Angular 18 released

New Angular changes that saves time and efficiency

Content :

Continue Content :

9- Internationalization.

10- A new builder package and deprecation.

11- No more donwleveling async/await (Experimental).

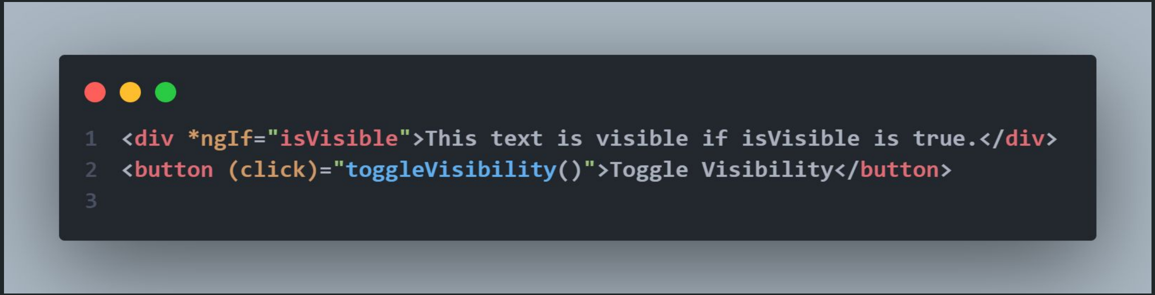
12- An new alias: ng dev => "ng dev" **command**

13- Typescript 5.4

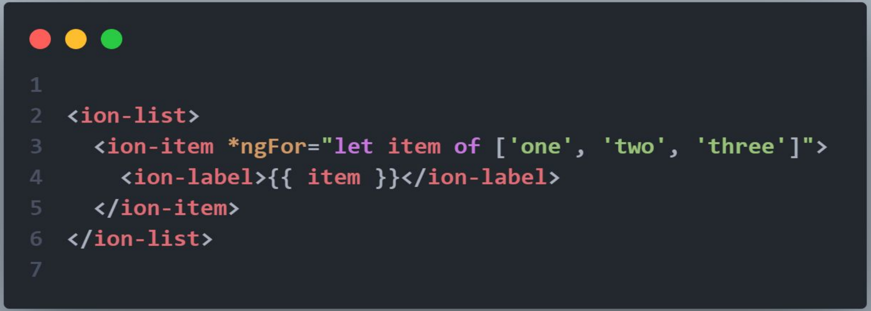
14- What can we expect in the future?

Old Control flow syntax

The following structural directives became optional:

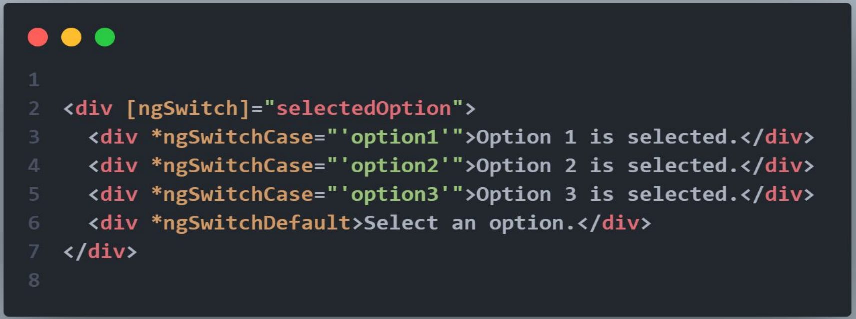


```html
<div *ngIf="isVisible">This text is visible if isVisible is true.</div>
<button (click)="toggleVisibility()">Toggle Visibility</button>
```



```html

<ion-list>
  <ion-item *ngFor="let item of ['one', 'two', 'three']">
    <ion-label>{{ item }}</ion-label>
  </ion-item>
</ion-list>
```

