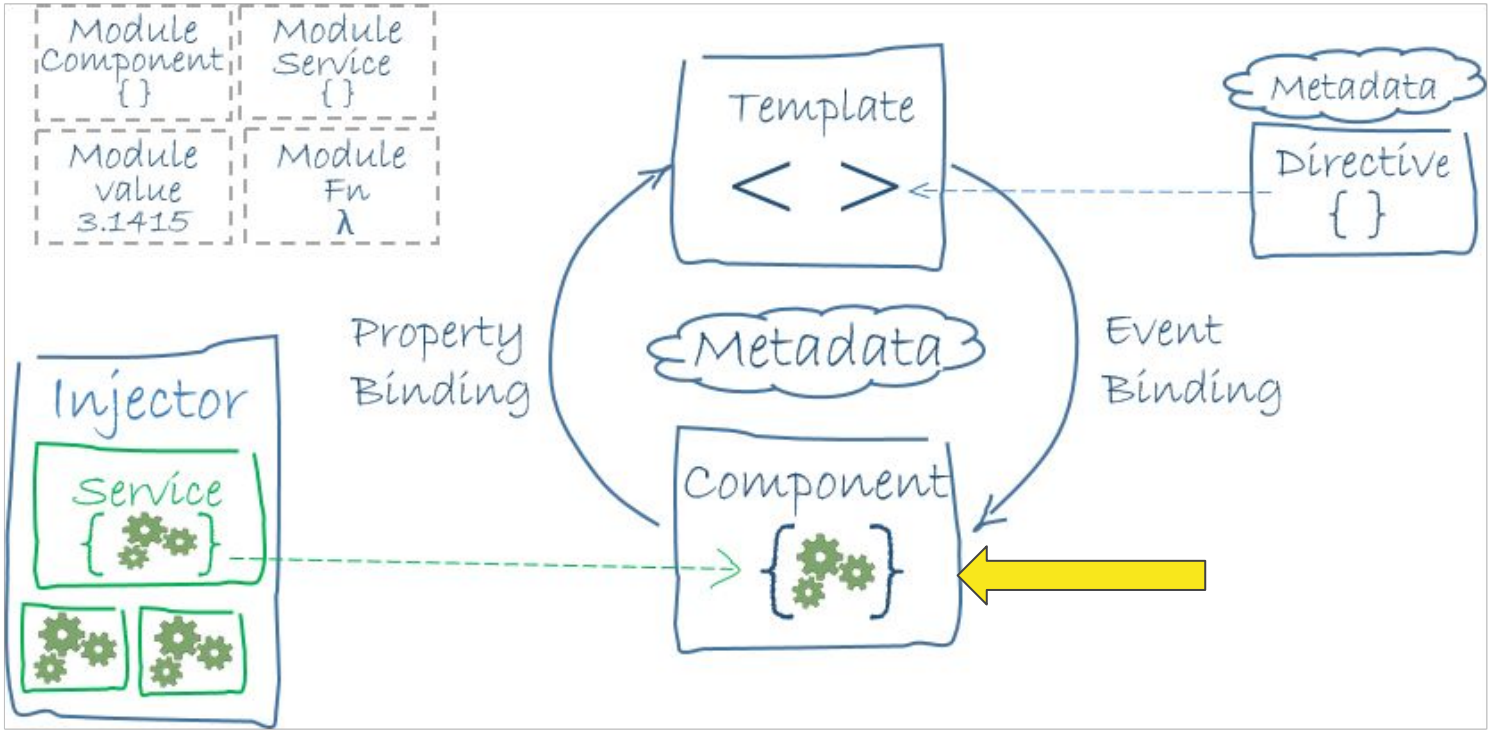
Angular

Components

Angular - Arquitectura



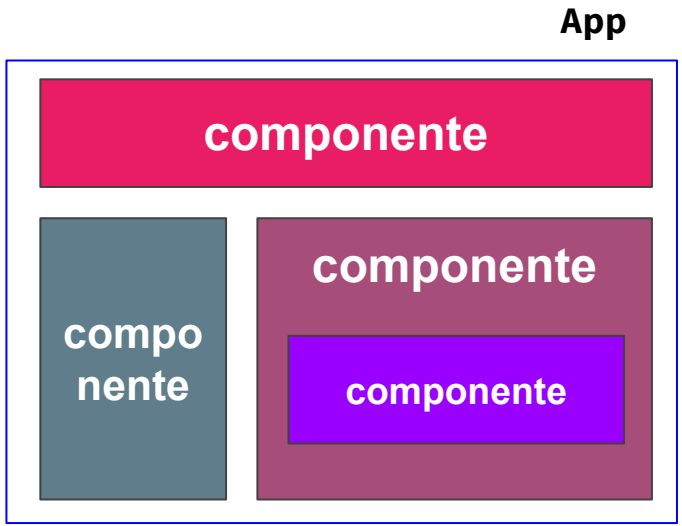
Qué es un Component?


FUSION
THE CHRIS GETHARD SHOW

Componentes en Angular

