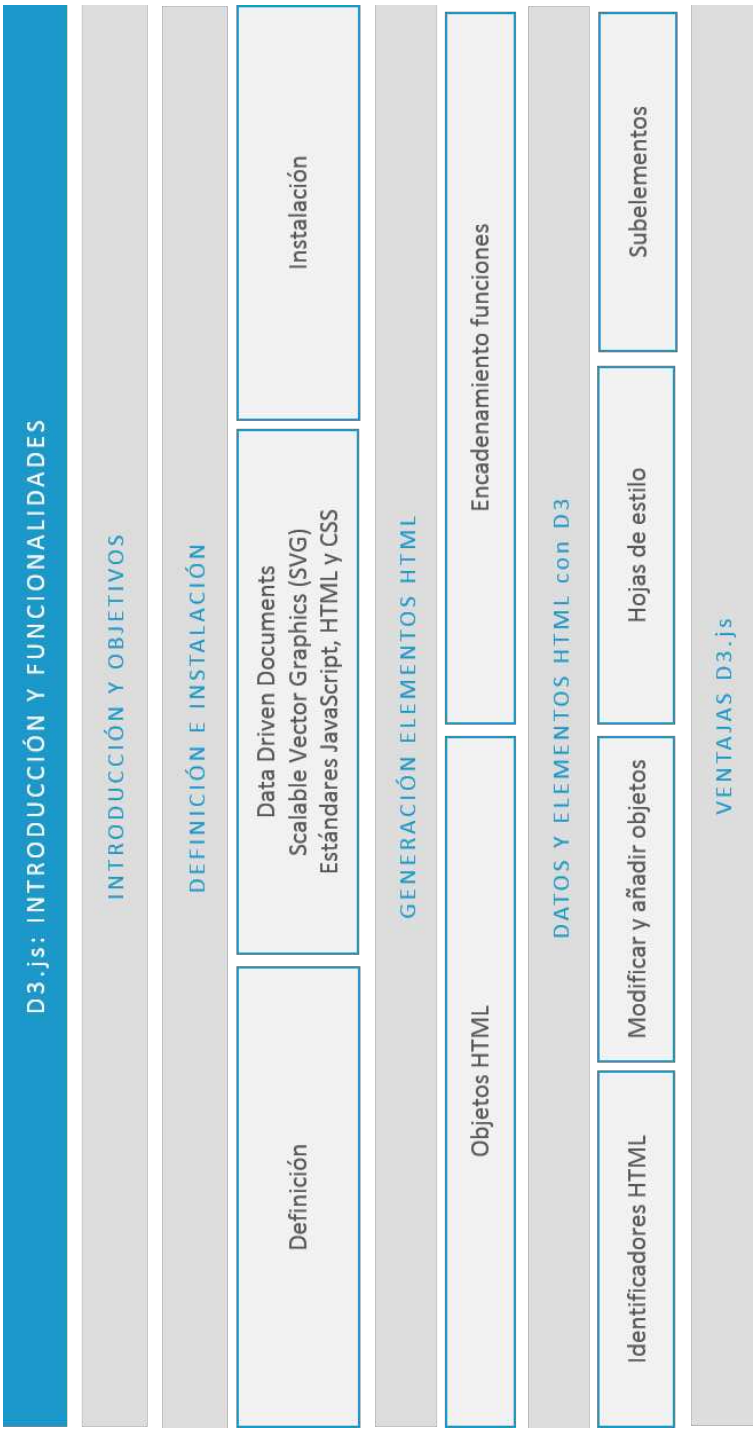
Herramientas de Visualización

Tema 3. D3.js. Introducción y funcionalidades

Índice

Esquema



3.1. Introducción y objetivos

En los próximos temas nos dedicaremos únicamente a la librería D3. ¿Por qué tantos temas a esta librería, qué tiene de especial? Vamos a conocer a fondo una librería enormemente flexible y que nos permitirá diseñar nuestras gráficas hasta el último detalle. Para ello nos basaremos en JavaScript como lenguaje de programación y abordaremos algunos conceptos avanzados que no suelen aparecer en la programación de páginas en HTML.

Figura 1. D3.js Data Driven Documents. Fuente: https://d3js.org/

Esta librería incluso nos permite crear las visualizaciones, aunque no nos consideremos buenos diseñadores. En GitHub podemos encontrar multitud de ejemplos en los que basarnos y que también podremos modificar.