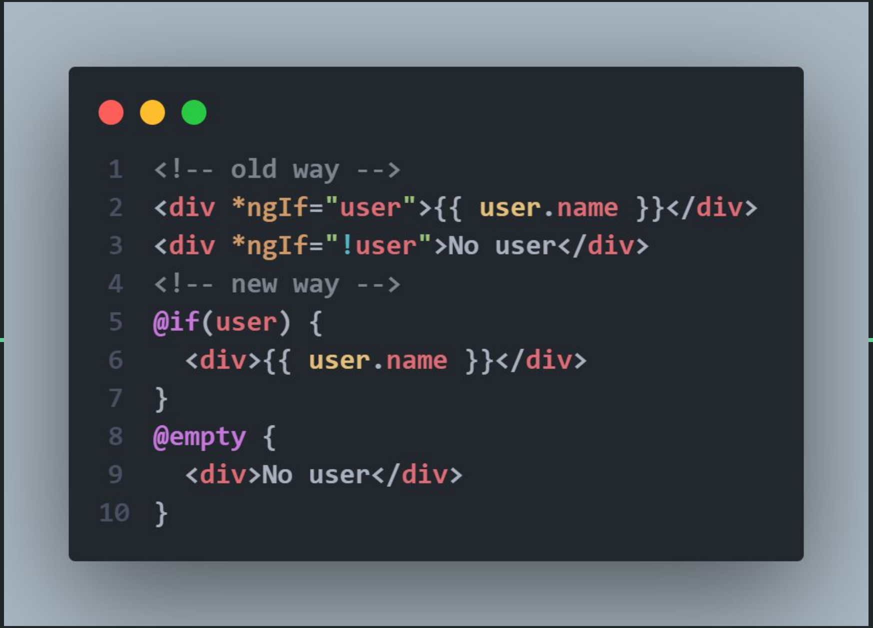


```html

<div [ngSwitch]="selectedOption">
  <div *ngSwitchCase="'option1'">Option 1 is selected.</div>
  <div *ngSwitchCase="'option2'">Option 2 is selected.</div>
  <div *ngSwitchCase="'option3'">Option 3 is selected.</div>
  <div *ngSwitchDefault>Select an option.</div>
</div>
```

New Ergonomic Control flow syntax

To migrate your application to this new control flow, a schematics is available.
$ ng g @angular/core:control-flow

