Hook: método que se llama cuando ocurre algo

#etiqueta: sirve para acceder a una etiqueta desde la propia vista o desde el ViewChild en código

```
<div>
    <label for="txtBox">Filtro:</label>
    <input #txtBox type="text" id="txtBox" name="txtBox">
    <button (click)="applyFilter(txtBox.value)">Buscar</button>
</div>
```

Input:

Html padre:

```
<app-header-component [direccionImagen]="direccionImagen"></app-header-component>
```

Ts hijo:

```
@Input() direccionImagen! : string;
```

Output:

Html padre:

```
<cabecera (eventoMostrarLibro)="mostrarLibro($event)"></cabecera>
```

Ts hijo:

```
@Output() eventoMostrarLibro = new EventEmitter<Object>();

enviarIdLibro(id : number) {
  this.eventoMostrarLibro.emit(id);
}
```

Proyección de contenido:

Html padre: (html en árbol normal)

```
<app-header2-component>
    <img src="{{direccionImagen}}" alt="Logo">
</app-header2-component>
```

Html hijo: (ng-content donde quiera proyectar la o las etiquetas del padre)

```
<ng-content></ng-content>
<div>
    <button>Botón 1</button>
    <button>Botón 2</button>
</div>
```

Para proyectar una etiqueta en concreto se usa select="" y dentro el nombre de etiqueta / .nombreClase / #nombreId

```
<ng-content select=".className"></ng-content>
```

Acceso a vista desde código:

Html hijo:

```
<img #childImage alt="Logo">
```

Ts hijo:

```
@ViewChild('childImage') childImage!: ElementRef;
```

Html padre:

```
<app-header3-component></app-header3-component>
```

Ts padre:

```
@ViewChild(Header3ComponentComponent) cabecera3! : Header3ComponentComponent;

ngAfterViewInit() {
  this.cabecera3.childImage.nativeElement.src = this.direccionImagen;
}
```

Directiva:

Html:

```
<p directiva>soy un texto con directiva</p>
```

Ts directiva:

```
import { Directive, ElementRef, HostBinding, HostListener, Renderer2 } from '@angular/core';

@Directive({
  selector: '[directiva]'
})
export class DirectivaDirective {

  constructor(private element: ElementRef, private renderer: Renderer2) { }

  @HostBinding('class.rojo') pulsado: boolean = false;

  @HostListener('mouseover') onMouseOver() {
    this.renderer.addClass(this.element.nativeElement, 'negrita');
    this.pulsado = true;
  }

  @HostListener('mouseout') onMouseOut() {
    this.renderer.removeClass(this.element.nativeElement, 'negrita');
    this.pulsado = false;
  }

}
```

Directiva parametrizada:

Html:

```
<a href="https://www.google.com" [directivaAParametrizada] = "{colorBorde: 'red', texto: 'Texto de enlace'}">https://www.google.com</a>
```

Ts directiva:

```ts
import { Directive, ElementRef, HostListener, Input, Renderer2 } from '@angular/core';

@Directive({
  selector: '[directivaAParametrizada]'
})
export class DirectivaAParametrizadaDirective {

  @Input('directivaAParametrizada') datos: any = {
    colorBorde: 'green',
    texto: 'https://angular.io/'
  };

  constructor(private element: ElementRef, private renderer: Renderer2) {}

  ngOnInit() {
    this.renderer.setProperty(this.element.nativeElement, 'textContent', this.datos.texto);
  }

  @HostListener('mouseenter') onMouseEnter() {
    this.renderer.setStyle(this.element.nativeElement, 'border', `1px solid ${this.datos.colorBorde}`);
    this.renderer.setStyle(this.element.nativeElement, 'text-transform', `uppercase`);
  }

  @HostListener('mouseout') onMouseOut() {
    this.renderer.removeStyle(this.element.nativeElement, 'border');
    this.renderer.removeStyle(this.element.nativeElement, 'text-transform');
  }
}
```

Directiva ngClass:

Html:

```html
<div [ngClass]="classNames">
```

Ts:

```ts
classNames: any = {
  panel: true,
  error: true
}
```

Directiva ngModel:

Html:

```html
[(ngModel)]="username"
```

Ts:

```ts
username: string = '';
```

Tubería:

Html:

```
<p>{{ texto | tuberia:['boo','damn','hell'] }}</p>
```

Ts tubería:

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

@Pipe({
  name: 'tuberia'
})
export class TuberiaPipe implements PipeTransform {

  transform(texto : string, palabrasInadecuadas : string[]): string {
    palabrasInadecuadas.forEach((palabra : string) => {
      texto = texto.replace(palabra, '****');
    });

    return texto;
  }

}
```

Formularios:

Html:

```
<form [formGroup]="form" (ngSubmit)="onSubmit()" >
    <label for="nombre">Nombre </label>
    <input type="text" name="nombre" id="nombre" formControlName="nombre" [ngClass]="{ 'bordeError': form.get('nombre')?.invalid && form.get('nombre')?.touched }">
    <p *ngIf="form.get('nombre')?.invalid && form.get('nombre')?.touched" class="error">Debe tener al menos 6 caracteres</p>
    <br><br>
    <label for="numero">Número de teléfono </label>
    <input type="text" name="numero" id="numero" formControlName="numero" [ngClass]="{ 'bordeError': form.get('numero')?.invalid && form.get('numero')?.touched }">
    <p *ngIf="form.get('numero')?.invalid && form.get('numero')?.touched" class="error">Debe tener 9 dígitos</p>
    <br><br>
    <button type="submit" [disabled]="!form.valid" [innerText]="edicion ? 'Confirmar' : 'Agregar'"></button>
</form>
```

Ts:

```
form! : FormGroup;

constructor(private formBuilder: FormBuilder) {
}

ngOnChanges(): void {
  this.actualizarDatos();
}

ngOnInit(): void {
  this.actualizarDatos();
}

actualizarDatos(): void {
  this.form = this.formBuilder.group({
    nombre: [this.nombre, [Validators.required, Validators.minLength(6)]],
    numero: [this.numero, [Validators.required, Validators.pattern(/^\d{9}$/)]]
  });
}

onSubmit(): void {
  if (this.form.valid) {
    this.edicion
    ? this.eventoEditarContacto.emit({nombre : this.form.value.nombre, numero : this.form.value.numero})
    : this.eventoNuevoContacto.emit(new Contacto(this.form.value.nombre, this.form.value.numero));

    this.nombre = undefined;
    this.numero = undefined;
    this.actualizarDatos();
  }
}
```

Observables:

Ts servicio:

```ts
//properties

private _books : Book[];
private _subject: BehaviorSubject<Object[]>;

constructor() {
  this._books = [];

  this._books.push(new Book("El corredor del laberinto", "James Dashner", "1", new Date(2010,7, 24)));
  this._books.push(new Book("Los juegos del hambre", "Suzanne Collins", "2", new Date(2008, 8, 14)));
  this._books.push(new Book("Percy Jackson", "Rick Riordan", "3", new Date(2005,5,28)));

  this._subject = new BehaviorSubject<Object[]>(this.books);
}

//getters

public get subject(): Observable<Object[]> {
  return this._subject.asObservable();
}

public get books(): Object[] {
  return this.getBooksCopy(this._books);
}

//methods

public getFilteredBooks(filter: string): void {
  let filteredBooks: Book[] = this._books.filter(book => book.title.includes(filter));
  this._subject.next(this.getBooksCopy(filteredBooks));
}
```

Ts componente:

```ts
books: any[];
observable: Observable<any[]>;

constructor(private service: Service) {
  this.books = service.books;

  this.observable = this.service.subject;
}
```

Html componente:

```html
<ul>
    <li *ngFor="let book of observable | async">
        <p>Título: {{book.title}}</p>
        <p>Autor/a: {{book.author}}</p>
        <p>ISBN: {{book.isbn}}</p>
        <p>Fecha de publicación: {{book.date | date: "dd/MM/YYYY"}}</p>
    </li>
</ul>
```

Para hacerlo sin la tubería async y suscribiéndose al observable desde código:

Ts servicio igual

Ts componente:

```ts
books: any[];

constructor(private service: Service) {
  this.books = service.books;

  this.service.subject.subscribe(books => this.books = books);
}
```

Routing:

Html padre:

```html
<router-outlet></router-outlet>
```

Ts app-routing.module:

```ts
const routes: Routes = [
  { path: 'loginPage', component: LoginPageComponent },
  {
    path: 'loggedPage',
    component: LoggedPageComponent,
    canActivate: [authenticationGuard]
  },
  { path: '', redirectTo: '/loginPage', pathMatch: 'full' },
];
```

Ts página desde la que quiero navegar a otra:

```ts
constructor(private router: Router, private authenticationService: AuthenticationService) {}

goToLoggedPage() {
  this.authenticationService.logIn(this.username, this.password);

  this.router.navigate(['/loggedPage'], {
    queryParams: {username: this.username, password: this.password}
  });
}
```

Ts página a la que navega:

```ts
constructor(private route: ActivatedRoute) {
  this.route.queryParams.subscribe(params => {
    this.username = params['username'];
    this.password = params['password'];
  });
}
```

Guard:

Ts app-routing.module:

```
{
  path: 'loggedPage',
  component: LoggedPageComponent,
  canActivate: [authenticationGuard]
},
```

Ts guard:

```
export const authenticationGuard: CanActivateFn = (route, state) => {
  const authenticationService: AuthenticationService = inject(AuthenticationService);
  const router: Router = inject(Router);

  if (authenticationService.isLoggedInUser()){
    return true;
  }
  else {
    router.navigate(['/loginPage']);
    return false;
  }
};
```