Accede a la galería de GitHub a través del siguiente enlace:

https://github.com/d3/d3/wiki/Gallery

Lo importante es que acabemos este curso siendo capaces de diseñar e implementar nuestras propias visualizaciones, ya sea desde cero o modificando una existente. En concreto este tema introducirá los **conceptos básicos** de D3.js. para ir profundizando en temas posteriores.

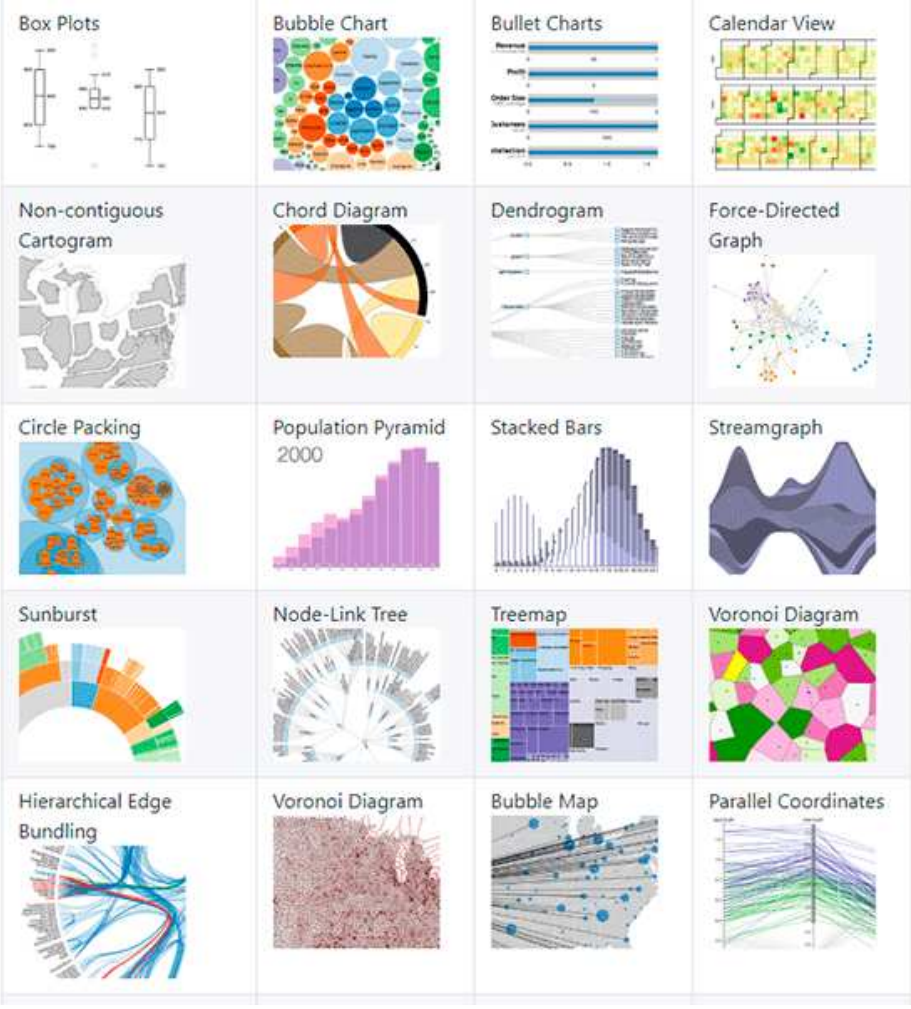


Figura 2. Ejemplos visualizaciones D3. Fuente: https://github.com/d3/d3/wiki/Gallery

Comenzamos con una frase de Barbara Sher:

«Puedes aprender cosas nuevas en cualquier momento de tu vida si estás dispuesto a

> ser un principiante. Si realmente aprendes a que te guste ser un principiante, el mundo entero se abre ante ti» (100 frases).

Te aconsejamos que en los primeros temas no intentes ir demasiado rápido. Lo importante es que los conceptos queden claros, pues serán muy útiles en visualizaciones más avanzadas y, si no se le presta la atención necesaria, más tarde puedes tener problemas.

Este tema es exclusivamente de introducción, no hay gran complicación tecnológica, pero es necesario para familiarizarnos con D3, por lo que sería interesante ver los recursos que recomendamos en este tema.

3.2. Definición e instalación

Antes de nada, empezaremos definiendo qué es D3.js:

> D3.js (*Data-Driven Documents*) es una librería JavaScript que utiliza datos digitales para impulsar la creación y control de gráficas dinámicas e interactivas en los navegadores.

