IIC2026
Visualización de Información

Hernán F. Valdivieso López
(2022 - 2 / Clase 09)

Antes de empezar... Revisión de contenidos (RC)

1. ¡No se olviden de las últimas actividades publicadas! En Canvas están los *links*.
2. Se acaba de publicar una actividad obligatoria. Consiste en **inventar una pregunta de alternativas** de algunos de los contenidos vistos en clases. En el enunciado se detalla cuáles contenidos pueden ser.
    - **Duración**: 3 semanas a partir de hoy.
    - **Intentos para responder**: ilimitados.
    - **Condición para obtener el punto RC**: cumplir las condiciones indicadas en el enunciado.
    - Pueden acercarse a **cualquier miembro del cuerpo docente** para tener *feedback* de esta actividad (solo *feedback*, no que le inventen la pregunta a ustedes).

Temas de la clase - Selecciones y Data joins en D3

1. Selecciones en D3.
    a. Comandos: `select` y `selectAll`.
    b. Comandos: `attr` y `style`.
    c. Method chaining.
    d. Comando: `append`.
2. Data joins en D3.
    a. En qué consiste el data joins.
    b. Agregar, actualizar y eliminar elementos con data joins.
    c. Comando: `join`.

Selecciones en D3

Selecciones en D3

Objetos en D3 que corresponden a una colección de elementos HTML.

`d3.selectAll("p")` → Una colección de todos los elementos con tag `<p>`.

2 formas de seleccionar:

- `select` → Selecciona el primer elemento de la búsqueda.
- `selectAll` → Selecciona todos los elementos de la búsqueda.

Selecciones en D3

Formas de seleccionar con `select`

- `d3.select("p")` → Por tag del elemento HTML.
- `d3.select(".class")` → Por la clase del elemento HTML.
- `d3.select("#id")` → Por el ID del elemento HTML.

Se aplica de igual forma al `selectAll`.

- `d3.selectAll("p")` → Por tag del elemento HTML.
- `d3.selectAll(".class")` → Por la clase del elemento HTML.
- `d3.selectAll("#id")` → Por el ID del elemento HTML.

Selecciones en D3 - Ejemplos

```
<body>
    <h1>Soy un título</h1>
    <p>Yo soy un párrafo</p>
    <p>¡Yo tambien :D!</p>
    <p class="uwu">¡Yo igual, pero tambien tengo clase</p>
    <p class="uwu">jejej yo tambien tengo clase</p>
    <h1 id="owo">Pero yo soy un título y tengo un ID único :P </h1>
</body>
```

Selecciones en D3 - Ejemplos

```
<body>
    <h1>Soy un título</h1>
    <p>Yo soy un párrafo</p>
    <p>¡Yo tambien :D!</p>
    <p class="uwu">¡Yo igual, pero tambien tengo clase</p>
    <p class="uwu">jejej yo tambien tengo clase</p>
    <h1 id="owo">Pero yo soy un título y tengo un ID único :P </h1>
</body>
```

```
d3.select("h1")
```

Selecciones en D3 - Ejemplos

```
<body>
    <h1>Soy un título</h1>
    <p>Yo soy un párrafo</p>
    <p>¡Yo tambien :D!</p>
    <p class="uwu">¡Yo igual, pero tambien tengo clase</p>
    <p class="uwu">jejej yo tambien tengo clase</p>
    <h1 id="owo">Pero yo soy un título y tengo un ID único :P </h1>
</body>
```

```
d3.select(".uwu")
```

Selecciones en D3 - Ejemplos

```
<body>
    <h1>Soy un título</h1>
    <p>Yo soy un párrafo</p>
    <p>¡Yo tambien :D!</p>
    <p class="uwu">¡Yo igual, pero tambien tengo clase</p>
    <p class="uwu">jejej yo tambien tengo clase</p>
    <h1 id="owo">Pero yo soy un título y tengo un ID único :P </h1>
</body>
```

```
d3.select("#owo")
```

