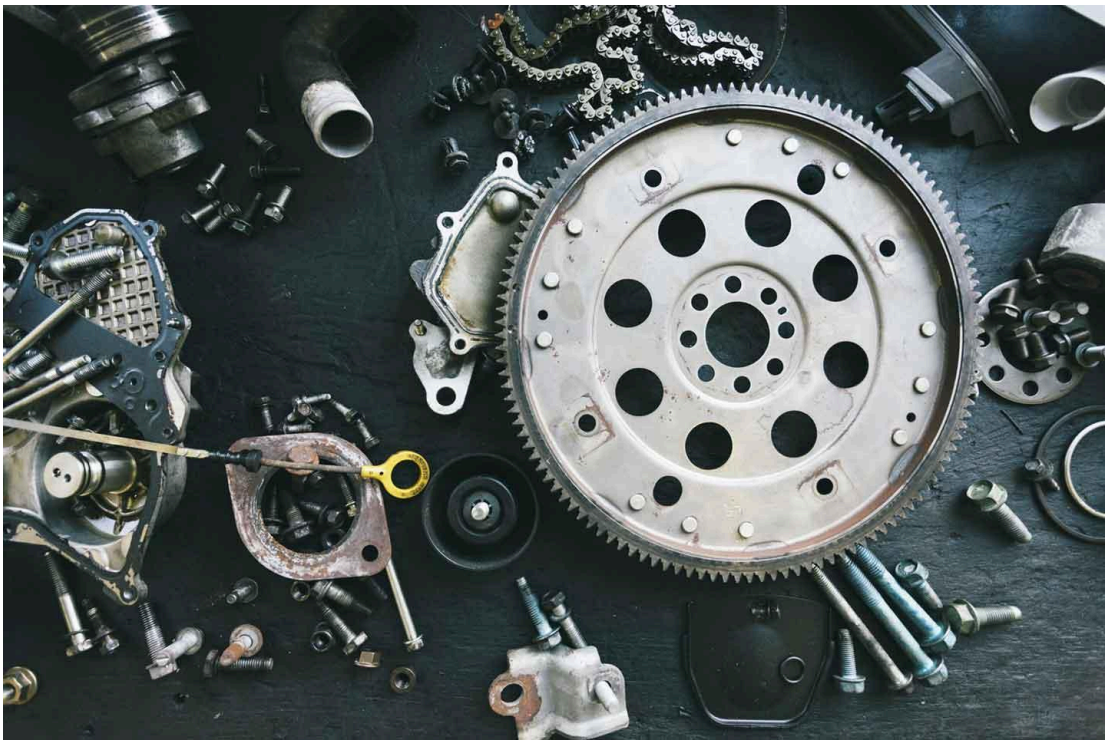
Ionic v8

DateTime



Progresa
Una formación de futuro

ÍNDICE

INTRODUCCIÓN

Nos muestra una interfaz de calendario y una rueda del tiempo, haciendo más fácil para los usuarios seleccionar horas y fechas.

Históricamente, manejar valores de fecha y hora en JavaScript, o incluso dentro de los inputs de HTML, siempre ha sido un desafío. Específicamente, el objeto Date de JavaScript es notoriamente difícil de analizar correctamente las cadenas de fecha y hora o de formatear los valores de fecha y hora. Aún peor es cómo diferentes navegadores y versiones de JavaScript analizan diferentes cadenas de fecha y hora de manera diferente, especialmente por localidad.

Afortunadamente, el input de fecha y hora de Ionic Framework ha sido diseñado para que los desarrolladores puedan evitar las trampas comunes, permitiendo a los desarrolladores manipular fácilmente los valores de fecha y hora y brindar al usuario un sencillo selector de fecha y hora para una excelente experiencia de usuario.

Formato de fecha y hora ISO 8601: YYYY-MM-DDTHH:mmZ

Ionic Framework utiliza el formato de fecha y hora ISO 8601 para su valor. El valor es simplemente una cadena, en lugar de usar el objeto Date de JavaScript. Usar el formato de fecha y hora ISO facilita la serialización y el análisis dentro de objetos JSON y bases de datos.