Es una herramienta compatible con W3C, haciendo uso de los estándares en gráficos SVG (*Scalable Vector Graphics*), JavaScript, HTML5 y CSS3 (*Cascading Style Sheets*). A diferencia de otras librerías D3, permite un gran control sobre el resultado visual final.

De esta definición resaltaremos varios conceptos que son relevantes:

Data driven documents

Este concepto es bastante importante si lo que estamos pensando es en **reutilizar visualizaciones existentes**. Hasta ahora, debíamos adaptar nuestra estructura de datos a las visualizaciones, mientras que aquí hay algo más de libertad.

Podemos adaptar nuestros datos o bien podemos cambiar la forma en la que D3 está leyendo los datos y es lo que se denomina **documentos orientados a datos**. D3 se creó con la intención de tener control absoluto sobre la visualización.

Scalable Vector Graphics (SVG).

No importa dónde mostremos nuestra visualización: pantallas y formatos grandes como un monitor o soportes más pequeños como los de un móvil. **Nuestra visualización se puede adaptar** a diferentes resoluciones.

Compatible con estándares

Es compatible con estándares como JavaScript, HTML5 y *Cascading Style Sheets* (CSS3), por lo que funciona en diferentes exploradores y diferentes dispositivos.

Como complemento a la definición anterior, podríamos decir que, a alto nivel, D3 es una librería que se caracteriza por las siguientes **funcionalidades**:

- *Loading*: carga información en la memoria de tu explorador.
- *Binding*: añade información a elementos existentes en tu HTML, vinculando los datos y los objetos HTML.
- *Transforming*: es capaz de leer datos y dar una respuesta visual basada en los datos. Para ello puede transformar el código HTML.
- *Transitioning*: basándose en la entrada del usuario, puede reaccionar y dar una respuesta diferente.