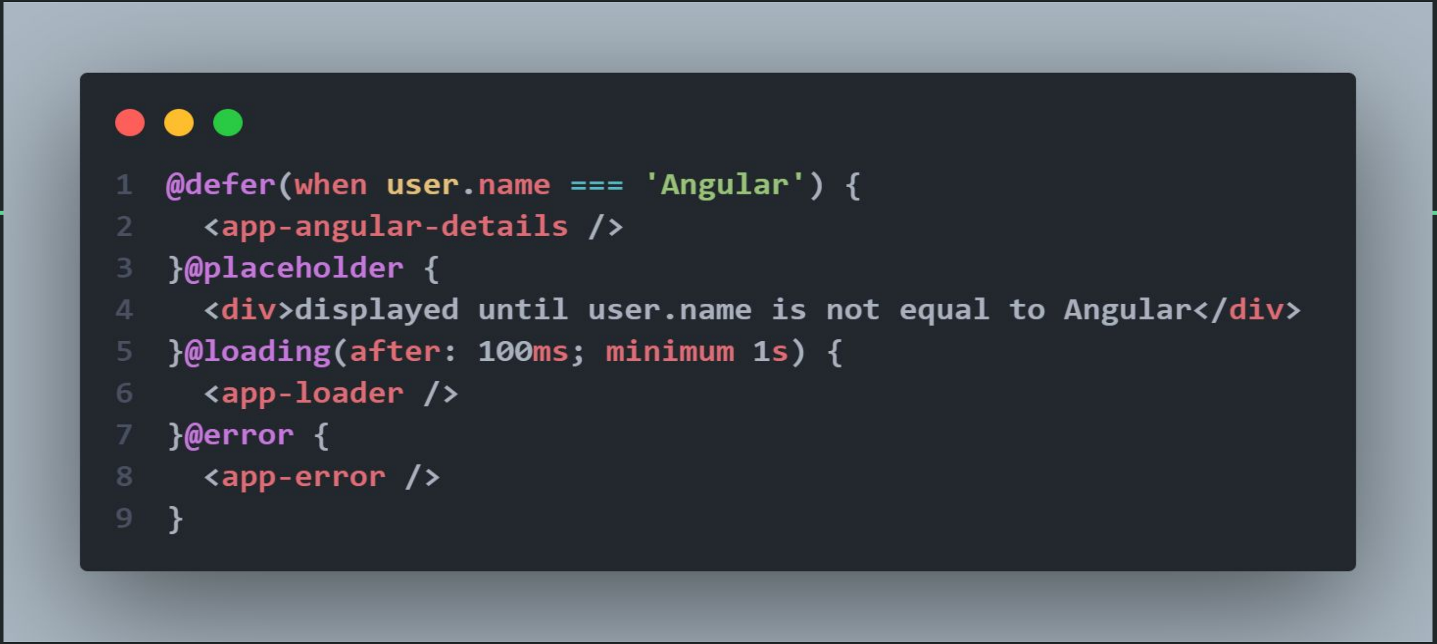


```
<!-- old way -->
<div *ngIf="user">{{ user.name }}</div>
<div *ngIf="!user">No user</div>
<!-- new way -->
@if(user) {
  <div>{{ user.name }}</div>
}
@empty {
  <div>No user</div>
}
```

Defer Control flow syntax

Template Level Lazy Loading

- ‘Defer’ was announced in previous version but now it’s stable, we will see how to take advantage of it in real world applications.



```
@defer(when user.name === 'Angular') {
  <app-angular-details />
}@placeholder {
  <div>displayed until user.name is not equal to Angular</div>
}@loading(after: 100ms; minimum 1s) {
  <app-loader />
}@error {
  <app-error />
}
```

Defer Control flow syntax

Template Level Lazy Loading

- The ‘@defer’ block will be lazy loaded when the case is matched or returned true.
- The **‘@placeholder’** block will be displayed as long as the ‘**@defer**’ block condition is not met

- The ‘**@loading**’ block will be displayed when the browser downloads the content of the ‘**@defer**’ block; in our case, the block loading will be displayed if the download takes more than 100ms, and will be displayed for a minimum duration of 1 second.

- The ‘**@error**’ block will be displayed if an error occurs while downloading the ‘**@defer**’ block.

Defer Summary

- Deferrable views are now stable. (Lazy)
- They enable developers to effortlessly improve their apps’ Core Web Vitals.
- You can use them in your applications and libraries.
- Built-in control flow is now stable and improved performance. We’ve seen significant adoption of this new syntax and, after addressing community feedback, we’re happy to announce this API as stable! (https://blog.angular.dev/).

Zone js

- Angular team is seeking making the zone.js optional.

- zone.js is declared in the polyfills.ts in Angular apps, It is used primarily for change detection in Angular by intercepting asynchronous operations. It provides a way to know when to update the UI by wrapping asynchronous operations and notifying Angular when these operations are complete.

