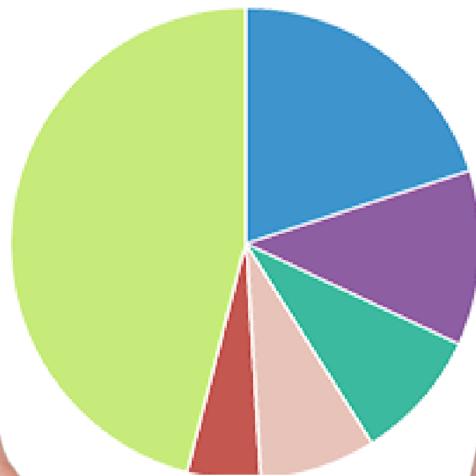


MEMORIA

LIBRERIA GRAFICOS CHART.JS



ÍNDICE

Introducción.....	2
Gráfica V1. De barras.....	2
Gráfica V2. De tarta.....	5
Gráfica V3. Cambio tipo de Gráfica.....	7
Gráfica V4. Controles Numéricos.....	11
Gráfica V5. Completa.....	17

INTRODUCCIÓN

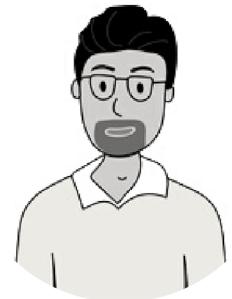


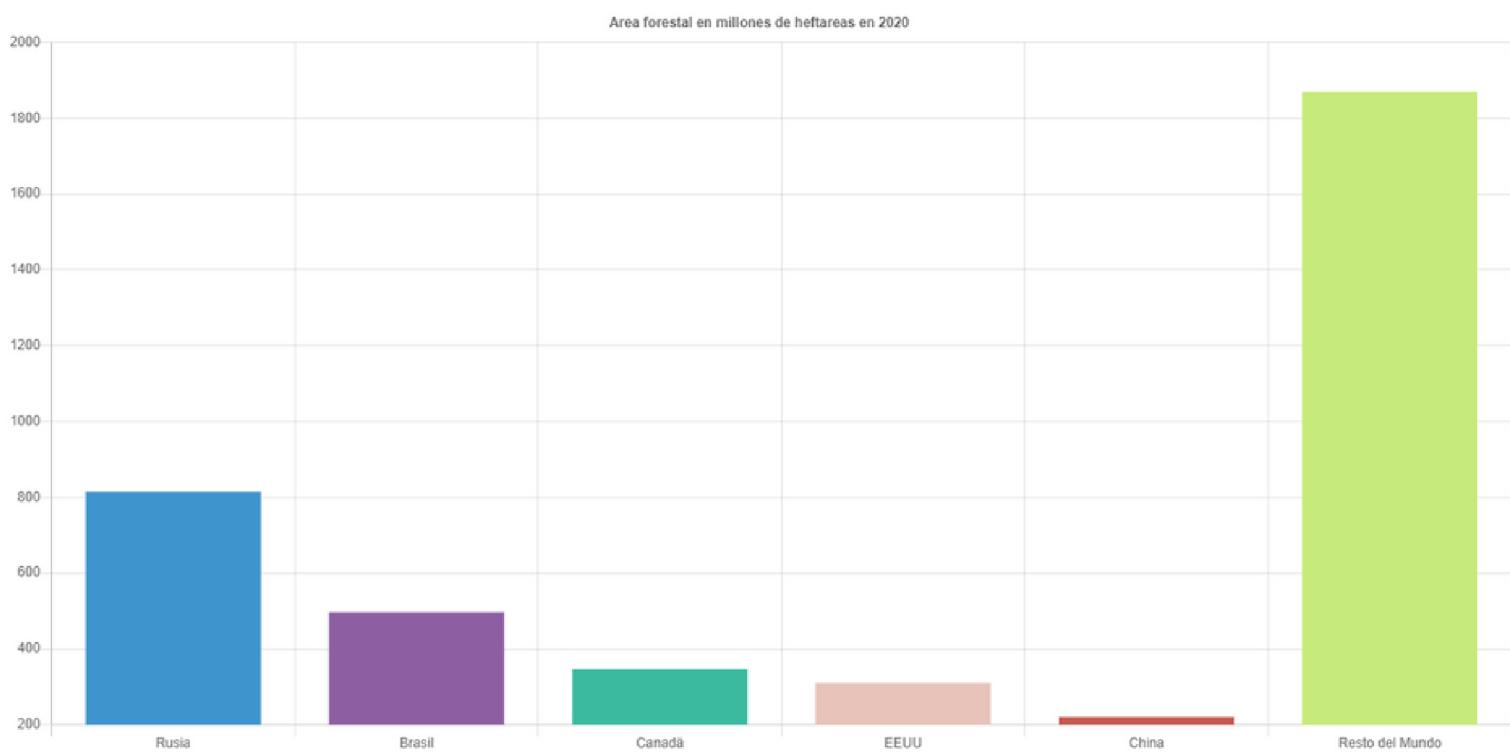
CHART.JS ES UNA LIBRERÍA POPULAR QUE SE PUEDE USAR PARA CREAR DIFERENTES TIPOS DE GRÁFICOS. LA LIBRERÍA TE PERMITE MEZCLAR DIFERENTES TIPOS DE GRÁFICOS Y TRAZAR DATOS EN ESCALAS FECHA TIEMPO, LOGARÍTMICA, O PERSONALIZADA CON FACILIDAD. LA LIBRERÍA TAMBIÉN SOPORTA ANIMACIONES QUE PUEDEN SER APLICADAS CUANDO SE CAMBIAN LOS DATOS O SE ACTUALIZAN COLORES.

MONTY SHOKEEN : COMENZANDO CON CHART.JS

VERSIÓN 1. GRÁFICA DE BARRAS

- En esta versión 1 hemos generado un sencillo gráfico de barras que se muestra al cargar la página. En él mostramos el área forestal existente en cada país en 2020 expresados en millones de hectáreas. La fuente de los datos es <https://www.fao.org/forest-resources-assessment/2020/es>

Area Forestal Hectareas (2020)



Fuente : <https://www.fao.org/forest-resources-assessment/2020/es>

Este gráfico está compuesto por este sencillo HTML junto a un también sencillo archivo de estilos .css.

