Angular Standards

Contents

Introduction

Orbis *Technologies* **develops** and maintains ***enterprise*** CCMS software, solutions, and services to organizations worldwide. Orbis provide solutions to clients across various industries that including manufacturing, healthcare, publishing, healthcare insurance, education industries and the federal government. Table 1

Thead	Thead	Thead	Thead
Well written software offers many advantages.	Well www.google.com written software offers many advantages.	Well written software offers many advantages.	Well written software offers many advantages.

Well written software offers many advantages.Â It will contain fewer bugs and will run more efficiently than poorly written programs.Â Since software has a life cycle and much of which revolves around maintenance, it will be easier for the original developer(s) and future keepers of the code to maintain and modify the software as needed. This will lead to increased productivity of the developer(s). The overall cost of the software is greatly reduced when the code is developed and maintained according to software standards.Â Table 1

The goal of these guidelines is to create uniform coding habits among software personnel in the engineering department so that reading, checking, and maintaining code written by different persons becomes easier. The intent of these standards is to define a natural style and consistency yet leave to the authors of the engineering department source code, the freedom to practice their craft without unnecessary burden.

Use Angular CLI

Angular CLI is one of the most powerful accessibility tools available when developing apps with Angular. Angular CLI makes it easy to create an application and follows best practices. Angular CLI is a command-line interface tool that is used to initialize, develop, scaffold, maintain, and test and debug Angular applications.

Instead of creating files and folders manually, use Angular CLI to generate new components, directives, modules, services, and pipes.

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

 # Check Angular CLI version

```
ng version
```

Maintain proper folder structure

Folder structure is an important factor to consider before initiating a project. The following folder structure will easily adapt to new changes during development.

-- app

|-- modules

|-- home

|-- [+] components

|-- [+] pages

|-- home-routing.module.ts

|-- home.module.ts

|-- core

|-- [+] authentication

|-- [+] footer

|-- [+] guards

|-- [+] http

|-- [+] interceptors

|-- [+] mocks

|-- [+] services

|-- [+] header

|-- core.module.ts

|-- ensureModuleLoadedOnceGuard.ts

|-- logger.service.ts

|

|-- shared

|-- [+] components

|-- [+] directives

|-- [+] pipes

|-- [+] models

|

|-- [+] configs

|-- assets

|-- scss

|-- [+] partials

|-- _base.scss

|-- styles.scss

Follow consistent Angular coding styles

Here are some rules to follow to ensure that your project follows the proper coding standard.

- **Limit files to 400 Lines of code.**
- **Define small functions and limit them to no more than 75 lines.**
- **Have consistent names for all symbols. The recommended pattern is feature.type.ts.**
- **If the values of the variables are intact, then declare it with â##constâ##.**
- **Use dashes to separate words in the descriptive name and use dots to separate the descriptive name from the type.**

- **Names of properties and methods should always be in lower camel case.**
- **Always leave one empty line between imports and modules, such as third party and application imports and third-party modules and custom modules.**

Use ES6 features

ECMAScript is constantly updated with new features and functionalities. The current version is ES6, which has lots of new features and functionalities that can be utilized in Angular.

Here are a few ES6 Features:

- **Arrow Functions**
- **String interpolation**
- **Object Literals**
- **Let and Const**
- **Destructuring**
- **Default**

Use trackBy along with ngFor

When using ngFor to loop over an array in templates, use it with a trackBy function, which will return a unique identifier for each DOM item.

When an array changes, Angular re-renders the whole DOM tree. When you use trackBy, Angular knows which element has changed and will only make DOM changes only for that element.

Use ngFor

```
<tr *ngFor="let item of listComponent;">{{item.id}}</li>
```

Now each time the changes occur, the whole DOM tree will be re-rendered.

Using trackBy function

```
<tr *ngFor="let item of listComponent ">
<td>{{ item.fullPath }}</td>
<td>{{ item.id && item.id.split('-').pop() }}</td>
<td>{{ item.userId }}</td>
</tr>
```

Load items

```
export class listComponent {
lockedItems =[];
getItems() {
this.lockedItems = items;
}
trackByFn(index, item) {
return index; // or item.id
}
}
```

Now it returns as a unique identifier for each item so only updated items will be re-rendered.

Break down into small reusable components

This is an extension of the single responsibility principle. Large components are very difficult to debug, manage, and test. If a component becomes large, break it down into more reusable smaller components to reduce duplication of the code, so that we can easily manage, maintain, and debug with less effort.

Angular best practices - parent component

Use Lazy Loading

Try to lazy load the modules in an Angular application whenever possible. Lazy loading loads something only when it is used. This reduces the size of the application load initial time and improves the application boot time by not loading unused modules.

```
import { WithoutLazyLoadedComponent } from './without-lazy-loaded.component';
{
path: 'without-lazy-loaded',
component: WithoutLazyLoadedComponent
}
```

```
{
path: 'lazy-load',
loadChildren: 'lazy-load.module#LazyLoadModule'
}
```

```
import { LazyLoadComponent } from './lazy-load.component';
@NgModule({
imports: [
RouterModule.forChild([
{
path: '',
component: LazyLoadComponent
}
])
],
declarations: [
LazyLoadComponent
]
})
export class LazyModule {
}
```

Use Index.ts

```
For example, we have webapp/src/app/store/actions/index.ts as
export * from './favorites.actions';
export * from './briefcase.actions';
export * from './clipboard.actions';
export * from './content.actions';
export * from './most-recents.actions';
export * from './node.actions';
export * from './repository.actions';
export * from './trashcan.actions';
import { Hero, HeroService } from '../heroes'; // index is implied
```

Avoid logic in templates

```
// template
Status: Unavailable// component
ngOnInit (): void {
this.aliases=resp.value;
}
```

```
// template
Status: Unavailable// component
ngOnInit (): void {
this.aliases=resp.value;
this.isUnavailable = this.aliases === 'inActive' || 'hold';
}
```

Cache API calls test commit jksingh

Use async pipe in templates

Without Using async pipe

```
//template
{{ text }}
```

Using async pipe

```
// template
{{ text | async }}
// component
this.text = observable.pipe(
map(value => value.item)
);
```

Declare safe strings

Normal String declaration

```
private taskName: string;
// We can assign any string to taskName.
this.taskName = 'Quick task;
```

Safe string declaration

```
private taskName: 'Quick task' | 'Simple editorial';
this. taskName = "Quick task ';
this.taskName = 'Simple editorial';
this. taskName = 'Workflow'; // This will give the below error
Type ""Workflow"" is not assignable to type "'Quick task" | "Simple editorial"'
```

Avoid any type when declaring constants and variables test 23 feb 2024

```
export interface HotFolder {
id: string;
folderName: string;
enabled: string;
processDefinitionName: string;
params: { [key: string]: string };
}
```

```
export class HotFoldersAddComponent implements OnInit
hotFolder: HotFolder;
constructor() {
}
ngOnInit() {
this.hotFolder = {
id: 12345,
folderName: ' /home/rsuite/test ',
enabled: true,
path: â##testâ##
}
}
}
```

State management

- **It enables sharing data between different components.**
- **It provides centralized control for state transition.**
- **The code is cleaner and more readable.**
- **The code is easier to debug when something goes wrong.**
- **Dev tools are available for tracing and debugging state management libraries.**

Use CDK Virtual Scroll