Selecciones en D3 - Ejemplos

```
<body>
    <h1>Soy un título</h1>
    <p>Yo soy un párrafo</p>
    <p>¡Yo tambien :D!</p>
    <p class="uwu">¡Yo igual, pero tambien tengo clase</p>
    <p class="uwu">jejej yo tambien tengo clase</p>
    <h1 id="owo">Pero yo soy un título y tengo un ID único :P </h1>
</body>
```

```
d3.selectAll("h1")
```

Selecciones en D3 - Ejemplos

```
<body>
    <h1>Soy un título</h1>
    <p>Yo soy un párrafo</p>
    <p>¡Yo tambien :D!</p>
    <p class="uwu">¡Yo igual, pero tambien tengo clase</p>
    <p class="uwu">jejej yo tambien tengo clase</p>
    <h1 id="owo">Pero yo soy un título y tengo un ID único :P </h1>
</body>
```

```
d3.selectAll(".uwu")
```

Selecciones en D3 - Ejemplos

```
<body>
    <h1>Soy un título</h1>
    <p>Yo soy un párrafo</p>
    <p>¡Yo tambien :D!</p>
    <p class="uwu">¡Yo igual, pero tambien tengo clase</p>
    <p class="uwu">jejej yo tambien tengo clase</p>
    <h1 id="owo">Pero yo soy un título y tengo un ID único :P </h1>
</body>
```

```
d3.selectAll("#owo")
```

Selecciones en D3 - Ejemplos

```
<body>
    <h1>Soy un título</h1>
    <p>Yo soy un párrafo</p>
    <p>¡Yo tambien :D!</p>
    <p class="uwu">¡Yo igual, pero tambien tengo clase</p>
    <p class="uwu">jejej yo tambien tengo clase</p>
    <h1 id="owo">Pero yo soy un título y tengo un ID único :P </h1>
</body>
```

```
d3.selectAll("p")
```

Selecciones en D3 - Ejemplos

```
<body>
    <div>
        <h1>Soy un título</h1>
        <h1>Soy otro título</h1>
    </div>
    <div>
        <h1>Soy un título</h1>
        <h1>Soy otro título</h1>
    </div>
    <p>Yo soy un párrafo</p>
    <h1>Yo no!!!</h1>
    <h1>Soy un título fuera de un contenedor</h1>
</body>
```

Quiero los H1 dentro de los divs

```
selection.selectAll("h1") ????
```

❌ Esto también incluirá los últimos h1

Solución:

```
const selection = d3.selectAll("div")
selection.selectAll("h1")
```

✅ Primero seleccionamos los div. Luego seleccionamos los h1 dentro de esos divs.

Selecciones en D3 - Attr y style

seleccion.attr(atributo, valor)

Define un atributo y su valor a cada elemento de la selección. Si ese atributo ya existe, sobreescribe su valor.

```
<body>
    <p>Mi párrafo es un anime: Full Metal Alchemists</p>
    <p class="uwu">Pero mi párrafo es un anime con clase: Gintama</p>
</body>
```

Antes

```
const seleccion = d3.selectAll("p");
seleccion.attr("class", "nani!!!");
```

```
<body>
    <p class="nani!!!">Mi párrafo es un anime: Full Metal Alchemists</p>
    <p class="nani!!!">Pero mi párrafo es un anime con clase: Gintama</p>
</body>
```

Después

Aprendizaje random: nani es una expresión en japonés que se puede entender como qué!!! (es muy ocupada en los anime y mangas).

Selecciones en D3 - Attr y style

seleccion.style(propiedad, valor)

Agrega una propiedad de CSS y su valor a cada elemento de la selección. Si esa propiedad del CSS ya existe, sobreescribe su valor.

```
<body>
    <p>Full Metal Alchemists</p>
    <p class="uwu" style="color: red;">Gintama</p>
</body>
```

Antes

```
const seleccion = d3.selectAll("p");
seleccion.style("color", "orange");
```

```
<body>
    <p style="color: orange;">Full Metal Alchemists</p>
    <p class="uwu" style="color:orange;">Gintama</p>
</body>
```

Después

Selecciones en D3 - Funciones en Attr y style

seleccion.style(propiedad, valor) - seleccion.attr(atributo, valor)

