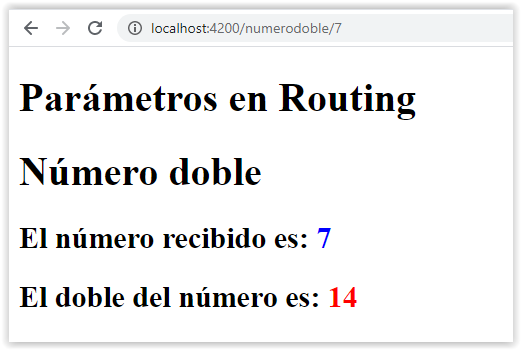
**RUTAS CON PARAMETROS ANGULAR**

* Para enviar parámetros a una ruta, debemos montar un ejemplo con Routing de angular y utilizar una determinada Sintaxis.
* Vamos a realizar una aplicación dónde enviaremos un número a la ruta y mostraremos en la página el doble del número pulsado.



Creamos un componente llamado **NumeroDoble** y lodejamos todo por defecto.

Para configurar el Router, debemos hacerlo desde el componente principal, por lo que iremos a la carpeta **app** y crearemos un nuevo archivo llamado **app.routing.ts**

**APP.ROUTING.TS**

//IMPORTAMOS LOS MODULOS DEL ROUTER DE ANGULAR

import { ModuleWithProviders } from "@angular/core";

import { Routes, RouterModule } from "@angular/router";

//AHORA DEBEMOS IMPORTAR LOS COMPONENTES QUE VAYAN A FORMAR PARTE

//DE LA NAVEGACION, ES DECIR, LOS QUE VISUALIZAREMOS

//DENTRO DE LAS ZONAS DE NUESTRO SITE

//IMPORTAMOS EL COMPONENTE DE NUMERO DOBLE

import { NumeroDobleComponent } from "./components/numero-doble/numero-doble.component";

//DEFINIMOS EL ARRAY DE RUTAS, ES DECIR, LA CONFIGURACION DE TODAS

//LAS RUTAS PARA NUESTRA APP, QUE SERA NUMERO DOBLE SIN PARAMETROS Y CON //PARAMETRO OPCIONAL.

const appRoutes: Routes = [

  { path: "numerodoble", component: NumeroDobleComponent },

  { path: "numerodoble/:numero", component: NumeroDobleComponent },

];

//AHORA EXPORTAMOS EL MODULO DEL ROUTER PARA PODER

//UTILIZARLO DENTRO DE ANGULAR

export const appRoutingProviders: any[] = [];

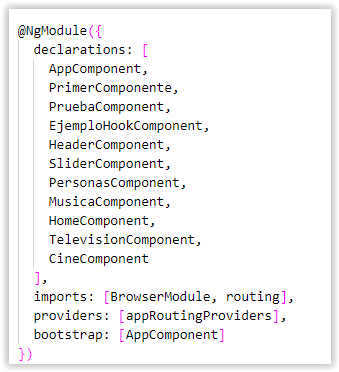
export const routing: ModuleWithProviders = RouterModule.forRoot(appRoutes);

A continuación, abrimos el módulo del componente principal de **app**, para declarar el routing que vayamos a utilizar. Abrimos el archivo **app.module.ts** y declaramos las dos constantes que hemos exportado en la clase de Routing, en la última línea.

**APP.MODULE.TS**

import { routing, appRoutingProviders } from "./app.routing";

Cargamos **routing** como módulo y **appRoutingProviders** como servicio dentro de la misma clase.



El siguiente paso será utilizar las rutas dentro de la página principal **app.component.html**

Para poder generar rutas, debemos utilizar la etiqueta **<router-outlet>**, que será la etiqueta encargada de buscar las rutas creadas en nuestro Routing.

**APP.COMPONENT.HTML**

<h1>Parámetros en Routing</h1>

<router-outlet></router-outlet>

Y podremos comprobar las rutas incluyendo los Path en la URL:

<http://localhost:4200/numerodoble>

<http://localhost:4200/numerodoble/14>



Como hemos visto, tenemos dos posibles rutas, lo que hace que el parámetro del **número** sea opcional.

Lo que haremos a continuación es modificar el código del componente **numerodoble** para capturar el valor del número recibido.

**NUMERODOBLE.COMPONENT.TS**

import { Component, OnInit } from "@angular/core";

import { Router, ActivatedRoute, Params } from "@angular/router";

@Component({

  selector: "app-numero-doble",

  templateUrl: "./numero-doble.component.html",

  styleUrls: ["./numero-doble.component.css"]

})

export class NumeroDobleComponent implements OnInit {

  constructor(private \_route: ActivatedRoute, private \_router: Router) {}

  ngOnInit(): void {

    //SE TRABAJA CON OBSERVABLE PORQUE

    //PODRIAN LLEGAR PARAMETROS QUE TARDASEN

    //EN SER RECIBIDOS

    this.\_route.params.subscribe((params: Params) => {

      console.log(params);

    });

  }

}

Y podremos visualizar los datos en console.



Ahora vamos a modificar el código del componente para tener propiedades para mostrar en la vista a partir de los parámetros.

