

# Laboratorio – Operaciones diversas con Angular

## 1. Display JSON Data in Table Tutorial

Angular muestra los datos del archivo JSON en una tabla. Un archivo JSON es un archivo útil que ayuda a almacenar estructuras de datos y objetos simples en formato JSON, típicamente conocido como JavaScript Object Notation.

Idealmente, es un formato de intercambio de datos común, básicamente utilizado para transferir datos entre una aplicación web y un servidor.

En este laboratorio, mencionaremos el procedimiento para mostrar datos de un archivo JSON en una tabla de Angular. Para lograr esta función, usted aprenderá cómo leer un archivo json en angular 12 a la inversa mostrar los datos json en una tabla HTML.

Una tabla HTML es beneficiosa para organizar la información o los datos, normalmente en filas y columnas o probablemente en una estructura compleja más amplia. Las tablas son ampliamente adoptadas en la investigación, el análisis de datos, etc.

En esta guía, crearás una aplicación básica de Angular, crearás un archivo de datos JSON en Angular e implementarás los datos json del archivo json a la tabla HTML.

Con la ayuda del esquema de Angular CLI, instalar la aplicación angular

```
ng new ng-demo
```

Entra en la carpeta del proyecto:

```
cd ng new ng-demo
```

Ejecuta el comando para instalar la última versión de Bootstrap en Angular.

```
npm install bootstrap --save
```

A continuación, añade la ruta CSS de Bootstrap en el archivo angular.json para habilitar el estilo.

```
"styles": [  
  "node_modules/bootstrap/dist/css/bootstrap.min.css",  
  "src/styles.scss"  
]
```

### Create JSON Data File

En este paso, tiene que crear un archivo users.json; además, debe añadir los objetos json proporcionados para crear un archivo json.

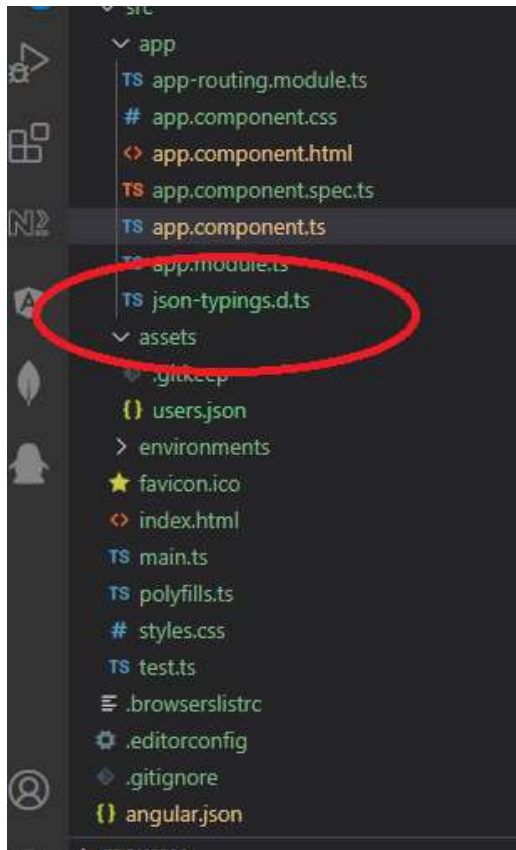
Abrir y acceder al código en src/app/assets/users.json file.

```
[{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz"
},
{
  "id": 2,
  "name": "Ervin Howell",
  "username": "Antonette",
  "email": "Shanna@melissa.tv"
},
{
  "id": 3,
  "name": "Clementine Bauch",
  "username": "Samantha",
  "email": "Nathan@yesenia.net"
},
{
  "id": 4,
  "name": "Patricia Lebsack",
  "username": "Karianne",
  "email": "Julianne.OConner@kory.org"
},
{
  "id": 5,
  "name": "Chelsey Dietrich",
  "username": "Kamren",
  "email": "Lucio_Hettinger@annie.ca"
}
```

```
},
{
  "id": 6,
  "name": "Mrs. Dennis Schulist",
  "username": "Leopoldo_Corkery",
  "email": "Karley_Dach@jasper.info"
},
{
  "id": 7,
  "name": "Kurtis Weissnat",
  "username": "Elwyn.Skiles",
  "email": "Telly.Hoeger@billy.biz"
},
{
  "id": 8,
  "name": "Nicholas Runolfsson",
  "username": "Maxime_Nienow",
  "email": "Sherwood@rosamond.me"
},
{
  "id": 9,
  "name": "Glenna Reichert",
  "username": "Delphine",
  "email": "Chaim_McDermott@dana.io"
},
{
  "id": 10,
  "name": "Clementina DuBuque",
  "username": "Moriah.Stanton",
  "email": "Rey.Padberg@karina.biz"
```

```
}  
]
```