En ambos casos, el valor puede ser una constante (como en los ejemplos anteriores) o una función.

```
<body>
    <p>Full Metal Alchemists</p>
    <p class="uwu" style="color: red;">Gintama</p>
</body>
```

Antes

```
const seleccion = d3.selectAll("p");
seleccion.style("fill", () => "orange");
```

```
<body>
    <p style="color: orange;">Full Metal Alchemists</p>
    <p class="uwu" style="color:orange;">Gintama</p>
</body>
```

Después

Selecciones en D3 - Funciones en Attr y style

seleccion.style(propiedad, valor) - seleccion.attr(atributo, valor)

En ambos casos, el valor puede ser una constante (como en los ejemplos anteriores) o una función.

```
<svg>
    <rect></rect>
    <rect></rect>
</svg>
```

Antes

```
const seleccion = d3.selectAll("rect");
seleccion.attr("x", (_, i, _) => i*100); // Segundo argumento → índice en la selección
```

```
<svg>
    <rect x="0"></rect>
    <rect x="100"></rect>
</svg>
```

Después

Selecciones en D3 - Method chaining

Métodos como `attr` y `style` retornan la misma selección.

```
const selection = d3.selectAll("rect");
selection.attr("x", 10);
selection.attr("y", 50);
selection.style("fill", "orange");
```

Es equivalente a:

```
d3.selectAll("rect").attr("x", 10).attr("y", 50).style("fill", "orange");
```

De forma más ordenada:

```
d3.selectAll("rect")
    .attr("x", 10)
    .attr("y", 50)
    .style("fill", "orange");
```

Selecciones en D3 - Append

seleccion.append(tag)

Crea un nuevo elemento dentro de cada elemento de la selección sobre la cual actúa con el tag definido en el método.

```
<body>
    <div></div>
    <div></div>
</body>
```

Antes

```
<body>
    <div>
        <h1></h1>
    </div>
    <div>
        <h1></h1>
    </div>
</body>
```

Después

```
const selection = d3.selectAll("div")
selection.append("h1")
```

Selecciones en D3 - Append

seleccion.append(tag)

Crea un nuevo elemento dentro de cada elemento de la selección sobre la cual actúa con el tag definido en el método.

```
<body>
    <div></div>
    <div></div>
</body>
```

Antes

```
<body>
    <div>
        <h1>uwu</h1>
    </div>
    <div>
        <h1>uwu</h1>
    </div>
</body>
```

Después

```
const selection = d3.selectAll("div")
selection.append("h1").text("uwu");
```

👀 **selection.text(texto)**

Agrega un texto dentro del elemento HTML. Notar que este texto no tiene un tag asociado.

Selecciones en D3 - Append

seleccion.append(tag)

Crea un nuevo elemento dentro de cada elemento de la selección sobre la cual actúa con el tag definido en el método.

```
<body>
    <div></div>
    <div></div>
</body>
```

Antes

```
<body>
    <div>
        <h1>uwu</h1>
        <h1>uwucito</h1>
    </div>
    <div>
        <h1>uwu</h1>
        <h1>uwucito</h1>
    </div>
</body>
```

Después

```
const selection = d3.selectAll("div")
selection.append("h1").text("uwu");
selection.append("h1").text("uwucito");
```

Selecciones en D3 - Append

seleccion.append(tag)

Crea un nuevo elemento dentro de cada elemento de la selección sobre la cual actúa con el tag definido en el método.

```
<body>
    <div></div>
    <div></div>
</body>
```

Antes

```
<body>
    <div>
        <h1 style="color:orange;">uwu</h1>
        <h1>uwucito</h1>
    </div>
    <div>
        <h1 style="color:orange;">uwu</h1>
        <h1>uwucito</h1>
    </div>
</body>
```

Después

```
const selection = d3.selectAll("div")
selection.append("h1").style("color", "orange").text("uwu");
selection.append("h1").text("uwucito");
```

Selecciones en D3 - Resumen de comandos

- **`d3.select(selector)`**: Selecciona el primer elemento según el `selector` indicado.
- **`d3.selectAll(selector)`**: Selecciona todos los elementos según el `selector` indicado.
- **`seleccion.style(propiedad, valor)`**: Agrega una `propiedad` de CSS junto con su `valor` a cada elemento de la selección.
- **`seleccion.attr(atributo, valor)`**: Agrega un `atributo` y su `valor` a cada elemento de la selección.
- **`seleccion.text(texto)`**: Agrega un `texto` dentro de cada elemento de la selección.
- **`seleccion.append(tag)`**: Agrega un nuevo elemento HTML (o svg) con el `tag` indicado dentro de cada elemento de la selección.

