Cápsula 3: Jerarquía en D3.js

Hola, bienvenidxs a una cápsula del curso Visualización de Información. En esta hablaré sobre representación de jerarquías mediante D3.js.

Como en muchas de las situaciones revisadas, D3.js provee un paquete especializado para tratar con datos jerárquicos: "d3-hierarchy". Este es un subpaquete muy variado, ya que provee una interfaz para generar varios *idioms* de los que revisamos en las cápsulas recién vistas.

Lo primero que revisaremos es funciones de utilidad para representar en un programa datos jerárquicos. Como siempre, contamos con la pregunta y decisión de cómo cargar datos y cuánto de esto se realiza tras bambalinas en un proceso de pre-procesamiento. El paquete provee dos formas de cargar datos jerárquicos, comenzaremos por el más simple.

Una alternativa, es que los datos que carguemos ya vengan organizados como una jerarquía. Una opción fácil es mediante JSON, ya que este tiene una estructura fácilmente recursiva, perfectamente podemos generar un archivo que tenga una especie de jerarquía armada ya.

En pantalla vemos un ejemplo de un archivo JSON ya estructurado como jerarquía. El objeto que cubre todo tiene nombre "A", y todos los ítems están dentro de un campo de este objeto, que aquí es una lista llamada "hijos". De forma similar, cada objeto en esa lista tiene una estructura similar: los objetos "B" y "C" tienen hijos cada uno, y sus hijos también tienen esa estructura, así sucesivamente.

Para este caso, la estructura jerárquica viene explícita en los datos, y podríamos usarlo de esta forma. El problema es que sería necesario navegar en esta jerarquía para obtener información relevante: como acceso fácil a todos los nodos, o acceso a todos los enlaces entre nodos.

Ahí es donde entra "d3.hierarchy", función que recibe un objeto ya estructurado y retorna una representación de árbol con muchas funcionalidades internas de utilidad. Recibe un objeto que contenga la jerarquía completa y una función de acceso para identificar cómo acceder a los hijos de cada nodo.

En base a eso, cada objeto que encuentre lo interpretará como un nodo, y en base a esa función determinará los hijos del nodo, para luego repetir el proceso de forma recursiva. En nuestro caso, el atributo que contenía hijos se llamaba "hijos". Si probamos esto e imprimimos el resultado en consola, nos encontraremos con una estructura de árbol armada.

La función retorna un nodo específico: la raíz del árbol. Podemos ver en la consola que el objeto retornado tiene atributos ya calculados. Por un lado está "children", que es un arreglo

con los nodos hijos directos del nodo actual. También está “parent” que sería una referencia al objeto del nodo padre, que en este caso no existe al tratarse de la raíz.

Pero también hay atributos numéricos que describen al nodo en el árbol. Por un lado está “height”, qué determina la altura a la cual el nodo está dentro de la jerarquía, mientras más niveles hacia arriba, máyor altura. En nuestro caso, tiene una altura igual a 4, por lo que hay hasta cuatro niveles de profundidad. Por otro lado “depth” es lo contrario, cuán hondo está el nodo en la jerarquía. En el caso es la raíz, que al estar en el primer nivel de la jerarquía, tiene profundidad 0.

Finalmente está “data” que contiene el objeto originalmente cargado. Si revisamos los nodos dentro de “children”, veremos la misma estructura repitiendose. Ambos hijos de la raíz tienen profundidad 1, pero solo “C” tiene altura al menos 3, mientras que “B” tiene altura 2. Esto porque solo en el caso de “C” hay tres niveles hacia abajo de profundidad, mientras que en la ramificación de “B” hay dos niveles.

Entonces podemos apreciar que “d3.hierarchy” genera un montón de cálculos al generar la estructura jerárquica. Pero también provee comportamiento, podemos llamar a los métodos que nos permiten acceder a cosas interesantes de la jerarquía directamente. Por ejemplo, en pantalla se acceden en orden: a todos los descendientes de la jerarquía, todos los enlaces, y todas las hojas.

El primero entrega un arreglo con el nodo en cuestión, todos sus hijos, los hijos de sus hijos, y así sucesivamente. Al llamarse desde la raíz, eso entrega todos los nodos de la red.

Los enlaces por otro lado son objetos que contienen información que vincula dos objetos. Por ejemplo, el primer enlace listado tiene de fuente, o “source”, al nodo raíz “A” y de objetivo, o “target”, al nodo “B”. El segundo, por otro lado, va desde “A” también, a “C”. Así se listan todas las relaciones entre nodos.

Finalmente las hojas son aquellos nodos de la jerarquía que no tiene hijos, que están “al final” del árbol. Aquí vemos a “D”, “I”, “J”; que podemos ver en el archivo original que no tienen hijos.

“d3-hierarchy” también proveen un montón de otros métodos interesantes, que puedes revisar en la documentación.

Ahora, de forma completamente alternativa tal vez los datos que le proveemos a nuestro programa vienen en forma tabular. Por ejemplo, en pantalla vemos un archivo CSV equivalente a la jerarquía recién vista, donde mediante dos columnas se especifican los nodos de forma separada. Cada uno expresa el nombre del nodo, y el nombre de su padre.

Para este caso es necesario generar jerarquía mediante referencias de objetos que no vienen ya organizados. Para eso, está “d3.stratify” que retorna un generador de jerarquías a partir de un arreglo de objetos. El generador recibe funciones que le dicen al objeto como determinar las relaciones jerárquicas como referencias.

Primero “id” determina cómo identificar cada nodo en el arreglo, en nuestro caso mediante el nombre. Luego, mediante “parentId” se especifica como cada objeto referencia al identificador de su padre. En base a estos dos métodos, el generador sabrá cómo producir una jerarquía.

Si se aplica sobre el arreglo de nodos cargado, que en nuestro caso es el archivo CSV antes mostrado, produce una jerarquía y retorna la raíz de esta, con las mismas funcionalidades producidas por “d3.hierarchy”.

Podemos imprimir el resultado de los mismos métodos y obtenemos la misma información.

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.

En la siguiente revisaremos cómo usar la estructura que recién acabamos de revisar para generar *idioms* visuales.¡Chao!