Por defecto, Angular no lee el archivo JSON en la aplicación. Así que tenemos que hacer algunas cosas extra para eso. Así que vamos a crear un archivo llamado 'json-typings.d.ts' dentro de la carpeta de la aplicación del proyecto. Añade el siguiente código en él.



```
declare module "*.json" {  
  const value: any;  
  export default value;  
}
```

## Create User Interface

En el paso anterior, hemos creado un archivo JSON, ahora entra en el archivo app.component.ts, importar el archivo UsersJson y crea la interfaz USERS.

```
import { Component } from '@angular/core';  
import UsersJson from './users.json';
```

```
interface USERS {  
    id: Number;  
    name: String;  
    username: String;  
    email: String;  
}
```

```
@Component({  
    selector: 'app-root',  
    templateUrl: './app.component.html',  
    styleUrls: ['./app.component.scss']  
})
```

```
export class AppComponent {  
  
    Users: USERS[] = UsersJson;  
  
    constructor(){  
        console.log(this.Users);  
    }  
}
```

## Create Bootstrap Table

En este paso, tienes que usar el componente UI de la tabla de bootstrap para mostrar los datos del archivo JSON.

Ahora, abra el archivo app.component.html, añada todo el código dado dentro del archivo angular html.

```
<div class="container mt-5">
```

```
<h2>Angular Display Data from Json File Example</h2>
```

```

<table class="table table-striped">
  <thead>
    <tr>
      <th>Id</th>
      <th>Name</th>
      <th>Username</th>
      <th>Email</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let user of Users">
      <td>{{ user.id }}</td>
      <td>{{ user.name }}</td>
      <td>{{ user.username }}</td>
      <td>{{ user.email }}</td>
    </tr>
  </tbody>
</table>

```

## Update tsconfig JSON

Antes de iniciar la aplicación, tiene que modificar su archivo tsconfig.json, definir el resolveJsonModule y el esModuleInterop dentro del objeto compilerOptions.

```

{
  "compileOnSave": false,
  "compilerOptions": {
    "resolveJsonModule": true,
    "esModuleInterop": true,

    ...

    ...

```

## Start Angular App

Ahora, puedes abrir el terminal y empezar a escribir el comando dado y pulsar enter para iniciar la aplicación angular:

```
ng serve
```

## Angular Display Data from Json File Example

| Id | Name                     | Username         | Email                     |
|----|--------------------------|------------------|---------------------------|
| 1  | Leanne Graham            | Bret             | Sincere@april.biz         |
| 2  | Ervin Howell             | Antonette        | Shanna@melissa.tv         |
| 3  | Clementine Bauch         | Samantha         | Nathan@yesenia.net        |
| 4  | Patricia Lebsack         | Karianne         | Julianne.OConner@kory.org |
| 5  | Chelsey Dietrich         | Kamren           | Lucio_Hettinger@annie.ca  |
| 6  | Mrs. Dennis Schulist     | Leopoldo_Corkery | Karley_Dach@jasper.info   |
| 7  | Kurtis Weissnat          | Elwyn.Skiles     | Telly.Hoeger@billy.biz    |
| 8  | Nicholas Runolfsdottir V | Maxime_Nienow    | Sherwood@rosamond.me      |
| 9  | Glenna Reichert          | Delphine         | Chaim_McDermott@dana.io   |
| 10 | Clementina DuBuque       | Moriah.Stanton   | Rey.Padberg@karina.biz    |

## 2. Angular URL Validation using Regular Expression Tutorial

En esta guía, compartiremos cómo validar una URL en la aplicación Angular 12 utilizando la expresión regular.

