# Transitions

* The start listener – a function that triggers at the start of your transition. By doing selection.each(“start”, function() ), you can trigger a function to run at the start of the transition.
* Note that the effects of this can be difficult to grasp initially. For example,

svg.append("circle")

.attr("cx", 10)

.attr("cy",100)

.attr("r", 10)

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

//Starting colour: black

//Result: black to blue transition

d3.select("circle")

.transition()

.delay(750)

.each("start", function() {d3.select(this).style("fill", "black"); } )

.style("fill", "blue");

//Starting colour: black

//Result: also black to blue transition

d3.select("circle")

.transition()

.delay(750)

.each("start", function() {d3.select(this).style("fill", "green"); } )

.style("fill", "blue");

Here, we change the fill of the circle to green when the animation starts. However, since we are changing the colour of the animation straight away, it appears we haven’t done anything.

Results are much more dynamic with something like:

d3.select("circle")

.transition()

.delay(750)

.each("start", function() {d3.select(this).attr("r", "100"); } )

.style("fill", "blue")

.each("end", function() {d3.select(this).attr("r", "10")} );

On the otherhand, we could just use the transition.styleTween() property:

d3.select("circle")

.transition()

.delay(500)

.duration(3000)

.styleTween("fill", function() { return d3.interpolate("green", "red"); } );

We can see how the circle changes from its starting black state to green when the transition starts. Increasing the duration makes this effect more visible.

To get styleTween working, you must specify both the start and end values, and the interpolator to use. The function d3.interpolate is smart – it can pick the type of interpolation that you need- and it does this by looking at the end value that the user specifies. If you choose a colour as the end value (as we did above), the interpolator of choice is d3.interpolateRgb ([https://github.com/d3/d3/wiki/Transitions#d3\_interpolateRgb](https://github.com/d3/d3/wiki/Transitions" \l "d3_interpolateRgb)). If we specify a number as the last value, then we use d3.interpolateNumber ([https://github.com/d3/d3/wiki/Transitions#d3\_interpolateNumber](https://github.com/d3/d3/wiki/Transitions" \l "d3_interpolateNumber)).

However, if we want to change the radius of our circle, we’ll have to use transition.attrTween ([https://github.com/d3/d3/wiki/Transitions#attrTween](https://github.com/d3/d3/wiki/Transitions" \l "attrTween)) instead. This is because now we’re changing an attribute of the circle, not its style. So we’d write

d3.select("circle")

.transition()

.duration(2000)

.delay(1000)

.attrTween("r", function () { return d3.interpolate(10, 50) } );

## Making your own interpolators

D3 contains a set of functions commonly used as interpolators. Sometimes, however, you just feel like coding your own.

Let’s assume that we want to make an interpolator to interpolate between the values 10 and 20. We’re going to need some variable to control how far through the interpolation we are. This value is called “t” by convention and will take values between 0 and 1.

So we get to work writing our function. We might come up with

function myInterpolator (t)

// t is between 0 and 1

// we want to interpolate between 10 and 20

{

return (10 + (20 - 10)\*t)

}

We can test this straightaway in the browser console. myInterpolator(1) will return 20, our end value, and myInterpolator(0) will return our starting value (0). Try it!

This is all well and great, but suppose we didn’t want to be restricted to hardcoding the values to interpolate between. What we need to do is make a closure (<https://en.wikipedia.org/wiki/Closure_(computer_programming)>. This will be a function that returns a function:

function interpolateNumber(a, b) {

return function(t) {

return a + t \* (b - a);

};

}

Let’s try it out:

var x = interpolateNumber(100,200)

x(0) //100

x(1) //200

x(0.4) //140

## Binding Data

Let’s say we want to bind data to our circle. We only have one circle, so we can do this with selection.datum([https://github.com/d3/d3/wiki/Selections#datum](https://github.com/d3/d3/wiki/Selections" \l "datum)):

svg.select("circle")  
.datum(300)

This will set the \_\_data\_\_ property of the circle to 300. We can also set the \_\_data\_\_ property manually with a bit of vanilla Javascript

document.getElementsByTagName("circle")[0].setAttribute("\_\_data\_\_", 300)

If we had multiple elements, we would use selection.data(https://github.com/d3/d3/wiki/Selections#data)

## Simple data join

svg.append("circle")

.attr("cx", 100)

.attr("cy",100)

.attr("r", 10)

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

svg.append("circle")

.attr("cx", 200)

.attr("cy",100)

.attr("r", 10)

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

svg.append("circle")

.attr("cx", 300)

.attr("cy",100)

.attr("r", 10)

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

var yValues = [50,100,150]

d3.selectAll("circle")

.data(yValues)

.attr("cy",function(d) { return d;});

## Simple Data Join with Transition

//with transition

d3.selectAll("circle")

.data(yValues)

.transition()

.duration(1000)

.attr("cy",function(d) { return d;});