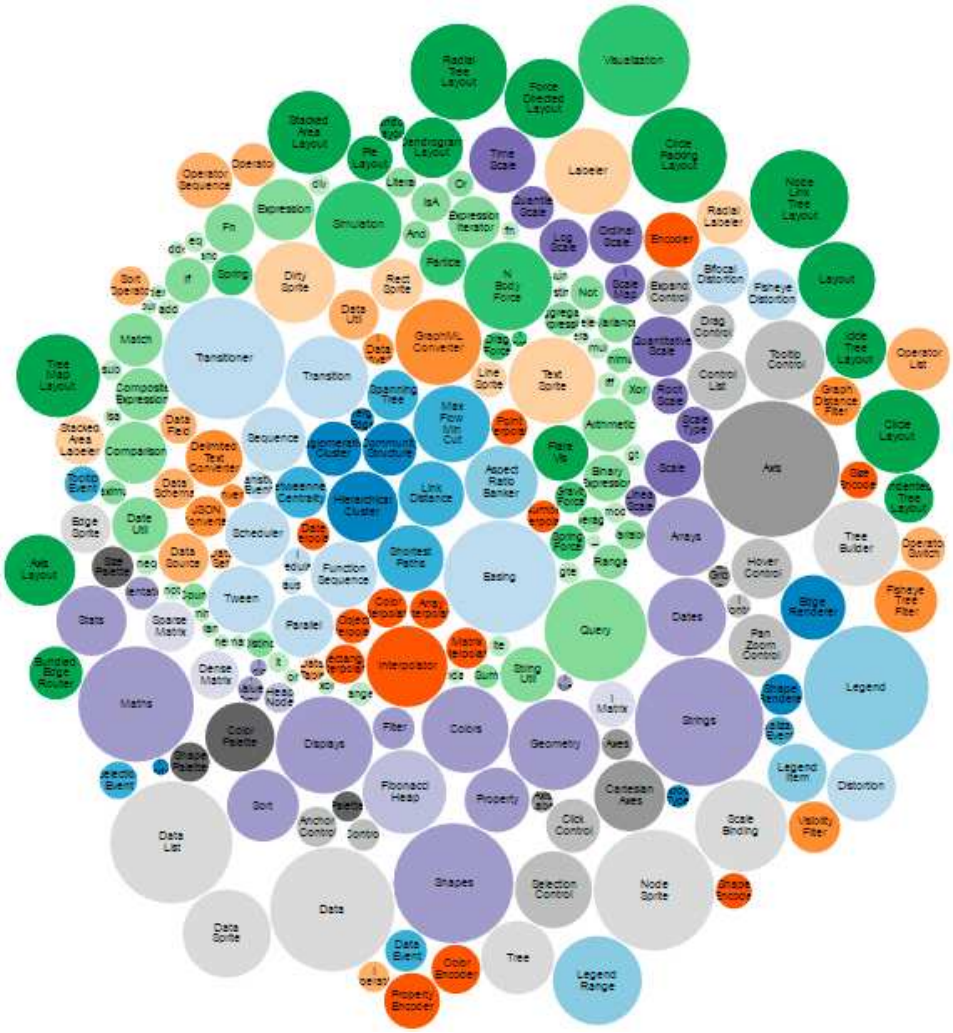


Figura 3. Ejemplo de visualización D3: Bubble Chart .Fuente:https://www.tutorialsteacher.com/d3js/what-is-d3js

Instalación

Para empezar a utilizar D3 no es necesario instalar ningún tipo de *software*. D3 es una librería *open source* por lo que está disponible de forma libre en su propia página web.

Accede a la página oficial de D3:

https://d3js.org/

Basta con llamar a la librería JavaScript de D3 desde nuestro propio código fuente. Se puede hacer de dos formas distintas:

- Incluir la librería D3 desde nuestro ordenador: descargamos la última versión disponible en d3js.org a través del fichero d3.zip. Lo descomprimimos y extraemos la librería, ubicándola en el directorio local que queramos para poder llamarla desde nuestro código en JavaScript.

- Incluir la librería D3 desde CDN (*Content Delivery Network*): enlazamos a la última versión de la librería con el siguiente código:

En los ejemplos utilizaremos la versión que se ofrece en la web oficial: http://d3js.org/d3.v3.min.js

Y este código será la base de nuestros ejercicios:

Código base para nuestros ejercicios

```
 <html>
<head>
<script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
</body>
</html>
```

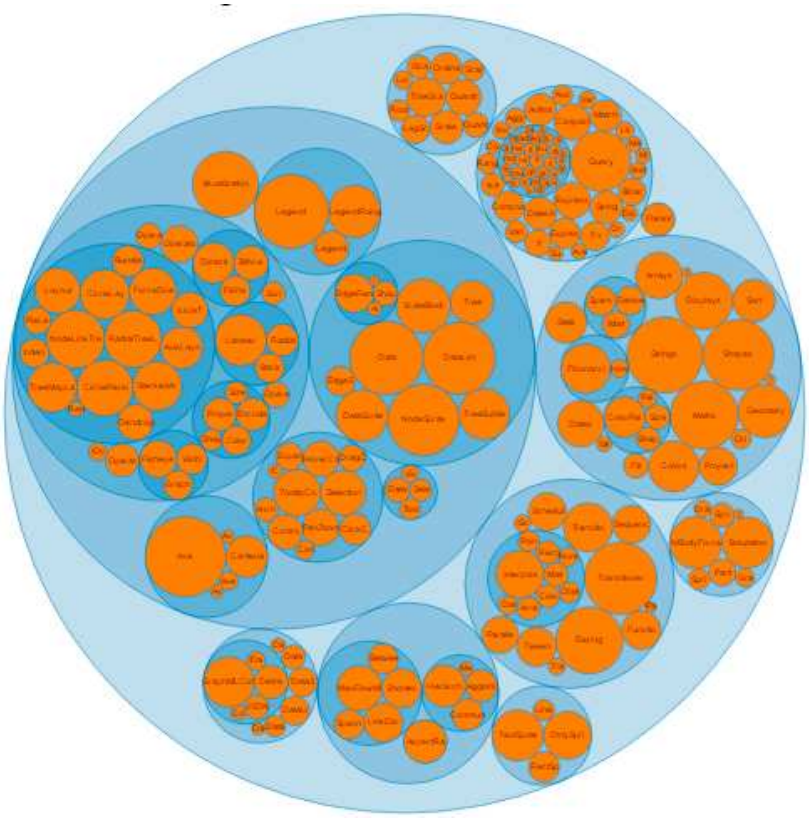


Figura 4. Ejemplo de visualización D3: *Circle Packing.*Fuente:https://www.tutorialsteacher.com/d3js/what-is-d3js

