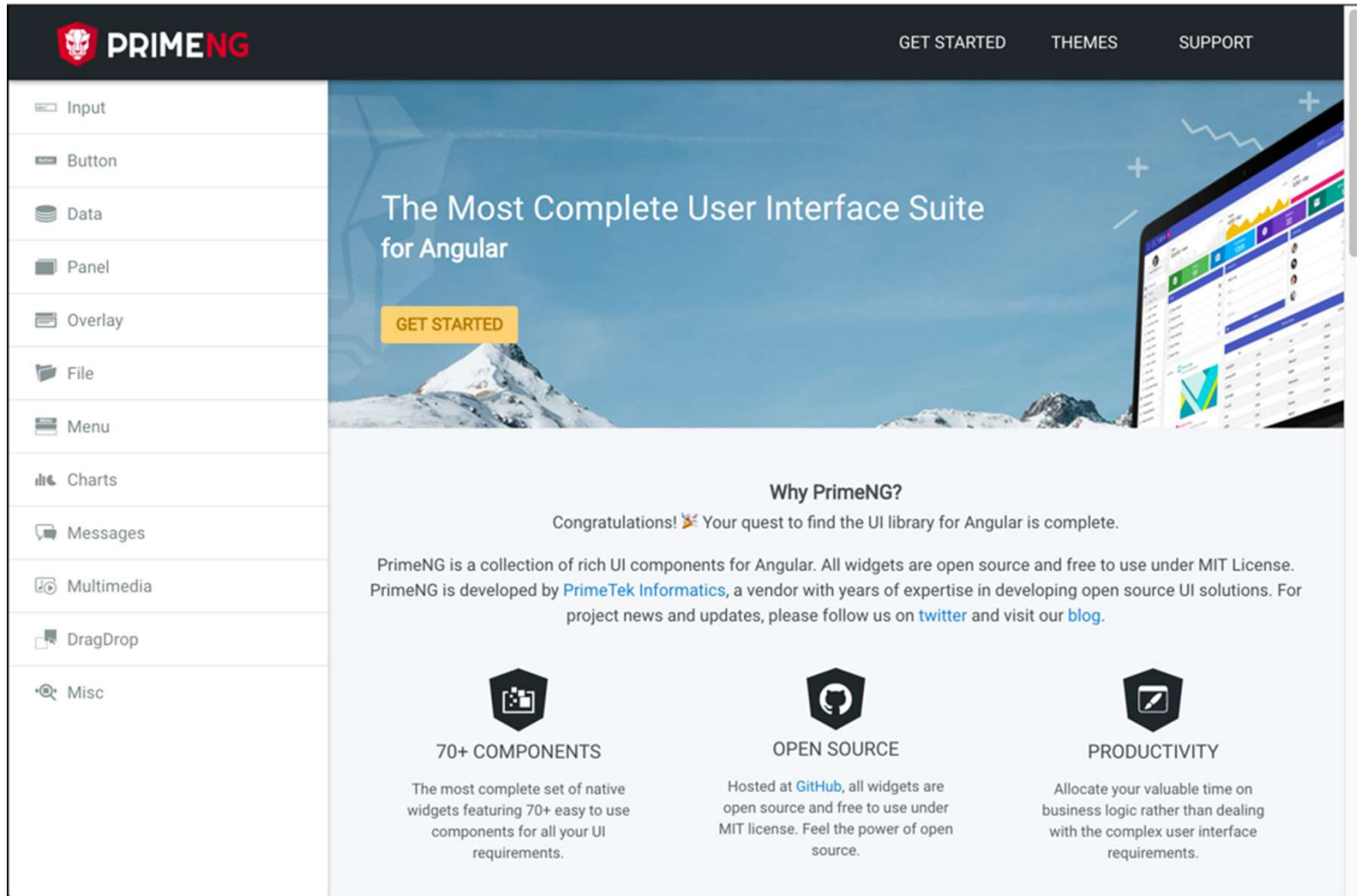
Angular Material – By Google (officially endorsed by Team Angular)



<https://material.angular.io/>

PrimeNG – Angular port of the PrimeFaces library

VERY Popular in the Java World. A LOT of components



The screenshot shows the PrimeNG website homepage. The header features the PrimeNG logo on the left and navigation links for 'GET STARTED', 'THEMES', and 'SUPPORT' on the right. A left sidebar lists various UI components: Input, Button, Data, Panel, Overlay, File, Menu, Charts, Messages, Multimedia, DragDrop, and Misc. The main content area has a hero section with the title 'The Most Complete User Interface Suite for Angular' and a 'GET STARTED' button. Below this is a 'Why PrimeNG?' section with a congratulatory message and a paragraph about the library's open-source nature and MIT license. At the bottom, three key features are highlighted: '70+ COMPONENTS', 'OPEN SOURCE', and 'PRODUCTIVITY', each with a brief description.