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <title>Superficie Forestal</title>
7          <script src="https://cdn.jsdelivr.net/npm/chart.js@latest/dist/Chart.min.js"></script>
8          <script src="chartScript.js"></script>
9          <link rel="stylesheet" href="chartStyles.css">
10     </head>
11     <body>
12         <h1>Área Forestal Hectáreas (2020)</h1>
13         <div class="canvasContainer">
14             <canvas id="myChart"></canvas>
15         </div>
16     </body>
17     <footer>
18         <p class="source">Fuente : https://www.fao.org/forestry-resources-assessment/2020/es</p>
19     </footer>
20 </html>
```

En él dentro de la sección <head> vemos que cómo se asocian los archivos de código nativos de Chart.js, luego nuestro archivo de código llamado, "chartScript.js" y nuestro archivo de estilos css "chartStyles.css".

Más abajo tenemos un título de página <h1>, después nuestro elemento <canvas> que es el más importante, pues en él se dibujarán las gráficas, y un pie donde simplemente ponemos un texto con el origen de los datos.

A continuación mostramos el contenido también muy sencillo del archivo chartStyles.css.

En él se definen dos estilos: la clase **.canvasContainer** donde definimos las propiedades gráficas para el canvas y la clase **.source** donde definimos las propiedades gráficas para el texto de la fuente de datos.

```
.canvasContainer{
    height:70%;           /*proporción del alto de la página que ocupa nuestro canvas*/
    width: 70%;            /*proporción del ancho de la página que ocupa nuestro canvas*/
}

.source{
    font-weight: lighter;  /*grosor de letra en el letrero de origen de datos*/
    left:0px;               /*posicionamiento a la izquierda del letrero de origen de datos*/
    bottom:0px;              /*posicionamiento parte inferior del letrero de origen de datos*/
}
```

Finalmente vemos el archivo chartScript.js con el código javascript para la configuración del chart

```
1 window.onload = function() {
2     //Obtenemos el elemento canva de la interfaz.
3     let canva = document.getElementById('myChart').getContext('2d');
4
5     //Configuramos el el gráfico utilizando el elemento canva para dibujarlo en él
6     let myChart = new Chart(canva, {
7         type: 'bar',
8         data: {
9             labels: ["Rusia", "Brasil", "Canadá", "EEUU", "China", "Resto del Mundo"],
10            datasets: [
11                {
12                    label: "Area Forestal",
13                    backgroundColor: ["#3e95cd", "#8e5ea2", "#3cba9f", "#e8c3b9", "#c45850", "#c7eb7a" ],
14                    data: [815, 497, 347, 310, 220, 1870]
15                }
16            ]
17        },
18    },
19    options: {
20        legend: { display: false },
21        title: {
22            display: true,
23            text: 'Area forestal en millones de hectáreas en 2020'
24        }
25    }
26 });
27 });
28 };
```

Podemos observar que todo el código de configuración del Chart está dentro del evento onLoad, esto provoca que se ejecute automáticamente en cuanto termina de cargarse todos los elementos HTML de la página.

- Primero obtenemos el elemento **canva** desde la interfaz.
- A continuación le asignamos la configuración de Chart.

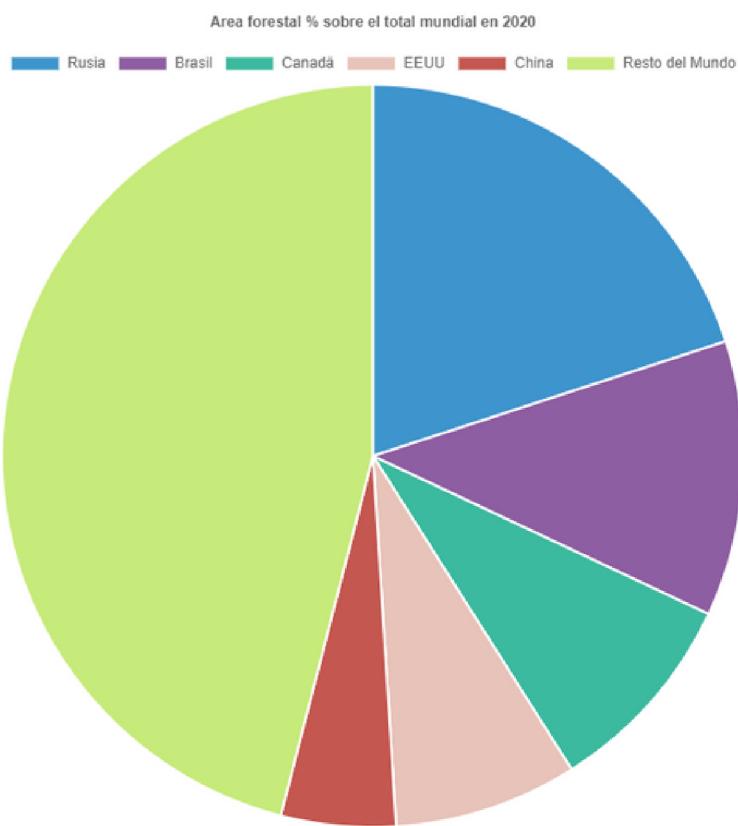
Dentro de la configuración del Chart hacemos lo siguiente:

- En el asignamos la propiedad **type** a 'bar' para configurar que sea un gráfico de barras.
- Le asignamos en **labels** el Array de etiquetas que indican el nombre de los países representados.
- Le asignamos a **label** el texto que utilizará la etiqueta emergente al pasar por encima con el ratón.
- Le asignamos a **backgroundcolor** en un Array los colores de las diferentes barras para cada país.
- Finalmente le asignamos un título en **title.text** que mostrará el gráfico en su parte superior.

VERSIÓN 2. GRÁFICA DE TARTA

- En esta versión 2 hemos querido probar otro tipo de gráfico. Para ello hemos generado un gráfico similar al anterior pero del tipo de tarta, utilizando los mismos datos pero esta vez será el porcentaje de cada país sobre el total de la masa forestal mundial. El gráfico de tarta es más indicado para porcentajes.

Area Forestal % (2020)



En este tipo de gráfico de tarta vemos que en vez de poner los nombres en el gráfico nos aporta una leyenda de colores en la parte superior con el nombre del país correspondiente.

Se puede apreciar claramente que entre esos 5 países representados juntos conservan más de la mitad de la masa forestal mundial.

Este gráfico está compuesto por este HTML prácticamente igual al de la versión 1.

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <title>Superficie Forestal</title>
7          <script src="https://cdn.jsdelivr.net/npm/chart.js@latest/dist/Chart.min.js"></script>
8          <script src="chartScript.js"></script>
9          <link rel="stylesheet" href="chartStyles.css">
10     </head>
11     <body>
12         <h1>Area Forestal Hectareas (2020)</h1>
13         <div class="canvasContainer">
14             <canvas id="myChart"></canvas>
15         </div>
16     </body>
17     <footer>
18         <p class="source">Fuente : https://www.fao.org/forestry-resources-assessment/2020/es</p>
19     </footer>
20 </html>
```

