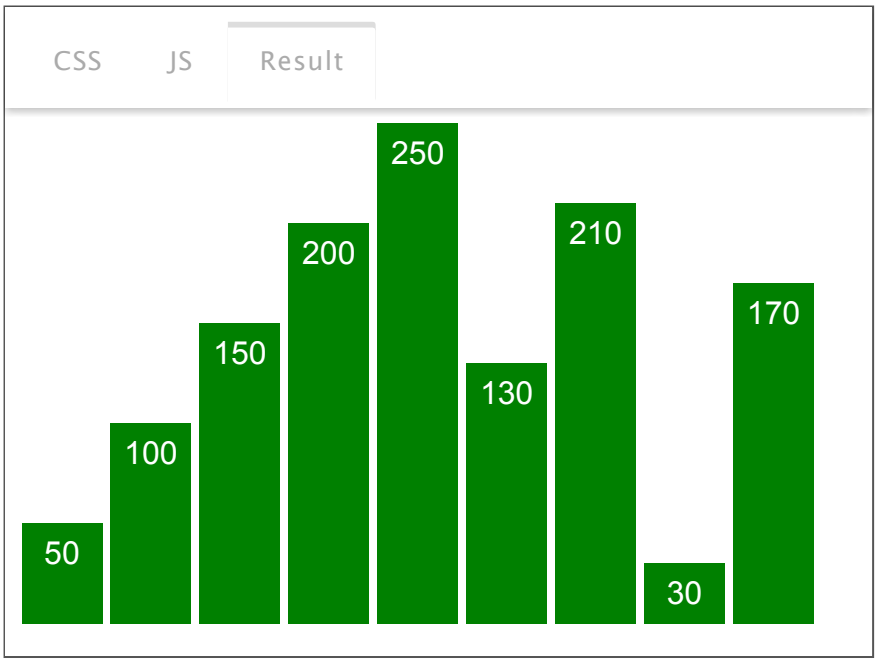
Brandon Morelli Seguir
Creador de @codeburstio - publicando con frecuencia tutoriales y artículos de desarrollo web.
Sígueme en Twitter también: @BrandonMorelli
4 de diciembre · 8 minutos de lectura

Introducción a la visualización de datos: cree su primer gráfico con D3.js

¡Aprenda los conceptos básicos de D3 y haga una tabla!

¡Hola a todos! Aquí hay un *codepen* de lo que vamos a aprender a construir en los próximos 15 minutos:



Código repetitivo

Puede usar el **codepen** anterior para seguirlo, o puede usar este archivo HTML repetitivo a continuación. Simplemente pondrías todo tu D3 dentro de las `<script>` etiquetas en la línea 13:

```
<! DOCTYPE html>
< html  lang = " en " >
< head >
    < meta  charset = " UTF-8 " >
    < meta  name = " viewport "  content = " width = device
    < meta  http-equiv = " Compatible con X-UA "  content =
    < script  src = " https://cdnjs.cloudflare.com/ajax/lib
    < script  src = " https://d3js.org/d3-selection-multi.v
    < title > D3 </ title >
</ head >
< cuerpo >
```

Introducción rápida a D3 y SVG

Si nunca has oído hablar de D3, esta es la definición directamente de la documentación :

> *D3 (Documentos controlados por datos o D3.js) es una biblioteca de JavaScript para visualizar datos utilizando estándares web. D3 te ayuda a dar vida a los datos usando SVG, Canvas y HTML. D3 combina poderosas técnicas de visualización e interacción*

Antes de comenzar con D3, es importante que comprenda qué son los *SVG* . *SVG* son **gráficos vectoriales escalables** . Son gráficos que están definidos en un formato XML y no pierden calidad cuando se amplían y cambian de tamaño.