PRIMENG

GET STARTED THEMES SUPPORT

Input
Button
Data
Panel
Overlay
File
Menu
Charts
Messages
Multimedia
DragDrop
Misc


The Most Complete User Interface Suite for Angular

[GET STARTED](#)

Why PrimeNG?


Congratulations! 🎉 Your quest to find the UI library for Angular is complete.

PrimeNG is a collection of rich UI components for Angular. All widgets are open source and free to use under MIT License. PrimeNG is developed by [PrimeTek Informatics](#), a vendor with years of expertise in developing open source UI solutions. For project news and updates, please follow us on [twitter](#) and visit our [blog](#).




70+ COMPONENTS

The most complete set of native widgets featuring 70+ easy to use components for all your UI requirements.



OPEN SOURCE

Hosted at [GitHub](#), all widgets are open source and free to use under MIT license. Feel the power of open source.



PRODUCTIVITY

Allocate your valuable time on business logic rather than dealing with the complex user interface requirements.

<https://www.primefaces.org/primeng/#/>

Ng-zorro = Ant Design, Developed by Alibaba

Originally created for React, now also loved by thousands of ng-developers

The screenshot displays the Ng-Zorro website interface. At the top, there's a navigation bar with the Ng-Zorro logo, a search bar, and links for Docs, Components, and Experimental. The main content area is titled "Ant Design of Angular" and describes it as an enterprise-class Angular UI component library. It features a list of features and a section for environment support, including browser and Electron compatibility. A sidebar on the left contains a table of contents, and a right sidebar lists additional resources like Features, Environment Support, and Version.

Ant Design of Angular

An enterprise-class Angular UI component library based on Ant Design, all components are open source and free to use under MIT license.

Features

- An enterprise-class UI design language for Angular applications.
- 60+ high-quality Angular components out of the box.
- Written in TypeScript with complete defined types.
- Support OnPush mode, high performance.
- Powerful theme customization in every detail.
- Internationalization support for dozens of languages.

Environment Support

- Modern browsers and Internet Explorer 11+ (with [polyfills](#))
- Server-side Rendering
- [Electron](#)

IE / Edge	Firefox	Chrome	Safari	Opera	Electron
-----------	---------	--------	--------	-------	----------

<https://ng.ant.design/docs/introduce/en>



"Why would I want to provide my Services in a Core Module?"

In other words, "Can't I just use plain files instead of configuring and importing yet another module?"

Answer – “Yes You Can”

- It is absolutely valid to use ‘just service files’.
- You can import them whenever/wherever you need

HOWEVER....

Because – singletons

- You may – or may not! – want to make sure a service is provided **as a Singleton**.
- You CAN NOT prevent that, by using 'just service files'
- Inside a `CoreModule` you can check if the module, and therefore the service – is already loaded.
- By using a `.forRoot()` method, and/or checking in the constructor
- For instance (from the docs): <https://angular.io/guide/singleton-services#prevent-reimport-of-the-greetingmodule>

Prevent Re-importing


Prevent reimport of the GreetingModule

Only the root AppModule should import the GreetingModule. If a lazy-loaded module imports it too, the app can generate [multiple instances](#) of a service.

To guard against a lazy loaded module re-importing GreetingModule, add the following GreetingModule constructor.

src/app/greeting/greeting.module.ts

```
constructor(@Optional() @SkipSelf() parentModule?: GreetingModule) {  
  if (parentModule) {  
    throw new Error(  
      'GreetingModule is already loaded. Import it in the AppModule only');  
  }  
}
```



For instance

```
export class SharedLibModule {  
  // Make sure this is a Singleton, i.e.  
  // Prevent this module from being loaded twice  
  constructor(@Optional() @SkipSelf() parentModule?: SharedLibModule) {  
    if (parentModule) {  
      throw new Error('Shared Library is already loaded. Import it in AppModule only');  
    }  
  }  
  ...  
}
```