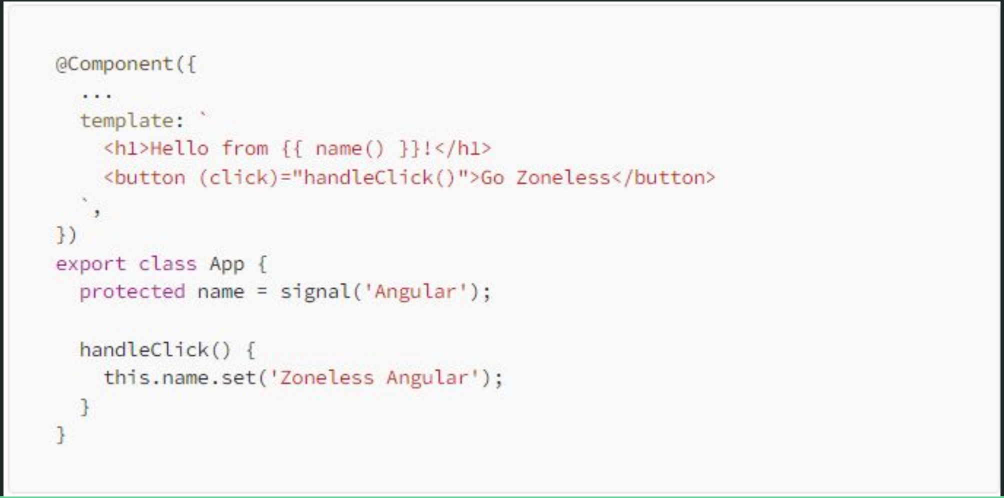
- Angular Team think zoneless Angular worth building, they mentioned that experience has revealed limitations such as:
  - Performance impact scale.
  - Maintainability challenges.
  - Cost of loading and initializing grows as APIs are introduced.

Zone js continued

- Changing a core part of the framework is a long-term project, they are investing in Angular the next 10 years.
- P.S What is expected is When you communicate state changes to Angular for example, by setting a signal used in template, the framework will schedule change detection automatically without the need for Zone.js.
- Moving forward, zoneless opens many doors for developers:
  - Improving composability for micro-frontends and interoperability with other frameworks.
  - Faster initial render and runtime.
  - Smaller bundle size and faster page loads.
  - More readable stack traces.
  - Simpler debugging.

Zone js continued

Example



```
@Component({
  ...
  template: `
    <h1>Hello from {{ name() }}!</h1>
    <button (click)="handleClick()">Go Zoneless</button>
  `,
})
export class App {
  protected name = signal('Angular');

  handleClick() {
    this.name.set('Zoneless Angular');
  }
}
```

- The best way to use zoneless in your components is with signals.
- With zone.js, Angular ran change detection any time application state might have changed. Without zones, Angular limits this checking to fewer triggers, such as signal updates.

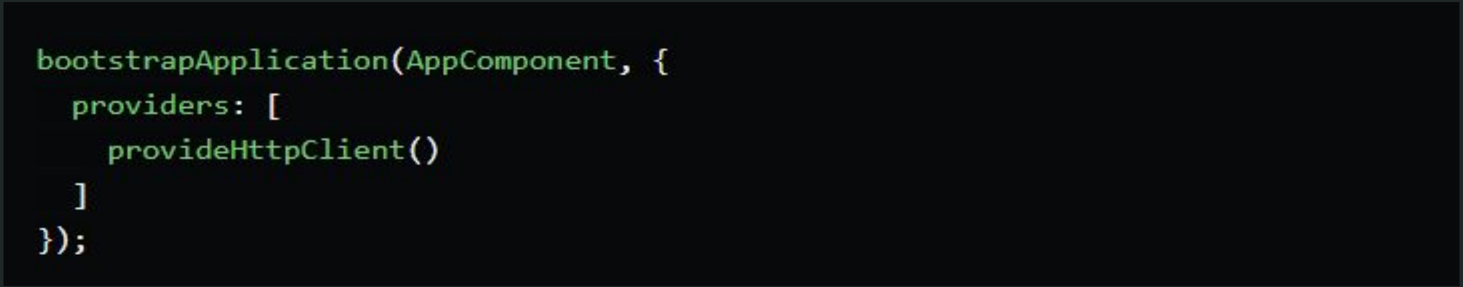
Zone js continued

Coalescing by default:

- In v18 event Coalescing is enabled by default, it’s configured in the NgZone provider in bootstrapApplication.
- Angular CDK and Angular Material have zoneless support enabled by default in v18.

