# Shadow Text

### API

#### useEl

Should ideally behave like a textElement: .text, .textAnchor, .letterSpacing, .style.opacity, .style.display, .style.fontSize, .style.fontFamily, ...

#### main

.style.fill, .style.opacity, .style.display, ...

#### light and shadow

Same as main, plus .x, .y

### Object Structure

main, light and shadow should be closures that hide a reference to the corresponding TextElement object. They should expose an API through which the element object can be manipulated. See fitbit-subclass-widget? Use separate constructor function or IIFE (see fitbit-subclass-widget)? Use prototype-chained objects for inheritance (see fitbit-polyline)?

widget, main, light and shadow APIs (including .style) might be able to extend/subclass/prototype from common (virtual) base classes.

#### Property Applicability

| **Property** | **widget (use)** | **main** | **light, shadow** |
| --- | --- | --- | --- |
| .style.opacity |  |  |  |
| .style.display |  |  |  |
| .style.fontSize |  |  |  |
| .style.fontFamily |  |  |  |
| .style.fill |  |  |  |
| .text |  |  |  |
| .textAnchor |  |  |  |
| .letterSpacing |  |  |  |
| .x | from <use> |  |  |
| .y | from <use> |  |  |

StyleCommon: opacity, display

StyleWidget extends StyleCommon: fontSize, fontFamily

StyleSubText extends StyleCommon: fill

Use class (with extend) to create objects that comprise the style API for widget and each subText.

### getBBox() DOMRect

| **Member** |  |  |
| --- | --- | --- |
| bottom | =y+height |  |
| height | fundamental |  |
| left | =x |  |
| right | =x+width |  |
| top | =y |  |
| width | fundamental |  |
| x | fundamental | reduce by max extent to which light or shadow is to left of main |
| y | fundamental | reduce by max extent to which light or shadow is above main |

Assume that mainEl always has display===’inline’.

Start with getBBox() from mainEl.

#### Adjustments caused by lightEl and shadowEl

If (display===’none’) 0; else...

x -= min(min(light.x,0), min(shadow.x,0))

### Integration Considerations

CSS style (even those in widgets/shadow-text/styles.css) take precedence over <set>, which implies CSS is applied AFTER <set>s have been executed. This surprised me, but I'm not sure it's anything we could change.

The CSS style selectors are fairly greedy, and may be applied to other elements with the same classNames. I think we addressed this somewhat in c-t.

If assigning a class to a s-t's <use>, we need to also include className 'widget-auto' or the widget won't be automatically created. This is logical and necessary to provide the option to bypass automatic creation, but can be a gotcha. (If a <use> doesn't specify a class, it's copied from that of the <symbol>, which is 'widget-auto'.)

When replacing a textEl with a shadowTextEl in a project, we have to change .style.fill to .main.style.fill. This is brutally logical, but reduces extent to which s-t can be a drop-in replacement for text. We could allow .style.fill on el, which only sets fill on main. Inconsistent?? Duplication?? Hypocrisy on my part??

widget.getBBox() returns the size of the text, including light and shadow if displayed. If you only want the size of main, call widget.main.getBBox().