Tan solo se diferencian en el título de la página mostrado en el `<h1>` para indicar que se trata de porcentajes.

El archivo .css también es exactamente igual que el de la versión 1, se utilizan los mismos dos estilos para el canvas y para el texto del origen de datos, así que omitiremos aquí incluir de nuevo la imagen.

Respecto al código .js con la configuración en esta versión 2:

```
1 ▼ window.onload = function() {
2
3     let chart = document.getElementById('myChart').getContext('2d');
4
5     let myChart = new Chart(chart, {
6         type: 'pie',
7         data: {
8             labels: ["Rusia", "Brasil", "Canadá", "EEUU", "China", "Resto del Mundo"],
9             datasets: [
10                 {
11                     label: "Area Forestal",
12                     backgroundColor: ["#3e95cd", "#8e5ea2", "#3cba9f", "#e8c3b9", "#c45850", "#c7eb7a" ],
13                     data: [20, 12, 9, 8, 5, 46]
14                 }
15             ],
16         },
17         options: {
18             legend: { display: true },
19             title: {
20                 display: true,
21                 text: 'Area forestal % sobre el total mundial en 2020'
22             }
23         }
24     });
25 };
```

Simplemente cambia la propiedad `type` que ahora está asignado como '`pie`' y esto lo configura como una tarta. La etiqueta del gráfico y los datos también son diferentes, como hemos dicho, ahora son porcentuales, más adecuados al formato tarta y no cifras de cantidad como en la versión 1.

VERSIÓN 3. CAMBIO DE TIPO DE GRÁFICA

- En esta versión 3 una vez hemos comprobado que la configuración de las gráficas es muy similar, y que dependen fundamentalmente del valor de la propiedad 'type', hemos querido introducir botones que cambien dinámicamente el tipo de gráfica mostrado.

Por ello cuando cargamos la página, vemos que aparecen los tres botones y ningún grafico cargado.

Opciones de gráficos

Fuente : <https://www.fao.org/forest-resources-assessment/2020/es>

Hemos eliminado por lo tanto el evento onLoad(), ahora se cargará cada gráfico bajo demanda al pulsar el usuario el botón correspondiente.

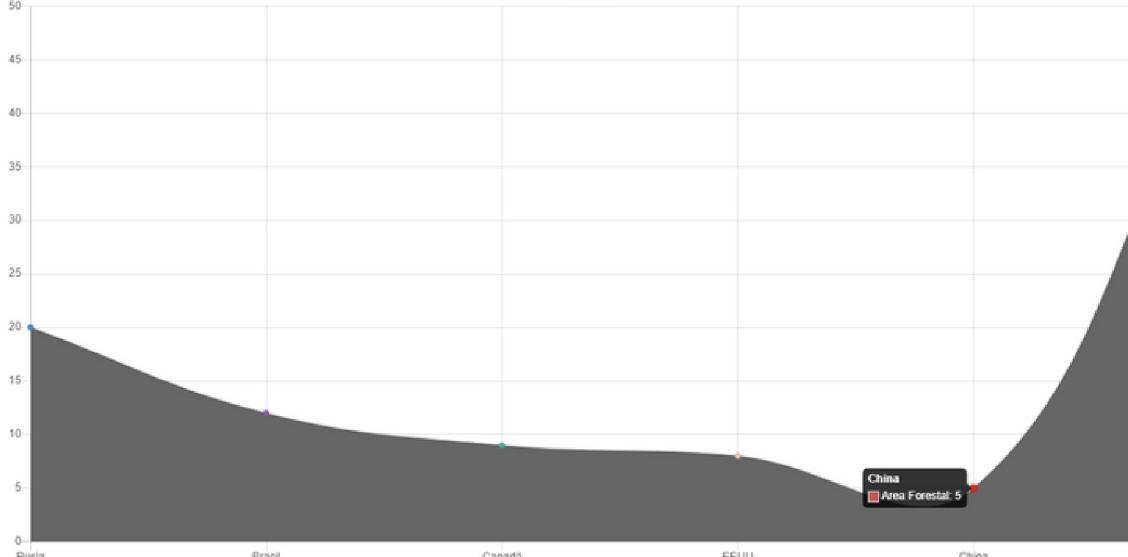
Y esto es lo que pasa cuando pulsamos un botón. Por ejemplo el que pone línea.

Área Forestal % (2020)

Opciones de gráficos

Área forestal % sobre el total mundial en 2020

Área Forestal



Fuente : <https://www.fao.org/forest-resources-assessment/2020/es>

Así conforme pulsamos cada uno de los tres botones nos carga un tipo de gráfico utilizando exactamente los mismos datos en todos los casos.

Es código html sigue siendo similar pero introduce las novedad de los tres botones de configuración de las gráficas.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Superficie forestal</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js@latest/dist/Chart.min.js"></script>
    <script src="chartScript.js"></script>
    <link rel="stylesheet" href="chartStyles.css">
  </head>
  <body>
    <h1>Area Forestal % (2020)</h1>

    <h2>Opciones de gráficos</h2>
    <button onclick="crearGraficoBarras()">Barras</button>
    <button onclick="crearGraficoTarta()">Tarta</button>
    <button onclick="crearGraficoLinea()">Línea</button>

    <div class="canvasContainer">
      <canvas id="myChart"></canvas>
    </div>
  </body>
  <footer>
    <p class="source">Fuente : https://www.fao.org/forestry-resources-assessment/2020/es</p>
  </footer>
```

Arriba del canvas podemos ver la configuración de los tres botones.

Todos tienen asociado su propio evento onclick para llamar a la función desarrollada en código javascript en el fichero chartScript.js y que crear el gráfico correspondiente.

Como todavía no hemos creado ningún estilo para estos nuevos botones el archivo .css sigue siendo igual al de la versión 1 y al de la versión 2, por lo tanto omitiremos de ponerlo de nuevo..

Finalmente vemos el archivo chartScript.js con el código javascript que respalda esta versión 3. Esta vez lo veremos por partes. Primero las constantes y variable global declaradas.

