D3

Originalartikel

Backup

<html> <h2>Intro</h2><hr/><p>

```
      <a href="http://mbostock.github.com/d3/">Data-Driven Documents</a>, or
D3 for short, is a new visualization library to build visualizations in SVG.
But in my opinion, it's also the best javascript multipurpose SVG library
when it comes to animation, interaction and above all for binding data to
graphics. The <a
href="https://groups.google.com/forum/#!forum/d3-js">community</a> is very
responsive, <a href="https://github.com/mbostock/d3">source code</a> is very
clean and the <a href="http://mbostock.github.com/d3/api/">API</a> is well
written. There is not a lot of <a
href="https://github.com/mbostock/d3/wiki">tutorials</a> to learn D3 yet.
But they are very good introductions and you can find plenty of <a
href="https://github.com/mbostock/d3/tree/master/examples">examples</a>
packaged with the source code or hidden in forums. But if you know SVG, you
already have a big head start. My goal was to write a simple introductory
tutorial covering the main aspects of D3, with very trivial examples but
with full length snippets for you to be up and running right now. So open
your favorite editor, a decent browser and be ready to copy-paste.
      </p><!-- ------------------------------------------------------------
---------------------------------- -->
    <h2>Simple example</h2><hr/><p>
      Let's start with a simple D3 example. It's an html file with a simple
<a href="http://www.w3schools.com/html5/tag_doctype.asp">html5 doctype</a>.
The D3 script will be loaded directly from the <a
href="https://github.com/mbostock/d3">D3 repository</a>, but you could as
well <a href="https://github.com/mbostock/d3/zipball/master">download a
copy</a> and load it locally. I've placed a <code>&lt;div&gt;</code> tag to
be used as a container for our visualization. The D3 script lay down just
before the closing <code>&lt;/body&gt;</code> tag to be sure the page is
fully loaded and ready to work with, and according to <a
href="http://developer.yahoo.com/performance/rules.html#js_bottom">best
practices</a>. This example shows the use of D3 for <a
href="http://www.w3schools.com/htmldom/default.asp">DOM</a> traversal, for
adding <a href="http://www.w3schools.com/svg/svg_intro.asp">SVG</a> and HTML
elements, for adding styles and attributes and for mouse <a
href="http://www.w3schools.com/jsref/dom_obj_event.asp">events</a> binding.
Just copy-paste this example in a file named "index.html" opened in your
favorite editor and in a recent browser. You can then replace the javascript
code by any snippets found in this tutorial.
      </p>
      <p>D3 simple example:</p>
      <pre>&lt;!DOCTYPE html&gt;
```

<html> <head>

```
  &lt;script type="text/javascript"
src="http://mbostock.github.com/d3/d3.js"&gt;&lt;/script&gt;
```

</head> <body>

```
  &lt;div id="viz"&gt;&lt;/div&gt;
  &lt;script type="text/javascript"&gt;
  var sampleSVG = d3.select("#viz")
      .append("svg")
      .attr("width", 100)
      .attr("height", 100);
  sampleSVG.append("circle")
      .style("stroke", "gray")
      .style("fill", "white")
      .attr("r", 40)
      .attr("cx", 50)
      .attr("cy", 50)
      .on("mouseover", function(){d3.select(this).style("fill",
"aliceblue");})
      .on("mouseout", function(){d3.select(this).style("fill", "white");});
  &lt;/script&gt;
```

</body> </html></pre>

```
      <p>SVG circle with hovering:</p>
      <p>
      This is the SVG generated by the code from the above example:
      </p>
      <pre>&lt;svg width="100" height="100"&gt;
```