Data joins

Data Joins

Vincular datos a elementos del SVG.

Podemos generar un vínculo de marcas y canales con datos mediante código.

```
const datos = [23, 45, 120, 64]

const datos = [“uwu”, “awa”, “owo”]

const datos = [
 {title: "Spy x Family", year: 2019},
 {title: "Kaguya-sama: Love Is War", year: 2019},
 {title: "Deaimon", year: 2016},
];
```

```
<svg>
    <rect></rect>
    <text></text>
    <circle></circle>
    <path></path>
</svg>
```

Data Joins - Comando data

selection.data(lista_datos)

Comando para vincular una selección de elementos a una lista de datos

```
const datos = [23, 45, 120, 64]
selection.data(datos)
```

Ejemplo:

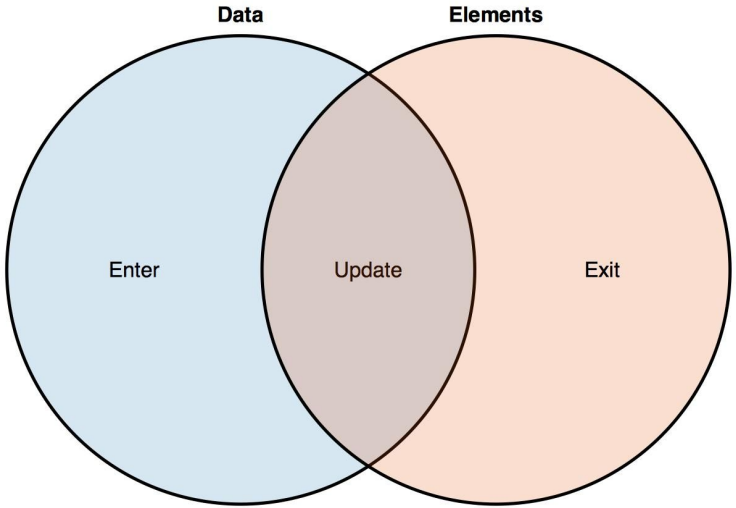
```
d3.selectAll("rect").data(datos)
```

Data Joins - Comando data

selection.data(lista_datos)

Cuando se ejecuta este comando, se generan **3 nuevas selecciones**:

- **Enter**: datos que no quedan asociados a ningún elemento.
- **Update**: datos asociados a algún elemento de la selección.
- **Exit**: elementos que no quedan asociados a ningún dato.



Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 1

Cantidad de datos == cantidad de elementos

```
const datos = [23, 45, 99, 64]
d3.select("svg").selectAll("rect").data(datos);
```

¿Qué ocurre?

```
<svg id="svg" width="400" height="250">
 <rect></rect> <!-- 23 -->
 <rect></rect> <!-- 45 -->
 <rect></rect> <!-- 99 -->
 <rect></rect> <!-- 64 -->
</svg>
```

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 1

Cantidad de datos == cantidad de elementos

```
const datos = [23, 45, 99, 64]
d3.select("svg").selectAll("rect").data(datos).attr("height", 40);
```

¿Qué ocurre?

```
<svg id="svg" width="400" height="250">
 <rect height=”40”></rect> <!-- 23 -->
 <rect height=”40”></rect> <!-- 45 -->
 <rect height=”40”></rect> <!-- 99 -->
 <rect height=”40”></rect> <!-- 64 -->
</svg>
```

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 1

Cantidad de datos == cantidad de elementos

```
const datos = [23, 45, 99, 64]
d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
```

¿Qué ocurre?