**NUMERODOBLE.COMPONENTS.TS**

import { Component, OnInit } from "@angular/core";

import { Router, ActivatedRoute, Params } from "@angular/router";

@Component({

  selector: "app-numero-doble",

  templateUrl: "./numero-doble.component.html",

  styleUrls: ["./numero-doble.component.css"]

})

export class NumeroDobleComponent implements OnInit {

  public numero: number;

  public doble: number;

  constructor(private \_route: ActivatedRoute, private \_router: Router) {}

  ngOnInit(): void {

    //SE TRABAJA CON OBSERVABLE PORQUE

    //PODRIAN LLEGAR PARAMETROS QUE TARDASEN

    //EN SER RECIBIDOS

    this.\_route.params.subscribe((params: Params) => {

      //COMPROBAMOS QUE TENEMOS PARÁMETROS EN LA PAGINA

      if (params.numero != null) {

        //SI TENEMOS NUMERO, ASIGNAMOS LA PROPIEDAD NUMERO

        //DE LA CLASE AL VALOR DEL PARAMETRO

        //A PESAR DE QUE SEA DECLARADO COMO

        //NUMERO EN TYPE SCRIPT, SIGUE VINIENDO

        //UN STRING, POR LO QUE CONVERTIMOS

        this.numero = parseInt(params.numero);

        //REALIZAMOS LA OPERACION PARA EL DOBLE

        this.doble = this.numero \* 2;

        console.log(params);

      } else {

        console.log("No hay parámetros");

      }

    });

  }

}

**NUMERODOBLE.COMPONENTS.HTML**

<h1>Número doble</h1>

<h2>

  El número recibido es:

  <span style="color:blue"> {{ numero }}</span>

</h2>

<h2>

  El doble del número es:

  <span style="color:red"> {{ doble }}</span>

</h2>

Por último, podremos utilizar un condicional para indicar si hemos recibido parámetros o no.

<h1>Número doble</h1>

<div \*ngIf="numero">

  <h2>

    El número recibido es:

    <span style="color:blue"> {{ numero }}</span>

  </h2>

  <h2>

    El doble del número es:

    <span style="color:red"> {{ doble }}</span>

  </h2>

</div>

<div \*ngIf="!numero">

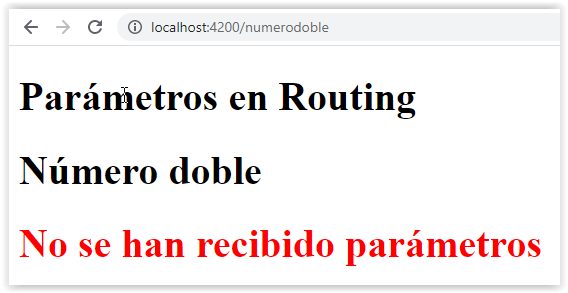
    <h1 style="color:red">

        No se han recibido parámetros

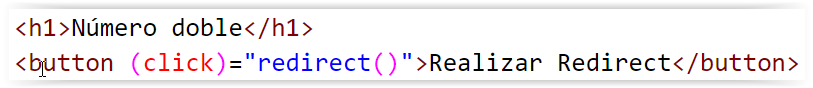
    </h1>

</div>

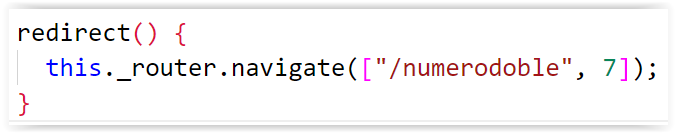
Y visualizaremos los cambios:



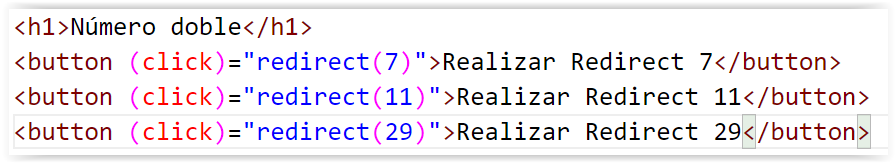
Ahora vamos a visualizar las redirecciones. Para ello, nos vamos a crear un botón y visualizaremos cómo podemos enviar datos a la página y redirigir.

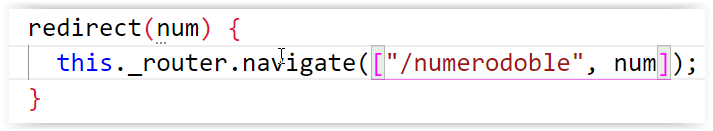


Y, en el código TS escribimos el método **redirect()** que se encargará de hacer la redirección a nuestra misma página.

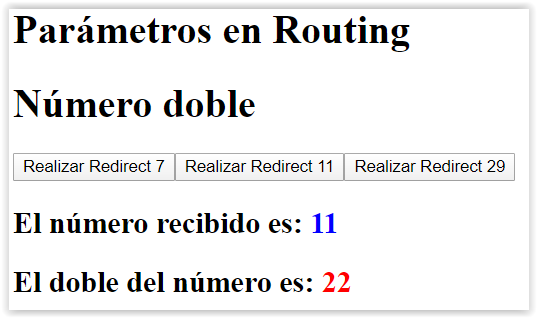


Ahora vamos a visualizar cómo enviar los parámetros a la función, que es cómo cualquier función de Javascript





Y veremos el resultado:



También podremos enviar parámetros con <a href>