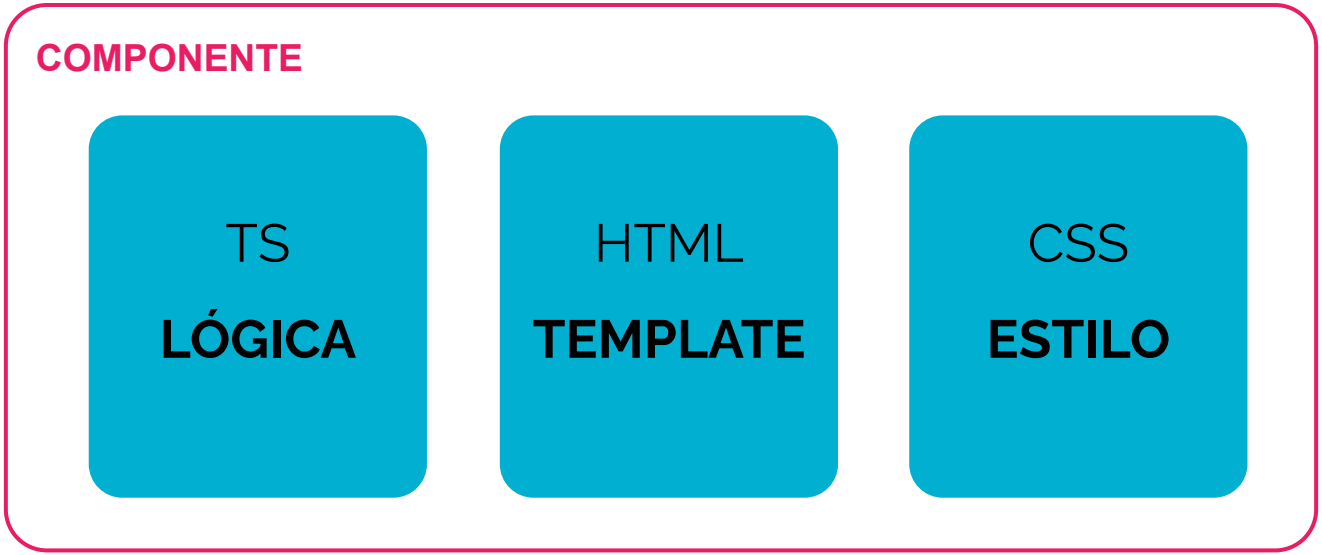
Los componentes son la manera en que construimos apps en Angular, dividiendo la funcionalidad en pequeñas piezas que luego al unirlas construyen una vista de un usuario.