```
2 var myChart = null;
3
4 // Las etiquetas son las que van en el eje X.
5 const etiquetas = ["Rusia", "Brasil", "Canadá", "EEUU", "China", "Resto del Mundo"];
6
7 // Podemos tener varios conjuntos de datos. Comencemos con uno
8 const datosSuperficieForestalPorcentaje = {
9     label: "Área Forestal",
10    backgroundColor: ["#3e95cd", "#8e5ea2", "#3cba9f", "#e8c3b9", "#c45850", "#c7eb7a"],
11    data: [20, 12, 9, 8, 5, 46]
12};
```

En este caso establecemos la variable global al fichero myChart que albergará el gráfico creado. Esto nos permitirá crearlo y eliminarlo antes de crearlo de nuevo al pulsar otro botón para facilitar su funcionamiento.

Vemos el primer ejemplo. La función que crea el gráfico de barras y que responde al click del botón “Barras”.

```
14 function crearGraficoBarras()
15 {
16     //Si existe el gráfico previamente lo eliminamos
17
18     if(myChart != null)
19         myChart.destroy();
20
21     //Obtenemos el elemento canva de la interfaz.
22
23     let canva = document.getElementById('myChart').getContext('2d');
24
25     myChart = new Chart(canva, {
26         type: 'bar',
27         data: {
28             labels: etiquetas,
29             datasets: [
30                 datosSuperficieForestalPorcentaje
31             ]
32         },
33         options: {
34             legend: { display: true },
35             title: {
36                 display: true,
37                 text: 'Área forestal % sobre el total mundial en 2020'
38             },
39             scales: {
40                 yAxes: [
41                     ticks: {
42                         beginAtZero: true
43                     }
44                 ],
45             }
46         }
47     });
48 }
49
50 };
```

En el vemos que fundamentalmente se hace lo siguiente:

- Primero si el gráfico está creado previamente, lo eliminamos.
- Luego obtenemos el elemento **canva** de la interfaz.
- A continuación configuramos el gráfico de barras, para ello utilizamos las constantes arriba declaradas, la de **etiquetas** y la de **datosSuperficieForestalPorcentaje** que configuran la parte de datos de la gráfica.

A continuación vemos en paralelo los métodos que responden al evento del botón **tarta** y **línea**.

