Cápsula 3: *Join* de datos personalizado

Hola, bienvenidxs a una cápsula del curso Visualización de Información. En esta hablaré más sobre un aspecto del *join* de datos que dejé pendiente hace unas cápsulas, el join de datos personalizado.

Primero recordemos que era el *join* de datos. Este es el proceso que realiza D3 al llamar al método “data” sobre una selección de elementos. Este proceso intenta efectuar un vínculo entre datos y los elementos en la selección. En este proceso aparecen tres selecciones resultantes: *enter*, *update*, y *exit*.

El flujo normal era agregar elementos para aquellos datos en *enter*; eliminar los elementos que sobran de *exit*, y actualizar aquellos elementos vinculados a datos exitosamente. Intentemos ver esto de forma visual.

Cuando comienza un programa en D3, la selección que creamos generalmente aún no contiene elementos. Una vez que se cargan datos en nuestro arreglo, intentaremos realizar el vínculo con elementos. La forma en que intenta hacer el vínculo es mediante una llave en el lado de elementos y de datos, por defecto, esta llave es el índice en su agrupación.

En esta primera situación, la llave de cada datos es su índice en el arreglo. Como no hay elementos en la selección, ningún dato queda vinculado, por lo que todos estarán en la selección *enter*. Y tras llamar a “append” sobre *enter*, se crean elementos para cada dato.

Ahora, digamos ejecutamos el *join* de datos con esta situación actualizada. Ahora, hay elementos que también pueden calcular su índice dentro del grupo para hacer vínculos. Como los índices coinciden entre todos los elementos y datos, entonces todos caen en la selección *update*.

Hasta el momento en nuestro ejemplo no ha sido problema, incluso considerando que agregamos nuevos datos. Esto es gracias a que agregamos nuevos datos siempre al final del arreglo de datos, por lo que el orden de los datos anteriores siempre es el mismo y agrega elementos siempre al final de los grupos.

Esto cambiará si alteramos el orden de los datos por si algo pasa, digamos, desaparecen elementos. Para verlo en práctica, agreguemos esta posibilidad a nuestro programa. Específicamente, que al hacer clic sobre una barra, esta se elimine.

Aquí agregamos el código que mediante clic elimine el dato asociado a la barra, y vuelva a llamar al *join* de datos que actualizará todo. También, aproveché de agregar transiciones sobre la selección *exit*, para que su altura disminuya y luego se remueve la barra asociada.

Si probamos esto, la agregación de nuevos elementos sigue funcionando bien. Si intentamos eliminar una barra al final, esta se elimina correctamente. Pero, si intentamos eliminar una barra en otra posición, cosas raras pasan. Primero, parece eliminarse la última barra. Además, todas las barras siguientes a la cliqueada cambian. ¿Qué puede estar ocurriendo?

El problema, es la vinculación durante el *join* de datos. Si un dato intermedio es eliminado, los índices de los datos cambian en relación a como estaban antes posicionados. Entonces al hacer la vinculación, todos estos elementos se vinculan con un dato con el cual no se relacionaron antes. Y el último elemento no tiene con quien relacionarse.

Entonces, todos los elementos vinculados quedarán en *update*, y siempre el último elemento queda en *exit*. El último se remueve, y las otras barras se adaptan al nuevo dato que se le asocia ahora. ¡Esto explica lo que vemos en nuestro ejemplo!

Pero no hay que desesperar, ya que D3 provee una solución. Lo que necesitamos es una forma de especificar cómo se hará la vinculación, de forma que elementos y datos que una vez vincularon, también lo hagan en *join* de datos posteriores. Por esto, la función “data” tiene otro argumento que permite especificar cómo obtener la llave de vinculación.

Lo que hará “data” con eso, es evaluar la función sobre elementos y datos por separado, y luego los vinculará según las llaves que coincidan entre ambos grupos. Aquí, uso la función que extrae el atributo categoría del dato.

Con eso, en *joins* de datos posteriores al inicial, los elementos pueden acceder a su dato actualmente asociado, y extraer la categoría. Los datos por su cuenta harán lo mismo, y así aquellos rectángulos que estaban asociados a un dato, se vincularán con el mismo de nuevo.

Un aspecto importante a considerar con esta especificación de llave, es que su valor debe ser único para cada dato y elemento. Si no lo es, potencialmente habrán coincidencias inesperadas entre elementos y datos. En nuestro ejemplo, se cumple porque el atributo categoría es único entre los datos.

Si probamos este código con la función agregada, vemos que la eliminación de barras funciona como esperábamos. Los elementos y datos están bien vinculados al fin.

Para tus programas, siempre te recomiendo especificar una función de llave para *joins*. Así te ahorrarás dolores de cabeza con comportamientos extraños. De todas formas, no todos los programas lo necesitan necesariamente, así que no es obligación.

También, ten ojo sobre qué puedes usar como llave para estas situaciones, y si tus datos no cuentan con un atributo identificador, intenta generarlo.

Con eso termina el contenido de esta cápsula. Recuerda que si tienes preguntas, puedes dejarlas en los comentarios del video para responderlas en la sesión en vivo de esta temática. ¡Chao!