JavaScript Avanzado

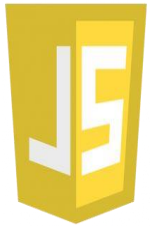
Mag. Armando Joel Donayre Cáceres

Tema: Elementos Angular

Logro de la unidad:

Al finalizar la unidad el participante aplica los frameworks NodeJS y Angular en la generación de soluciones web.

Utilidad: Conocer distintos frameworks relacionado al desarrollo con js



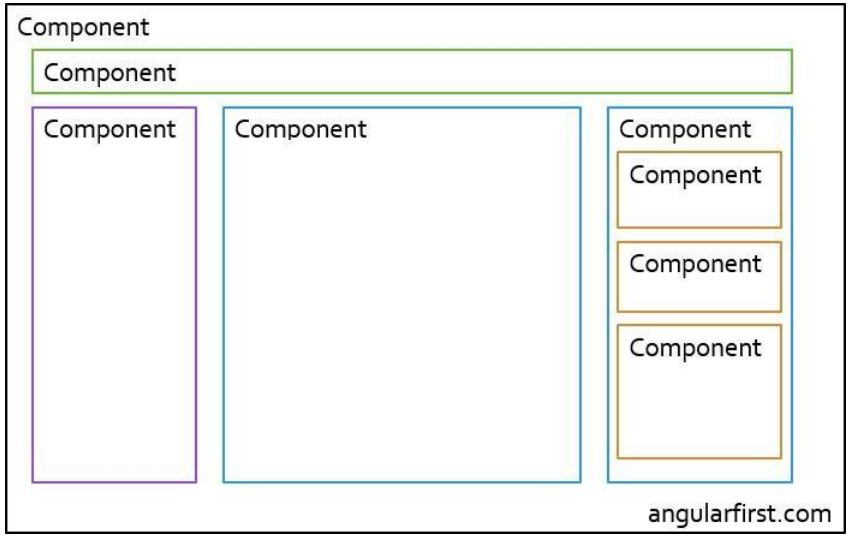
¿Qué hicimos la última clase?

Angular - Módulos

- Organizan la aplicación en bloques funcionales.
- Cada aplicación tiene, como mínimo, un módulo: el módulo principal o root, llamado **AppModule (App.module.ts**)
- En el módulo se declaran los componentes y, si es necesario, se importan otros módulos necesarios para su funcionamiento

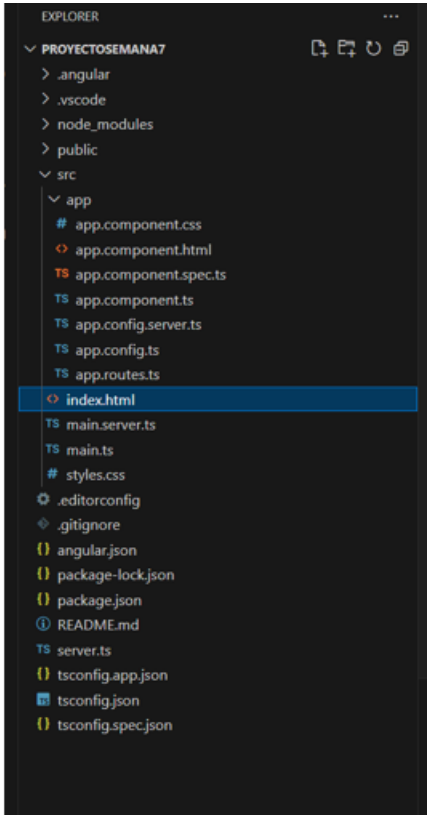
Angular - Componentes

- Angular.io: *Los componentes definen **vistas**, que son conjuntos de elementos de pantalla que Angular puede elegir y modificar según la lógica y los datos de su programa.*
- Se podría definir como una **etiqueta HTML** creada por el programador que consta de una vista y de una lógica:
  - **Vista**: Es una plantilla (*template*) HTML que puede incorporar elementos propios de angular.
  - **Lógica**: Clase Typescript asociada a la vista
- Son los principales bloques de construcción de una aplicación Angular
- Cada componente gestiona una pequeña parte de UI



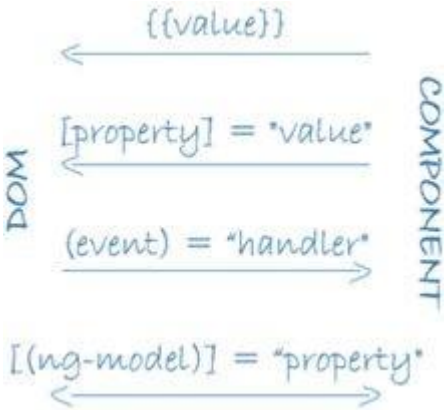
Componentes en Angular

- Componentes tienen tres partes:
- HTML (vista).
- CSS (estilos).
- TypeScript (lógica)