3.3. Elementos básicos de D3.js. Generando elementos HTML

Empezaremos con un ejemplo fácil. Hemos dicho que una de las características de D3 es que es capaz de transformar el código HTML y vincular datos con objetos HTML. ¿Cómo podemos crear un objeto HTML y añadir un texto con D3?

```
<html>
<head>
<script language="javascript" type="text/javascript"
src="http://d3js.org/d3.v3.min.js"></script>
<script type="text/javascript">
function draw(){
d3.select("body").append("p").text("Párrafo nuevo");
}
</script>
</head>
<body onload="draw()">
</body>
</html>
```

Si copiamos este texto en nuestro editor de código veremos que lo que hace es **generar un nuevo elemento** en nuestro HTML. Añade un párrafo nuevo cuando el HTML está cargado, , que llama a su vez a la función , que es la que realiza la acción en sí.

La **sintaxis de D3** sigue el concepto de encadenamiento o *chaining*. Se trata de concatenar varias funciones una seguida de otra, también es habitual utilizarla en Java, por ejemplo.

Pero ¿qué es lo que hace D3 con esta sentencia HTML?

- : selecciona el elemento body de nuestro HTML.
- : indica que añadirá algo a este elemento y define qué tipo de elemento HTML.

- : Define el texto que añadirá, es decir, el contenido texto.

De hecho, tendría el mismo efecto que utilizar lo siguiente:

```
var body = d3.select("body");
var p = body.append("p");
p.text("Párrafo nuevo");
```

3.4. Trabajando con datos reales y elementos en el HTML

Ahora veremos cómo trabajar con elementos dentro del HTML.

```
<html>
<head>
<script language="javascript" type="text/javascript"
src="http://d3js.org/d3.v3.min.js"></script>
<script type="text/javascript">
function draw(){
d3.select("p#target").text("Añadiendo contenido");
}
</script>
</head>
<body onload="draw()">
<p id="target"></p>
</body>
</html>
```

Esta estructura es muy parecida a la versión anterior, pero hemos utilizado las etiquetas o elementos habituales en HTML con las que hemos estado trabajado en temas anteriores.

Modificar elementos HTML y atributos

Pero D3 no solo nos permite añadir texto, sino también modificar los elementos HTML y sus atributos.

Añadir un atributo

Por ejemplo, queremos añadirle un atributo a nuestro elemento p llamado y asignarle el valor “ ”. Seguidamente lo imprimimos por pantalla.Si miramos nuestro contenido, el resultado final en el HTML es algo muy simple: nos muestra el texto “Contenido: primer_descriptor”. Y si miramos el código HTML en la consola de nuestro

explorador, podemos ver algo parecido a esto:

```
function draw(){
d3.select("p#target").text("Contenido: ");
// Establece atributo "descr" con el valor "primer descriptor" en
el elemento p
d3.select("p").attr("descr", "primer_descriptor");
// Obtiene el atributo "descr" en el elemento p
d3.select("p#target").text(d3.select("p").text()+d3.select("p").attr("descr"));

}
```

Si miramos nuestro contenido, el resultado final en el HTML es algo muy simple: nos muestra el texto . Y si miramos el código HTML en la consola de nuestro explorador, podemos ver algo parecido a esto:

```
<html>
<head>...</head>
<body onload="draw()">
<p id="target" descr="primer_descriptor">Contenido:
primer_descriptor</p>
</body>
</html>
```

Como puedes observar, hemos añadido un atributo a nuestro elemento en el propio código HTML.

Uso de estilos

Otro uso interesante de los elementos HTML a través de D3 es el uso de los estilos y lo podemos hacer a través de la función .

Por ejemplo, cambiarle el tamaño de la fuente:

```
d3.select("p#target").style("font-size", function(){return
"40px";});
```

Ahora mismo, dentro del código de la función solo devuelve 40px, pero podríamos

asignarle un valor dinámico dependiendo de alguna otra variable o del valor de un dato, lo que demuestra las enormes posibilidades que ofrece D3 al poder crear dinámicamente objetos HTML.