Afortunadamente, los SVG son muy fáciles de usar. Echemos un vistazo a un código de muestra que crea un rectángulo rojo:

```
<svg>
  <rect width = '50 'height =' 200 'style =' fill: red '/>
</ svg>
```

Como puede ver, el código es bastante directo: tenemos una `SVG` etiqueta y dentro de esa `SVG` etiqueta tenemos una `rect` etiqueta de cierre automático . La `rect` etiqueta se usa para crear, lo adivinaste, rectángulos. Dentro de nuestra `rect` etiqueta solo necesitamos definir el `width` , y `height` , pero también he definido un color para nuestro rectángulo a través del `style` atributo. Cuando se coloca en un archivo html, obtenemos esto:

Increíble. Un rectángulo. D3 es simplemente una biblioteca que nos permite construir y hacer cosas **realmente** increíbles con SVG (entre otros).

Listo? ¡Empecemos!

> *¿Desea aprender más visualización de datos y JavaScript avanzado?* ***Consulte el Bootcamp avanzado de desarrolladores web aquí*** .

Hacer un rectángulo en D3

Ahora vamos a recrear nuestro rectángulo exacto que creamos en SVG, pero vamos a hacerlo con D3.

Lo primero que debemos hacer es utilizar el `select()` método. Con este método podemos pasar una referencia al nodo DOM que queremos seleccionar. En pocas palabras, podemos seleccionar un elemento dentro de nuestro HTML utilizando este método. Seleccionaremos el `body` de nuestra página web:

```
d3.select ('cuerpo')
```

Increíble. Ahora que hemos seleccionado el elemento en el que queremos poner nuestra visualización, podemos comenzar a crearlo. Primero agregaremos en nuestra `<svg>` etiqueta:

```
    .append ('svg')
```

A continuación, dentro de nuestra `<svg>` etiqueta, necesitamos agregar una `<rect>` etiqueta para crear nuestro rectángulo:

```
    .append ('rect')
```

Ahora que ya hemos creado nuestro rectángulo, podemos añadir en nuestro `width` , `height` y ninguna `styles` .

```
    .attr ('ancho', 50). // establecer ancho
    .attr ('alto', 200) // establecer alto
    .style ('relleno', 'azul'); // establecer color
```

¡Y eso es! Ahora, cuando actualizamos la página, se nos presenta exactamente el mismo rectángulo que creamos antes (con un color diferente, por supuesto)

Ahora, obviamente esto fue más trabajo para usar D3 de lo que era crear el rectángulo desde cero con SVG. Eso es porque este fue un ejemplo ultra simplista. Una vez que entramos en ejemplos complejos, es mucho más fácil usar D3.

Vamos a hacer un cuadro

Un diagrama de barras es esencialmente solo un montón de rectángulos. Entonces, para crear uno, solo necesitamos un montón de rectángulos de diferentes longitudes. Es un poco más complejo que eso, pero vamos a saltar y comenzar a construir.

Lo primero que debemos hacer es crear nuestro `svg` lienzo. Queremos que nuestro lienzo se coloque dentro del cuerpo de nuestro sitio web, y vamos a construir un lienzo que sea `400px` ancho y `250px` alto

```
const w = 400;
const h = 250;
```

```
let svg = d3.select ('body')
            .append ('svg')
            .attr ('width', w)
            .attr ('height', h);
```

Increíble. Ahora tenemos una variable llamada `svg` que hace referencia a nuestro `svg` elemento recién creado .

Ahora necesitamos seleccionar los rectángulos dentro de nuestro `svg` elemento. Podemos hacer esto usando el `selectAll` método. Este método seleccionará todos los `rect` elementos dentro de nuestro `svg` .

Pero recuerde, todavía no hemos `rect` creado ninguno . ¡Está bien! Esta parte puede ser algo confusa, pero esencialmente le estamos diciendo a D3 que tenga paciencia mientras creamos estos rectángulos:

```
svg.selectAll ('text')
```

Ahora podemos agregar nuestros datos al `svg` lienzo usando el `data()` método:

Usaremos los siguientes datos:

```
const data = [50,100,150,200,250,130,210,30,170];
```

Podemos encadenar esto usando:

```
.data (datos)
```

Increíble. Ahora necesitamos decirle a D3 que cree elementos de marcador de posición para cada uno de nuestros puntos de datos. Hacemos esto usando `.enter()`

Así es como se ve nuestro código hasta ahora:

```
svg.selectAll ('rect')
  .data (data)
  .enter ()
```

Todo lo que esté más allá de este punto se aplicará a todos nuestros nueve puntos de datos individualmente.