Si usted no sabe cómo agregar la validación de la URL utilizando el patrón regex en la aplicación angular. ¡No te preocupes! Vamos a explicar cómo crear un formulario con entrada de texto utilizando los formularios reactivos en angular. Este campo de entrada sólo aceptará la URL.

Después de completar esta guía, usted tendrá un conocimiento profundo de la validación de patrones de URL en angular. Este pequeño ejemplo funcionará adecuadamente con casi todas las versiones de angular, ya sea 8,9,10,11, o 12.

Se comenzara el ejemplo creando el proyecto, por ello, se ejecuta el siguiente comando.

```
ng new ng-demo
```

Después de ejecutar el comando sugerido, se generará un proyecto esqueleto dentro de la carpeta ng-demo con un montón de archivos.

Accesar al directorio del proyecto de la aplicación.

```
cd ng-demo
```

### Add Reactive Forms Module

En el siguiente paso, es necesario importar el módulo de formularios reactivos, por lo tanto, ir a src/app/app.module.ts y añadir el código proporcionado en el archivo.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
```

```
import { ReactiveFormsModule, FormsModule } from
 '@angular/forms';
```

```
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
```



```

    BrowserModule,
    AppRoutingModule,
    ReactiveFormsModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})

```

```
export class AppModule { }
```

## Update TypeScript Template

En este paso, tienes que importar FormBuilder, FormGroup, Validators de '@angular/forms', luego definir el formulario usando el FormGroup y usar el patrón del número de móvil usando el regex y vincularlo al método submit.

Entonces, abre y actualiza el código en el archivo *src/app/app.component.ts*.

```

import { Component } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from
 '@angular/forms';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})

export class AppComponent {

  public myForm: FormGroup;
  constructor(private formBuilder: FormBuilder) {
    this.myForm = formBuilder.group({
      url: ['', [Validators.required,
Validators.pattern('(https?://)?([\\da-z.-]+)\\.([a-
z.]{2,6})[/\\w .-]*/?')]]
    })
  }
}

```

```

get m(){
  return this.myForm.controls;
}
onSubmit(){
  console.log(this.myForm.value);
}
}

```

## Update HTML Template

En este paso tienes que actualizar la plantilla HTML, abrir el archivo *src/app/app.component.html*, crear el formulario usando la directiva reactive form, también definir la validación requerida con la validación del patrón regex de la url..

```

<div class="container mt-5">

  <h2>Angular Pattern URL Validation Example</h2>

  <form [formGroup]="myForm" (ngSubmit)="onSubmit()" novalidate>

    <div class="form-group">

      <label>Enter URL</label>

      <input type="text" formControlName="url" class="form-control mb-2">

      <div *ngIf="m.url.touched && m.url.invalid" class="alert alert-danger">

        <div *ngIf="m.url.errors?.required">Please provide url</div>

        <div *ngIf="m.url.errors?.pattern">Please provide valid url</div>

      </div>

    </div>

    <div class="d-grid mt-3">

      <button class="btn btn-dark" [disabled]="!myForm.valid" type="submit">Store</button>

    </div>

```

```
</form>
</div>
```

## Start Angular App

Ahora, debe iniciar el servidor de desarrollo de aplicaciones utilizando el comando dado, asegúrese de ejecutar el siguiente comando.

```
ng serve
```

Abra el navegador y utilice la url dada para ver la aplicación.

`http://localhost:4200`

## Angular Pattern URL Validation Example

Enter URL

Please provide valid url

Store

## 3. How to Export Data to Excel File in Angular Application

Mientras se trabaja con las aplicaciones web, a menudo tenemos que exportar datos en pdf, excel, JSON, etc para generar facturas, informes y otros datos analíticos para el usuario. Ya he publicado un artículo para generar archivos PDF en la aplicación Angular

En este artículo, vamos a ver cómo exportar datos en la aplicación Angular.

### Step 1: Create an Angular app using Angular CLI

Ejecute el siguiente comando para crear una nueva aplicación Angular.

```
ng new angular-app
```

Una vez que la nueva aplicación creada con éxito, / ahora se mueven a la carpeta del proyecto con el comando `cd./angular-app`

### Step 2: Installed needed packages

En la tarea requerida, necesitamos instalar el módulo npm `xlsx`. Instálelo después de escribir el siguiente comando en el terminal.