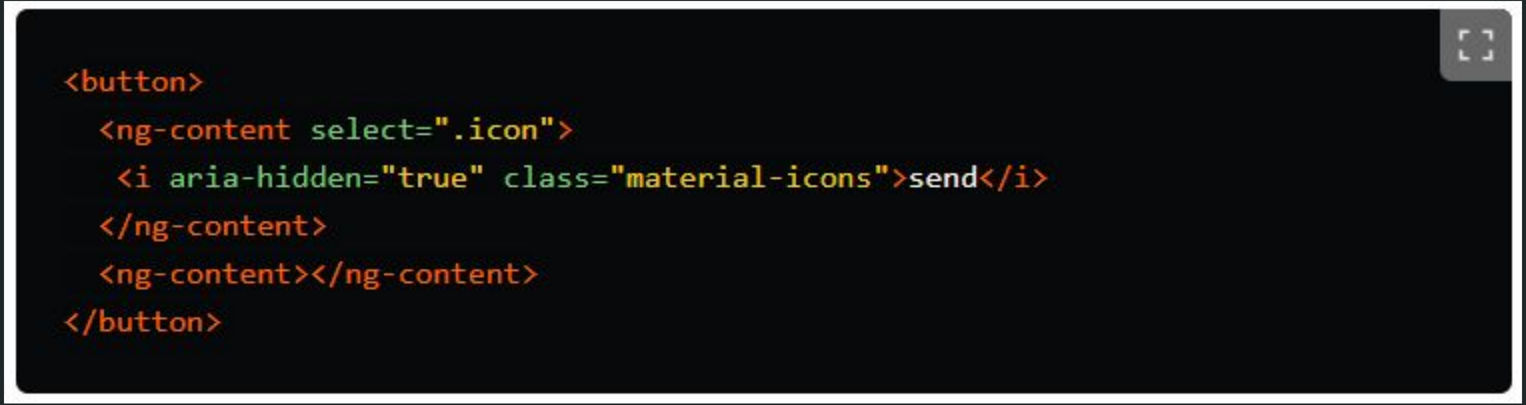
HttpClient Module Deprecation

- Since version 14 of Angular and the arrival of standalone components, modules have become optional in Angular, and now it's time to see the first module deprecated:
  => HttpClient Module
- This module was in charge of registering the HttpClient singleton for your entire application, as well as registering interceptors.
- This module can easily be replaced by the `providerHttpClient()` function, with options to support XSRF (XSRF-token) and JSONP (Only get Requests).
- This function has a twin sister for testing: `provideHttpClientTesting()`.
- migrate your application : `ng update @angular/core @angular /cli` command, a request will be made to migrate the HttpClient Module if used in the application



```
bootstrapApplication(AppComponent, {
  providers: [
    provideHttpClient()
  ]
});
```

ng-content fallback



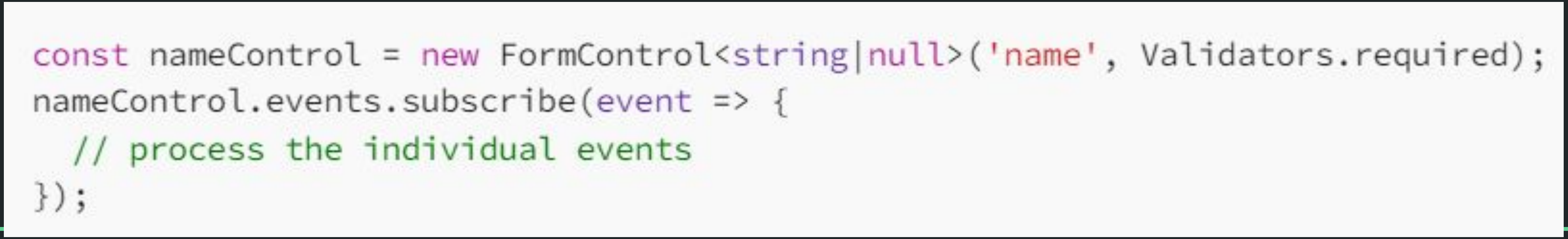
```
<button>
  <ng-content select=".icon">
   <i aria-hidden="true" class="material-icons">send</i>
  </ng-content>
  <ng-content></ng-content>
</button>
```

