Construyendo tu propio pipe 16/20

RECURSOS MARCADORES

Para poder afirmar que estás **construyendo tu propio Pipe**, es necesario hacer uso del CLI de Angular con el comando `ng generate pipe test-name` o en su forma corta con `ng g p test-name` .

Mi primer “pipe” en Angular

De la misma manera que lo hace con los servicios y componentes, el CLI creará un archivo `.ts` que contiene el código del pipe y un archivo `.spec.ts` para sus respectivas pruebas unitarias.

```
// pipes/test-name.pipe.ts
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'testName'
})
export class TestNamePipe implements PipeTransform {
  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
```

```
  }
}
```

El pipe ya trae algo de código y configuraciones por defecto. Tendrás que cambiar los `unknown` por el tipeado que necesites dependiendo el pipe que estés generando.

Es importante observar el decorador `@Pipe()` que contiene el nombre del pipe, para así poder llamarlo en tu HTML.
Y no olvides importar el pipe en el módulo de tu aplicación para que este pueda ser utilizado.

```
// app.module.ts
import { TestNamePipe } from './pipes/test-name.pipe';
@NgModule({
  declarations: [
    // ...
    TestNamePipe,
  ],
  imports: [
    // ...
  ],
  providers: [],
  bootstrap: [ /* ... */ ]
})
export class AppModule { }
```

Currency pipe personalizado

Vamos a crear tu propio currency pipe. Para esto harás uso del objeto global de Javascript Intl. Si no conocías este objeto, puedes leer más al respecto de su Especificación de ECMAScript.

El global Jacascript *Intl*, proporciona una API de internacionalización para el formateo de monedas y fechas, entre otras funcionalidades más.

Veamos un ejemplo de cómo utilizarlo:

```
// pipes/custom-currency.pipe.ts
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({
  name: 'customCurrency'
})
export class CustomCurrencyPipe implements PipeTransform {

  transform(value: number, ...args: string[]): string {
    const divisa = args[0];

    if (divisa == 'USD')
      return new Intl.NumberFormat('en-us', { style: 'currency', currency: 'USD'
}).format(value);
    else
      return new Intl.NumberFormat('es-ar', { style: 'currency', currency: 'ARS'
}).format(value);
  }

}
```

El pipe recibe un parámetro del tipo `number` y usando el segundo parámetro `args` puedes capturar las variables que necesites pasarle. En este ejemplo, lo empleamos para configurar el tipo de divisa.
Verificamos qué divisa se está pasando por parámetro. Si es **USD**, empleando el objeto `Intl` damos formato a la moneda para que sea **$1,000.00** mientras que con cualquier otra divisa sea **$1.000,00**. Nota la pequeña diferencias de intercambiar el punto y la coma para formatear el número y los decimales.

Finalmente, utiliza tu nuevo pipe en el HTML de la siguiente manera:

```
<div>
    {{ 1000 | customCurrency:'USD' }}
</div>
```

Los pipes utilizan un concepto llamado Memoization para ahora en tiempo de ejecución guardando el resultado de las funciones en memoria.
Emplea tus propios pipes siempre que sea posible para optimizar tu aplicación.

Ver código fuente del proyecto

Contribución creada con los aportes de Kevin Fiorentino.

Lecturas recomendadas

GitHub - platzi/angular-componentes at 13-step
https://github.com/platzi/angular-componentes/tree/13-step

date-fns - modern JavaScript date utility library
https://date-fns.org/v2.23.0/docs/formatDistance

Clases relacionadas

Crea tu propio pipe

Curso de Backend con NestJS