Data Binding

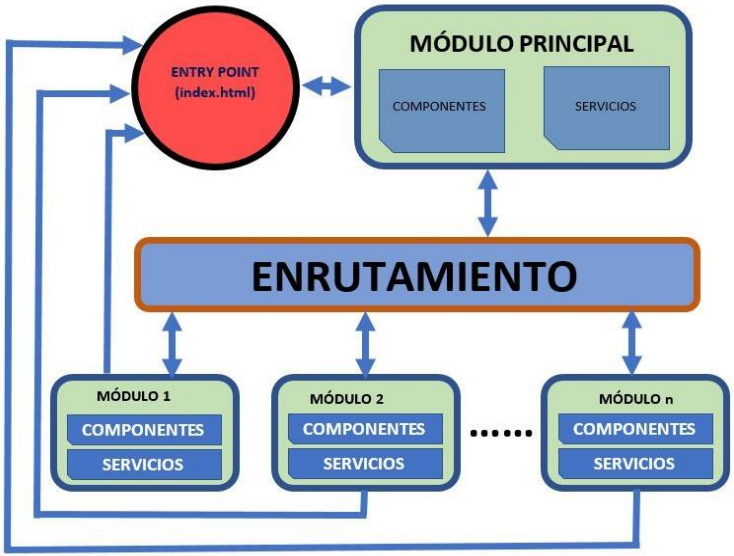
- Los componentes deben ser dinámicos: permitir modificar datos, responder a interacciones de usuario y reaccionar ante eventos.
- El **data binding** una técnica en la que los datos permanecen sincronizados entre el componente y la vista.
    - Siempre que el usuario actualiza los datos en la vista, Angular actualiza el componente.
    - Cuando el componente obtiene nuevos datos, Angular actualiza la vista.



Enrutamiento y navegación

RouterModule

- El **RouterModule** es el módulo que gestiona el enrutamiento y la navegación en Angular.
- Las aplicaciones Angular son SPA, es decir, sólo existe una página, pero múltiples vistas.
- El Router se encargará de procesar las rutas (URL's del navegador) y determinar cuál será la vista que deba mostrar en cada dirección.
- Gestiona la navegación **de una vista a otra vista** mediante la interpretación de la URL.
- Además, permite:
  - Pasar parámetros a la vista
  - Proteger determinadas rutas frente a usuarios no autorizados mediante *Guards*.



https://httpmasters.es/2018/05/02/estructura-de-una-aplicacion-angular-5/

Universidad Tecnológica del Perú

Pipes

Pipes

- Son funciones simples que se usan para transformar datos.
- Aceptan un valor de entrada y devuelven un valor transformado (sin modificar el original).
- Pipes integradas:
  - DatePipe
  - UpperCasePipe
  - LowerCasePipe
  - CurrencyPipe
  - DecimalPipe
  - PercentPipe
  - SlicePipe
  - AsyncPipe
  - JsonPipe
- https://angular.io/api?query=pipe

PIPE DATE

En aplicaciones reales, los datos a menudo provienen de bases de datos, APIs, o sistemas externos que almacenan las fechas en formatos que no son directamente amigables para los usuarios. Un formato común es el ISO 8601 (2024-10-02T00:00:00Z), que es eficiente para el almacenamiento y las operaciones lógicas en el backend, pero no es adecuado para mostrar al usuario.

CASO:

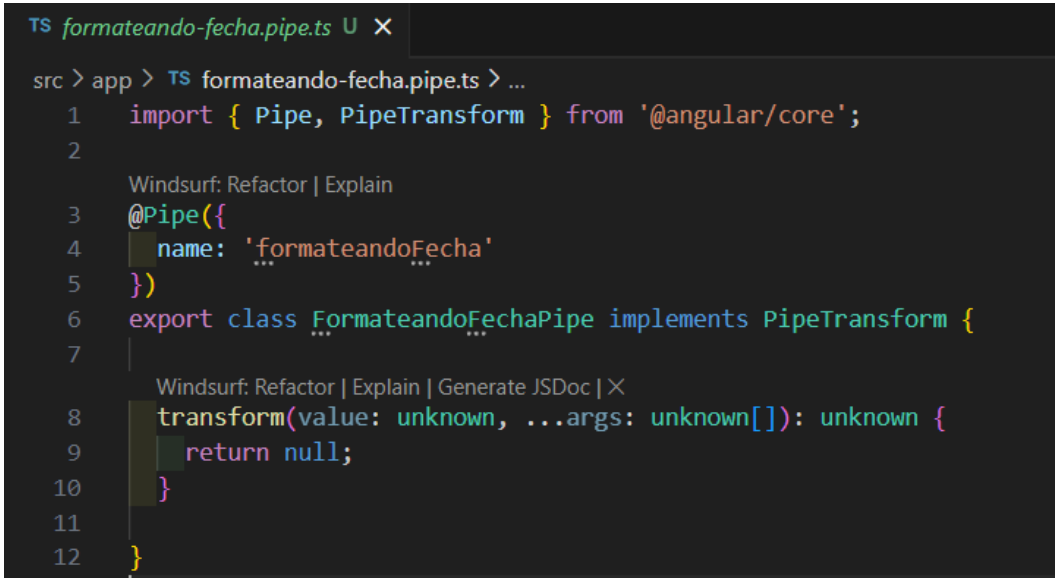
Imagina que desarrollas un sistema de reservas para un hotel. La base de datos almacena las fechas en formato 2024-10-02T00:00:00, pero los usuarios finales esperan ver la fecha en un formato más legible como 02 de Octubre de 2024. Usar el pipe date en Angular puede resolver este problema sin tener que cambiar los datos en tu backend.