- The icon send will be displayed if no element with the icon class is provided when using the button component
- Before You couldn't give it a default content.
- Since version 18, this is no longer the case. You can have content inside the tag that will be displayed if no content is provided by the developer.

Form Events: a way to group the event of the form

Unified control state change events:

- `FormControl`, `FormGroup` and `FormArray` classes from Angular forms now expose a property called `events`, which allows you to subscribe to a stream of events for this form control. Using it you can track changes in value, touch state, pristine status, and the control status.

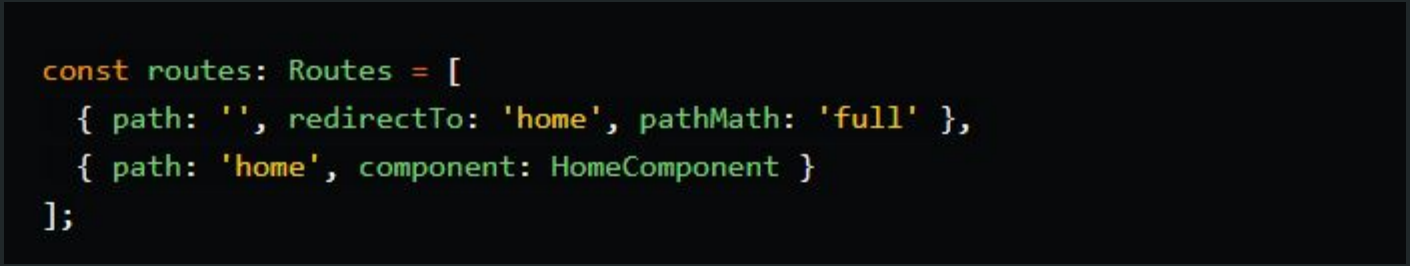


```
const nameControl = new FormControl<string|null>('name', Validators.required);
nameControl.events.subscribe(event => {
  // process the individual events
});
```

- This feature request was by the community itself and now it has over 440 thumb up on GitHub.

Routing: redirect as a function

- Before the latest version of Angular `redirectTo` property took its value only a character string.

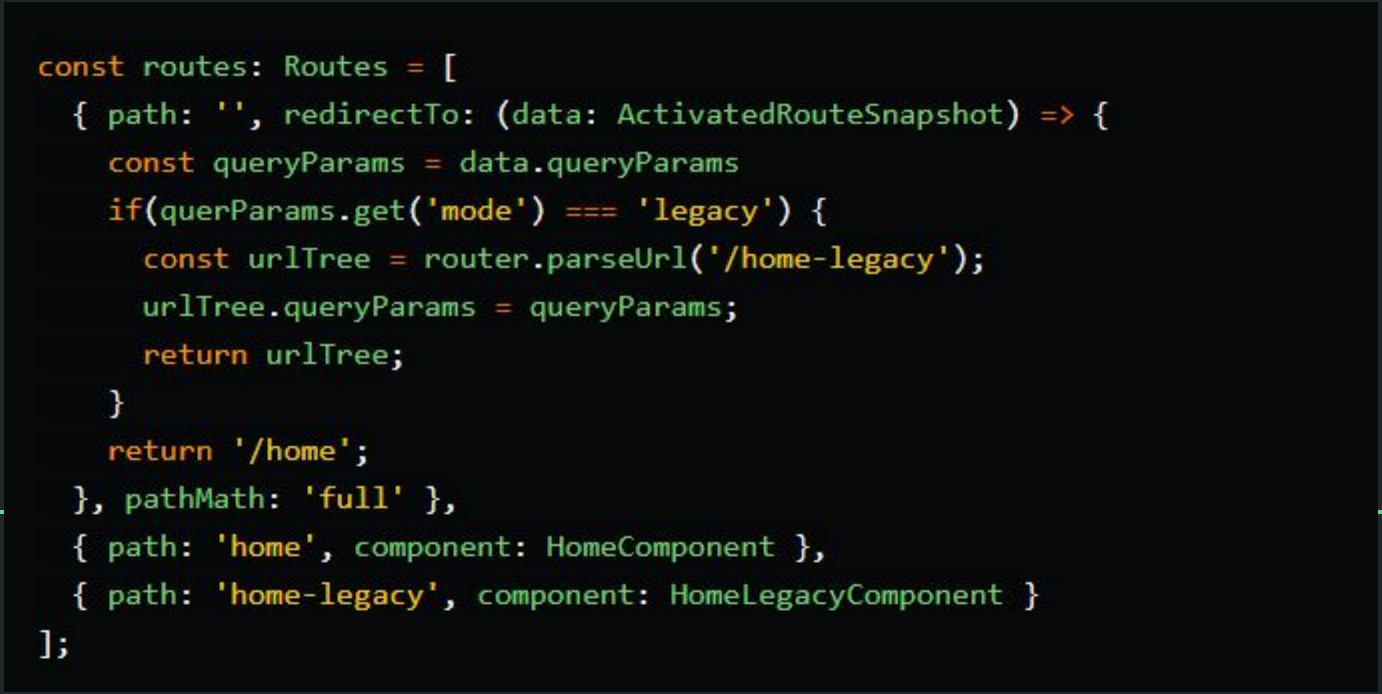


```
const routes: Routes = [
  { path: '', redirectTo: 'home', pathMath: 'full' },
  { path: 'home', component: HomeComponent }
];
```

- It is now possible to pass a function with this property. This function takes `ActivatedRouteSnapshot` as a parameter, allowing you to retrieve `queryParams` or params from the url.

- Another interesting point is that this function is called in an injection context, making it possible to inject services.

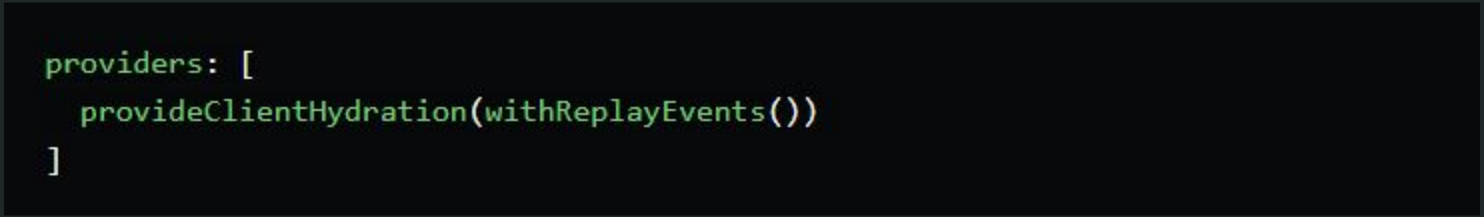
Routing: redirect as a function



```
const routes: Routes = [
  { path: '', redirectTo: (data: ActivatedRouteSnapshot) => {
    const queryParams = data.queryParams
    if(querParams.get('mode') === 'legacy') {
      const urlTree = router.parseUrl('/home-legacy');
      urlTree.queryParams = queryParams;
      return urlTree;
    }
    return '/home';
  }, pathMath: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'home-legacy', component: HomeLegacyComponent }
];
```