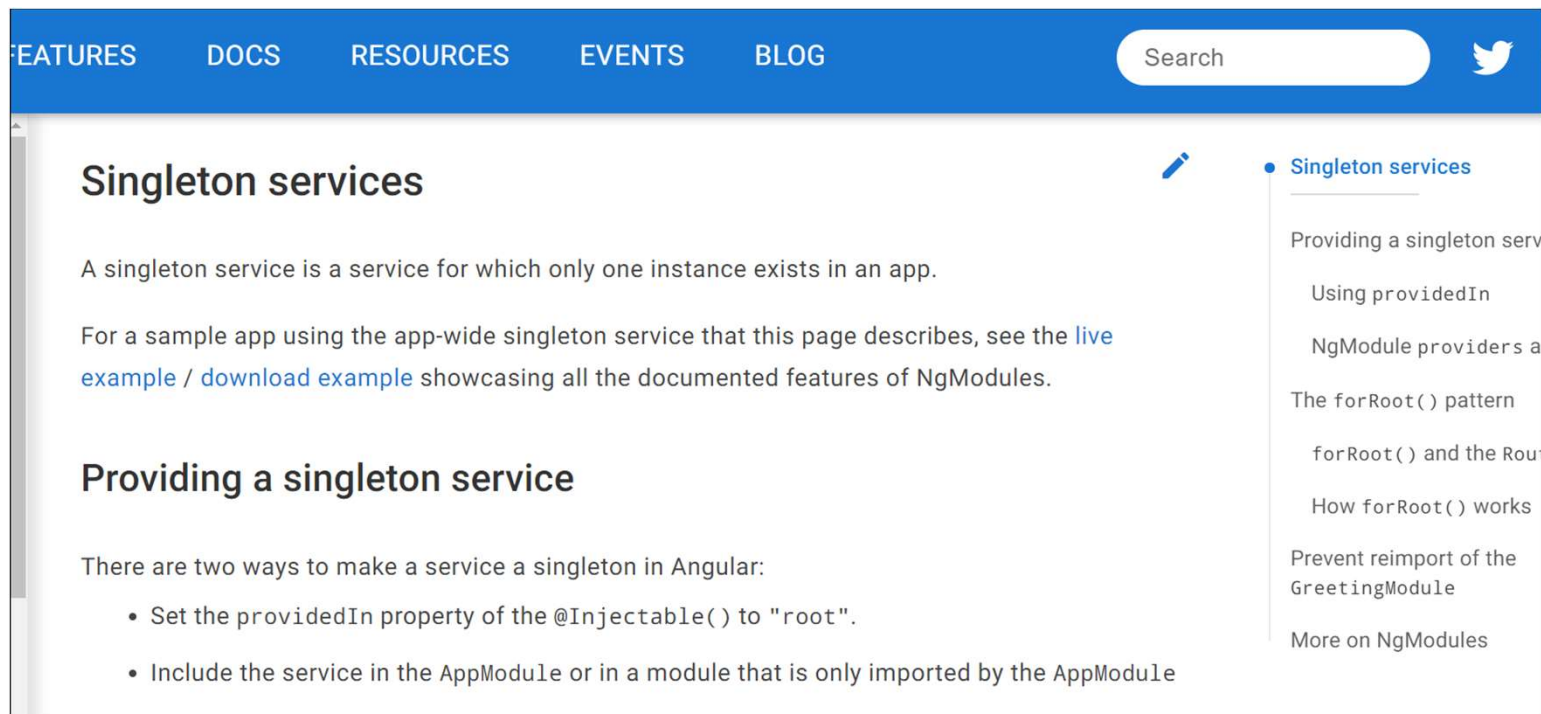


Using `.forRoot()` configuration

Configure your Singleton Services with a custom object

Configure your module

- We'll cover that tomorrow, when creating and using Angular Libraries
- If you can't wait, check <https://angular.io/guide/singleton-services> :-)



The screenshot shows the Angular.io website's 'Singleton services' guide. The page has a blue header with navigation links: FEATURES, DOCS, RESOURCES, EVENTS, and BLOG. A search bar and a Twitter icon are on the right. The main content area is titled 'Singleton services' and includes a sub-section 'Providing a singleton service'. The text explains that a singleton service has only one instance in an app and provides a live example and download link. A list of two ways to create a singleton service is shown: setting the providedIn property to 'root' and including the service in the AppModule. A right-hand sidebar contains a table of contents for the 'Singleton services' section.

Singleton services

A singleton service is a service for which only one instance exists in an app.

For a sample app using the app-wide singleton service that this page describes, see the [live example](#) / [download example](#) showcasing all the documented features of NgModules.

Providing a singleton service

There are two ways to make a service a singleton in Angular:

- Set the `providedIn` property of the `@Injectable()` to `"root"`.
- Include the service in the `AppModule` or in a module that is only imported by the `AppModule`

Singleton services

- Providing a singleton service
- Using `providedIn`
- NgModule providers and `providedIn`
- The `forRoot()` pattern
- `forRoot()` and the Router
- How `forRoot()` works
- Prevent reimport of the `GreetingModule`
- More on NgModules

*Other
questions?*

Agenda - 3 days - Thematic



- Day 1: Architecture
 - Composing Applications with multiple modules
 - ...
- Day 2: **Store & Observables**
 - Creating observables from scratch
 - State management and @ngrx/store w/ observables
 - Concepts, State, Action, Reducer, Dispatcher, Effect, Http
 - More on observables / Operators...

Next up

- Day 3: **Miscellaneous**
- More on observables
- Angular Performance tips/tricks/tools
- Unit Testing w/ Karma & Jasmine, Cypress as alternative
- Angular Enterprise applications – monorepo....or not
- Creating and Publishing Angular Libraries [to NPM]
- More on Angular Schematics
- Q & A