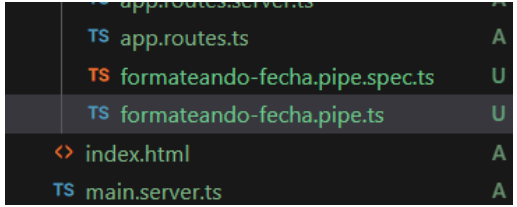
Ejemplo de este pipe

Creando en nuestra terminal un pipe:

ng generate pipe FormateandoFecha

En tu archivo customDatePipe edita lo sgte:

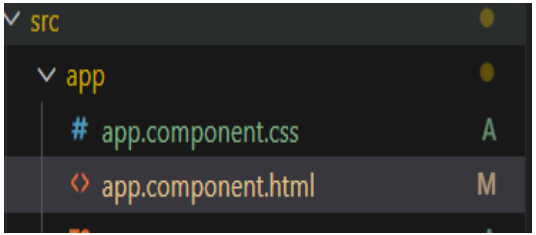




app.component.html M | formateando-fecha.pipe.spec.ts U | app.component.ts 1, M | formateando-fecha.pipe.ts U

src > app > formateando-fecha.pipe.ts > ...

```ts
import { Pipe, PipeTransform } from '@angular/core';

Windsurf: Refactor | Explain
@Pipe({
  name: 'formateandoFecha'
})
export class FormateandoFechaPipe implements PipeTransform {

  Windsurf: Refactor | Explain | Generate JSDoc | X
  transform(value: any): string {

    if (!value) return "No hay nada que transformar";

    const fecha = new Date(value);

    return `${fecha.getDate()}/${fecha.getMonth() + 1}/${fecha.getFullYear()}`;
  }
}
```



app.component.html M X | formateando-fecha.pipe.spec.ts U | app.component.ts 1, M | formateando

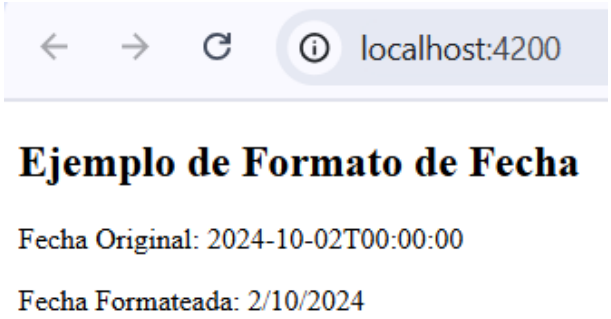
src > app > app.component.html > ...

Go to component

```
1  <h2>Ejemplo de Formato de Fecha</h2>
2  <p><span>Fecha</span> Original: {{ '2024-10-02T00:00:00' }}</p>
3  <p><span>Fecha</span> Formateada: {{ '2024-10-02T00:00:00' | formateandoFecha }}</p>
4
```

Ahora lo podemos usar en HTML, como por ejemplo:

```
<p>Fecha Original: {{ '2024-10-02T00:00:00' }}</p>
<p>Fecha Formateada: {{ '2024-10-02T00:00:00' | customDate }}</p>
```



CONCLUSIONES

Los pipes son útiles cuando quieres cambiar la presentación de un valor sin tener que modificar los datos originales en tu lógica de negocio.
Podemos usarlos para:
Formatear fechas (date).
Redondear números (number).
Cambiar texto a mayúsculas o minúsculas (uppercase, lowercase).
O crear tus propios pipes para hacer transformaciones personalizadas, como formatear fechas, ajustar el formato de monedas, y más.
Los pipes en Angular nos ayudan a transformar los datos para que sean más legibles o amigables para los usuarios, sin cambiar el valor original.