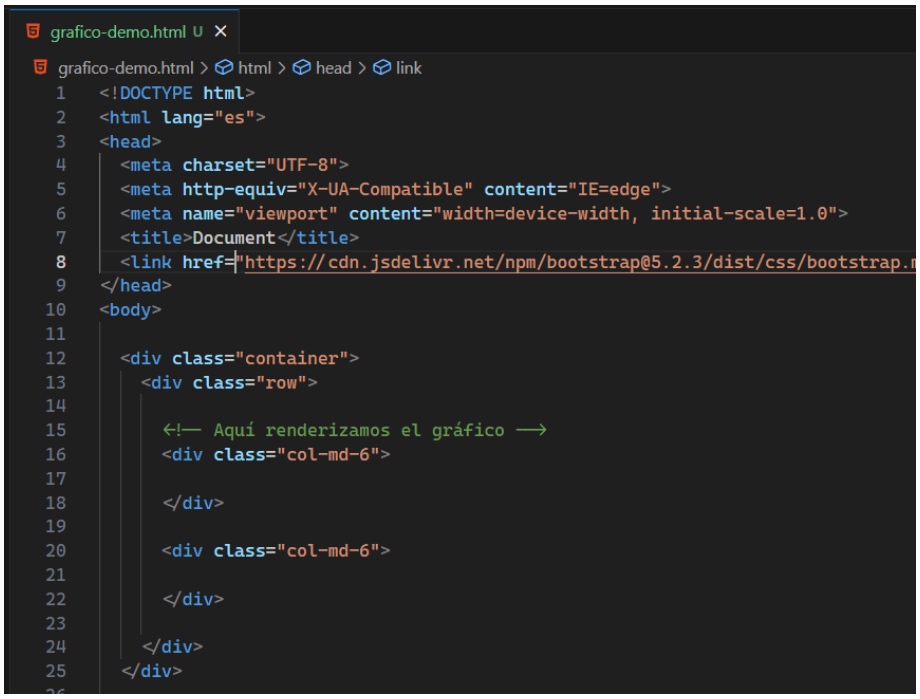
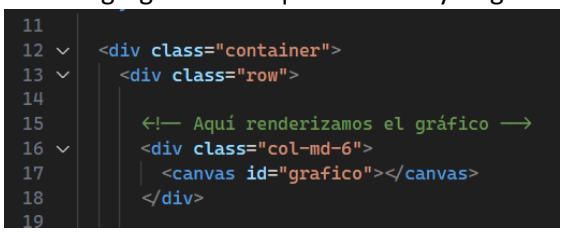
APRENDIENDO CHART JS

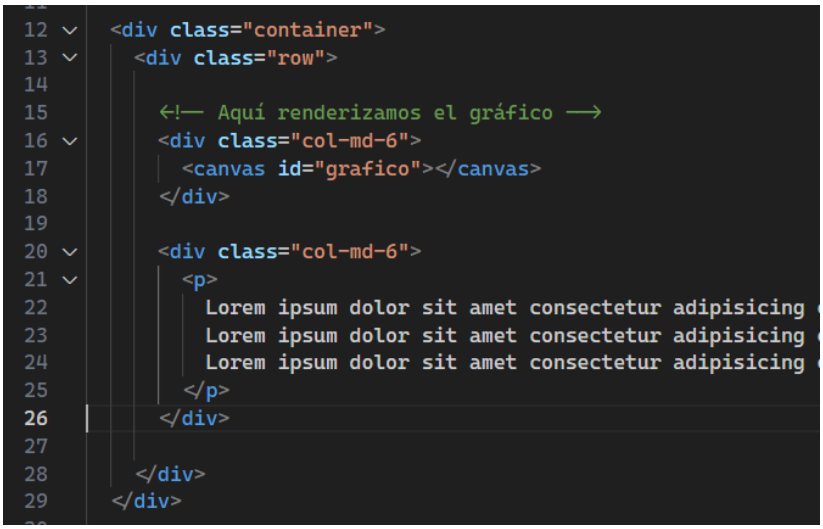
1. Diseñe un nuevo archivo HTML (**grafico-demo.html**), para este ejemplo, cree una fila y divida en dos columnas iguales (Bootstrap 5.x)



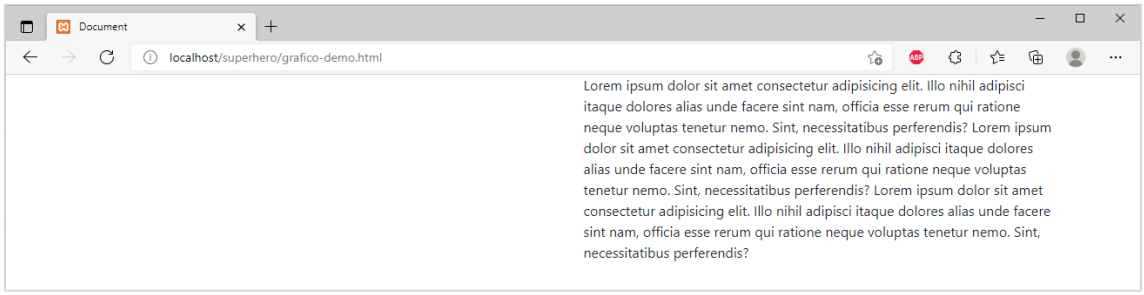
2. En la primera columna agregue una etiqueta **canvas** y asígnele un ID