```
npm i xlsx --save
```

### Step 3: Create a Dummy list of users

Aquí usaremos una lista ficticia de usuarios, en el escenario real, los datos se obtendrán del servidor.

Añade el siguiente código en el archivo `app.component.ts`.

```
userList = [  
  
  {  
  
    "id": 1,  
  
    "name": "Leanne Graham",  
  
    "username": "Bret",  
  
    "email": "Sincere@april.biz"  
  
  },  
  
  {  
  
    "id": 2,  
  
    "name": "Ervin Howell",  
  
    "username": "Antonette",  
  
    "email": "Shanna@melissa.tv"  
  
  },  
  
  {  
  
    "id": 3,  
  
    "name": "Clementine Bauch",  
  
    "username": "Samantha",  
  
    "email": "Nathan@yesenia.net"
```

```

    },
    {
      "id": 4,
      "name": "Patricia Lebsack",
      "username": "Karianne",
      "email": "Julianne.OConner@kory.org"
    },
    {
      "id": 5,
      "name": "Chelsey Dietrich",
      "username": "Kamren",
      "email": "Lucio_Hettinger@annie.ca"
    }
  ]

```

#### Step 4: Update listing with the export button

Ahora vamos a actualizar la parte de la plantilla. Ponga el siguiente código en el archivo `app.component.html` con el siguiente código.

```

<div style=" margin: auto; width: 50%;">
  <button (click)="exportexcel()">Export to Excel</button>

  <table id="excel-table">
    <tr>
      <th>Id</th>
      <th>Name</th>
      <th>Username</th>

```

```

        <th>Email</th>
    </tr>
    <tr *ngFor="let item of userList">
        <td>{{item.id}}</td>
        <td>{{item.name}}</td>
        <td>{{item.username}}</td>
        <td>{{item.email}}</td>
    </tr>
</table>
</div>

```

### Step 5: Add method in app.component.ts file to export excel file

```

import { Component } from '@angular/core';
import * as XLSX from 'xlsx';

```

```

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'angular-app';
  fileName= 'ExcelSheet.xlsx';
  userList = [
    {
      .....
      .....
    }
  ]
  exportexcel(): void
  {

```

```

/* pass here the table id */
let element = document.getElementById('excel-table');
const ws: XLSX.WorkSheet =XLSX.utils.table_to_sheet(element);

/* generate workbook and add the worksheet */
const wb: XLSX.WorkBook = XLSX.utils.book_new();
XLSX.utils.book_append_sheet(wb, ws, 'Sheet1');

/* save to file */
XLSX.writeFile(wb, this.fileName);

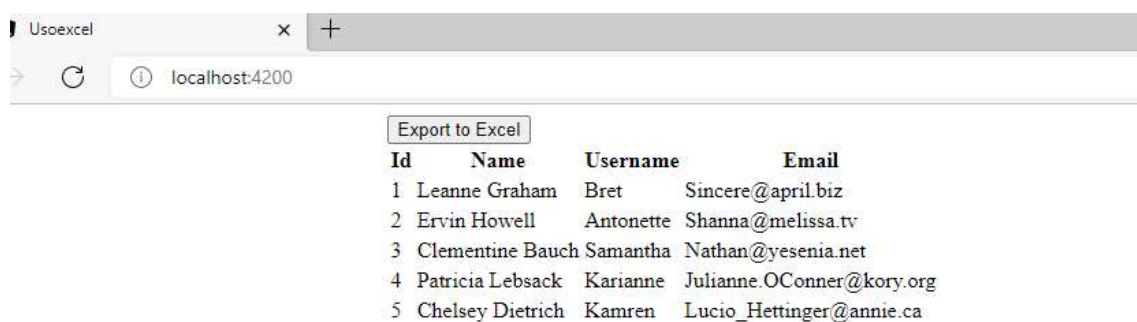
}
}

```

## Step 6: Run the App

Finalmente, hemos terminado con todos los pasos necesarios, ejecuta la aplicación con el siguiente comando en la terminal