Lo primero que haremos es agregar un rectángulo para cada punto de datos y establecer algunos atributos en esos rectángulos. Como voy a establecer múltiples atributos para cada rectángulo, voy a usar el `.attrs({})` método que nos permite aplicar múltiples atributos a la vez con un objeto:

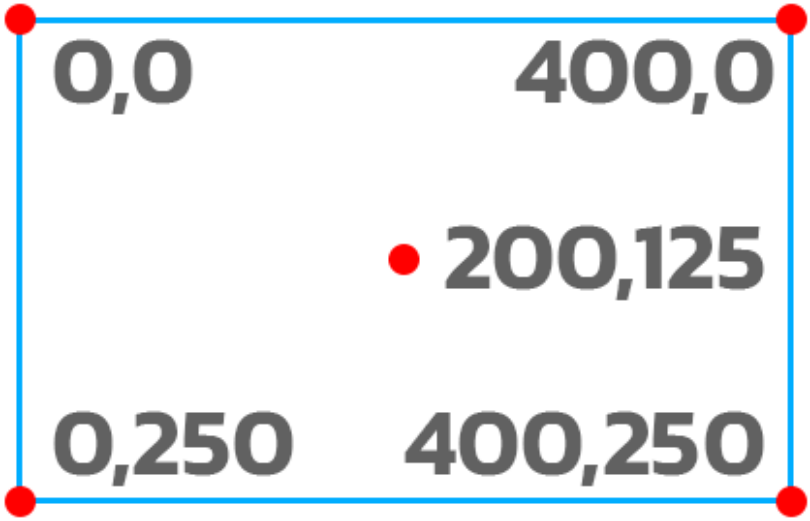
```
.append ('rect')
    .attrs ({
      // atributos aquí
    })
```

Los cinco atributos que necesitaremos para cada rectángulo son `x`, `y`, `width`, `height`, y `fill`.

- `x` - el valor inicial (izquierda-derecha) de cada rectángulo
- `y` - el valor inicial (arriba-abajo) de cada rectángulo
- `height` - la altura de cada rectángulo desde `y`
- `width` - el ancho de cada rectángulo desde `x`
- `fill` - el color de relleno de cada rectángulo

Antes de que podamos trabajar en estos atributos, debemos entender cómo funciona una grilla SVG. Recuerda desde antes que nuestro elemento SVG tiene 400 px de ancho y 250 px de alto. Una cuadrícula SVG siempre comienza en la esquina superior izquierda, que es el punto 0,0 y se mueve hacia abajo / hacia la derecha desde allí. Este es probablemente un punto de partida diferente de la mayoría de los sistemas basados en cuadrículas con los que ha trabajado en el pasado.

Echa un vistazo a este dibujo crudo para visualizar mejor esto:



Guay. Ahora que puede visualizar dónde están los puntos de datos en nuestro lienzo SVG, volvamos a crear nuestros atributos para cada rectángulo:

Nuestros `x` valores son bastante directos. Si el ancho de nuestra `svg` era `100px` y tuviéramos 10 elementos en total, cada uno sería `100 / 10 = 10` . Podemos aplicar el mismo concepto aquí. Nuestro `x` valor debe ser nuestro ancho dividido por la cantidad de puntos de datos. Entonces podemos multiplicar este valor multiplicado por el índice del rectángulo con el que estamos trabajando para asegurar que cada rectángulo se coloque correctamente:

```
x: (d, i) => i * (con data.length)
```

Nuestro `y` valor es un poco contra-intuitivo. Recuerde que la cuadrícula SVG comienza en la parte superior, no en la inferior. Esto significa que nuestro `y` valor será el alto del SVG menos el punto de datos actual:

```
y: d => h - d
```