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23"></rect> <!-- 23 -->
 <rect height="40" width="45"></rect> <!-- 45 -->
 <rect height="40" width="99"></rect> <!-- 99 -->
 <rect height="40" width="64"></rect> <!-- 64 -->
</svg>
```

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 1

Cantidad de datos == cantidad de elementos

```
const datos = [23, 45, 99, 64]
d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);
```

¿Qué ocurre?

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
 <rect height="40" width="99" y="100"></rect> <!-- 99 -->
 <rect height="40" width="64" y="150"></rect> <!-- 64 -->
</svg>
```

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 1

Cantidad de datos == cantidad de elementos

```
const datos = [23, 45, 99, 64]
const update = d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);
```

Por defecto, `.data(...)` va a retornar la selección de **update.** La guardaremos en una variable llamada `update` para no olvidarnos de esto.

¿Qué ocurre?

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
 <rect height="40" width="99" y="100"></rect> <!-- 99 -->
 <rect height="40" width="64" y="150"></rect> <!-- 64 -->
</svg>
```

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 2

Cantidad de datos < cantidad de elementos

```
const datos = [23, 45]
const update = d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);
```

¿Qué ocurre?

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
 <rect></rect> <!-- ?? -->
 <rect></rect> <!-- ?? -->
</svg>
```

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 2

Cantidad de datos < cantidad de elementos

```
const datos = [23, 45]
const update = d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);

const sinDatos = update.exit()
```

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
 <rect></rect> <!-- ?? -->
 <rect></rect> <!-- ?? -->
</svg>
```

exit nos da acceso a todos los elementos de la selección anterior **sin datos asociados.**

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 2

Cantidad de datos < cantidad de elementos

```
const datos = [23, 45]
const update = d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);

const sinDatos = update.exit().remove()
```

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
</svg>
```

Hacemos `.remove()` para eliminar esos datos del SVG.

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 3

Cantidad de datos > cantidad de elementos

```
const datos = [23, 45, 99, 64]
const update = d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);
```

¿Qué ocurre?

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
 ?? <!-- 99 -->
 ?? <!-- 64 -->
</svg>
```

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 3

Cantidad de datos > cantidad de elementos

```
const datos = [23, 45, 99, 64]
const update = d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);

const sinElementos = update.enter()
```

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
 ?? <!-- 99 -->
 ?? <!-- 64 -->
</svg>
```

enter nos da acceso a todos los datos **sin elementos asociados.**

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 3

Cantidad de datos > cantidad de elementos

```
const datos = [23, 45, 99, 64]
const update = d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);

const sinElementos = update.enter().append("rect")
```

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
 <rect></rect> <!-- 99 -->
 <rect></rect> <!-- 64 -->
</svg>
```

Hacemos `.append(tag)` para agregar un elemento a cada dato de esta selección.

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 3

Cantidad de datos > cantidad de elementos

```
const datos = [23, 45, 99, 64]
const update = d3.select("svg").selectAll("rect").data(datos).attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);

const sinElementos = update.enter().append("rect").attr("height", 40)
  .attr("width", (d, i, _) => d)
  .attr("y", (d, i, _) => i * 50);
```

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
 <rect height="40" width="99" y="100"></rect> <!-- 99 -->
 <rect height="40" width="64" y="150"></rect> <!-- 64 -->
</svg>
```

¿En verdad tenemos que replicar todo el código?

Data Joins - Comando data

```
// Caso original
<svg id="svg" width="400" height="250">
 <rect></rect>
 <rect></rect>
</svg>
```

Caso 3

Cantidad de datos > cantidad de elementos

```
const datos = [23, 45, 99, 64]
const update = d3.select("svg").selectAll("rect").data(datos);
const sinElementos = update.enter().append("rect");

update.merge(sinElementos).attr("height", 40)
                          .attr("width", (d, i, _) => d)
                          .attr("y", (d, i, _) => i * 50);
```

```
<svg id="svg" width="400" height="250">
 <rect height="40" width="23" y="0"></rect> <!-- 23 -->
 <rect height="40" width="45" y="50"></rect> <!-- 45 -->
 <rect height="40" width="99" y="100"></rect> <!-- 99 -->
 <rect height="40" width="64" y="150"></rect> <!-- 64 -->
</svg>
```

¿En verdad tenemos que replicar todo el código?