Modificar varios elementos HTML al mismo tiempo

Ahora aplicaremos estos conceptos en varios elementos HTML al mismo tiempo. Para ello, utilizaremos uno de tantos ejemplos que podemos encontrar en el libro *Data Visualization with D3.js Cookbook* (Zhu, 2013) en el que los ejemplos son de libre acceso y se pueden descargar. El ejemplo es el siguiente:

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="styles.css"/>
<script language="javascript" type="text/javascript"
src="http://d3js.org/d3.v3.min.js"></script>
<script type="text/javascript">
function draw(){
d3.selectAll("div")
.attr("class", "red box");
}
</script>
</head>
<body onload="draw()">
<div></div>
<div></div>
<div></div>
</body>
</html>
```

La hoja de estilo a utilizar está definida en el archivo styles.css. En él, la clase y la clase están definidas de la siguiente forma:

```
.box {
width: 200px;
height: 200px;
margin: 40px;
```

```
float: left;
text-align: center;
border: #969696 solid thin;
padding: 5px;
}
.red {
background-color: #e9967a;
color: #f0f8ff;
}
.blue {
background-color: #add8e6;
color: #f0f8ff;
}
```

Aplicando las clases y a los diferentes elementos se obtiene el siguiente resultado:

Figura 5. Resultado de dibujar 3 cajas.

Si queremos **iterar** sobre cada elemento para hacer algo diferente, lo podemos hacer de la siguiente manera:

```
d3.selectAll("div")
.attr("class", "red box")
.each(function (d, i) {
d3.select(this).append("h1").text(i);
});
```

Aquí resaltaremos lo que se diferencia de lo anterior:

```
each(function (d, i) {
```

```
d3.select(this).append("h1").text(i);
```

La función itera sobre los diferentes elementos, siendo d el elemento en sí, es decir, el elemento div referenciado dentro de la función como . Y lo que hacemos es añadir un elemento con el número de la iteración .

Figura 6. Resultado de dibujar 3 cajas con texto.

Subelementos

Podemos trabajar también con **subelementos**. En el siguiente ejemplo, veremos cómo seleccionar un subelemento dentro de otro elemento. Cambiemos nuestro con lo siguiente:

```
<section id="section1">
<div>
<p>blue box</p>
</div>
</section>
<section id="section2">
<div>
<p>red box</p>
</div>
</section>
```

Y el código de la función por este:

```
d3.select("#section1 > div")
.attr("class", "caja azul");
```

```
d3.select("#section2")
.select("div")
.attr("class", "caja roja");
```

En el código anterior podemos observar dos maneras diferentes de seleccionar un subelemento:

- identificador > elemento html:

d3.select("#section1 > div")

- doble select:

d3.select("#section2")

.select("div")

El código completo queda de la siguiente forma:

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="styles.css"/>
<script language="javascript" type="text/javascript"
src="http://d3js.org/d3.v3.min.js"></script>
<script type="text/javascript">
function draw(){
d3.select("#section1 > div")
.attr("class", "blue box");
d3.select("#section2")
.select("div")
.attr("class", "red box");
}
</script>
</head>
<body onload="draw()">
<section id="section1">
<div>
<p>caja azul</p>
</div>
</section>
```

```
<section id="section2">
<div>
<p>caja roja</p>
</div>
</section>
</body>
</html>
```

Y el resultado final es el siguiente:

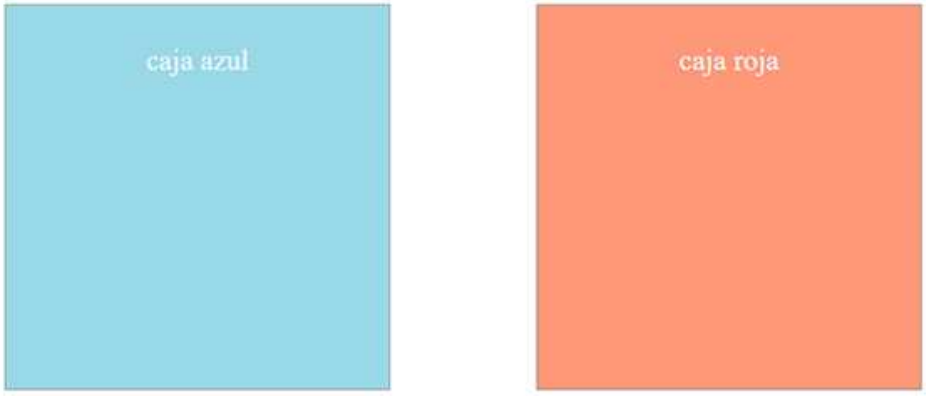


Figura 7. Resultado dibujar 2 cajas de diferente color y con texto.