<circle style=„stroke: #808080; fill: #F0F8FF; “ r=„40“ cx=„50“ cy=„50“></circle>
</svg></pre>

```
      <p>
      Why not just write this SVG fragment directly? There are several
reasons to use D3 to generate SVG and HTML markup:
      </p><ul><li>Add, remove and modify multiple elements from an existing
DOM</li>
      <li>Dynamically add attributes and styles according to a function</li>
      <li>Animate attributes and styles</li>
      <li>Bind data to automatically add elements when needed</li>
      <li>Benefits from a lot of helper functions</li>
      </ul><!-- ------------------------------------------------------------
---------------------------------- --><h2>Select, append, bind
event</h2><hr/><p>
      D3, like <a href="http://jquery.com/">JQuery</a> and other javascript
frameworks, simplify the selection of an element in the DOM but also in the
SVG DOM. The static method <code>d3.select()</code> can take any <a
```

```
href="http://www.w3schools.com/cssref/css_selectors.asp">CSS selector</a> as
an argument. The <code>d3.append()</code> method is similar, but is used to
add a new element as a child of the selected element. To append an SVG
element, you can prefix it by the SVG <a
href="http://www.w3schools.com/xml/xml_namespaces.asp">namespace</a> like
this: <code>d3.append("circle")</code>.
      </p>
      <p>
      D3 uses a <a
href="http://en.wikipedia.org/wiki/Declarative_programming">declarative</a>
style of programming. You declare what you want to select, then your
modifications on the DOM, CSS styles, HTML and SVG attributes, texts,
animations, and events. D3 will loop through your selection set and apply
the modifications for you, hiding just enough of the DOM manipulation
ugliness to leave you with the fun part and with full control on the
process.
      </p>
      <p>
      Binding events is just as simple. Select a DOM element and use the
<code>.on()</code> method with the event you want as a string for the first
attribute and a callback function for the second attribute, like in the
example: <code>.on("mouseover", function(){d3.select(this).style("fill",
"white");})</code>. You will often see this type of anonymous function as an
argument in D3.
      </p>
```

<!– ———————————————————————————— ->

```
    <h2>Animating</h2><hr/><p>
      All attributes and styles can be animated using
<code>.transition()</code>, <code>.delay()</code> and
<code>.duration()</code>. D3 will recognize the type of the value and will
interpolate numbers as well as colors, matrices, paths and percentages.
      </p>
      <p>Simple animation:</p>
      <pre>var sampleSVG = d3.select("#viz")
  .append("svg")
  .attr("width", 100)
  .attr("height", 100);
```

sampleSVG.append(„circle“)

```
  .style("stroke", "gray")
  .style("fill", "white")
  .attr("r", 40)
  .attr("cx", 50)
  .attr("cy", 50)
.transition()
  .delay(100)
  .duration(1000)
  .attr("r", 10)
```

```
  .attr("cx", 30)
  .style("fill", "black");</pre>

<!– ———————————————————————————————— ->

    <h2>Animation chaining</h2><hr/><p>
      One way to chain animations is by using <code>.delay()</code> to start
one animation after the other. This example shows one way of using this
technique on a mouse press. The <code>mousedown</code> event pass the
current context, in this case the element where the event occurred, as an
argument for the callback function that can use it under the name
<code>this</code>. Notice how the duration of the first transition is
inherited in the second. The main idea here is to delay the second
transition for the duration of the first one so it is only triggered after.
      </p>
      <p>Animation chaining trigged by the mouse:</p>
      <pre>var sampleSVG = d3.select("#viz")
  .append("svg")
  .attr("width", 100)
  .attr("height", 100);
```

sampleSVG.append(„circle“)

```
  .style("stroke", "gray")
  .style("fill", "white")
  .attr("r", 40)
  .attr("cx", 50)
  .attr("cy", 50)
  .on("mouseover", function(){d3.select(this).style("fill", "aliceblue");})
  .on("mouseout", function(){d3.select(this).style("fill", "white");})
  .on("mousedown", animate);
```

function animate() {

```
  d3.select(this).transition()
      .duration(1000)
      .attr("r", 10)
    .transition()
      .delay(1000)
      .attr("r", 40);
```

};</pre>

```
      <p>
      A better way of chaining animation is to listen to the
<code>end</code> event with the <code>.each();</code> method to detect the
end of each animation. See <a href="http://bl.ocks.org/1125997">this
example</a> for a way to pass arguments to the function and to call it
recursively.
      </p>
```

```
      <div class="legend" readability="6">Animation chaining using the
<code>end</code> event:</div>
      <pre>var sampleSVG = d3.select("#viz")
  .append("svg")
  .attr("width", 100)
  .attr("height", 100);
```

sampleSVG.append(„circle“)

```
  .style("stroke", "gray")
  .style("fill", "white")
  .attr("r", 40)
  .attr("cx", 50)
  .attr("cy", 50)
  .on("mouseover", function(){d3.select(this).style("fill", "aliceblue");})
  .on("mouseout", function(){d3.select(this).style("fill", "white");})
  .on("mousedown", animateFirstStep);
```

function animateFirstStep(){

```
  d3.select(this)
    .transition()
      .delay(0)
      .duration(1000)
      .attr("r", 10)
      .each("end", animateSecondStep);
```