- Cada parte de nuestro app va a ser un componente.
- Son reutilizables :)



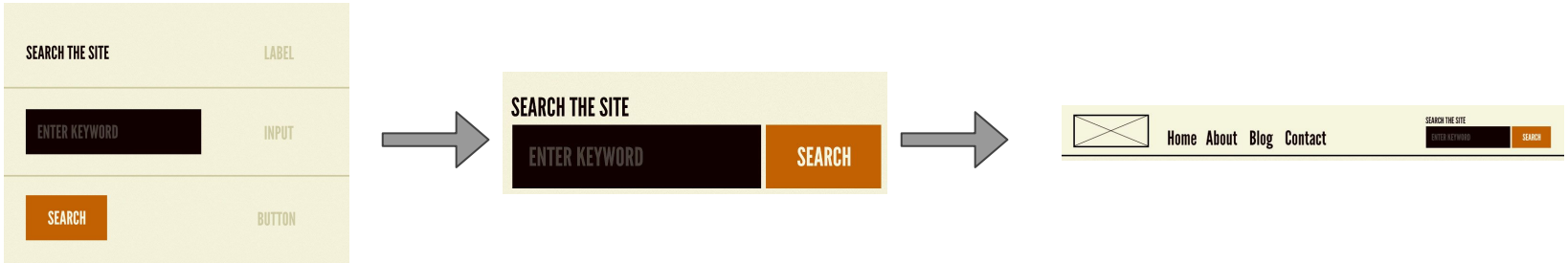
Componentes

Un componente **Angular** está compuesto por tres partes:



Componentes y atomicidad

Los componentes siguen un **diseño atómico.**



https://blog.maestriajs.com/blog/angular/Angular-Crear-Componentes/

Estructura de un Componente

En Angular, los componentes son una clase acompañado con un **@decorator**

app.component.ts

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app works!';
}
```

index.html

```html
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Brewery</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root>Loading...</app-root>
</body>
</html>
```

selector: Como vamos a incluir nuestro componente.
templateUrl: El html que vamos a cargar en nuestro componente.
styleUrls: Las hojas de estilo de nuestro componente.

Cargar un componente

- Para poder cargar el componente, lo tenemos que declarar en el **app.module.ts**

```
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Declaro todos los componentes que voy a usar

Este es el root component

angular-cli lo hace por nosotros cuando creamos un componente nuevo!

Template del Componente

- El template es mayormente HTML.
- También se usan **directives** de Angular para poder hacer diferentes cosas. (Coming soon!)
- Para mejorar la separación, lo ponemos en otro archivo.

Manos a la obra

Nuestro primer componente

- El primer componente que vamos a necesitar es “App Component” (que ya esta creado)

app.component.ts

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app works!';
}
```

app.component.html

```html
<h1>
  {{title}}
</h1>
```

Interpolation

- Las llaves **{{ }}** nos permiten usar propiedades definidas en el componente y mostrarlas en el template.

app.component.ts

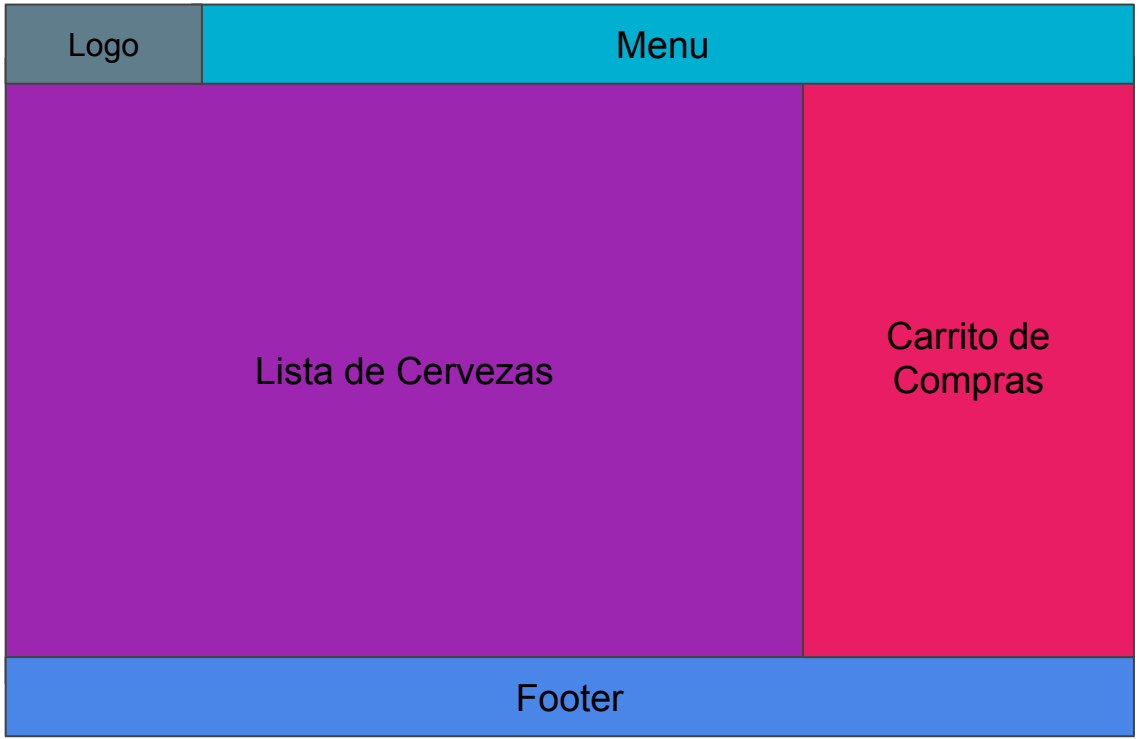
```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app works!';
}
```

app.component.html

