d3 – making things appear when you click them

There’s something oddly satisfying about making things appear through a mouse click. It’s like the webpage has recognised your supremacy over it and is bowing to accommodate your wishes. You are its master. You have power.

Are you a person who likes to feel powerful?

Read on. We’re going to create something truly amazing.

**Our aim: to click inside a rectangle, and have dancing people appear.**

Why dancing people, you ask?

Why not. There’s plenty of other tutorials that focus on creating circles. Dancing people are much more exciting.

First steps

I’m going to assume that you know where to write your d3 code. If you don’t, visit this (http://www.jeromecukier.net/blog/2012/09/04/getting-to-hello-world-with-d3/) link.

Like all basic d3.js visualisations, we need to create a SVG element to hold our graphics wizardry. This svg element needs to be specified with two attributes: height and width.

We can make our code flexible by defining variables for our height and width, rather than hard-coding the values. This has the advantage that if we want to change the dimensions of the SVG element later, we only need to change the code in one place. We’ll also create a variable called “padding” – this will act as space between the edge of the SVG element and the rectangle.

var svgWidth = 900

var svgHeight = 500

var padding = 5

Now we can create the svg element. We can store the result in a variable so that we can access it easily later.

var svg = d3.select("body").append("svg")

.attr("height", svgHeight)

.attr("width", svgWidth);

We can now add the rectangle inside the SVG element. What we will do is enclose the rectangle inside a HTML group element, and give the group an id name of “back”. This has a few advantages:

* If we want to extend our project later (perhaps by adding more background rectangles in a funky pattern), then this provides a convenient place to add the other background objects to.
* If we do in fact add other objects to the group, then we can style all of them easily. Using CSS selectors to select all elements inside the group with id of “back” is a breeze.

Let’s add the group and the rectangle. We’ll also specify the x and y coordinates of the rectangle as variables for convenience.

We’re going to want only the border of the rectangle to show. To achieve this, we can use d3’s syntax to set “fill” as “none” in the style of the SVG rectangle.

var rectx = 5

var recty = 5

svg.append("g")

.attr("id", "back")

.append("rect")

.attr("x", rectx)

.attr("y", recty)

.attr("width", svgWidth - padding )

.attr("height", svgHeight - padding)

.style("fill", "none")

.style("stroke", "black");

## Adding mouse events

It’s easy to add mouse events. To make something happen when we click on our rectangle, we can use the selection.on ([https://github.com/d3/d3/wiki/Selections#on](https://github.com/d3/d3/wiki/Selections" \l "on)) syntax:

svg.on("click", function()

{

//do something

});

Remember: we want to make dancing people appear when we click.

I’ve harvested an array of hilarious dancers (in gif format) using the magic of Google Images. Storing the web addresses of each gif in an array is a convenient way of accessing these gifs later on.

Of course, if the url for a gif changes, then the links aren’t going to point towards the correct image anymore. You could get around this by storing the gifs locally, but I’m going to use the web addresses because this project is only a bit of fun.

Here’s our array:

dancers = [

"http://static.tumblr.com/f20418d4dac3303f5653efc4a178f44a/905zxhy/iQEmjbvys/tumblr\_static\_tumblr\_mjatuzs0lg1rb4km6o1\_500.gif",

"http://www.playcast.ru/uploads/2016/03/21/17930826.gif",

"http://vomzi.com/wp-content/uploads/2016/02/latest-dancing-gifs-187.gif",

"http://bestanimations.com/Music/Dancers/3d-animated-girl-dancing-5.gif",

"http://www.degaston.org/Jeannette/dancing\_rocking\_woman.gif",

"https://66.media.tumblr.com/504cb94fe0f41e563f435ee2f833dcb5/tumblr\_n47d7f7uFv1rt5pgzo1\_400.gif",

"http://media0.giphy.com/media/3s7nLx9Gxhf32/giphy.gif",

"http://25.media.tumblr.com/62b196c124113db72bf3203a89db8bfb/tumblr\_n02qsfqLSJ1stmohyo1\_500.gif",

"http://media1.giphy.com/media/zVIlIw2PeRAS4/giphy.gif"

]

Then to make the dancers appear:

svg.on("click", function()

{

var coords = d3.mouse(this)

x = coords[0]

y = coords[1]

svg.append("svg:image")

.attr("xlink:href", dancers[Math.floor(Math.random()\*dancers.length)])

.attr("x",x)

.attr("y",y)

.attr("width", 40)

.attr("height", 100);

});

There’s a lot going on here, so I’ll break it down:

* Firstly, we get the coordinates of the mouse click with var coords = d3.mouse(this). The coordinates are stored in a two dimensional array, with the x coordinate as the first element and the y coordinate as the second element. We extract these in the next two lines and give each its own variable.
* Next, we add the gif to the page with svg.append("svg:image"). This kind of acts like a temporary image container, at least until we tell it the address of the image to display. We can do this by setting the “xlink:href” attribute of the image. So if we wrote something like

svg.append("svg:image")

.attr("xlink:href",”www.sweetstuff.com/deliciouscake.jpg”)

then the image at that hyperlink would display.

* We actually want to do something a bit cooler than display only one particular gif upon mouseclick– we're actually going to choose one dancer at random from our array. The line   
  .attr("xlink:href", dancers[Math.floor(Math.random()\*dancers.length)])

does this for us. We generate a random number between 0 and the length of the dancers array minus one, and then we select the corresponding array element.

- The final four lines simply set the attributes of our added gif. The height and width of the gif are somewhat arbitrary – you can make this as big or as small as you like.

And there you have it! You can now bask in your power and make your minions dance all over the screen for you.

Wasn't that fun?