

---

# IIC2026

# Visualización de Información

— Hernán F. Valdivieso López —  
(2023 - 2 / Clase 06)

---

# Temas de la clase - Data join 2 en D3

## 1. *Data joins*

- a. Actualizar elementos (*update*)
- b. Actualizar y crear elementos (*update y enter*)
- c. Eliminar elementos (*exit*)
- d. Actualizar y eliminar al mismo tiempo (*update y exit*)

## 2. Data join avanzado

- a. Personalizar *enter*, *update* y *exit*
- b. Múltiples *joins* por dato

# Data joins

Parte 1  
(recordatorio *flash*)

---

# 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},  
];
```



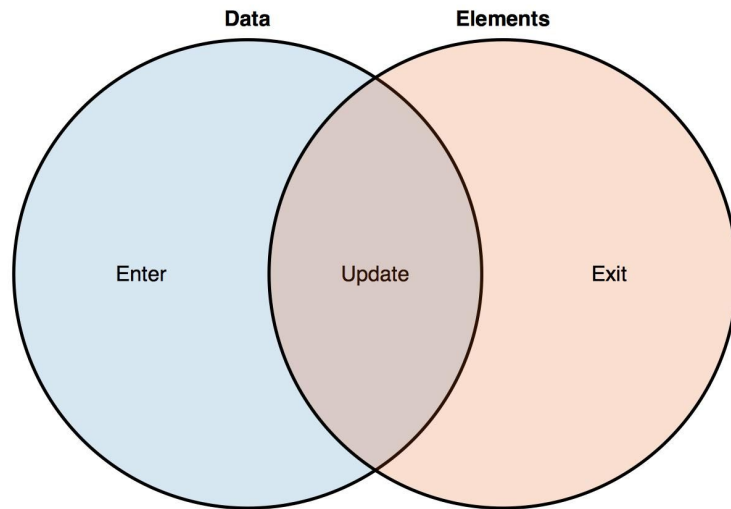
```
<svg>  
  <rect></rect>  
  <text></text>  
  <circle></circle>  
  <path></path>  
</svg>
```

# Data Joins - Comando data

`selection.data(lista_datos)`

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 join

// Caso original

```
<svg id="vis" width="400" height="250">  
</svg>
```

## Agregar elementos vinculados a datos

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

¿Qué ocurre?

```
<svg id="vis" 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

## Parte 2

---

# Data Joins - Comando join

## Actualizar elementos

```
const datos = [99, 64]
d3.select("#vis").selectAll("rect").data(datos)
```

¿Qué ocurre?

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

```
// Caso original
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect>
  <rect height="40" width="45"></rect>
</svg>
```



# Data Joins - Comando join

## Actualizar elementos

```
const datos = [99, 64]
d3.select("#vis").selectAll("rect").data(datos).join("rect")
```

```
// Caso original
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect>
  <rect height="40" width="45"></rect>
</svg>
```

join verá que ya hay 2 elementos, así que no creará más.

¿Qué ocurre?

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

# Data Joins - Comando join

## Actualizar elementos

```
const datos = [99, 64]
```

```
d3.select("#vis").selectAll("rect").data(datos).join("rect").attr("height", 12)
```

```
// Caso original
```

```
<svg id="vis" width="400" height="250">  
  <rect height="40" width="23"></rect>  
  <rect height="40" width="45"></rect>  
</svg>
```

join verá que ya hay 2 elementos, así que no creará más.

¿Qué ocurre?

```
<svg id="vis" width="400" height="250">  
  <rect height="12" width="23"></rect> <!-- 99 -->  
  <rect height="12" width="45"></rect> <!-- 64 -->  
</svg>
```

# Data Joins - Comando join

## Actualizar elementos

```
const datos = [99, 64]
d3.select("#vis").selectAll("rect").data(datos).join("rect").attr("height", 12)
    .attr("width", (dato, i, _) => dato)
```

```
// Caso original
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect>
  <rect height="40" width="45"></rect>
</svg>
```

join verá que ya hay 2 elementos, así que no creará más.

¿Qué ocurre?

```
<svg id="vis" width="400" height="250">
  <rect height="12" width="99"></rect> <!-- 99 -->
  <rect height="12" width="64"></rect> <!-- 64 -->
</svg>
```

# Data Joins - Comando join

## Actualizar elementos

```
const datos = [99, 64]
d3.select("#vis").selectAll("rect").data(datos).join("rect").attr("height", 12)
  .attr("width", (dato, i, _) => dato)
  .attr("y", (d, i, _) => i * 50);
```

¿Qué ocurre?

```
<svg id="vis" width="400" height="250">
  <rect height="12" width="99" y="0"></rect> <!-- 99 -->
  <rect height="12" width="64" y="50"></rect> <!-- 64 -->
</svg>
```

```
// Caso original
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect>
  <rect height="40" width="45"></rect>
</svg>
```

join verá que ya hay 2 elementos, así que no creará más.