A continuación se muestran algunos ejemplos de formatos ISO 8601 que se pueden usar con ion-datetime:

Descripción	Format	Datetime Value Example
Año	YYYY	1994
Año y mes	YYYY-MM	1994-12
Fecha completa	YYYY-MM-DD	1994-12-15
Fecha y hora	YYYY-MM-DDTHH:mm	1994-12-15T13:47
Fecha y hora UTC	YYYY-MM-DDTHH:mm:ssZ	1994-12-15T13:47:20Z
Fecha desfasada	YYYY-MM-DDTHH:mm:ssTZD	1994-12-15T13:47:20+05:00
Horas y minutos	HH:mm	13:47

Hay que tener en cuenta que el año es siempre de cuatro dígitos, los milisegundos (si se agregan) son siempre de tres dígitos y todos los demás son siempre de dos dígitos. Por lo tanto, el número que representa enero siempre tiene un cero a la izquierda, como 01. Además, la hora está siempre en el formato de 24 horas, por lo que 00 es 12am en un reloj de 12 horas, 13 significa 1pm y 23 significa 11pm.

La documentación del Datetime está en la página:

https://ionicframework.com/docs/api/datetime

Preparación

Componente

Para empezar a trabajar con el datetime vamos a necesitar una página donde usarlo. Creamos la página en la **terminal**:

```
ionic g page pages/Datetime --spec=false
```

Icono

```
 addIcons({grid, … , searchOutline, calendarNumber});
}
```

Json

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

```
,{
 "nombre": "Datetime",
 "ruta": "/datetime",
 "icono": "calendar-number"
}
```

Dependencia

Además vamos a tener que instalar un paquete de configuraciones de formato de fechas. En la **terminal**:

npm i date-fns

Y en el tsconfig.json modificamos

```
"compilerOptions": {
 …
 "module": "es2020",
 "lib": ["es2018", "dom", "esnext"],
 "useDefineForClassFields": false
},
```

Ejemplo

Lógica

Vamos a empezar haciendo unas configuraciones en nuestro **datetime.page.ts**:

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { IonContent, IonDatetime, IonHeader, IonItem, IonLabel, IonModal, IonSegment, IonSegmentButton, IonText, IonTitle, IonToolbar } from '@ionic/angular/standalone';
import {format, parseISO} from "date-fns";

@Component({
 selector: 'app-datetime',
 templateUrl: './datetime.page.html',
 styleUrls: ['./datetime.page.scss'],
 standalone: true,
   imports: [IonContent, IonHeader, IonTitle, IonToolbar, CommonModule, FormsModule, IonSegment, IonSegmentButton, IonLabel, IonItem, IonText, IonModal, IonDatetime]
})
export class DatetimePage {
  modes = ['date', 'date-time', 'month', 'month-year', 'time', 'time-date', 'year'];
 selectedMode = 'date';
 dateValue = format(new Date(), 'yyyy-MM-dd') + 'T11:00:00.000Z';
 formattedString = '';

 constructor() {
   this.setToday();
 }
 setToday() {
   this.formattedString = (format(parseISO(
       format(new Date(),
         'yyyy-MM-dd') + 'T11:00:00.000Z'),
     'HH:mm, MMM d, yyyy'));
 }
```

```
 dateChanged(value: any) {
    this.dateValue = value;
      this.formattedString = format(parseISO(value), 'HH:mm, MMM d,
yyyy');
 }
}
```

VISUALIZACIÓN

Ahora que tenemos la parte lógica, en **datetime.page.html** vamos crear la parte visual:

```
<ion-header>
 <ion-toolbar color="primary">
   <ion-title class="ion-text-center">Datetime</ion-title>
 </ion-toolbar>
 <ion-toolbar>
   <ion-segment [(ngModel)]="selectedMode" scrollable="true">
     @for (mode of modes;track $index){
       <ion-segment-button [value]="mode">
         <ion-label>{{mode}}</ion-label>
       </ion-segment-button>
     }
   </ion-segment>
 </ion-toolbar>
</ion-header>

<ion-content>

 <ion-item id="open-modal">
   <ion-label>Date</ion-label>
   <ion-text slot="end">
     {{ formattedString }}
   </ion-text>
 </ion-item>