El `width` de cada rectángulo será el ancho dividido por la cantidad total de rectángulos. También voy a agregar un pequeño relleno para separar un poco nuestros rectángulos:

```
const padding = 4;
```

```
ancho: w / data.length - relleno,
```

El `height` de nuestro rectángulo es simplemente nuestro punto de datos para ese rectángulo:

```
altura: d => d
```

Finalmente, el `fill` de nuestro rectángulo será simplemente verde:

```
llenar: 'verde'
```

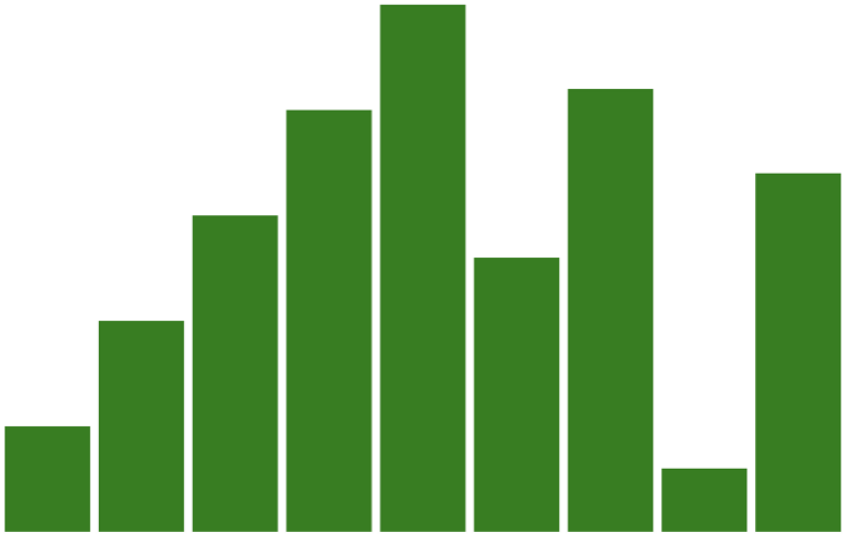
Si juntamos todo hasta ahora ... aquí está el código:

```
const w = 400, h = 250;
const padding = 4;
const data = [50,100,150,200,250,130,210,30,170];
```

```
let svg = d3.select ('body')
            .append ('svg')
            .attr ('width', w)
            .attr ('height', h);


svg.selectAll ('rect')
  .data (data)
  .enter ()
    .append ('rect')
    .attrs ({
      x: (d, i) => i * (w / data.length),
      y: d => h - d,
      ancho: w / data.length - relleno,
      altura: d => d,
      relleno: 'verde'
});
```

Y si ejecutamos este código obtenemos:



¡Increíble! Tenemos un gráfico de barras básico construido. Pero no hemos terminado todavía. Ahora vamos a agregar algo de texto superpuesto a nuestro gráfico.

Agregar el texto

Agregar nuestro texto comienza con un código casi idéntico a la creación de los rectángulos: '

```
svg.selectAll ('text')
  .data (data)
  .enter ()
    .append ('text')
```

Recuerde, este código dice a D3:

1. Seleccione todos los `text` elementos dentro de nuestro `svg` elemento. ¿Oh? No tenemos ninguno? Guay. Esta bien.
2. Toma esta información, úsala!
3. Aplique esta información a todos y cada uno de los `text` elementos. ¿Oh? No tenemos ninguno? Cool, crea marcadores de posición. Para cada marcador de posición, ejecute el código restante.
4. En realidad crea nuestros `text` elementos para cada elemento en nuestro SVG.

Ahora que hemos creado nuestros `text` elementos, necesitamos configurar nuestro texto y luego agregar algunos atributos.

Establecer el texto es bastante directo en esta instancia:

```
.text (d => d)
```

Y solo necesitamos dos atributos: la ubicación de nuestro texto `x` y su `y` ubicación.