# Data Joins - Comando join

## Actualizar y crear elementos

```
const datos = [99, 64, 22]  
d3.select("#vis").selectAll("rect").data(datos)
```

¿Qué ocurre?

```
<svg id="vis" width="400" height="250">  
  <rect height="40" width="23"></rect> <!-- 99 -->  
  <rect height="40" width="45"></rect> <!-- 64 -->  
  ?? <!-- 22 -->  
</svg>
```

```
// Caso original  
<svg id="vis" width="400" height="250">  
  <rect height="40" width="23"></rect>  
  <rect height="40" width="45"></rect>  
</svg>
```

# Data Joins - Comando join

## Actualizar y crear elementos

```
const datos = [99, 64, 22]
d3.select("#vis").selectAll("rect").data(datos).join("rect")
```

```
// Caso original
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect>
  <rect height="40" width="45"></rect>
</svg>
```

join verá que ya hay 2 elementos pero son 3 datos → creará uno más.

¿Qué ocurre?

```
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect> <!-- 99 -->
  <rect height="40" width="45"></rect> <!-- 64 -->
  <rect></rect> <!-- 22 -->
</svg>
```

# Data Joins - Comando join

## Actualizar y crear elementos

```
const datos = [99, 64, 22]
d3.select("#vis").selectAll("rect").data(datos).join("rect").attr("height", 12)
  .attr("width", (dato, i, _) => dato)
  .attr("y", (d, i, _) => i * 50);
```

```
// Caso original
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect>
  <rect height="40" width="45"></rect>
</svg>
```

join verá que ya hay 2 elementos pero son 3 datos → creará uno más.

¿Qué ocurre?

```
<svg id="vis" width="400" height="250">
  <rect height="12" width="99" y="0"></rect> <!-- 99 -->
  <rect height="12" width="64" y="50"></rect> <!-- 64 -->
  <rect height="12" width="20" y="100"></rect> <!-- 22 -->
</svg>
```

# Data Joins - Comando join

## Eliminar elementos

```
const datos = [99]
d3.select("#vis").selectAll("rect").data(datos)
```

¿Qué ocurre?

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

```
// Caso original
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect>
  <rect height="40" width="45"></rect>
</svg>
```



# Data Joins - Comando join

## Eliminar elementos

```
const datos = [99]
d3.select("#vis").selectAll("rect").data(datos).join("rect")
```

// Caso original

```
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect>
  <rect height="40" width="45"></rect>
</svg>
```

join verá que ya hay 2 elementos pero es solo 1 dato → eliminará 1 rect.

¿Qué ocurre?

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

# Data Joins - Comando join

## Eliminar elementos y actualizar

```
const datos = [99]
d3.select("#vis").selectAll("rect").data(datos).join("rect").attr("height", 12)
  .attr("width", (dato, i, _) => dato)
  .attr("y", (d, i, _) => i * 50);
```

¿Qué ocurre?

```
<svg id="vis" width="400" height="250">
  <rect height="12" width="99" y="0"></rect> <!-- 99 -->
</svg>
```

// Caso original

```
<svg id="vis" width="400" height="250">
  <rect height="40" width="23"></rect>
  <rect height="40" width="45"></rect>
</svg>
```

join verá que ya hay 2 elementos pero es solo 1 dato → eliminará 1 rect.

# Data Joins - Ejemplo

Estudiaremos un ejemplo donde agregaremos rectángulos naranjos en función de diferentes botones:

- Botón 1 hará data join de `rect` con la lista `[1, 2, 3]`.
- Botón 2 hará data join de `rect` con la lista `[5, 6]`.
- Botón 3 hará data join de `rect` con la lista `[5, 10]`.
- Botón 4 eliminará todos los `rect` del `html`.

# Data Joins

Vamos al código    
(ejemplo1)

# Data joins

Avanzado

---

# Data Joins Avanzado

Vamos a personalizar cuando agregamos datos, actualizamos y eliminamos datos.

Para esto, en vez de escribir `join("rect")`, escribimos 3 funciones dentro del join.

- Una para los elementos nuevos asociados a datos
- Otra para los elementos existentes asociados a datos
- Una última para los elementos sin dato asociado.

```
d3.selectAll("rect").data(datos).join(  
  (enter) => enter.append("rect"), // función 1: selección de datos sin elementos  
  (update) => update,              // función 2: selección de elementos ya existentes  
  (exit) => exit.remove(),          // función 3: selección de elementos que perdieron su dato  
);
```

**Los nombres "enter", "update" y "exit" son solo nombre para facilitar la división de las 3 funciones. Pueden llamarse "nana", "nene" y "nini", y funcionará**