Server Side Rendering: two new awesome feature

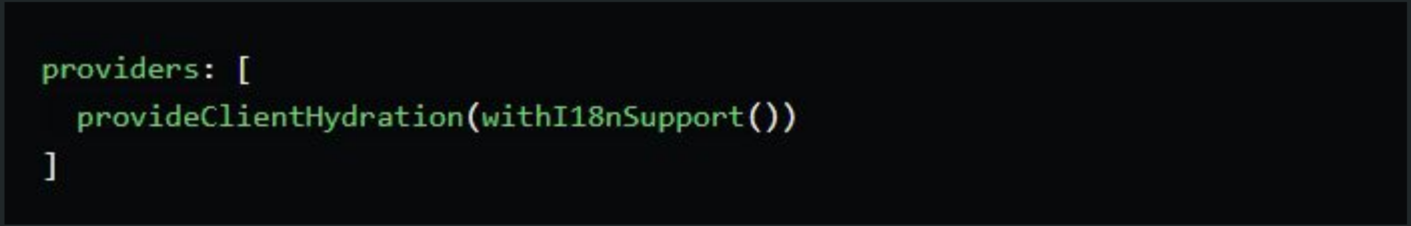
- Event Reply :
  - During **hydration** phase, no response to an interaction can be sent, so user interaction is lost until hydration is complete.
  - Angular is able to record user interactions during this hydration phase and replay them once the application is fully loaded and interactive.
  - To unlock this feature, still in developer preview, you can use the **ServerSideFeature** `withReplayEvents` function.



```
providers: [
  provideClientHydration(withReplayEvents())
]
```

Server Side Rendering: two new awesome feature

- Internationalization :
  - Please note that this feature is still in development preview and can be activated using the `withI18nSupport` function.



```
providers: [
  provideClientHydration(withI18nSupport())
]
```

Internationalization

- Angular recommends using the INTL native javascript API for all matters concerning the internationalization of an Angular application.
- With this recommendation, the function helpers exposed by the @angular/common package have become deprecated. As a result, functions such as `getLocaleDateFormat`, for example, are no longer recommended.

A new builder package and deprecation

- Up until now, and since the arrival of vite in Angular, the builder used to build the Angular application was in the package: `@angular-devkit/build-angular`. This package contained Vite, Webpack and Esbuild. A package that's far too heavy for applications that in future will use only Vite and Esbuild.
- With this potential future in mind, a new package containing only Vite and Esbuild was created under the name @angular/build, When migrating to Angular 18, an optional schematic can be run if the application does not rely on webpack (e.g. no Karma-based unit testing). This schematic will modify the angular.json file to use the new package and update the package.json by adding the new package and deleting the old one.

- When you migrate to Angular 18, these dependencies will be added automatically if you wish to use the new package.

No more donwleveling async/await

- Zone js does not work with the Javascript feature async/await.
- Angular's CLI transforms code using `async/await` into a "regular" Promise. This transformation is called downleveling, just as it transforms Es2017 code into Es2015 code.

- With the arrival of applications no longer based on Zone Js, even if this remains experimental for the moment, Angular will no longer downlevel if Zone Js is no longer declared in polyfills.ts.
- The application build will therefore be a little faster and a little lighter.

An new alias: ng dev

From now on, when the `ng dev` command is run, the application will be launched in development mode.

In reality, the ng dev command is an alias for the `ng serve` command.

This alias has been created to align with the Vite ecosystem, particularly the `npm run dev` command.

Typescript 4.5

Microsoft also Announcing TypeScript 5.4 (Clickable)

Feel free to visit the Microsoft devblogs website to review the new changes added to TS v5.4.

References

1- What'new in Angular 18 - DEV Community (clickable)

2- Angular v18 is now available!

3- Announcing TypeScript 5.4

Our Team created a repo that will record any changes to the new v18 of Angular, visit the below GitHub repo for reviewing code changes and examples regarding the new way of Angular.

https://github.com/OSRoot/angular-18-updates