3. La tercera columna es solo un espacio que puede ser utilizado para cualquier otro propósito, así que construya un párrafo con muchas líneas de prueba:



Hasta el momento debería tener



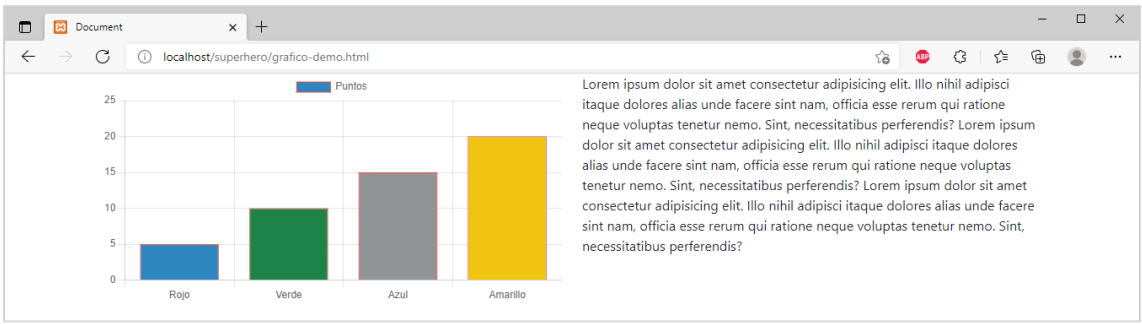
4. Incluya un bloque JavaScript, haga uso del CDN e ingrese las siguientes instrucciones:

   CDN:
   https://cdn.jsdelivr.net/npm/chart.js

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script>
  document.addEventListener("DOMContentLoaded", () => {

    //Objeto para referencias el canvas
    const lienzo = document.getElementById("grafico");

    const graficoBarras = new Chart(lienzo, {
      type: 'bar',
      data: {
        labels: ['Rojo', 'Verde', 'Azul', 'Amarillo'],
        datasets: [
          {
            borderColor: '#E74C3C',
            backgroundColor: ['#2E86C1', '#1D8348', '#909497', '#F1C40F'],
            label: 'Puntos',
            data: [5, 10, 15, 20, 25],
            borderWidth: 1
          }
        ]
      }
    });
```

Los atributos **borderColor** y **backgroundColor** NO son obligatorios. Si los incluye, puede elegir la gama de colores que desee: https://htmlcolorcodes.com/es/

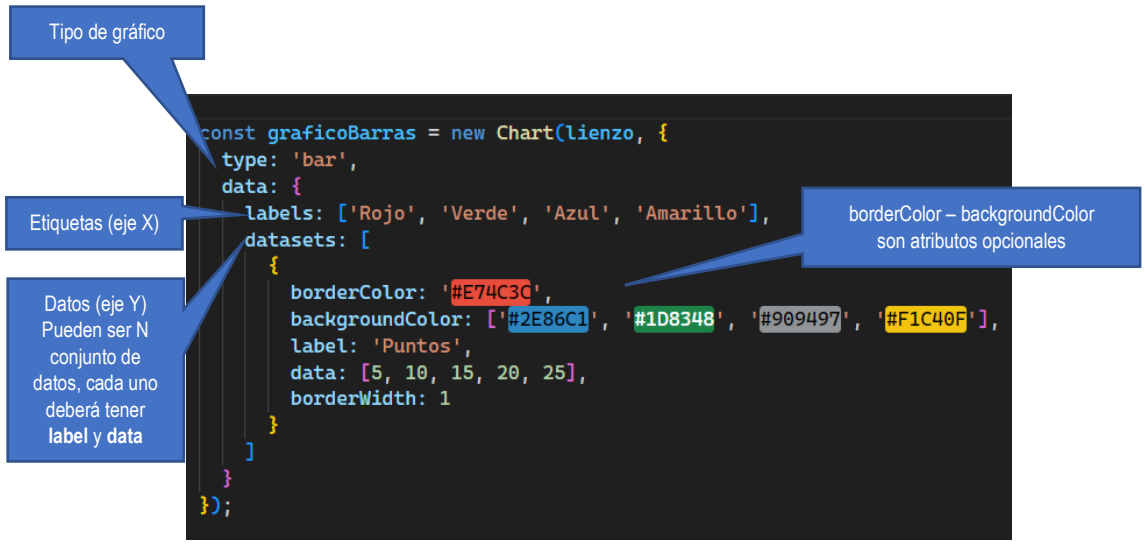
5. En el navegador



IMPORTANTE:

- Los atributos definidos con:
  - Atributo: ‘’ Cadena
  - Atributo: 1 Entero
  - Atributo: {} Objetos
  - Atributo: [] Colecciones
  - Atributo: [{}] Colección de objetos
  - Atributo: [‘’] Colección de cadenas

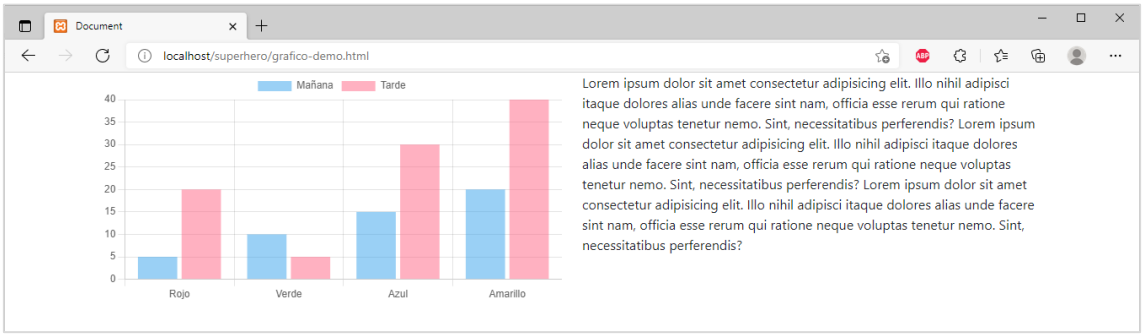
- Analizando código anterior:



6. Antes de continuar y a fin de mantener el código lo más limpio posible, eliminaré borderColor, backgroundColor y borderWidth (si Ud. Desea lo puede mantener). Para comprender mejor el concepto de datasets, agregue un nuevo set de datos:

```
<script>
  document.addEventListener("DOMContentLoaded", () => {

    //Objeto para referencias el canvas
    const lienzo = document.getElementById("grafico");

    const graficoBarras = new Chart(lienzo, {
      type: 'bar',
      data: {
        labels: ['Rojo', 'Verde', 'Azul', 'Amarillo'],
        datasets: [
          {
            label: 'Mañana',
            data: [5, 10, 15, 20, 25]
          },
          {
            label: 'Tarde',
            data: [20, 5, 30, 40, 10]
          }
        ]
      }
    });
```

7. El resultado en el navegador



ACTUALIZANDO DATOS

Vamos a copiar el archivo anterior (grafico-demo.html) y crearemos el archivo grafico-update.html, a partir de este momento, trabajaremos con este último

1. Empezaremos agregando un formulario y 4 cajas de texto, además de un botón actualizar, estos elementos se ubicarán donde anteriormente estaba el párrafo con el texto de prueba:

```
<div class="container">
  <div class="row">

    <!— Aquí renderizamos el gráfico —>
    <div class="col-md-6">
      <canvas id="grafico"></canvas>
    </div>

    <!— Ahora construiremos un formulario —>
    <div class="col-md-6">
      <form action="" autocomplete="off" class="mt-2">
        <div class="form-floating mb-3">
          <input type="number" class="form-control" placeholder="Valor rojo" id="rojo">
          <label for="rojo">Valor rojo</label>
        </div>
        <div class="form-floating mb-3">
          <input type="number" class="form-control" placeholder="Valor verde" id="verde">
          <label for="verde">Valor verde</label>
        </div>
        <div class="form-floating mb-3">
          <input type="text" class="form-control" placeholder="Valor azul" id="azul">
          <label for="azul">Valor azul</label>
        </div>
        <div class="form-floating mb-3">
          <input type="text" class="form-control" placeholder="Valor amarillo" id="amarillo">
          <label for="amarillo">Valor amarillo</label>
        </div>
      </form>
      <button type="button" id="actualizar" class="btn btn-info">Actualizar</button>
    </div>

  </div> <!— Fin row —>
</div> <!— Fin container —>
```

2. El siguiente paso será crear un objeto para manejar al botón

```
document.addEventListener("DOMContentLoaded", () => {

  //Objeto para referencias el canvas
  const btActualizar = document.querySelector("#actualizar");
  const lienzo = document.getElementById("grafico");
```

3. Agregue una nueva función que permita leer el valor de cada caja de texto, convirtiendo el valor a entero, estos constituirán un nuevo arreglo que reemplazará al existente.

```
function actualizarData(){
  const nuevoArreglo = [
    parseInt(document.querySelector("#rojo").value),
    parseInt(document.querySelector("#verde").value),
    parseInt(document.querySelector("#azul").value),
    parseInt(document.querySelector("#amarillo").value)
  ];

  //Asignamos el nuevo datos
  graficoBarras.data.datasets[0].data = nuevoArreglo;
  graficoBarras.update();
}

//Evento que dispara al método
btActualizar.addEventListener("click", actualizarData);
```

Fuera del método actualizarData, utilice el evento click del botón para desencadenar esta acción

4. Resultado: