Cápsula 2: Zoom sobre *idioms* I

Hola, bienvenidxs a una cápsula del curso Visualización de Información. En esta implementaremos zoom sobre un *idiom* específico.

Es difícil hacer una generalización de todas las posibles formas de aplicar zoom sobre *idioms* hechos en D3, así que nos concentraremos en un ejemplo, a modo de ver como aparecen distintas consideraciones a tener cuando trabajamos con zoom.

Comenzaremos con el código en pantalla, que genera un *dataset* de 1000 objetos aleatorios. Luego, a partir de esos datos, genera un gráfico de dispersión con márgenes, de forma muy similar a ejemplos que ya hemos revisado.

Si lo probamos, vemos el siguiente resultado. Como los datos son aleatorios, cambian en cada ejecución, pero las escalas y espacio se ajusta a eso. Al inspeccionar esto, mediante el navegador, podemos apreciar que el SVG se organiza en tres elementos “g” contenedores: uno para los puntos, y uno para cada uno de los ejes. Respectivamente, hay variables para cada contenedor en el código.

La idea entonces es poder realizar zoom y navegar dentro de este gráfico. Como hay 1000 puntos, es difícil identificar cada uno por separado. Para eso, podemos crear un comportamiento de zoom como lo hicimos en la cápsula pasada, y aplicarlo sobre el SVG actual.

Este solo permitirá zoom mediante acercamiento, sin traslación, y en la función a gatillar aplicaremos una transformación sobre la selección de puntos en pantalla, así se aplicará la transformación asociada. Si lo probamos, vemos que funciona, pero hay dos problemas. Primero, los puntos se pasan de su espacio y cruzan ejes e incluso la región de márgenes. Por otro lado, las escalas se mantienen igual a pesar de que se realice zoom.

Partiremos por el problema de los puntos. En la última cápsula esto no ocurrió porque usamos todo el SVG como espacio disponible, y si hay elementos fuera de su región visible, estos simplemente no se muestran. Ahora, nos encontramos con una situación especial, solo una región específica dentro del SVG se dedica como espacio de puntos, y el transformarlos con zoom permite que salgan de esa región, aún siendo visibles.

Una forma de solucionar esto es mediante CSS, existe una propiedad llamada “clip-path” que permite definir la región visible de un elemento, y oculta todo lo que esté fuera de ella. Hay varias formas de definir formas para usar como “clip-path”, y una es SVG.

Al agregar un elemento de tipo “clipPath”, se crea una definición de forma para usar en otro elemento. En el código crearemos uno, con “id” igual a “clip”, y dentro agregamos un rectángulo con las dimensiones de la región de los puntos. Nota que esto no agrega

visualmente un rectángulo, ya que estando dentro de “clipPath”, este solo queda como definición.

Ahora, eso no es suficiente, hace falta indicar qué elementos usarán ese rectángulo como límite visual. Una buena alternativa es el elemento “g” que contendrá a todos los puntos. Esta se ubica en la esquina que considera el margen, y todos sus hijos deberán respetar el *clip*. Si escribimos su atributo “clip-path” con una referencia al elemento de “id” “clip” que definimos, entonces se aplicará.

Si lo probamos, ¡vemos que funciona! Los puntos respetan el rectángulo definido. Intenta cambiando las dimensiones del rectángulo en “clipPath”, y ve que ocurre.

Ahora nos queda el asunto de los ejes. Que estos no cambien ante nuestro zoom tiene sentido con nuestro código, ya que en la función a gatillar en zoom no realizamos nada sobre los ejes, sólo los círculos. Es necesario alterarlos de alguna forma para que se acomoden a nuestras posiciones.

Si lo pensamos en términos de escalas primero, podemos notar que lo que hace falta cambiar es el dominio de la escala, el rango se mantiene idéntico. Específicamente, determinar a partir de la transformación qué intervalo de valores del dominio ahora serían visibles en el mismo rango.

El objeto de transformación que usamos para desplazar los puntos no solo tiene propiedades, también tiene métodos. Específicamente, tiene métodos “rescaleX” y “rescaleY” que reciben una escala, y a partir de la transformación generan una escala con dominio actualizado.

Con esto, es posible actualizar los objetos de eje correspondientes a esas escalas, y correspondientemente mostrarlos en pantalla. Si probamos esto, ¡vemos que ahora sí! Es posible navegar el gráfico de dispersión y actualizar los ejes de forma correspondiente.

Hay algunas restricciones sobre qué escalas pueden hacer esto. Deben ser escalas continuas, ya que esto permite calcular de forma inversa qué valores espaciales ahora caben en la región. Para otras opciones, habría que buscar alternativas de transformación o ingeniárselas mediante cómputo geométrico.

Y con esto 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!