```
53  function crearGraficoTarta()
54  {
55      if(myChart != null)
56          myChart.destroy();
57
58      let canva = document.getElementById('myChart').getContext('2d');
59
60      myChart = new Chart(canva, {
61          type: 'pie',
62          data: {
63              labels: etiquetas,
64              datasets: [
65                  datosSuperficieForestalPorcentaje
66              ],
67          },
68          options: {
69              legend: { display: true },
70
71              title: {
72                  display: true,
73                  text: 'Área forestal % sobre el total mundial en 2020'
74              },
75
76              scales: {
77                  yAxes: [{{
78                      ticks: {
79                          beginAtZero: true
80                      }
81                  }],
82              },
83          }
84      });
85  };
86
88  function crearGraficoLinea()
89  {
90      if(myChart != null)
91          myChart.destroy();
92
93      let canva = document.getElementById('myChart').getContext('2d');
94
95      myChart = new Chart(canva, {
96          type: 'line',
97          data: {
98              labels: etiquetas,
99              datasets: [
100                  datosSuperficieForestalPorcentaje
101             ],
102         },
103         options: {
104             legend: { display: true },
105
106             title: {
107                 display: true,
108                 text: 'Área forestal % sobre el total mundial en 2020'
109             },
110
111             scales: {
112                 yAxes: [{{
113                     ticks: {
114                         beginAtZero: true
115                     }
116                 }],
117             },
118         }
119     });
120
121 }
```

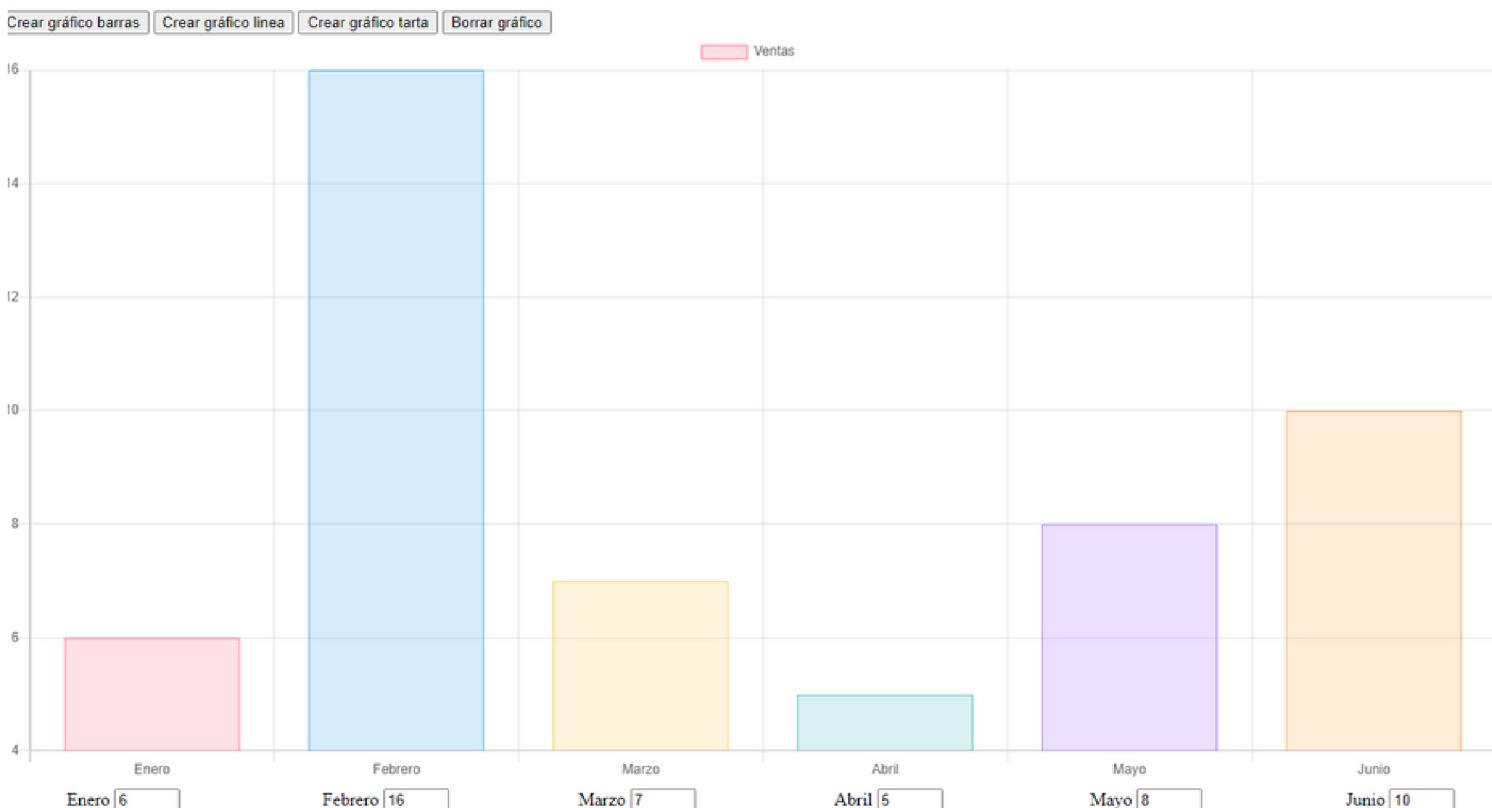
De esta forma se observa con claridad que los métodos son iguales entre sí (también con el anterior que es el de barras y que solo diferenciandolos por la propiedad **type** ya estamos obteniendo gráficos totalmente de diferente naturaleza).

Así con el uso de estos tres botones se puede apreciar que el cambio de uno a otro es sencillo y gráficamente da una sensación más dinámica al visionado de datos.

VERSIÓN 4. CONTROLES NUMÉRICOS

- Para esta versión 4, partiendo de la versión 3, nos hemos propuesto añadir en la parte de los botones un botón más con el texto “Borrar gráfico”. También hemos añadido en la parte baja un campo numérico para cada mes donde s.

En este caso el gráfico cargará los datos simulados de ventas de una empresa



En la parte alta, el botón “Borrar grafica” eliminará la gráfica que en ese momento esté representada y por lo tanto reseteará la página al estado inicial de carga, con el **canva** vacío.

En la parte baja, los controles numéricos van acompañados del nombre de cada mes y contienen el dato representado en la gráfica para ese mes. En esta versión permiten también que el usuario dinámicamente modifique el dato en el control numérico de cualquier mes y automáticamente el dato se actualizará en la gráfica en ejecución.

En esta versión vemos como el HTML se va haciendo más extenso.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ventas</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js@latest/dist/Chart.min.js"></script>
    <script src="chartScript.js"></script>
    <link rel="stylesheet" href="chartStyles.css">
</head>
<body>
    <button onclick="crearGraficoBarras()">Crear gráfico barras</button>
    <button onclick="crearGraficoLinea()">Crear gráfico linea</button>
    <button onclick="crearGraficoTarta()">Crear gráfico tarta</button>
    <button onclick="borrarGrafico()">Borrar gráfico</button> ←
    <div class="canvasContainer">
        <canvas id="myChart"></canvas>
    </div>
    <table id="tableOperationControls">
        <tr>
            <td class="tdNumberTitle"></td>
            <td class="tdNumber">
                <label for="numberEnero" class="labelData">Enero</label>
                <input type="number" id="numberEnero" class="numberData" onchange="numberChanged(this,0)">
            </td>
            <td class="tdNumber">
                <label for="numberFebrero" class="labelData">Febrero</label>
                <input type="number" id="numberFebrero" class="numberData" onchange="numberChanged(this,1)">
            </td>
            <td class="tdNumber">
                <label for="numberMarzo" class="labelData">Marzo</label>
                <input type="number" id="numberMarzo" class="numberData" onchange="numberChanged(this,2)">
            </td>
            <td class="tdNumber">
                <label for="numberAbril" class="labelData">Abril</label>
                <input type="number" id="numberAbril" class="numberData" onchange="numberChanged(this,3)">
            </td>
            <td class="tdNumber">
                <label for="numberMayo" class="labelData">Mayo</label>
                <input type="number" id="numberMayo" class="numberData" onchange="numberChanged(this,4)">
            </td>
            <td class="tdNumber">
                <label for="numberJunio" class="labelData">Junio</label>
                <input type="number" id="numberJunio" class="numberData" onchange="numberChanged(this,5)">
            </td>
        </tr>
    </table>
</body>
</html>
```



Observar como en los botones creados al principio del body hemos creado uno más con el texto crear gráfico.

En la parte baja,vemos una **tabla** con 1 fila y 6 columnas que contienen los 6 controles numéricos definidos para cada mes. Todos ellos llaman en el evento onChange() a la función "numberChanged(this, 0)" en donde **this** es el propio control que llama y el parámetro numérico corresponde a su índice de 0 a 5, para que cuando un valor cambie el evento numberChanged sea del mes que se trata por su índice.

Respecto al archivo de estilos .css sí que observamos algunas modificaciones.