```
npm start
```



The screenshot shows a web browser window with a table of user data. The table has columns: Id, Name, Username, and Email. The data is as follows:

| Id | Name             | Username  | Email                     |
|----|------------------|-----------|---------------------------|
| 1  | Leanne Graham    | Bret      | Sincere@april.biz         |
| 2  | Ervin Howell     | Antonette | Shanna@melissa.tv         |
| 3  | Clementine Bauch | Samantha  | Nathan@yesenia.net        |
| 4  | Patricia Lebsack | Karianne  | Julianne.OConner@kory.org |
| 5  | Chelsey Dietrich | Kamren    | Lucio_Hettinger@annie.ca  |

An 'Export to Excel' button is located above the table. An arrow points from this button to a download menu that shows 'ExcelSheet.xlsx' with a download icon and a link to 'Abrir archivo'.

Below the browser window, the same data is shown in an Excel spreadsheet format. The columns are labeled 'Id', 'Name', 'Username', and 'Email'. The data rows are identical to the table above.

## 4. How to Create/Generate QR Code in Angular 12 Application

En este código, se aprenderá a generar o crear un código QR en la aplicación Angular 12 tomando la ayuda de la biblioteca angular2-qrcode.

*El angular2-qrcode es un componente que puedes integrar fácilmente en tu proyecto. Se basa en qrious para generar códigos QR*

En las aplicaciones modernas, muchos trabajos se realizan mediante el escaneo de los códigos QR, ya sea realizar pagos, compartir datos, obtener detalles del producto o recuperar la ubicación.

Sólo hay que sacar el móvil del bolsillo y mantenerlo frente al escáner y el resto del trabajo lo hace automáticamente la aplicación. No está mal decir que los códigos QR se han convertido en una parte esencial de nuestras vidas.



En cuanto a la compatibilidad, los códigos QR son compatibles con las principales plataformas como iOS, Android y Windows.

*Un código QR (código de respuesta rápida) es un tipo de código de barras matricial (o bidimensional) diseñado por primera vez en 1994 para la industria automovilística japonesa. Un código de barras es una etiqueta óptica legible por máquina que contiene información sobre el artículo al que se adhiere. En la práctica, los códigos QR suelen contener datos de un localizador, identificador o rastreador que apunta a un sitio web o una aplicación.*

A continuación, se creará una aplicación Angular desde cero y se verá cómo implementar códigos QR en la aplicación Angular. Se convertirán los datos proporcionados por el usuario en un código QR.

En primer lugar, se crea la aplicación a usar:

```
ng new angular-qr-code-app-example
```

Answer some CLI questions:

```
# ? Would you like to add Angular routing? Yes
```

```
# ? Which stylesheet format would you like to use? CSS
```

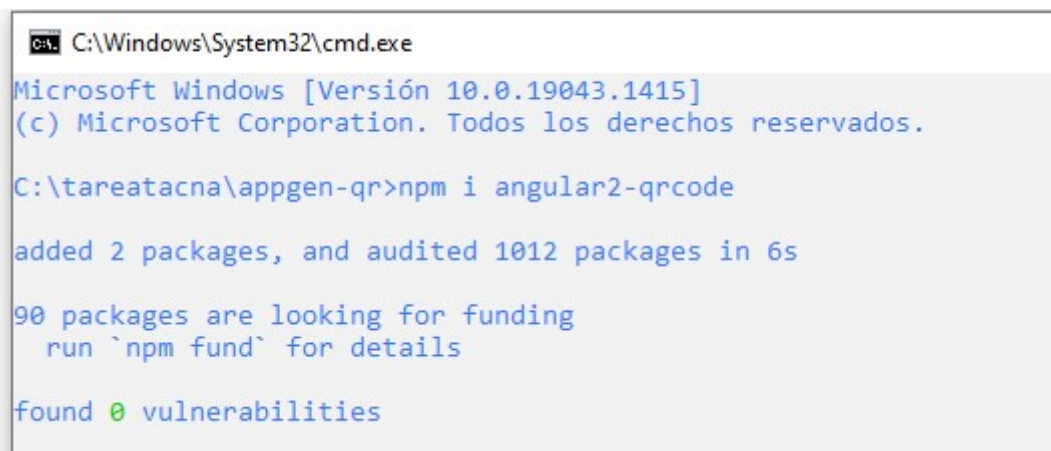
Se ingresa a la raíz del proyecto:

```
cd angular-qr-code-app-example
```

### Instalar el módulo de código QR de Angular 12

Bien, este tutorial está incompleto sin la instalación de la librería angular2-qrcode. Para instalar el paquete qrcode de Angular 2 en angular, ejecuta el siguiente comando.

