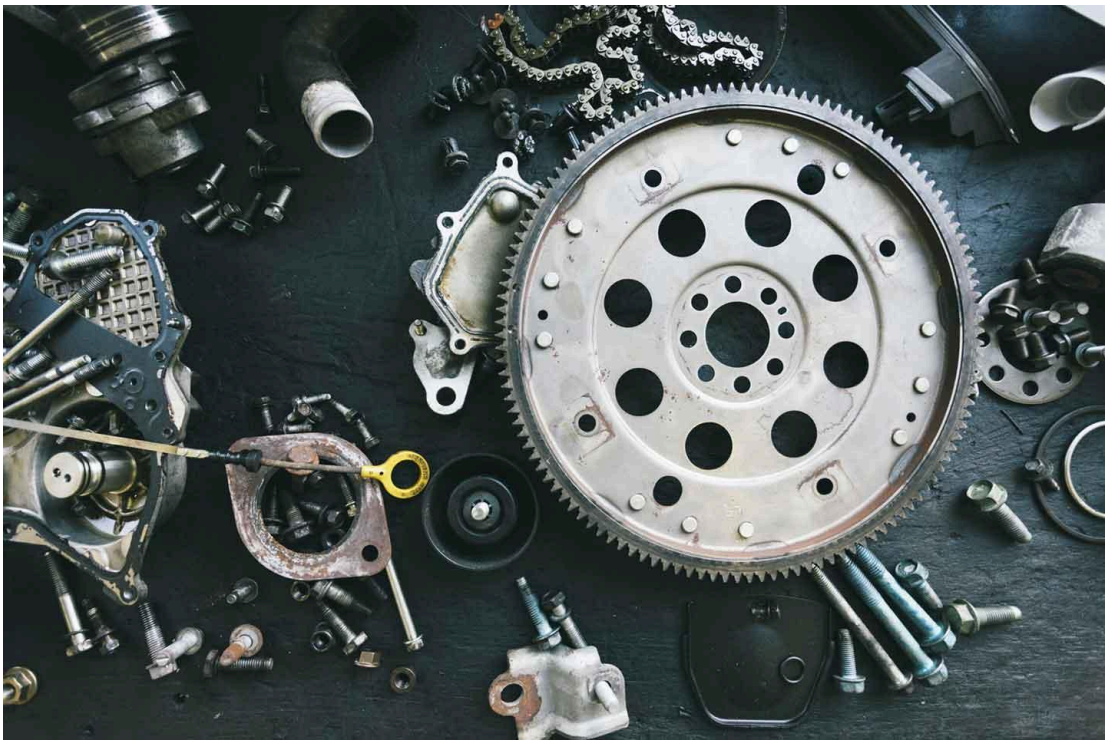
# Ionic v8

*Popover*



Progresa
Una formación de futuro

# ÍNDICE

# Introducción

Es un diálogo que aparece en la parte superior de la página actual. Puede usarse para cualquier cosa, pero generalmente se usa para funciones que no caben en la barra de navegación. Hay 2 maneras de usar el popover: en el propio html o mediante un controlador. Vamos a ver estas 2 formas.

La documentación del Popover está en la página:
https://ionicframework.com/docs/api/popover

# Presentación

## Componente

Para empezar a trabajar con el popover vamos a necesitar una página donde usarlo y otra que cargará el contenido. Creamos las páginas en la **terminal**:

**ionic g page pages/Popover --spec=false**
**ionic g page pages/Popover-data --spec=false**

## Icono

Añadimos en nuestro **app.component.ts** el icono que queremos usar para el menú en la lista de iconos(addIcons).

```
addIcons({grid, …,  caretDownCircle, flame});
```

## JSON

Añadimos en nuestro **menu.json** el nuevo elemento que queremos que aparezca tanto en el menú como en la página de inicio.

```
,{
 "nombre": "Popover",
 "ruta": "/popover",
 "icono": "flame"
}
```