 <ion-modal trigger="open-modal">
   <ng-template>
     <ion-content>
       <ion-datetime
```

```
          #datetime
          [presentation]="selectedMode"
          size="cover"
          first-day-of-week="1"
          (ionChange)="dateChanged(datetime.value)"
          showDefaultButtons="true"
          [value]="dateValue">
        </ion-datetime>
      </ion-content>
    </ng-template>
  </ion-modal>
</ion-content>
```

Estilos

Y unos estilos para mejorar el aspecto visual. En **datetime.page.scss**:

```
ion-modal{
 --background: rgba(44,39,45,0.2);

 &::part(content) {
   backdrop-filter: blur(6px);
 }

 ion-content{
   --background: transparent;

   --padding-top: 25vh;
   --padding-start: 20px;
   --padding-end: 20px;

   ion-datetime {
     border-radius: 8px;
   }
 }
}
```

Explicación

```
dateValue = format(new Date(), 'yyyy-MM-dd') + 'T11:00:00.000Z';
```

Establecemos una variable donde guardamos la fecha:

- yyyy: Año con formato de 4 dígitos.
- MM: Mes con formato de 2 dígitos.
- dd: Días con formato de 2 dígitos.

```
<ion-datetime
        #datetime
        [presentation]="selectedMode"
        size="cover"
        first-day-of-week="1"
        (ionChange)="dateChanged(datetime.value)"
        showDefaultButtons="true"
        [value]="dateValue">
```

`#datetime`: Es la forma de identificar este datetime, para sacar el valor después.

`[presentation]="selectedMode"`: Para decir que tipo de calendario vamos a mostrar.

`first-day-of-week="1"`: El día de la semana en el que queremos que comience el calendario.

`showDefaultButtons="true"`: Para mostrar los botones por defecto del calendario.

`(ionChange)="dateChanged(datetime.value)"`: Evento que se activa al cambiar el calendario.

`[value]="dateValue">`: El valor inicial del calendario.

```
&::part(content) { backdrop-filter: blur(6px); }
```

Para emborronar el fondo.

Resaltado de Fechas Específicas

Utilizando la propiedad **highlightedDates**, los desarrolladores pueden estilizar fechas particulares con texto personalizado o colores de fondo. Esta propiedad puede definirse como una matriz de fechas y sus colores, o como una función de devolución de llamada que recibe una cadena ISO y devuelve los colores a utilizar.

Al especificar colores, se puede usar cualquier formato de color CSS válido. Esto incluye códigos hexadecimales, rgba, variables de color, etc.

Para mantener una experiencia de usuario consistente, el estilo de las fechas seleccionadas siempre sobrescribirá los resaltados personalizados.

Esta propiedad solo es compatible cuando **preferWheel="false"** y se utiliza una presentation de **"date", "date-time" o "time-date"**.

En **datetime.page.html** vamos a añadir un nuevo datepicker con una colección de fechas marcadas:

```
</ion-modal>

 <ion-datetime presentation="date"
[highlightedDates]="highlightedDates"></ion-datetime>

</ion-content>
```

Y la función que añade las fechas señaladas en **datetime.page.ts**:

```
formattedString = '';

highlightedDates = (isoString: any) => {

 const date = new Date(isoString);

 const utcDay = date.getUTCDate();

 // Coloreamos los días múltiplos de 5 de rosa con texto morado

 if (utcDay % 5 === 0) {

   return {

     textColor: '#800080',

     backgroundColor: '#ffc0cb',

   };

 }
```

```
 // Coloreamos los días múltiplos de 3 de secondary con texto
secondary contrast
 if (utcDay % 3 === 0) {
   return {
     textColor: 'var(--ion-color-secondary-contrast)',
     backgroundColor: 'var(--ion-color-secondary)',
   };
 }
 return undefined;
};


constructor() {
 this.setToday();
}
```

Y por último estilizamos el datepicker en **datetime.page.scss**:

```
ion-datetime {
 margin-left: auto;
 margin-right: auto;
 --background: #e0ecfd;
 --background-rgb: 224, 236, 253;

 border-radius: 16px;
}
```