Cápsula 1: Cargado de datos en D3

Hola, bienvenidxs a una cápsula del curso Visualización de Información.

En las cápsulas de este grupo construiremos sobre el ejemplo base de un programa en D3 que hemos usado hasta el momento, y revisaremos distintas funcionalidades que nos provee D3 para construir visualizaciones. En esta hablaré específicamente sobre cargado de datos en D3.

Hasta el momento, utilizamos un arreglo de números definido en el programa para crear barras. Este método no es suficiente para casos generales, ya que quisieramos utilizar datos reales. Ahora veremos funciones de D3 que permiten **cargar archivos de datos** directamente.

D3 provee varias funciones para cargar archivos, cada una por tipo o formato de archivo. Una de ellas es **"d3.csv"**. Esta recibe la ruta a un archivo, ya sea de un archivo local en tu computador, o incluso una dirección web. Aquí le entrego la ruta a un archivo "datos.csv", que está en el mismo directorio que mi programa.

"d3.csv" lo que hará es intentar leer el archivo en la ruta, y de encontrarlo, procesa su contenido de forma de producir un arreglo de objetos de JavaScript. En el caso de CSV, como los datos tienen forma de tabla con columnas, **la función hace el trabajo de procesar estos campos y retornar algo con sentido para JavaScript**.

"d3.csv" como función retorna un objeto de JavaScript conocido como promesa. Las promesas son un mecanismo moderno de JavaScript de procesar funciones asíncronas, es decir, que no necesariamente terminan de ejecutarse cuando se llama en el programa, porque pueden demorar mucho tiempo.

No iremos tanto en detalle sobre cómo funcionan las promesas, pero para tratar el resultado de una podemos encadenar las funciones "then" y "catch". "then" recibe una función que se ejecutará si hay éxito, en este caso si los datos se procesan sin problemas, y lo recibe como argumento en la función entregada.

Si hay un error en el proceso, esto se captura en la función entregada a "catch". En este caso, se va a imprimir en consola el resultado exitoso de cargado de datos o de error en cargado.

Si intentamos ejecutar esto abriendo directamente el archivo y abrimos la consola, ¡nos encontraremos con un error probablemente! Este es un error CORS, que es bastante recurrente en desarrollo web. Este tipo de error es una medida de seguridad que tienen los navegadores para que no se soliciten recursos locales de forma maliciosa. En general es bueno porque nos protege, pero en este caso nos complica el desarrollo de programas.

Hay varias formas de dar vuelta este error. Una opción es instalar una extensión en el navegador que levanta estos chequeos. Otra alternativa, es simular un servidor local para abrir los documentos que desarrollamos.

Hay varias formas de levantar servidores locales. Aquí lo haré ejecutando un método de Python en una terminal en el directorio de donde está el proyecto: “python3 -m http.server”. Mientras esto esté en ejecución, podemos acceder a un puerto específico, en este caso es el 8000 por defecto, y permite abrir un documento HTML en uso.

Si abrimos la consola, podemos encontrar el arreglo de datos procesado por “d3.csv”. Nos entrega un arreglo de objetos con propiedades que siguen el nombre de las columnas. **Averigua una forma de evitar el error CORS en tu caso, ya sea por extensión o levantando un servidor local usando Python u otra herramienta. Si tienes dudas, nos puedes avisar.**

Los datos se cargan, pero hay un asunto extraño. Por naturaleza, CSV guarda todo como texto, por lo que así se cargan los datos, se cargan como texto. En este caso, imaginamos el atributo “frecuencia” como numérico, entonces haría falta transformarlo a un objeto número.

Este tipo de procesamiento es común. Suele ocurrir en objetos de fecha temporal por ejemplo que no tienen una presentación temporal cuando se cargan y es necesario crear objetos de fecha manualmente. Aquí agrego una función que *parsea* un objeto, y para cada uno transforma el atributo frecuencia a un número utilizando “map”.

Es posible llamarlo inmediatamente al recibir datos para transformarlo, pero también las funciones de cargado como “d3.csv” admiten un segundo argumento de función de transformación que aplicará sobre cada objeto cargado. Así, nos retorna el objeto bien procesado.

Finalmente, ahora que si cargan los datos, podemos hacer uso de la función que genera las barras. Un detalle de esto es que la función esperaba un arreglo de números, no de objetos. ¡Hay que cambiar el acceso de atributos!

Como sigue siendo un arreglo, acceder mediante “length” al largo aún funciona. Pero, la sentencia que define el alto de las barras en base al valor del dato debe cambiar, porque ahora el dato es un objeto, no un número. **Ahora, el atributo que buscamos vincular al canal altura es “frecuencia”, por lo que accedemos a tal atributo.** Y si lo probamos, ¡funciona!

Ahora contamos con un programa que carga datos de un archivo y determina la altura de las barras con sus atributos. Aquí usamos “d3.csv” de ejemplo, pero D3 tiene métodos para varios otros formatos típicos. Por ejemplo, existe “d3.json”. Aquí cargo un archivo equivalente al anterior pero en formato JSON.

“d3.csv” y “d3.json” son funciones del módulo “d3-fetch” de D3. **D3 se divide en subpaquetes de funcionalidades**, y este es el que define funciones de cargado de datos. Puedes encontrar en la documentación oficial descripciones de otras funciones para otros formatos.

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!