```html
<h1>
  {{title}}
</h1>
```

Proyecto “The Brewery” - Diseño



Pensemos en Componentes

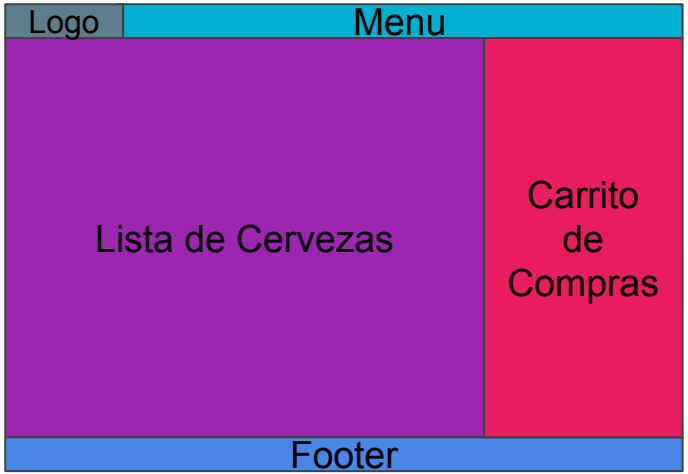
- Si una app de Angular se construye en componentes, que componentes se les ocurre que creemos?

Pensemos el app.component template

Miremos el layout de cómo debería ser la homepage de nuestra App Angular.

- Qué sería un componente?
- Qué sería contenido estático?

Componentes	Contenido Estatico

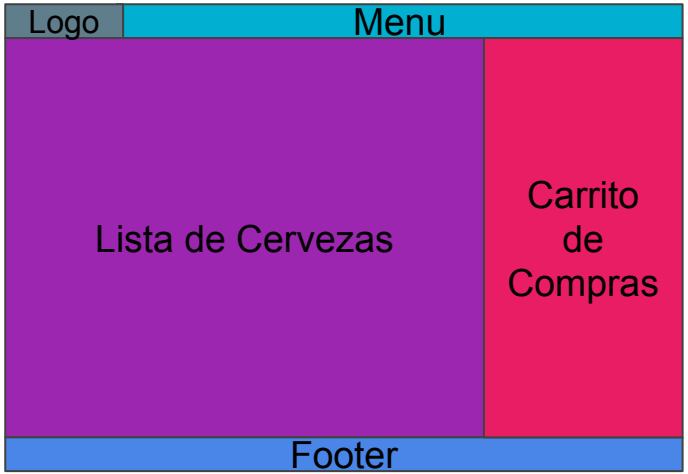


Pensemos el app.component template

Hagamos un diagrama de cómo debería ser la homepage de nuestra App Angular.

- Qué sería un componente?
- Qué sería contenido estático?

Componentes	Contenido Estatico
• Lista de cervezas • Carrito de compras	• Logo • Menu • Footer



Hagamos la Home Page

- Vamos a usar Bootstrap (lo incluimos)
- Armemos la estructura de columnas
- Pongamos el contenido estático

Creemos el componente - Lista de Cervezas

Creemos el componente nuevo usando *angular-cli*

- Ejecutamos

  ng generate component beer-list

  ó la versión corta:

  ng g c beer-list

- Esto nos crea una carpeta **beer-list** con todo lo que necesita nuestro componente.

Revisemos el componente

- Nos lo creo con el selector **app-beer-list**
- Buena práctica usar prefijos por app, para evitar colisiones con bibliotecas
- El template solo muestra el “**beer-list works!**”
- angular-cli nos agregó todas las dependencias en app.module.
- Probemoslo, agregandolo en el lugar del grid que dejamos listo.