Fundamentalmente clases aplicadas a la tabla de los controles numéricos para su alineación.

```
1  .canvasContainer{  
2      height:70%;           /*proporcion del alto de la pagina que ocupa nuestro canvas*/  
3      width: 70%;           /*proporcion del ancho de la pagina que ocupa nuestro canvas*/  
4  }  
5  
6  .tdNumberTitle{  
7      width: 4%;            /*Ancho de la primera celda de la tabla. Esta vacia, solamente supone un margen*/  
8  }  
9  
10 .tdNumber{  
11     min-width: 80px;        /*Maximo del ancho en el caso de que se redimensione*/  
12     width: 215px;           /*Ancho de cada celda*/  
13     white-space: nowrap;    /* Evita que se rompa la linea entre label y numeric */  
14     display: inline-block;   /* Muestra siempre en una linea los dos elementos*/  
15     margin-right: 10px;      /* Espacio entre la etiqueta y el input */  
16  }  
17  
18 .numberData{  
19     width: 50px;            /*Ancho del control numerico*/  
20     min-width: 20px;         /*Minimo del ancho en el caso de que se redimensione*/  
21  }  
22
```

Como novedades:

La clase **tdNumberTitle** se aplica a la primera columna que antecede a los controles y sirve de margen respecto a la izquierda del primer control.

La clase **tdNumber** se aplica a cada una de las celdas que aloja un control numérico y controla su disposición en linea sus márgenes y su ancho fundamentalmente.

La clase **numberData** se aplica al mismo control y establece fundamentalmente su ancho.

Con respecto al archivo chartscript.js en esta versión 4 también ha habido cambios.

Hemos refactorizado los métodos la creación de los graficos, para ello hemos sacado como constante la configuración de datos del chart que como vimos en la versión anterior era igual en todos los casos.

Así es como quedan las constantes y la variable global en el archivo .js.

```
1  var myChart = null;
2
3  const ventaPorMeses = [12, 19, 3, 5, 8, 3];
4
5  const dataVentas = {
6      labels: ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio'],
7      datasets: [
8          {
9              label: 'Ventas',
10             data: ventaPorMeses,
11             backgroundColor: [
12                 'rgba(255, 99, 132, 0.2)',
13                 'rgba(54, 162, 235, 0.2)',
14                 'rgba(255, 206, 86, 0.2)',
15                 'rgba(75, 192, 192, 0.2)',
16                 'rgba(153, 102, 255, 0.2)',
17                 'rgba(255, 159, 64, 0.2)'
18             ],
19             borderColor: [
20                 'rgba(255, 99, 132, 1)',
21                 'rgba(54, 162, 235, 1)',
22                 'rgba(255, 206, 86, 1)',
23                 'rgba(75, 192, 192, 1)',
24                 'rgba(153, 102, 255, 1)',
25                 'rgba(255, 159, 64, 1)'
26             ],
27             borderWidth: 1
28         }]
29     };
30 
```

Y así es como queda cada uno de los eventos de botón que crea las gráficas

```
function crearGraficoBarras() {
    if(myChart != null)
        myChart.destroy();

    let canva = document.getElementById('myChart').getContext('2d');

    myChart = new Chart(canva, {
        type: 'bar',
        data: dataVentas,
        options: {
            scales: {
                y: {
                    beginAtZero: true
                }
            }
        }
    });

    inicializaNumberControls();
} 
```

Se puede apreciar un diseño mucho más despejado con solo extraer la configuración de los datos en una constante.

Al final se llama a inicializaNumberControls(), metodo que asigna el valor de los datos de la gráfica a los controles numéricos para que sean mostrados. Luego lo veremos.

Como siempre los 3 métodos de creación son iguales, solo cambia la propiedad **type** por lo tanto omitiré los otros dos (el de linea y el de tarta).

A continuación hemos añadido la función **borrar gráfico**

```
96 ▼ function borrarGrafico(input, mes) {  
97  
98 ▼   if(myChart != null) {  
99       myChart.destroy();  
100      myChart = null;  
101    }  
102  
103    vaciaNumberControls();  
104  };  
105
```

Muy sencilla. Comprueba si el chart ha sido creado y si es así lo destruye.

En todo caso llama al método **vaciaNumberControls** que reseteará los controles numéricos de los meses tras el borrado del gráfico.

A continuación tenemos la función **numberChanged** que responde al evento **onChanged** de cada uno de los controles numéricos en el HTML mostrado más arriba.

```
105  
106 ▼ function numberChanged(input, mes) {  
107  
108 ▼   if(myChart != null) {  
109     myChart.data.datasets[0].data[mes] = input.value;  
110     myChart.update();  
111   }  
112 };  
113
```

También muy sencilla. Comprueba si el chart ha sido creado y si es así le asigna el dato que tenemos en el control que emite el evento (primer parámetro) al dato correspondiente según el mes (segundo parámetro que corresponde con el índice del mes).

Para finalizar hemos creado dos funciones para el manejo de los controles numéricos.

```
114 ▼ function inicializaNumberControls(){
115
116     let numberControlEnero = document.getElementById('numberEnero');
117     numberControlEnero.value = ventaPorMeses[0];
118
119     let numberControlFebrero = document.getElementById('numberFebrero');
120     numberControlFebrero.value = ventaPorMeses[1];
121
122     let numberControlMarzo = document.getElementById('numberMarzo');
123     numberControlMarzo.value = ventaPorMeses[2];
124
125     let numberControlAbril = document.getElementById('numberAbril');
126     numberControlAbril.value = ventaPorMeses[3];
127
128     let numberControlMayo = document.getElementById('numberMayo');
129     numberControlMayo.value = ventaPorMeses[4];
130
131     let numberControlJunio = document.getElementById('numberJunio');
132     numberControlJunio.value = ventaPorMeses[5];
133 }
134
135 ▼ function vaciaNumberControls(){
136
137     let numberControlEnero = document.getElementById('numberEnero');
138     numberControlEnero.value = "";
139
140     let numberControlFebrero = document.getElementById('numberFebrero');
141     numberControlFebrero.value = "";
142
143     let numberControlMarzo = document.getElementById('numberMarzo');
144     numberControlMarzo.value = "";
145
146     let numberControlAbril = document.getElementById('numberAbril');
147     numberControlAbril.value = "";
148
149     let numberControlMayo = document.getElementById('numberMayo');
150     numberControlMayo.value = "";
151
152     let numberControlJunio = document.getElementById('numberJunio');
153     numberControlJunio.value = "";
154 }
155
```

inicializaNumberControls() que será llamada justo después de crear la gráfica e inicializa todos los valores de los campos numéricos según el array **ventasPorMeses** que es el utilizado para la configuración tras la creación de la propia gráfica.

vaciaNumberControls() que será llamada justo al destruir la gráfica de la página y quedarse está en blanco. Este método vacía todos los campos numéricos.