No🙅‍♂️. Usamos el comando `merge`.

Data Joins - Ejemplo clase 7

```
// Caso original
<body>
</body>
```

```
const datos = [150, 256, 130, 0, 11, 420, 235];
// Encontrar el body en el DOM y agregar un elemento SVG. Luego definir ancho y largo
const svg = d3.select("body").append("svg");
svg.attr("width", 50 + datos.length * 100).attr("height", 500);
// Agregar “rect” y personalizar según los datos que tenemos
svg
  .selectAll("rect")
  .data(datos)
  .enter()
  .append("rect")
  .attr("width", 50)
  .attr("fill", "orange")
  .attr("height", (d) => d)
  .attr("x", (_, index) => 50 + index * 100);
```

Vinculamos rectángulos a los datos.

Como no hay ningún `rect` al inicio. Vamos directamente con la selección de **`enter`** y le hacemos `append` de un `rect` a cada dato.

Personalizamos cada `rect`

Data Joins - Comando Join

Desde **D3.js V5** se creó este comando para facilitar el proceso de data joins.

- Aunque en internet abundan ejemplos donde no usan este comando 😓 .

```
const datos = [23, 45, 99, 64]
const update_and_enter = d3.selectAll("rect").data(datos).join("rect");
```

`.join("rect")` hará 3 acciones de forma interna:

1. Todos los elementos `rect` que no tengan asociado un dato les hace `.remove()`.
2. Todos los datos que no tengan asociado un elemento, les hace `append("rect")`
3. Retorna `update.merge(enter)`.

```
update_and_enter.attr("height", 40)
    .attr("width", (d, i, _) => d)
    .attr("y", (d, i, _) => i * 50);
```

Podemos personalizar de inmediato todos los rect que correspondan

Data Joins - Comando Join

¿Si quiero trabajar con cada conjunto por separado primero?

En vez de entregar un string, podemos definir 3 funciones que se encargará de procesar cada selección.

```
const update_and_enter = d3.selectAll("rect").data(datos).join(
    (enter) => enter.append("rect"), // función 1: sólo recibe selección de enter
    (update) => update,              // función 2: sólo recibe selección de update
    (exit) => exit.remove(),         // función 3: sólo recibe selección de exit
);
```

En la próxima ayudantía (no la de esta semana) verán ejemplos de usar `join` con estas 3 funciones.

Data Joins - Ejemplo de estudio para la casa

Míni actividad formativa (no entrega puntos de RC).

1. Revise el `ejemplo_5.html` y `programa_5.js`. Luego responda las siguientes preguntas:
    a. ¿Qué está haciendo el código?
    b. ¿Qué sucede si comentamos/eliminamos la línea 13?
    c. ¿Qué sucede si comentamos/eliminamos las líneas 16 y 21?
2. Revise el `ejemplo_6.html` y `programa_6.js`. Luego responda las siguientes preguntas:
    a. ¿Este código hace algo distinto al ejemplo 5?
    b. ¿Qué pasó con la parte de eliminar los elementos de la selección **`exit`**?
    c. ¿Por qué ya no hacemos **`merge`**?

*comentar: agregar // al inicio de la línea.

Próximos eventos

Próxima clase

- Clase de color.
- Desde dicha clase tendrán 1 semana para la actividad bonus que adelanté hace un par de semanas (el test de daltonismo).

Ayudantía de mañana

- Repasar Javascript.
- Usar D3 (sin data join) para confeccionar SVG.

No olviden

La Tarea 1 se entrega **este viernes a las 20:00**. Tienen hasta 4 días para entregas atrasadas (con descuento de 5 décimas por día) y si necesitan más de esos 4 días, contactarme a mi o la ayudante de bienestar para evaluar una flexibilidad.

IIC2026
Visualización de Información

Hernán F. Valdivieso López
(2022 - 2 / Clase 09)