}; function animateSecondStep(){

```
  d3.select(this)
    .transition()
      .duration(1000)
      .attr("r", 40);
```

};</pre>

Animation chaining trigged by a mouse press:

```
    <h2>Binding data</h2><hr/><p>
      D3 means Data-Driven Document because of the way it can bind data to
graphics. Here is a simple example adding as much elements as there is
members in the dataset. Once again, anonymous functions are used to
dynamically calculate values for styles and attributes. Those functions
provide two arguments: the data bound to the element and the index of the
element in the selection set. Follow <a
href="http://mbostock.github.com/d3/tutorial/circle.html">this tutorial</a>
directly from the master to learn how to add, update and remove elements on
the fly.
      </p>
      <p>Adding element from data:</p>
      <pre>var dataset = [],
```

```
  i = 0;
```

for(i=0; i</pre>

```
      <p>One SVG circle generated for each data member:</p>
```

<!– ———————————————————————————— –>

```
    <h2>Binding two-dimensional data</h2><hr/><p>
      To bind data made from an array of arrays, often called two-
dimensional array, we must use a little trick. We will build a simple html
table. First we add a <code>&lt;table&gt;</code> tag, then
<code>&lt;tr&gt;</code> tags to add rows. For this, we bind the dataset to
an empty <code>&lt;tr&gt;</code> selection and we then append as many
<code>&lt;tr&gt;</code> tags as there is members in the first dimension of
the dataset by chaining the two methods <code>.enter().append()</code>. But
when it's time to bind data to the <code>&lt;td&gt;</code> tags to add
columns, you have to bind the sub-array with this trick:
<code>.data(function(d){return d;})</code>.
      </p>
      <p>Table generated by data binding:</p>
      <pre>var dataset = [],
```

tmpDataset = [], i, j; for (i = 0; i </pre>

```
      <p>HTML table generated from a 2D array:</p>
      <h2>Loading data from a file</h2><hr/><p>
      For the last examples, we generated arbitrary data. But D3 can also
help you load a data file, for example a <a
href="http://en.wikipedia.org/wiki/Comma-separated_values">Comma Separated
Value</a> (CSV) file using <a
href="http://en.wikipedia.org/wiki/Ajax_(programming)">Ajax</a>. This
example will work as is on a web server, <a
href="http://www.uniformserver.com/">local</a> or remote. But you could also
bypass the <a
href="http://en.wikipedia.org/wiki/Same_origin_policy">security
restrictions</a> of your browser that prevent you from loading data from a
local file, for example using the command switch <code>--allow-file-access-
from-files</code> in Google Chrome. If you understand what this is, you
probably also know the risks. If not, use a web server. Anyway to load a CSV
file with D3, you just need to use the <code>d3.csv()</code> method. But
here, I will use an equivalent method, <code>d3.text()</code>, to load it as
plain text and use a d3 helper to parse the CSV.
      </p>
      <p>Table from external CSV file:</p>
      <pre>d3.text("auto_mpg_tmp.csv", function(datasetText) {
```

var parsedCSV = d3.csv.parseRows(datasetText); var sampleHTML = d3.select(„#viz“)

```
  .append("table")
```

```
  .style("border-collapse", "collapse")
  .style("border", "2px black solid")
  .selectAll("tr")
  .data(parsedCSV)
  .enter().append("tr")
  .selectAll("td")
  .data(function(d){return d;})
  .enter().append("td")
  .style("border", "1px black solid")
  .style("padding", "5px")
  .on("mouseover", function(){d3.select(this).style("background-color",
"aliceblue")})
  .on("mouseout", function(){d3.select(this).style("background-color",
"white")})
  .text(function(d){return d;})
  .style("font-size", "12px");
```

});</pre>

```
      <p>Content of the external CSV file:</p>
      <pre>car
name,miles/gallon,cylinders,displacement,horsepower,weight,acceleration,mode
l year,origin
```