```
<div class="row">
  <div class="col-md-8">
    <beer-list>Loading...</beer-list>
  </div>
  <div class="col-md-4 well">Carrito</div>
</div>
```

https://angular.io/guide/styleguide#component-custom-prefix

Modifiquemos el Template

- Necesitamos una lista de cervezas, pero es una lista o una tabla?

Modifiquemos el Template

Supongamos que para la tabla de cervezas necesitamos

- **Nombre:** El nombre comercial de la cerveza.
- **Tipo:** El estilo de cerveza.
- **Precio:** EL precio por unidad.
- **Stock:** La cantidad de cervezas en stock.

Nombre	Tipo	Precio	Stock
Negra Juerte	Porter	20	100
Red Red Wine	Barley Wine	40	1200
La Rubia	Golden Ale	60	1500

Flujo de datos entre JS y DOM

Binding de variables

- Ahora vamos a hacer el binding de una cerveza.
- Creamos una propiedad en el componente.
- Usamos interpolación para mostrarlo en la tabla.

```
export class BeerListComponent implements OnInit {

  beer = {
    name: "Negra Fuerte",
    style: "Porter",
    price: 12,
    stock:100
  };
  constructor() { }
```

```
  </thead>
  <tbody>
    <tr>
      <td>{{beer.name}}</td>
      <td>{{beer.style}}</td>
      <td>{{beer.price}}</td>
      <td>{{beer.stock}}</td>
    </tr>
  </tbody>
</table>
```

Agregar una imagen

- Cómo hacemos para agregar una imagen a nuestras cervezas?
- Listemos los pasos:
  - TBC

Beer-list Component Template

- Podemos hacerlo con interpolación.

```
<td><img src="{{beer.image}}" alt="{{beer.name}}"/></td>
<td>{{beer.name}}</td>
<td>{{beer.style}}</td>
```

- Hay una sintaxis especial que también se puede usar

```
<td><img [src]="beer.image" [alt]="beer.name"/></td>
<td>{{beer.name}}</td>
<td>{{beer.style}}</td>
```

Property Binding

El property binding nos permite relacionar componentes con propiedades del DOM.

- Usamos corchetes [] en lugar de llaves {}.
- Los corchetes le dicen a Angular que setee esta propiedad del elemento a la propiedad de nuestro componente.
- Si la propiedad del componente cambia, entonces cambia el DOM.

```
<td><img [src]="beer.image" [alt]="beer.name"/></td>
<td>{{beer.name}}</td>
<td>{{beer.style}}</td>
```

LISTO!
Nuestro primer componente Angular

Ejercicio

Crear el componente del carrito (o el segundo componente de tu página)

Qué pasa si creo una clase css para poner estilo a la tabla, cuyo nombre está usado en el app component?

Hagamos la prueba

- Creemos la clase **resaltar**, en app.component.css
- Tratemos de usarla en app.component.html y en beer-list.component.html

- Probemos crear una class **resaltar** en beer-list.component.css y usemosla en beer-list.component.html

Estilos en Componentes

- Los estilos que creemos en las hojas de estilos de nuestros componentes, se auto limitan a nuestro componente.
- No tenemos que preocuparnos si estamos pisando otro estilo.

A trabajar un poco...

- Creen el componente del Carrito (cart).
- Hagan el HTML del template.
- Prueben crear una misma clase de css en app.component.css y en cart.component.css, para ver como se auto limita el estilo al componente.

Referencias

- Angular.io - Getting Started - Documentacion Oficial
- Branch en el Repositorio