ARTG 6900 Week 7

Interaction

Previous Weeks

We learned about the architecture of a complex visualization project
- Modular code
- Relationship between representation (“views”) and data (“model”)
- Event architecture

This Week

Interaction patterns
- Tooltip
- Selection with line graph
- Brush
- Zoom
- Scroll

d3.brush

1. Setting up a brush behavior

Very much similar to how we constructed our modules

```
var brush = d3.brush() //returns a function
```

1. Setting up a brush behavior

Invoke brush function with a selection.

```
var brush = d3.brush(); //returns a function
brush(selection); //returns a function
```

1. Setting up a brush behavior

Invoke brush function with a selection.

```
var brush = d3.brush(); //returns a function
brush(selection); //returns a function
```

...which is the same as

```
selection.call(brush);
```

2. Brush events

As brushing occurs, three events occur in sequence:
`start` - `brush` - `end`

```
var brush = d3.brush()
    .on('start', callback)
    .on('brush', callback)
    .on('brush.foo', callback)
    .on('end', callback)
; //returns a function
selection.call(brush);
```

2. Brush events

In each of the event callbacks, we have access to the following event arguments through `d3.event`

```
d3.event
    .target //brush function itself
    .type // ‘start’, ‘brush’, ‘end’
    .selection //numerical extent of the brush
    .sourceEvent

this //the <g> element (not the selection!)
```

3. Moving the brush programmatically (i.e. not with mouse or touch)

brush.move is a function that allows to programmatically move the brush to some arbitrary extent

```
brush.move(group, [x0,x1]);
```

...which is the same as

can be a transition too

```
group.call(brush.move, [x0,x1]);
```

Please review the following API

```
d3.brush()
brush.on
brush.extent
brush.move
```

d3.zoom

1. Setting up a zoom behavior

Very much similar to brush, construct a zoom function and invoke it with a selection as the argument

```
var zoom = d3.zoom() //returns a function
zoom(selection);
selection.call(zoom);
```

One caveat: what should selection be?

1. Setting up a zoom behavior

Very much similar to brush, construct a zoom function and invoke it with a selection as the argument

```
var zoom = d3.zoom() //returns a function
zoom(selection);
selection.call(zoom);
```

One caveat: what should selection be?

2. Zoom events

As zoom occurs, three events are emitted in succession: start, zoom, end

```
var zoom = d3.zoom() //returns a function
    .on('start',callback)
    .on('zoom',callback)
    .on('end',callback)
selection.call(zoom);
```

2. Zoom events

In these zoom events, we have access to the following event arguments

```
d3.event
    .type
    .target
    .sourceEvent
    .transform //the transform state

this
```

2. Zoom events and transform state

The transform state object: `d3.event.transform` has these attributes:

`x` --> translate by x
`y` --> translate by y
`k` --> scale by k

so that point [a,b] pre-zoom will become [a*k+x, b*k+y] post zoom

3. Zoom programmatically

Using zoom.transform function, we can manually zoom an element without mouse or touch events

```
zoom.transform(selection, transformState);
```

same as

```
selection.call(zoom.transform, transformState);
```

3. Zoom programmatically

In practice, we can construct a desired transformState using d3.zoomIdentity

```
selection.call(zoom.transform, d3.zoomIdentity);

selection.call(zoom.transform,
    d3.zoomIdentity.translate(x,y)
    .zoom(k)
);
```

Please review the following API

```
d3.zoom
zoom.on
zoom.transform
transform.translate
transform.scale
d3.zoomIdentity
```

Next Steps

Next Steps

Think about the role of interaction in your final project: what is useful, how do we implement it?

How do these intereactions relate to the larger event architecture? (You'll get to practice this with the assignment)

Review bootstrap and its many UI elements

Final project check-in: Wednesday, 3/1