„chevrolet chevelle malibu“,18,8,307,130,3504,12,70,1 „buick skylark 320“,15,8,350,165,3693,11.5,70,1 „plymouth satellite“,18,8,318,150,3436,11,70,1</pre>

```
      <p>Table generated by the data from a CSV file:</p>
```

<!– ———————————————————————————————— <h2>Loading data from another website</h2><hr />

```
      <p>Here is a little more advanced example querying real-time Google
stock quotes data from YAHOO finance. There are some challenges here
unrelated to D3. The first is <a
href="http://en.wikipedia.org/wiki/Cross-site_scripting">Cross-domain
scripting</a>. One way to get data from another website is to pass the
request to a proxy on the same server as your script. Here is the PHP proxy
code. </p>
      <div class="legend">PHP Cross-Site Scripting code for YAHOO
Finance:</div>
      <pre>&lt;?php
```

$ticker = $_GET["url"]; $url = 'http://chartapi.finance.yahoo.com/instrument/1.0/'.$ticker.'/chartdata;type=quote;range=1d/csv/'; $headers = $_GET['headers']; $mimeType =$_GET['mimeType']; $session = curl_init($url); curl_setopt($session, CURLOPT_HEADER, ($headers == "true") ? true : false); curl_setopt($session, CURLOPT_FOLLOWLOCATION, true); curl_setopt($session, CURLOPT_RETURNTRANSFER, true); $response = curl_exec($session); if ($mimeType != "")header("Content-Type: ".$mimeType); echo $response;

curl_close($session); ?></pre>

```
      <p>But the D3 part should be more obvious. A timer request data every
10 seconds through the PHP proxy. YAHOO reply with CSV data that we
transform to our needs. We can see here some D3 helper functions like
<code>d3.min()</code>, <code>d3.max()</code> and
<code>d3.scale.linear()</code> to transform our data to graphical
coordinates. The line is a <a
href="http://www.w3schools.com/svg/svg_path.asp">SVG path</a> that can be
part of a complete line chart. You can see the Yahoo finance Flash version
<a href="http://finance.yahoo.com/q?s=goog&ql=1">here</a>.</p>
      <div class="legend">Line chart component from real-time stock quotes
data:</div>
      <pre>var w = 400,
  h = 300;
```

var vizSVG = d3.select(„#viz“)

```
      .append("svg")
      .attr("width", w)
      .attr("height", h)
      .append("path");
```

getData(); setInterval(function() {

```
  getData();
```

}, 10000); function getData(){

```
  d3.text("http://christopheviau.com/d3_tutorial/proxy.php?url=GOOG",
function(csv) {
      var csvArray = d3.csv.parseRows(csv).slice(15);
      var data = [];
      var date = new Date();
      for(var i=0; i&lt;csvArray.length; i++){
          date.setTime(csvArray[i][0]*1000);
          data.push({
              "date":  date.toGMTString(),
              "quote": parseFloat(csvArray[i][1])
          });
      }
      buildChart(data);
  });
```

} function buildChart(data){

```
  var len = data.length;
  var dataMin = d3.min(data, function(d){return d.quote;});
  var dataMax = d3.max(data, function(d){return d.quote;});
  var x = d3.scale.linear()
```

```
      .domain([0, len])
      .range([0, w]);
  var y = d3.scale.linear()
      .domain([dataMax, dataMin])
      .range([0, h]);
  var svgPath = "M";
  for (var i = 0; i < len; i++) {
      svgPath += x(i)+" "+y(data[i].quote);
      if(i!=data.length-1) svgPath += "L";
  }
  d3.select("#viz path")
      .attr("d", svgPath)
      .attr("stroke", "black")
      .attr("fill", "none");
```

}</pre>

Google stock quotes data updating every 10 secondes:

```
      <div id="viz7"></div>
```

–> That's about it. I hope this tutorial was useful for you. If you want to learn more, continue to other great tutorials!   Christophe Viau In partnership with
 </html>

From:
https://schnipsl.qgelm.de/ - **Qgelm**

Permanent link:
https://schnipsl.qgelm.de/doku.php?id=wallabag:d3

Last update: **2021/12/06 15:24**