# EJEMPLO

## VISUALIZACIÓN

Y empezamos a dar contenido, primero **popover.page.html**:

```
<ion-header>
 <ion-toolbar color="primary">
   <ion-buttons slot="start">
     <ion-back-button defaultHref="/"></ion-back-button>
   </ion-buttons>
   <ion-title class="ion-text-center">Popover</ion-title>
   <ion-buttons slot="end">
     <ion-button id="trigger-header">
                              <ion-icon    name="person-add-outline"
slot="icon-only"></ion-icon>
     </ion-button>
     <ion-popover trigger="trigger-header">
       <ng-template>
         <ion-content class="ion-padding">Header Popover!</ion-content>
       </ng-template>
     </ion-popover>
   </ion-buttons>
 </ion-toolbar>
</ion-header>
```

```
<ion-content class="ion-padding">
 <ion-button expand="full" (click)="presentPopover($event)">
   <ion-icon name="person-add-outline" slot="start"></ion-icon>
   <ion-label>Popover Controller</ion-label>
 </ion-button>
 <p>{{ message }}</p>
</ion-content>

<ion-footer>
 <ion-toolbar color="primary">
   <ion-buttons slot="start">
     <ion-button id="trigger-footer">
       <ion-icon name="person-add-outline" slot="icon-only"></ion-icon>
     </ion-button>
     <ion-popover trigger="trigger-footer">
       <ng-template>
         <ion-content class="ion-padding">Footer Popover!</ion-content>
       </ng-template>
     </ion-popover>
   </ion-buttons>
   <ion-title class="ion-text-center">Footer</ion-title>
 </ion-toolbar>
</ion-footer>
```

Con esto hemos definido un popover en el header y otro en el footer mediante el método inline. Mediante este método no hace falta más, pero no hay control sobre el popover para interactuar. En la parte del central, en el content, vamos a ver como realizar el popover con controlador.

## Lógica

En el **popover.page.ts**:

```
import {Component, inject, OnInit} from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { IonBackButton, IonButton, IonButtons, IonContent,
IonFooter, IonHeader, IonIcon, IonLabel, IonPopover, IonTitle,
IonToolbar } from '@ionic/angular/standalone';
```

```
import {PopoverController} from "@ionic/angular";
import {PopoverDataPage} from "../popover-data/popover-data.page";

@Component({
 selector: 'app-popover',
 templateUrl: './popover.page.html',
 styleUrls: ['./popover.page.scss'],
 standalone: true,
providers: [PopoverController],
    imports:    [IonContent,    IonHeader,    IonTitle,    IonToolbar,
CommonModule,  FormsModule,  IonButtons,  IonBackButton,  IonButton,
IonIcon, IonPopover, IonLabel, IonFooter]
})
export class PopoverPage {
        private         popoverController:         PopoverController
=inject(PopoverController);
 message = '';

 constructor() {}

 async presentPopover(e: Event) {
   const popover = await this.popoverController.create({
     component: PopoverDataPage,
     event: e,
   });

   await popover.present();

   const result = await popover.onDidDismiss();
   if(result.data) {
               this.message   =   `Popover   dismissed   with   item:
${result.data.item}`;
   }
 }
}
```

Y en **app.component.ts** añadimos el icono usado en el popover:

```
 addIcons({grid, … , flame, personAddOutline});
}
```

## Contenido del popover

Y la definición de la página de contenido del popover, **popover-content.page.html**:

```
<ion-content>

 <ion-list>
   <ion-item *ngFor="let item of items;let i=index;" detail="true"
             (click)="onClick( i + 1 )">
     <ion-label>Item {{ i + 1 }}</ion-label>
   </ion-item>
 </ion-list>

</ion-content>
```

Y la lógica del propio popover en **popover-content.page.ts**:

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import {IonContent, IonHeader, IonItem, IonLabel, IonList, IonTitle,
IonToolbar} from '@ionic/angular/standalone';
import {PopoverController} from "@ionic/angular";

@Component({
 selector: 'app-popover-data',
 templateUrl: './popover-data.page.html',
 styleUrls: ['./popover-data.page.scss'],
 standalone: true,
providers: [PopoverController],
    imports:    [IonContent,    IonHeader,    IonTitle,    IonToolbar,
CommonModule, FormsModule, IonList, IonItem, IonLabel]
})
export class PopoverDataPage  {
 private popoverCtrl: PopoverController = inject(PopoverController);
 items = Array(40);
 constructor() { }
 onClick( valor: number) {
   this.popoverCtrl.dismiss({ item: valor });
 }
}
```

*Explicación*

```
<ion-popover trigger="trigger-header">
```

Con trigger decimos el id que disparará este popover.

```
 <ng-template>
```

Como pasaba en el modal, representa la plantilla que contendrá el popover.

```
 component: PopoverContentPage
```

El componente que cargaremos dentro del popover.