Dando un paso más, también podríamos reemplazar toda la sección y dejarla completamente limpia . Para ello incluimos el código D3 siguiente dentro de la función :

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="styles.css"/>
<script language="javascript" type="text/javascript"
src="http://d3js.org/d3.v3.min.js"></script>
<script type="text/javascript">
function draw(){
var body = d3.select("body");
body.append("section")
.attr("id", "section1")
.append("div")
.attr("class", "blue box")
```

```
.append("p")
.text("caja azul dinámica");
body.append("section")
.attr("id", "section2")
.append("div")
.attr("class", "red box")
.append("p")
.text("caja roja dinámica");
d3.select("#section1 > div")
.attr("class", "blue box");
d3.select("#section2")
.select("div")
.attr("class", "red box");
}
</script>
</head>
<body onload="draw()">
</body>
</html>
```

¿Cuáles son las principales diferencias?

Primero creamos una variable y le asignamos el resultado de la función . Esto es lo mismo que hemos hecho al principio del tema.

Para crear la primera caja añadimos un elemento . Le añadimos un atributo identificador llamado . A este elemento , le añadimos un elemento y le asignamos la clase . A este elemento le añadimos un elemento párrafo o y le asignamos el texto caja azul dinámica. La caja roja sigue el mismo patrón de creación.

3.5. Ventajas de D3.js

Para terminar este tema, enunciaremos algunas de las ventajas del uso de D3 como herramienta de visualización.

Hemos visto a través de los ejemplos que es necesario conocimientos de JavaScript, aunque el esfuerzo necesario tiene su recompensa:

- D3.js es una biblioteca de JavaScript. Por lo tanto, puede ser usada con cualquier marco JS de su elección como Angular.js, React.js, etc…
- D3 se centra en los datos, por lo que es la herramienta más apropiada y especializada para la visualización de datos.
- D3 es de código abierto, te permite trabajar con el código fuente y añadir tus propias características.
- Funciona con estándares web, por lo que no se necesita ninguna otra tecnología o *plugin* que no sea un navegador para hacer uso de D3.
- D3 trabaja con estándares web como HTML, CSS y SVG, no se requiere ninguna nueva herramienta de aprendizaje o depuración para trabajar en D3.
- D3 proporciona un control completo sobre su visualización para personalizarla de la manera que se desee. Esto le da una ventaja sobre otras herramientas populares como Tableau o QlikView para usos específicos.
- Dado que D3 es ligero y funciona directamente con los estándares web, es extremadamente rápido y funciona bien con grandes conjuntos de datos.

En definitiva, una potente librería que, aunque requiere una curva de aprendizaje en términos de programación general, a cambio aporta una gran flexibilidad y posibilidades casi infinitas.

3.6. Referencias bibliográficas

100 frases. *Frase de Barbara Sher*. https://100frases.com/autor/barbara-sher/puedes-aprender-cosas-nuevas-en-cualquier-momento--5b6e09f373e77aa867125151

TutorialsTeacher. *D3.js Tutorial*. https://www.tutorialsteacher.com/d3js

Zhu, N. Q. (2013). *Data Visualization with D3.js Cookbook*. [S. l.]: Packt Publishing. https://github.com/NickQiZhu/d3-cookbook

D3 for the Impatient: Interactive Graphics for Programmers and Scientists

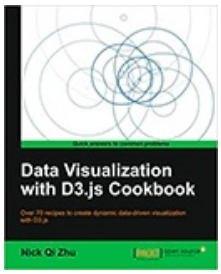
Janert, P. K. (2013). *D3 for the Impatient: Interactive Graphics for Programmers and Scientists.* [S. l.]: O'Reilly.

Este libro está recomendado para quienes entienden lo básico de HTML, CSS y JavaScript y quieran explorar rápidamente la extensa, pero a menudo abrumadora, documentación de referencia sobre D3.js.

Philipp K. Janert, autor de *Data Analysis with Open Source Tools,* proporciona una concisa hoja de ruta para esta librería, incluyendo sus convenciones y conceptos fundamentales. *D3.js para los impacientes* es conciso y completo.

Data Visualization with D3.js Cookbook

Zhu, N. Q. (2013). *Data Visualization with D3.js Cookbook.* [S. l.]: Packt Publishing. https://github.com/NickQiZhu/d3-cookbook



Convierte tus datos digitales en gráficos dinámicos con este libro repleto de guías y ejemplos útiles sobre la biblioteca de JavaScript D3, que te ayudarán a resolver problemas reales de visualización.