El `y` valor es bastante directo. Queremos la altura del SVG menos la altura del punto de datos, exactamente la misma `y` fórmula que utilizamos para nuestro rectángulo. Luego agrego `20` píxeles adicionales como relleno.

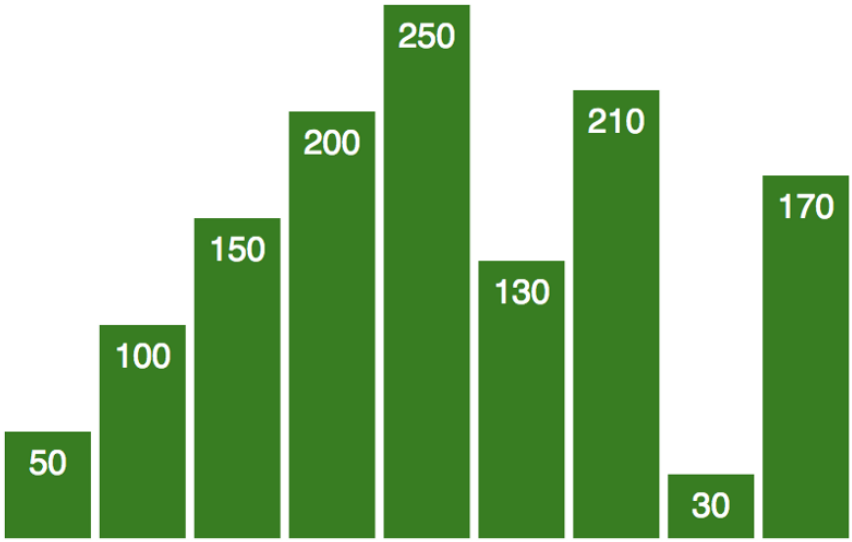
El `x` valor es un poco más complicado e implica un poco de matemáticas. Debido a que agregamos relleno, no podemos simplemente usar la `i * (w / data.length)` fórmula que usamos para el rectángulo. Tenemos que agregar en este relleno y luego dividir por dos para obtener el punto central de cada rectángulo:

```
x: (d, i) => i * (w / data.length) + (w / data.length -
relleno) / 2,
y: (d) => h - d + 20
```

Una vez que hayamos hecho esto, podemos agregar un poco de CSS para embellecer nuestro texto. A continuación, voy a cambiar la familia de fuentes, ponderar la fuente y anclar el texto en el centro:

```
texto {
  font-family: sans-serif;
  fill: #ffffff;
  texto-ancla: medio;
}
```

¡Y terminamos!



Así es como se ve nuestro código final:

```
const  w  =  400 , h =  250 ;
const  padding  =  4 ;
const  data  = [ 50 , 100 , 150 , 200 , 250 , 130 , 210 , 3

dejar svg =  d3 . seleccionar ( ' cuerpo ' )
             . append ( ' svg ' )
             . attr ( ' ancho ' , w)
             . attr ( ' altura ' , h);

svg . selectAll ( ' rect ' )
  . datos (datos)
  . enter ()
    . añadir ( ' rect ' )
    . attrs ({
      x : ( d , i ) => i * (w /  datos . de longitud ),
      y :  d  => h - d,
      ancho : w /  datos . longitud  - relleno,
      altura :  d  => d,
      llenar :  ' verde '
});
```

¡Increíble! Acabas de construir tu primer gráfico con D3. Sé que puede ser un poco abrumador al principio, pero una vez que descubres lo básico, solo se vuelve más fácil.

Notas de cierre:

Gracias por leer y espero que haya sido útil. Si estás listo para aprender JavaScript y visualización de datos más avanzados, echa un vistazo a **The Ultimate Guide to Learning Full Stack Web Development en 6 meses** .

Publico 4 artículos sobre desarrollo web cada semana. Considere **ingresar su correo electrónico aquí** si desea que lo agreguen a mi lista de correo electrónico semanal o síganme en **Twitter** .

Si esta publicación fue útil, haga clic en el botón aplaudir abajo varias veces para mostrar su apoyo. ⬇⬇