The Ellipsis DSL

The Ellipsis Domain-Specific Language (DSL) is a small language embedded in JavaScript that enables programmatic authoring of narratives. This appendix provides code samples (see Figure 1) of the DSL in use. The DSL uses a method chaining convention similar to jQuery and D3.js: setter methods return the original object, enabling chained calls.

Wrapping a visualization: Figure 1(a) shows a snippet of the visualization wrapper used for the *Budget Forecasts* example. It exposes the visualization's data source, identifies its "stage" element using a CSS selector and dimensions, and enumerates constants and parameters. The code to draw the visualization is specified in `render`, which is called every time a parameter value is changed. Figure 1(b) shows how a visualization parameter can be bound to an extra-visualization control widget.

Defining annotations: The Ellipsis DSL provides a built-in palette of annotation types including ellipses, lines, arrows, rectangles, and labels. Figure 1(c) shows how one of these built-in annotation types can be instantiated and visual properties such as position and style specified. To allow for more complex annotation types, custom annotation templates can be defined as shown in Figure 1(d), which provides an embedded YouTube video. Each template consists of a unique identifier, along with `enter` and `exit` functions that are responsible for adding and removing the annotation from the display. If additional parameters are needed to instantiate a template (e.g., the identifier for a YouTube video), these may be defined in the template's method signatures and then specified on instantiation, as in Figure 1(e).

Creating scenes: Figure 1(f) illustrates how a scene is built. The `set` method sets the value of a visualization parameter; the `add` method includes an annotation. The first argument identifies the target visualization, allowing a scene to coordinate multiple independent visualizations. Paths through the narrative are determined using scene transitions, shown in 1(g). Transitions can be defined using a wildcard (`*`), indicating all possible origin-destination scene pairs, or as an array of origin and destination scene identifiers.

Using triggers: In Figures 1(h), (i), and (j), annotations are added to the scene only when triggered. Here, the triggers fire if (h) the `year` parameter is 1995, (i) the element with ID `showme` is clicked, and (j) two seconds has elapsed after the previous annotation was added.

```
var vis = el.vis('budgetForecast')
  .data(fullDataObj)
  .stage('#stage', 700, 400)

  .const('minYear', 1980)
  .const('scaleX', function() {...})

  .param('year', d3.range(1980, 2011))
  .param('plotForecasts', [true, false])

  .render(function() { /* Draw vis */ });
```
A

```
el.vis('budgetForecast')
    .bind('year', function() {
        // Update slider value
    })
```
B

```
var c = el.annotation('circle')
  .radius(10)
  .center([625, 30])
  .style('fill', 'firebrick')
```
C

```
el.annotation.def('youTubeComic')
    .enter(function(vidId, panel) {
        var vis = this.vis();
        ...
    })
    .exit(function(vidId, panel) {
        var data = this.data();
        ...
    })
```
D

```
var y = el.annotation('youTubeComic')
  .data([1, 2, 3])
  .args('NCc9XudToy8', 53)
```
E

```
el.scene('scene_1')
    .set('budgetForecast', 'year', 2010)
    .set('dowJones', 'year', 2010)

    .add('budgetForecast',
         el.annotation('circle')
             .radius(10)
             .center([625, 30])
             .style('fill', 'firebrick'))

    .add('dowJones', function() { ... })
```
F

```
el.scene.transition('*', '*',
  function(fromScene, toScene) {
    $('#caption-' + fromScene.id()).hide();
    $('#caption-' + toScene.id()).show();
});
```
G

```
el.scene('scene_2')
    .add('budgetForecast',
         annotationObj,
         el.trigger('budgetForecast')
             .where('year')
             .is(1995))
```
H

```
    .add('budgetForecast',
         annotationObj,
         el.trigger('#showme')
             .on('click'))
```
I

```
    .add('budgetForecast',
         annotationObj,
         el.trigger
             .afterPrev(2000))
```
J

Figure 1. Examples of Ellipsis DSL statements.