# Data Joins Avanzado

Los nombres "enter", "update" y "exit" son solo nombre para facilitar la división de las 3 funciones. Pueden llamarse "nana", "nene" y "nini", y funcionará

```
d3.selectAll("rect").data(datos).join(  
  (enter) => enter.append("rect"), // función 1: selección de datos sin elementos  
  (update) => update,             // función 2: selección de elementos ya existentes  
  (exit) => exit.remove(),        // función 3: selección de elementos que perdieron su dato  
);
```

```
d3.selectAll("rect").data(datos).join(  
  (nana) => nana.append("rect"), // función 1: selección de datos sin elementos  
  (nene) => nene,                 // función 2: selección de elementos ya existentes  
  (nini) => nini.remove(),        // función 3: selección de elementos que perdieron su dato  
);
```

# Data Joins Avanzado

```
d3.selectAll("rect").data(datos).join(  
  enter => enter.append("rect")  
    .attr("height", 20)  
    .attr("x", 0)  
    .attr("fill", "orange")  
    .attr("width", d => d * 10)  
    .attr("y", (d, i) => i * 40)  
  ,  
  update => update.attr("width", d => d * 10).attr("x", 110).attr("fill", "red")  
  ,  
  exit => exit.attr("x", 220).attr("fill", "green")  
);
```

enter será la selección de datos que no están vinculados a elementos. Lo primero es agregarles un rect a cada dato y luego los personalizamos.



# Data Joins Avanzado

```
d3.selectAll("rect").data(datos).join(  
  enter => enter.append("rect")  
    .attr("height", 20)  
    .attr("x", 0)  
    .attr("fill", "orange")  
    .attr("width", d => d * 10)  
    .attr("y", (d, i) => i * 40)  
  
  ,  
  update => update.attr("width", d => d * 10).attr("x", 110).attr("fill", "red")  
  
  ,  
  exit => exit.attr("x", 220).attr("fill", "green")  
);
```

update será la selección de elementos que quedaron vinculados a un dato.  
Solo necesitamos actualizar sus atributos

# Data Joins Avanzado

```
d3.selectAll("rect").data(datos).join(  
  enter => enter.append("rect")  
    .attr("height", 20)  
    .attr("x", 0)  
    .attr("fill", "orange")  
    .attr("width", d => d * 10)  
    .attr("y", (d, i) => i * 40)  
  
  ,  
  update => update.attr("width", d => d * 10).attr("x", 110).attr("fill", "red")  
  
  ,  
  exit => exit.attr("x", 220).attr("fill", "green")  
);
```

exit será la selección de elementos que NO quedaron vinculados a un dato. Podemos personalizar sus atributos

# Data Joins Avanzado

```
d3.selectAll("rect").data(datos).join(  
  enter => enter.append("rect")  
    .attr("height", 20)  
    .attr("x", 0)  
    .attr("fill", "orange")  
    .attr("width", d => d * 10)  
    .attr("y", (d, i) => i * 40)  
  
  ,  
  update => update.attr("width", d => d * 10).attr("x", 110).attr("fill", "red")  
  
  ,  
  exit => exit.remove()  
);
```

exit será la selección de elementos que NO quedaron vinculados a un dato. Podemos personalizar sus atributos • **eliminar esos elementos**

# Data Joins Avanzado

Vamos al código    
(ejemplo2)

# Data Joins Avanzado

Si tuviéramos más información por dato. Por ejemplo:

```
[  
  { nombre: "Alex", nota_promedio: 52, desviacion: 3 },  
  { nombre: "Fran", nota_promedio: 60, desviacion: 9 },  
  { nombre: "Luis", nota_promedio: 40, desviacion: 16 },  
  { nombre: "Pepe", nota_promedio: 90, desviacion: 10 }  
]
```

Y queremos hacer un mini gráfico de barra que incluya el nombre del estudiante y la desviación.

🤔 ¿Cómo lo hacemos?

# Data Joins Avanzado

Si tuviéramos más información por dato. Por ejemplo:

```
[  
  { nombre: "Alex", nota_promedio: 52, desviacion: 3 },  
  { nombre: "Fran", nota_promedio: 60, desviacion: 9 },  
  { nombre: "Luis", nota_promedio: 40, desviacion: 16 },  
  { nombre: "Pepe", nota_promedio: 90, desviacion: 10 }  
]
```

Y queremos hacer un mini gráfico de barra que incluya el nombre del estudiante y la desviación.

🤔 ¿Cómo lo hacemos?

**Múltiples append dentro del data join**

# Data Joins Avanzado + Contenedores (g)

Vamos al código    
(ejemplo3)

# Próximos eventos

## Próxima clase

- Teoría cuando trabajamos con datos tabulares

## Próxima ayudantía

- Construir una visualización más sofisticada (*full data join*)

## Tarea 1

- Se entrega **este viernes a las 20:00**. Tienen hasta 3 días para entregas atrasadas (con descuento de 5 décimas por día), es decir, hasta el lunes a las 20:00.



---

# IIC2026

## Visualización de Información

— Hernán F. Valdivieso López —  
(2023 - 2 / Clase 06)

---