VERSIÓN 5. VERSIÓN FINAL (COMPLETA)

- Para esta versión 5 y última, partiendo de la versión 4, nos hemos propuesto añadir dos botones que acompañarán a cada control numérico y que añadirán o quitarán uno al valor del dato de forma ágil haciendo click con el ratón.

También hemos añadido un control de selección de color para cada mes en la parte de abajo del todo.



Como se puede apreciar también hemos aplicado un estilo a los botones de arriba de los diferentes tipos de chart, y hemos añadido 3 tipos nuevos de Chart : Donut, Burbujas y Área Polar.

Con el control de color de la parte de debajo, el usuario podrá cambiar dinámicamente en ejecución el color asignado a ese mes. Al seleccionar el usuario un nuevo color, se actualizarán tanto la zona de la gráfica correspondiente como el fondo de los botones de añadir o quitar que tiene encima y el control numérico correspondientes a ese mes. De esta forma quedan los controles añadidos por nuestra parte mucho más integrados con el diseño del gráfico

La versión completa del HTML quedá así:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ventas</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js@latest/dist/Chart.min.js"></script>
    <script src="chartScript.js"></script>
    <link rel="stylesheet" href="chartStyles.css">
</head>
<body>
    <button class="chart-button" onclick="crearGraficoBarras()">Barras</button>
    <button class="chart-button" onclick="crearGraficoLinea()">Línea</button>
    <button class="chart-button" onclick="crearGraficoTarta()">Tarta</button>
    <button class="chart-button" onclick="crearGraficoDonut()">Donut</button>
    <button class="chart-button" onclick="crearGraficoBurbujas()">Burbujas</button>
    <button class="chart-button" onclick="crearGraficoAreaPolar()">Área Polar</button>
    <button class="chart-button" onclick="borrarGrafico()">Borrar gráfico</button>
    <div class="canvasContainer">
        <canvas id="myChart"></canvas>
    </div>
    <table id="tableOperationControls">
        <tr>
            <td class="tdNumberTitle"></td>
            <td class="tdNumber">
                <button name='btnEneroAdd' id='btnEneroAdd' class="chart-operation-button btnEnero" onclick="addOne(0)">+</button>
                <input type="number" id="numberEnero" class="numberData" onchange="numberChanged(this,0)">
                <button name='btnEneroSubtract' id='btnEneroSubtract' class="chart-operation-button btnEnero" onclick="subtractOne(0)">-</button>
            </td>
            <td class="tdNumber">
                <button name='btnFebreroAdd' id='btnFebreroAdd' class="chart-operation-button btnFebrero" onclick="addOne(1)">+</button>
                <input type="number" id="numberFebrero" class="numberData" onchange="numberChanged(this,1)">
                <button name='btnFebreroSubtract' id='btnFebreroSubtract' class="chart-operation-button btnFebrero" onclick="subtractOne(1)">-</button>
            </td>
            <td class="tdNumber">
                <button name='btnMarzoAdd' id='btnMarzoAdd' class="chart-operation-button btnMarzo" onclick="addOne(2)">+</button>
                <input type="number" id="numberMarzo" class="numberData" onchange="numberChanged(this,2)">
                <button name='btnMarzoSubtract' id='btnMarzoSubtract' class="chart-operation-button btnMarzo" onclick="subtractOne(2)">-</button>
            </td>
            <td class="tdNumber">
                <button name='btnAbrilAdd' id='btnAbrilAdd' class="chart-operation-button btnAbril" onclick="addOne(3)">+</button>
                <input type="number" id="numberAbril" class="numberData" onchange="numberChanged(this,3)">
                <button name='btnAbrilSubtract' id='btnAbrilSubtract' class="chart-operation-button btnAbril" onclick="subtractOne(3)">-</button>
            </td>
            <td class="tdNumber">
                <button name='btnMayoAdd' id='btnMayoAdd' class="chart-operation-button btnMayo" onclick="addOne(4)">+</button>
                <input type="number" id="numberMayo" class="numberData" onchange="numberChanged(this,4)">
                <button name='btnMayoSubtract' id='btnMayoSubtract' class="chart-operation-button btnMayo" onclick="subtractOne(4)">-</button>
            </td>
            <td class="tdNumber">
                <button name='btnJunioAdd' id='btnJunioAdd' class="chart-operation-button btnJunio" onclick="addOne(5)">+</button>
                <input type="number" id="numberJunio" class="numberData" onchange="numberChanged(this,5)">
                <button name='btnJunioSubtract' id='btnJunioSubtract' class="chart-operation-button btnJunio" onclick="subtractOne(5)">-</button>
            </td>
        </tr>
    </table>
    <table class="tableOperationColors">
        <tr>
            <td class="tdColorTitle"></td>
            <td class="tdColor">
                <input type="color" id="colorPickerEnero" onchange="colorChanged(this, 0)">
            </td>
            <td class="tdColor">
                <input type="color" id="colorPickerFebrero" onchange="colorChanged(this, 1)">
            </td>
            <td class="tdColor">
                <input type="color" id="colorPickerMarzo" onchange="colorChanged(this, 2)">
            </td>
            <td class="tdColor">
                <input type="color" id="colorPickerAbril" onchange="colorChanged(this, 3)">
            </td>
            <td class="tdColor">
                <input type="color" id="colorPickerMayo" onchange="colorChanged(this, 4)">
            </td>
            <td class="tdColor">
                <input type="color" id="colorPickerJunio" onchange="colorChanged(this, 5)">
            </td>
        </tr>
    </table>
</div>
</body>
</html>
```

Cabe destacar:

- Podemos ver que en los botones del principio del body, ahora hay 7. 6 que crean un tipo diferente de gráfico (barras, linea, tarta, donut, burbujas y area polar) y un botón para borrar. Con lo cual hemos añadido tres nuevos tipos con respecto a la versión anterior.
- Debajo de la tabla de numeric controls vemos una segunda tabla similar pero que alberga controles de tipo colorPicker (selectores de color), una para cada mes del gráfico.

En el archivo de estilos .css de esta versión 5 tenemos además de lo anterior de la versión 4 los siguientes estilos creados. Para los botones de selección de Chart, verdes de la parte superiores

```
.chart-button {  
    display: inline-block;  
    margin: 0 auto;  
    padding: 10px;  
    color: black;  
    margin-bottom: 30px;  
    background-color: green;  
    border: none;  
    border-radius: 5px;  
    cursor: pointer;  
    width: 150px;  
}
```

Para los botones redondos de añadir y restar de cada mes.