```
npm install angular2-qrcode
```



```
cmd C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1415]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\tareatacna\appgen-qr>npm i angular2-qrcode

added 2 packages, and audited 1012 packages in 6s

90 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

### Importar y registrar QRCodeModule en el módulo de la aplicación

En general, para generar los códigos QR usando el módulo `angular2-qrcode`, necesitamos importar y registrar el paquete `QRCodeModule` en el archivo principal del módulo de la aplicación de Angular.

Además, importaremos el `FormsModule` para interactuar con los elementos de entrada del formulario HTML.

Añade el código en el archivo `app.module.ts`.

```
import { BrowserModule } from '@angular/platform-browser';
```

```
import { NgModule } from '@angular/core';
```

```
import { AppRoutingModuleModule } from './app-routing.module';
```

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

```
import { QRCodeModule } from 'angular2-qrcode';
```

```
import { FormsModule } from '@angular/forms'
```

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

TS app.module.ts M X

```
src > app > TS app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6
7  import { QRCodeModule } from 'angular2-qrcode';
8  import { FormsModule } from '@angular/forms'
9
10
11  @NgModule({
12    declarations: [
13      AppComponent
14    ],
15    imports: [
16      BrowserModule,
17      AppRoutingModule,
18      FormsModule,
19      QRCodeModule
20    ],
21    providers: [],
22    bootstrap: [AppComponent]
23  })
24  export class AppModule { }
25
```

## Implementar el generador de códigos QR en Angular 12

Todo gracias al componente angular2-qrcode, se puede implementar muy rápidamente el generador de códigos QR en angular con solo añadir el atributo <qr-code>.

```
<qr-code [value]=" https://larepublica.pe/" ></qr-code>
```

### Algunas propiedades del código QR

El Código QR ofrece varias propiedades que se pueden asociar a la directiva <qr-code>.

value (valor): El tipo de propiedad toma un valor de cadena y convierte la cadena en código QR; el valor por defecto es "".

size: Esta propiedad establece la altura y la anchura del componente del código QR, el tipo de propiedad es un número y el valor por defecto es 100.

level: El tipo de propiedad es String; el valor por defecto se establece en L principalmente se utiliza para el nivel de Corrección QR ('L', 'M', 'Q', 'H').

background: El tipo de prop es String; el valor por defecto es blanco. Se utiliza principalmente para configurar el color de fondo.

backgroundAlpha: Se utiliza para establecer la opacidad del fondo, definida en forma numérica, y el valor por defecto es 1.0.

foreGround: Se utiliza para ajustar el color de primer plano, el tipo de propiedad es String, y el valor por defecto es negro.

foregroundAlpha: Establece la opacidad del primer plano. El valor por defecto es 1.0 y se define en forma numérica.

mime: El tipo de valor es String, utilizado para configurar el tipo mime de la imagen de salida. Además, el valor por defecto es image/png.

padding: Principalmente configura el relleno en el Código QR; el número es el tipo de propiedad con el valor por defecto es null.

canvas: El tipo de valor es booleano y se utiliza para la salida de un elemento de lienzo si se establece como verdadero. Sin embargo, el valor por defecto es falso.

## Almacenar datos JSON en el Código QR

En este paso, aprenderemos a incluir datos JSON en el componente QR Code utilizando el método `JSON.stringify()` y el componente qr-code, se coloca el siguiente código en *app.component.ts*.

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

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

```
export class AppComponent {

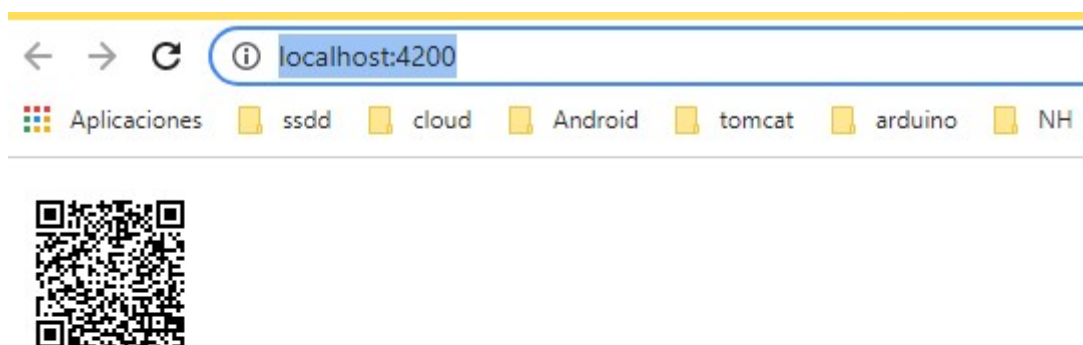
  data = [{
    'name': 'Jorge Guerra',
    'profile': 'Software Developer',
    'email': 'jguerra@gmail.com',
    'hobby': 'coding'
  }]
  dataToString = JSON.stringify(this.data);
```

}

```
src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'appgen-qr';
10
11    data = [{
12      'name': 'Jorge Guerra',
13      'profile': 'Software Developer',
14      'email': 'jguerra@gmail.com',
15      'hobby': 'coding'
16    }]
17
18    dataToString = JSON.stringify(this.data);
19
20  }
21
```

Se puede incluir el código en el archivo *app.component.html*.

`<qr-code [value]="dataToString"></qr-code>`



Probando con un escaner QR:



### Generar QR-Code dinámico en Angular

Aquí está la versión rápida integrada con las propiedades de los códigos QR para actualizar dinámicamente los valores del componente QR Code en Angular.

Se coloca el siguiente código en el archivo *app.component.html*.

```
<h2>Dynamically Update QR Code Values</h2>
<div>QR Code: {{ qrCodeVal }}</div>
<div>Level: {{ level }}</div>
<div>Width: {{ width }}</div>
<br />
<br />

<div>
  <qr-code [value]="qrCodeVal" [size]="width" [level]="level"></qr-code>
</div>

<h4>Change QR Code Info</h4>
<label>
  Update name string: <input [(ngModel)]="qrCodeVal"
  placeholder="name" />
```

```

</label>
<br />
<br />

<button (click)="updateQrInfo('John Doe')">Update to "John Doe"</button>
<button (click)="updateQrInfo('Ryan Duff')">Update to "Ryan Duff"</button>
<button (click)="updateQrInfo('Jerry Maguire')"> update to "Jerry Maguire"</button>

<br />
<br />
<strong>Update Width</strong>:
<button (click)="updateWidth(500)">500</button>
<button (click)="updateWidth(400)">400</button>
<button (click)="updateWidth(300)">300</button>
<button (click)="updateWidth(200)">200</button>

<br />
<br />
<strong>Update Level</strong>:
<button (click)="updateLevel('L')">L</button>
<button (click)="updateLevel('M')">M</button>
<button (click)="updateLevel('Q')">Q</button>
<button (click)="updateLevel('H')">H</button>

```

Incorpore el siguiente código en el archivo *app.component.ts*.

```

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']

```

```
})
```

```
export class AppComponent {  
  public qrCodeVal: string;  
  public level: "L" | "M" | "Q" | "H";  
  public width: number;  
  
  constructor() {  
    this.level = "L";  
    this.qrCodeVal = "QR code string value";  
    this.width = 200;  
  }  
  updateLevel(newValue: "L" | "M" | "Q" | "H") {  
    this.level = newValue;  
  }  
  updateQrInfo(newValue: string) {  
    this.qrCodeVal = newValue;  
  }  
  updateWidth(newValue: number) {  
    this.width = newValue;  
  }  
}
```

A continuación, inicie el siguiente comando para probar la aplicación:

```
ng serve --open
```

De esta manera estará completo el generador de códigos QR de Angular 12 con un ejemplo



## Actualizar dinámicamente los valores del código QR

QR Code: Jerry Maguire  
Level: Q  
Width: 300



Cambiar la información del código QR

Actualizar la cadena nombre:

Actualizar Ancho:

Actualizar Nivel:

Al correr el programa, se prueban las opciones, aquí se muestra el QR escogiendo los datos "Jerry Maguire", "300", "Q"