Data Driven Documents D3.js: Tips and Tricks

Maclean, M. (S. f.). *Data Driven Documents D3.js: Tips and Tricks* (v.3.x) [S. l.]: Leanpub. https://leanpub.com/D3-Tips-and-Tricks/read#leanpub-auto-what-do-you-need-toget-started

Este libro, escrito por Malcolm Maclean y de libre acceso, ofrece consejos y trucos para utilizar d3.js. Es un libro que te ayudará a empezar, pero también incluye algunas tareas avanzadas.

Creando un mapa sencillo con D3

Morales, A. (15 de marzo de 2016). Creando un mapa sencillo con D3. *Mapping GIS*. https://mappinggis.com/2015/03/creando-un-mapa-sencillo-con-d3/

Este artículo explica cómo crear un mapa sencillo en D3 que muestre los países del mundo.

Learn D3.js in 5 minutes

Nehal, S. (6 de abril de 2018). Learn D3.js in 5 minutes. *Free Code Camp*. https://www.freecodecamp.org/news/learn-d3-js-in-5-minutes-c5ec29fb0725/

Introducción a la creación de representaciones visuales con tus datos.

Test

1. ¿Qué significa el concepto *chaining* en D3?

   A. Las funciones se pueden concatenar una detrás de otra separándolas con un punto.

   B. Las funciones dependen unas de otras, y solo puedes llamarlas de una manera seguida.

   C. Las funciones permiten crear elementos HTML.

   D. Ninguna de las anteriores.

2. Si ejecutamos la sentencia :

   A. Seleccionamos el elemento p con identificador target en el HTML y añadimos el texto indicado entre comillas.

   B. Seleccionamos el elemento p con identificador target en el HTML y reemplazamos el texto existente con el indicado entre comillas.

   C. Seleccionamos el elemento target con identificador p.

   D. Ninguna de las anteriores.

3. Si ejecutamos la sentencia

   A. Seleccionamos el elemento `del HTML, añadimos un elemento p y le asignamos el texto entre comillas.`

   B. Seleccionamos el elemento `del HTML, con identificador y le asignamos el texto entre comillas.`

   C. No tiene una sintaxis correcta y no funciona.

   D. Ninguna de las anteriores.

4. Con D3, ¿podemos modificar las clases CSS de los elementos HTML?

A. Sí.

B. No, porque figuran un fichero independiente.

C. No, porque D3 solo me permite trabajar con los datos.

D. No, porque solo puedo modificar el código HTML

5. Con D3, ¿podemos modificar el estilo de un elemento HTML como, por ejemplo, el tamaño de la fuente, con una sentencia tipo ?

A. Sí.

B. No, porque las características de los objetos residen en las hojas de estilo.

C. No, porque no existe una forma de modificar solo un identificador de estilo.

D. No, porque los elementos de las hojas de estilo no son accesibles por D3.

6. ¿Qué hace la siguiente sentencia: ?

A. Devuelve todos los elementos div del HTML con el atributo clase que contenga .

B. Asigna a todos los elementos div del HTML la clase .

C. Asigna a un elemento div del HTML la clase .

D. Ninguna de las anteriores.

7. D3 permite trabajar tanto con los datos a visualizar como con elementos HTML:

A. No, solo con datos.

B. No, solo con elementos HTML.

C. No, porque los elementos u objetos HTML deben codificarse antes de su visualización por el navegador.

D. Sí. Es una de sus principales ventajas como herramienta de visualización.

8. ¿Qué hace la siguiente sentencia: *?*

A. Itera por todos los elementos y les añade un elemento con el texto , donde es el contador de la iteración.

B. Itera por todos los elementos y les añade un elemento con el texto , donde es el texto que el elemento HTML contiene.

C. Itera por todos los elementos y les añade un elemento con el texto , donde es una constante fija previamente definida.

D. Ninguna de las anteriores.

9. Identifica si obtenemos el mismo resultado con la sentencia que con el siguiente código:

```
var body = d3.select("body");
var p = body.append("p");
p.text("Párrafo nuevo");
```

A. Sí.

B. No, porque la primera sentencia no es correcta.

C. No, porque aunque la primera sentencia es correcta, el código siguiente no lo es.

D. No, porque tanto la línea de código individual como el grupo de código son incorrectos.

10. ¿Qué hace la sentencia d ?

A. Le asigna vacío al elemento con identificador .

B. Devuelve el contenido del elemento con identificador .

C. Devuelve el contenido del elemento , siendo este de formato texto.

D. Le asigna un texto al elemento .