```
54 .chart-operation-button {  
55     font-size: 1.2em;  
56     font-weight: bold;  
57     margin: 0 auto;  
58     padding: 10px;  
59     margin-bottom: 30px;  
60     color: navy;  
61     border: none;  
62     border-radius: 40px;  
63     cursor: pointer;  
64     width: 40px;  
65     height: 40px;  
66 }  
67  
68  
69 .btnEnero{  
70     border: 2px solid black;  
71     background-color: #e6f2ff;  
72 }  
73
```

Y para la tabla que estructura los colores de selección de color picker color

```
54
55 .tdColorTitle{
56 |   text-align: center;
57 |   width: 5.7%;
58 }
59
60 .tdColor {
61 |   min-width: 140px;
62 |   width: 215px;
63 }
64
65 input[type="color"] {
66 |   width: 100px;
67 |   height: 50px;
68 |   background-color: black;
69 |   border-radius: 20px;
70 }
71
```

Finalmente el código javascript.

Aquí el código ha ido cogiendo mayor complejidad, por lo cual he ido incrementando las variables y constantes utilizadas para facilitar configuraciones de cara al desarrollo.

```
1  /*////////////////////////////////////////////////////////////////////////*/
2  //                               CONSTANTES                                //
3  //                               Y                                     //
4  //                               VARIABLES COMUNES                      //
5  //                               //                                     //
6  //                               //                                     //
7  ///////////////////////////////////////////////////////////////////*/
8  const BARRAS = 'bar';
9  const LINEA = 'line';
10 const TARTA = 'pie';
11 const DONUT = 'doughnut';
12 const AREA_POLAR = 'polarArea';
13 const BURBUJAS = 'bubble';
14
15 const CHART = 'myChart';
16
17 const TITLE = 'Ventas';
18
19 const ENERO = 'Enero';
20 const FEBRERO = 'Febrero';
21 const MARZO = 'Marzo';
22 const ABRIL = 'Abril';
23 const MAYO = 'Mayo';
24 const JUNIO = 'Junio';
25
26 const JANUARY = 0;
27 const FEBRUARY = 1;
28 const MARCH = 2;
29 const APRIL = 3;
30 const MAY = 4;
31 const JUNE = 5;
32
33 const R = 0;
34 const G = 1;
35 const B = 2;
36 const A = 3;
37
38 const BLACK = '#000000';
39
40 const meses = [ENERO, FEBRERO, MARZO, ABRIL, MAYO, JUNIO];
41 const months = [JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE];
42
43 var opacityBack = 0.2;
44
45 var myChart = null;
```

Estas son las clases de creación de gráficos, que como se puede ver, se han refactorizado y todas ellas llaman al mismo único método que crea todos los gráficos, sea cual sea el tipo.

```
48 ///////////////////////////////////////////////////////////////////
49 //          //
50 //      FUNCIONES DE CREACIÓN          //
51 //          //
52 //          //
53 ///////////////////////////////////////////////////////////////////*/
54
55 function crearGraficoBarras() {
56     crearGraficoPorTipo(BARRAS);
57 };
58
59
60 function crearGraficoLinea() {
61     crearGraficoPorTipo(LINEA);
62 };
63
64
65 function crearGraficoTarta() {
66     crearGraficoPorTipo(TARTA);
67 };
68
69
70 function crearGraficoDonut() {
71     crearGraficoPorTipo(DONUT);
72 };
73
74
75 function crearGraficoAreaPolar() {
76     crearGraficoPorTipo(AREA_POLAR);
77 };
78
79
80 function crearGraficoBurbujas() {
81     crearGraficoPorTipo(BURBUJAS);
82 };
83
84
85 function crearGraficoPorTipo(tipo){
86
87     if (myChart != null) {
88
89         myChart.config.type = tipo;
90         myChart.update();
91     }
92     else{
93
94         let data = generateRandomNumberList(6, 1, 20);
95         let ctx = document.getElementById(CHART).getContext('2d');
96         let colors = ['rgb(255, 99, 132)',  

97             'rgb(54, 162, 235)',  

98             'rgb(255, 206, 86)',  

99             'rgb(75, 192, 192)',  

100            'rgb(153, 102, 255)',  

101            'rgb(255, 159, 64)'];
102
103        let backColors = rgbListToRGBASTring(colors, opacityBack);
104        let borderColors = rgbListToRGBASTring(colors, 1);
105
106        myChart = new Chart(ctx, {
107            type: tipo,
108            data: {
109                labels: meses,
110                datasets: [{  

111                    label: TITLE,  

112                    data: data,  

113                    backgroundColor: backColors,  

114                    borderColor: borderColors,  

115                    borderWidth: 1
116                }]
117            },
118            options: {
119                scales: {
120                    yAxes: [{  

121                        display: true,
122                        ticks: {
123                            beginAtZero: true
124                        }
125                    }]
126                }
127            }
128        });
129
130        var tabla = document.getElementById("tableOperationControls");
131
132        if(tabla == null){
133
134            console.log('Se ha producido un NULL al recoger la tabla de Control de Operaciones');
135        }
136        else{
137            tabla.removeAttribute("hidden");
138        }
139
140        initializeInputNumbersData(data);
141
142        initializeColorList(colors, backColors, borderColors);
143    }
144};
```

Esto demuestra lo unificada que está el código de creación.

Esto demuestra lo unificada que está el código de creación de charts.

CONCLUSIÓN

La experiencia de uso de chartjs ha sido satisfactoria.

Es una librería de rápido aprendizaje, tan solo una semana es suficiente para un dominio de manejos suficiente. La configuración tan unificada de los charts facilita mucho que el aprendizaje y el avance sea rápido de uno a otro y por supuesto a nivel gráfico permite definir muchos detalles que facilitan la personalización de las gráficas y ajustarla a las necesidades del proyecto.

Para dudas y pruebas puedes ver y descargar el código de este trabajo en la dirección de GitHub

<https://github.com/rafamar/chartLibraryDemo/>