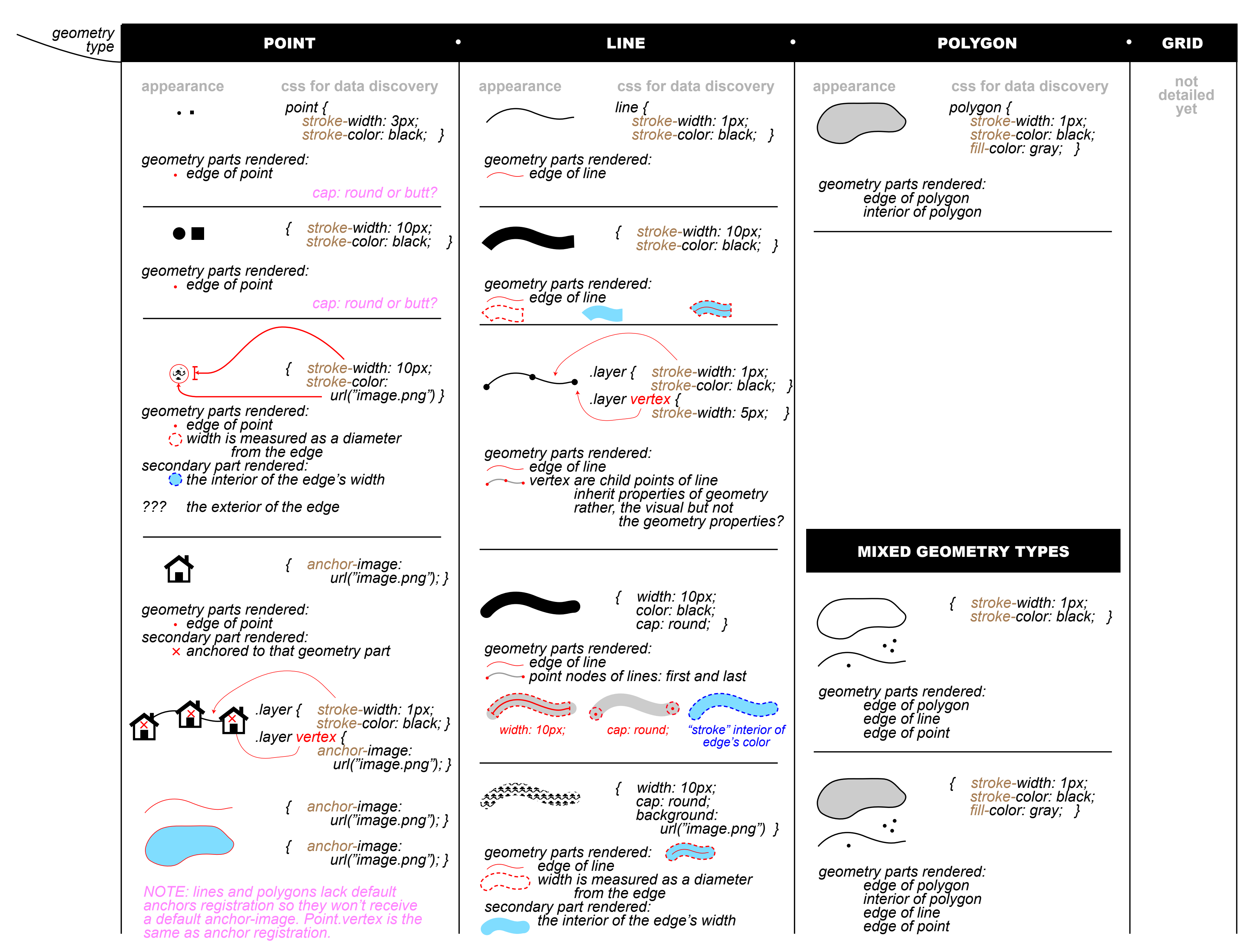


POINT

```
point {
    stroke-width: 3px;
    stroke-color: black;  }
```

geometry parts rendered:
- edge of point

cap: round or butt?

```
{   stroke-width: 10px;
    stroke-color: black;   }
```

geometry parts rendered:
- edge of point

cap: round or butt?

```
{   stroke-width: 10px;
    stroke-color:
        url("image.png") }
```

geometry parts rendered:
- edge of point
- width is measured as a diameter from the edge

secondary part rendered:
- the interior of the edge's width

??? the exterior of the edge

```
{   anchor-image:
        url("image.png"); }
```

geometry parts rendered:
- edge of point

secondary part rendered:
- × anchored to that geometry part

```
.layer {   stroke-width: 1px;
           stroke-color: black; }
.layer vertex {
           anchor-image:
             url("image.png"); }
```

```
{   anchor-image:
        url("image.png"); }

{   anchor-image:
        url("image.png"); }
```

NOTE: lines and polygons lack default anchors registration so they won't receive a default anchor-image. Point.vertex is the same as anchor registration.

LINE

```
line {
    stroke-width: 1px;
    stroke-color: black;  }
```

geometry parts rendered:
- edge of line

```
{   stroke-width: 10px;
    stroke-color: black;   }
```

geometry parts rendered:
- edge of line

```
.layer {   stroke-width: 1px;
           stroke-color: black;   }
.layer vertex {
           stroke-width: 5px;    }
```

geometry parts rendered:
- edge of line
- vertex are child points of line inherit properties of geometry rather, the visual but not the geometry properties?

```
{   width: 10px;
    color: black;
    cap: round;   }
```

geometry parts rendered:
- edge of line
- point nodes of lines: first and last

```
{   width: 10px;
    cap: round;
    background:
        url("image.png")  }
```

geometry parts rendered:
- edge of line
- width is measured as a diameter from the edge

secondary part rendered:
- the interior of the edge's width

POLYGON

```
polygon {
    stroke-width: 1px;
    stroke-color: black;
    fill-color: gray;   }
```

geometry parts rendered:
- edge of polygon
- interior of polygon

MIXED GEOMETRY TYPES

```
{   stroke-width: 1px;
    stroke-color: black;   }
```

geometry parts rendered:
- edge of polygon
- edge of line
- edge of point

```
{   stroke-width: 1px;
    stroke-color: black;
    fill-color: gray;   }
```

geometry parts rendered:
- edge of polygon
- interior of polygon
- edge of line
- edge of point

GRID

not detailed yet

CSS [geometry] appearance attributes bubble UP from points

defaults (mandated):

point:
    cap: round;   cap: round or butt?
line:
    cap: butt;
polygon:
    cap: butt;

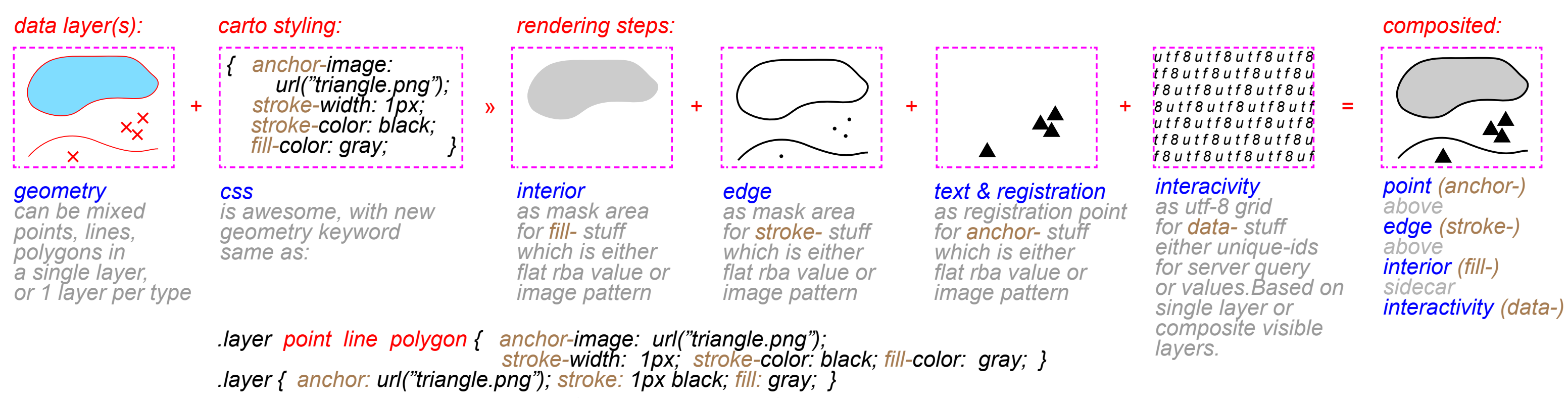
geometry (where)	css (part)	rendering (pivot)	ink/color (with)
point	anchor	registration	rgb
line	stroke	edge	gradient
polygon	fill	interior	image

.layer::attachment

is a repeated version of this for the same geometry

option to draw interior, edge, and registration per feature or per layer
(effectively makes 3 attachment layers)
question of how carto less.js loops attachment versus cascadenik's inner outer looping
question: what is difference between anchor and vertex?

raster and 3d are are combinations of the other geometry types
edge and interior have children and can reference them in the styles. Plus vertex and ringsinner.



```
{   anchor-image:
        url("triangle.png");
    stroke-width: 1px;
    stroke-color: black;
    fill-color: gray;         }
```

geometry
can be mixed points, lines, polygons in a single layer, or 1 layer per type

css
is awesome, with new geometry keyword same as:

interior
as mask area for fill- stuff which is either flat rba value or image pattern

edge
as mask area for stroke- stuff which is either flat rba value or image pattern

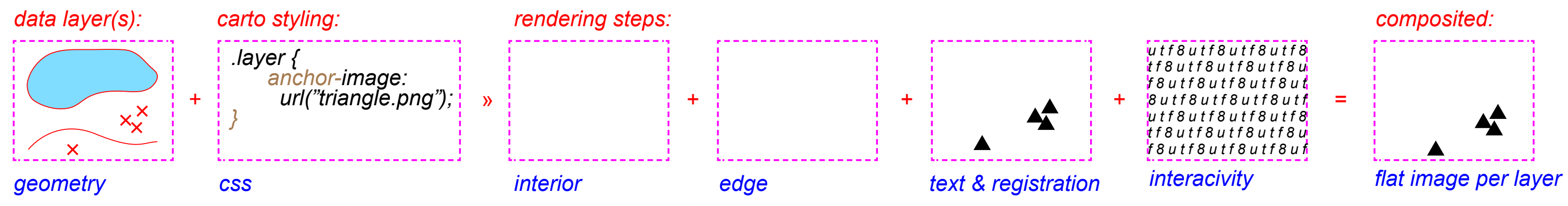
text & registration
as registration point for anchor- stuff which is either flat rba value or image pattern

interacivity
as utf-8 grid for data- stuff either unique-ids for server query or values.Based on single layer or composite visible layers.

point (anchor-)
above
edge (stroke-)
above
interior (fill-)
sidecar
interactivity (data-)

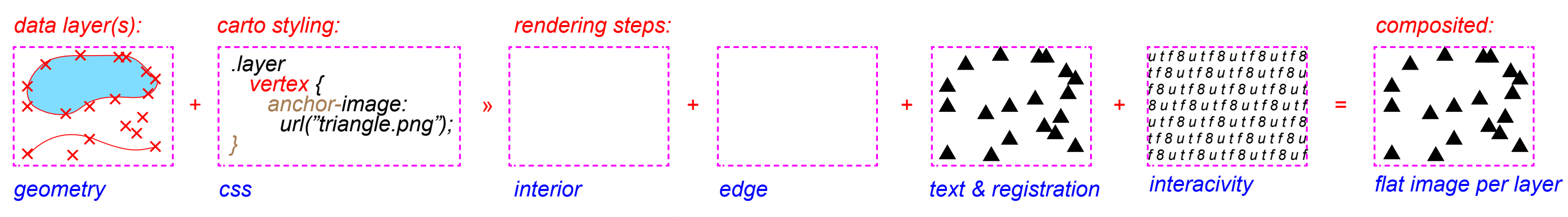
```
.layer  point  line  polygon {   anchor-image:  url("triangle.png");
                                 stroke-width:  1px;  stroke-color: black; fill-color:  gray;  }
.layer {  anchor: url("triangle.png"); stroke:  1px black; fill: gray;  }
```

note: point geometry are treated as if they have an edge for stroking.



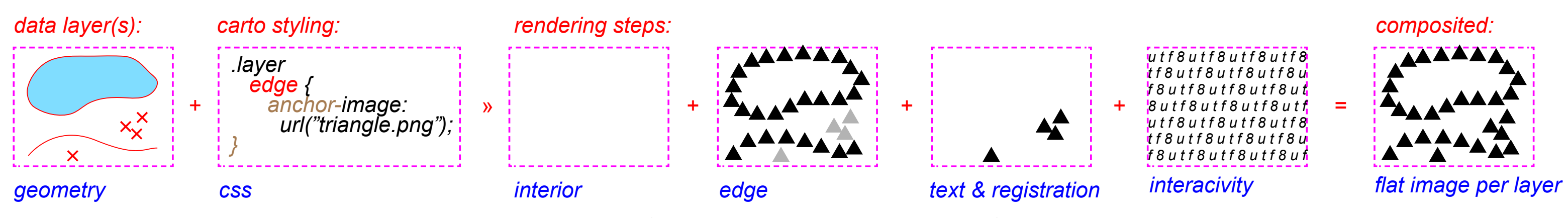
```
.layer {
    anchor-image:
      url("triangle.png");
}
```

```
.layer  point {  anchor-image: url("triangle.png");  }
.layer  registration {  anchor-image: url("triangle.png");  }
```



```
.layer
   vertex {
     anchor-image:
       url("triangle.png");
}
```

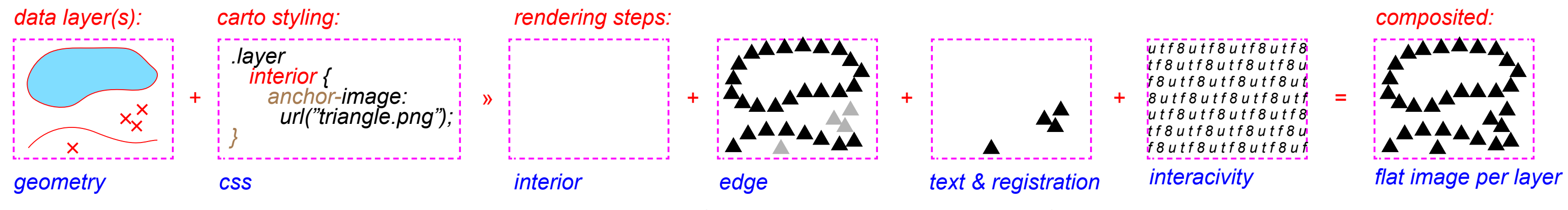
```
.layer  point.vertex  line.vertex  polygon.vertex {  anchor-image: url("triangle.png");  }
```



```
.layer
   edge {
     anchor-image:
       url("triangle.png");
}
```

```
.layer  point.edge  line.edge  polygon.edge  {  anchor-image: url("triangle.png");  }
```

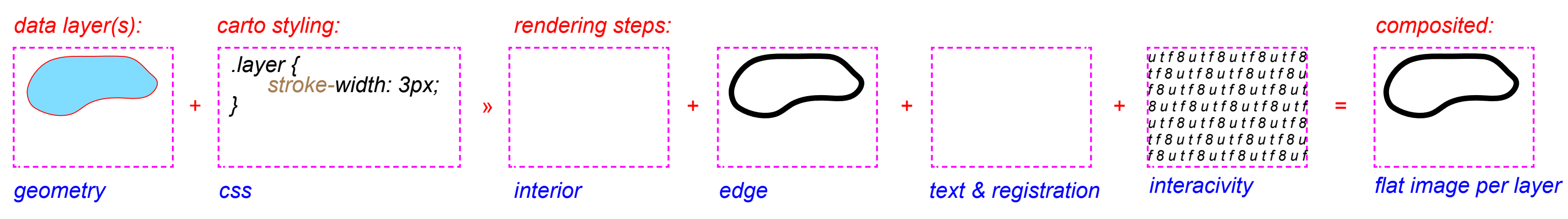
note: edge of point and default point registration are the same, skipped for edge and drawn in registration for consistent stacking
note: alignment tag for horizontal or tangent along line
note: repeat controls spacing offset between crowded pattern and sparse pattern
question: anchor-image or anchor-color? We're using color everywhere else, and allowing that to be a rgb, url, or gradient...



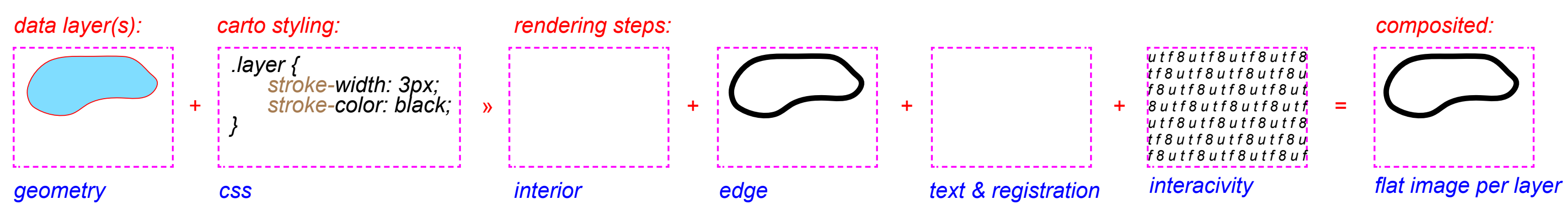
```
.layer
   interior {
     anchor-image:
       url("triangle.png");
}
```

```
.layer  point.interior  line.interior  polygon.interior  {  anchor-image: url("triangle.png");  }
```

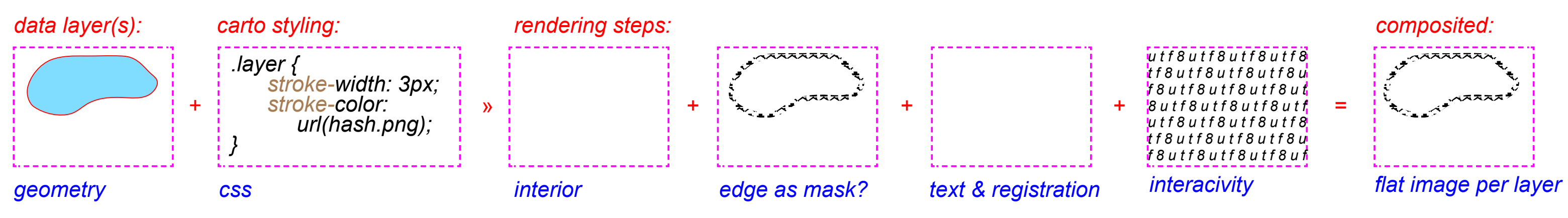
note: edge of point and default point registration are the same, skipped for edge and drawn in registration for consistent stacking
note: alignment tag for horizontal or tangent along line



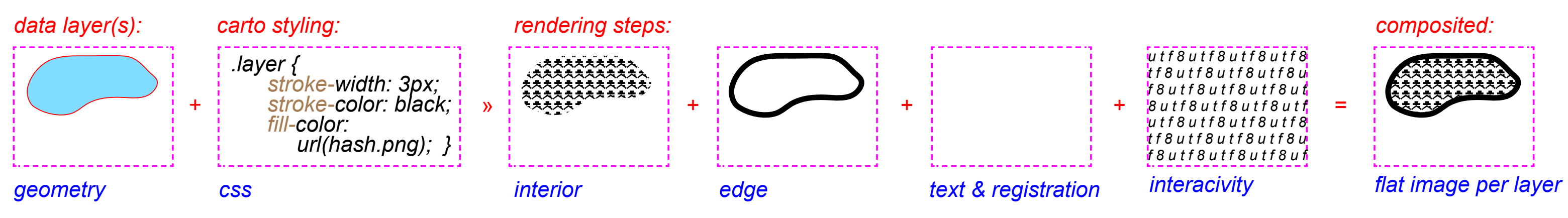
```
.layer {
    stroke-width: 3px;
}
```



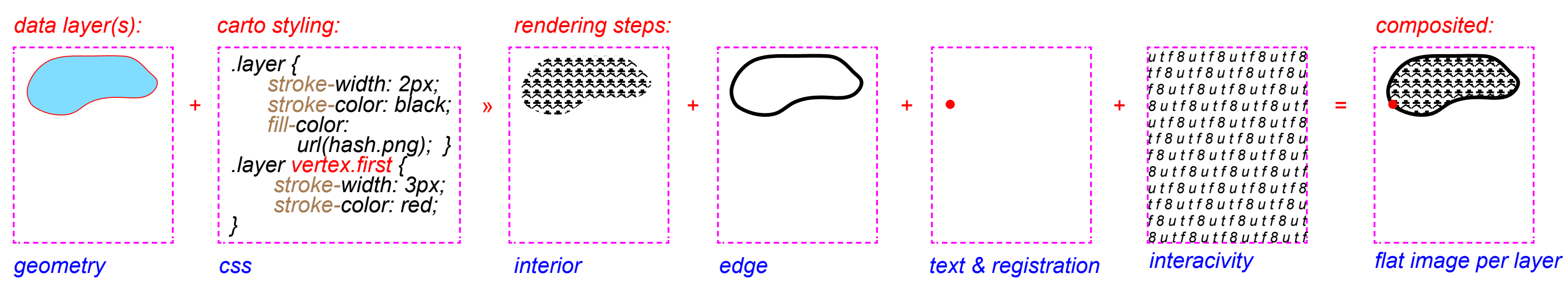
```
.layer {
    stroke-width: 3px;
    stroke-color: black;
}
```



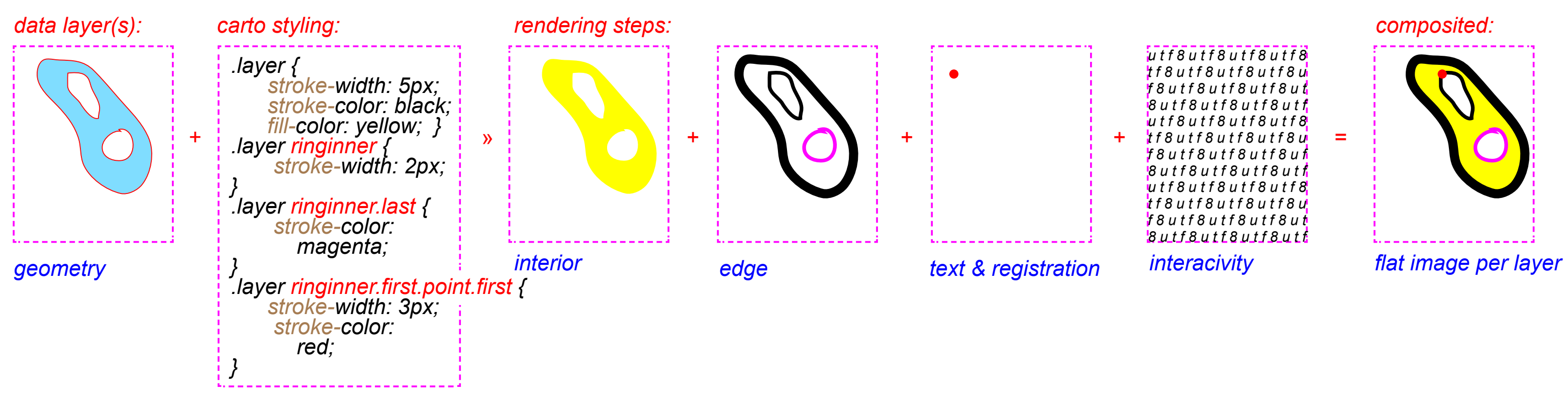
```
.layer {
    stroke-width: 3px;
    stroke-color:
        url(hash.png);
}
```



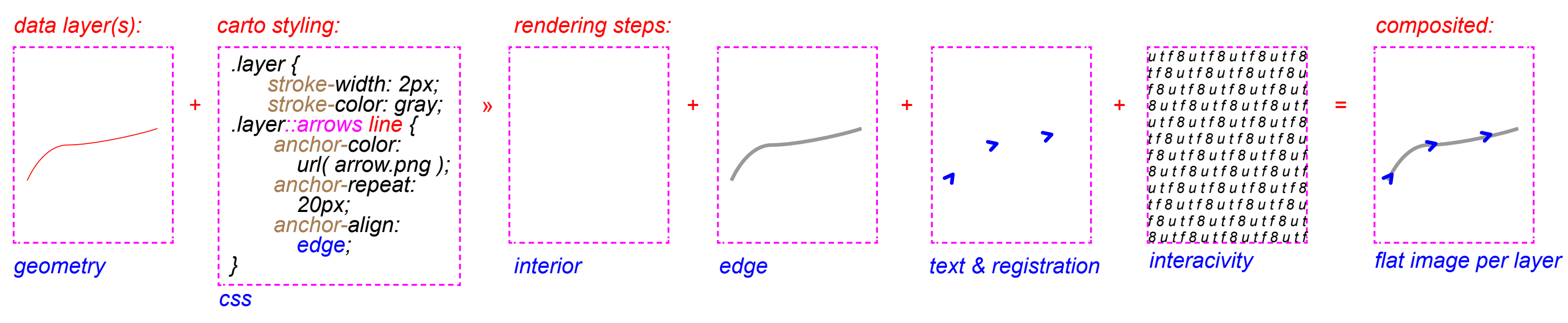
```
.layer {
    stroke-width: 3px;
    stroke-color: black;
    fill-color:
        url(hash.png);  }
```



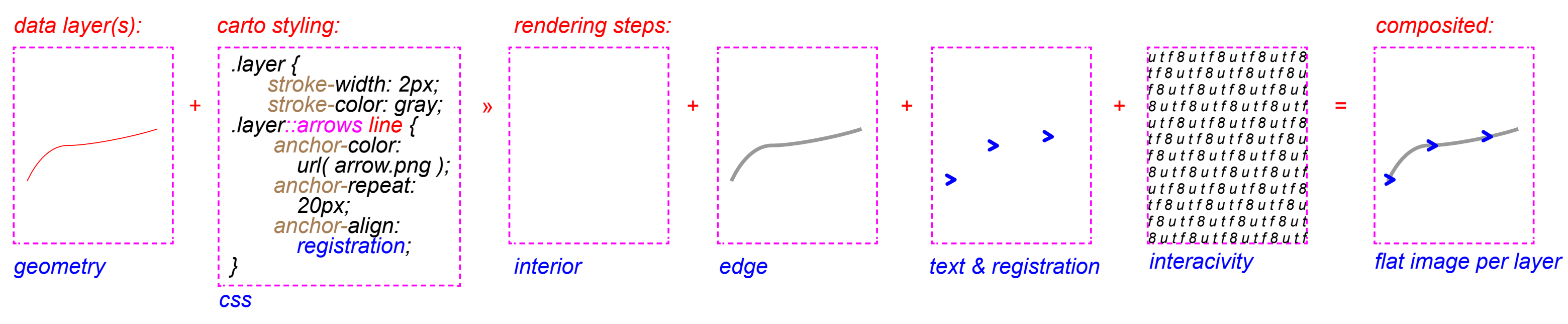
```
.layer {
    stroke-width: 2px;
    stroke-color: black;
    fill-color:
        url(hash.png);  }
.layer vertex.first {
     stroke-width: 3px;
     stroke-color: red;
}
```



```
.layer {
    stroke-width: 5px;
    stroke-color: black;
    fill-color: yellow;  }
.layer ringinner {
     stroke-width: 2px;
}
.layer ringinner.last {
     stroke-color:
        magenta;
}
.layer ringinner.first.point.first {
    stroke-width: 3px;
    stroke-color:
        red;
}
```

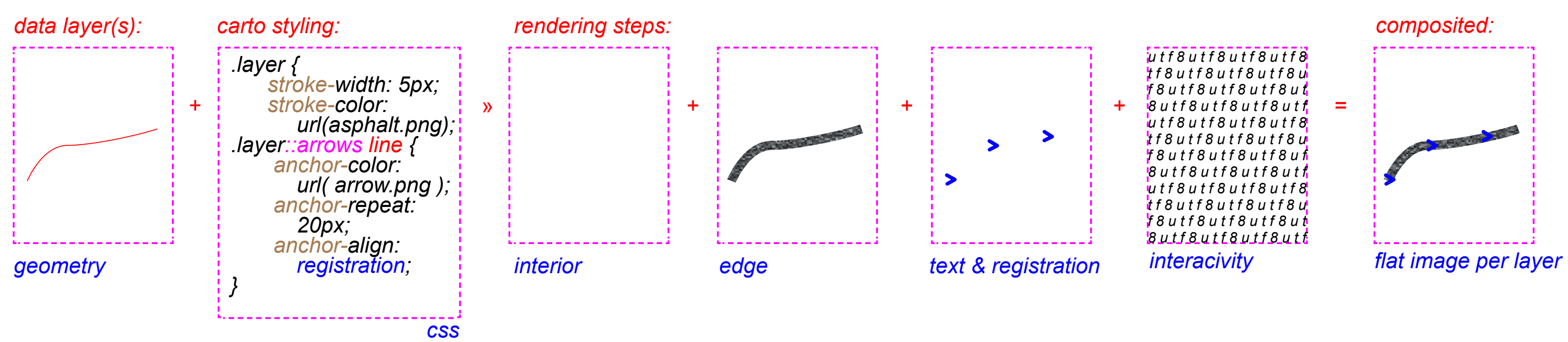


```
.layer {
    stroke-width: 2px;
    stroke-color: gray;
.layer::arrows line {
     anchor-color:
        url( arrow.png );
     anchor-repeat:
        20px;
     anchor-align:
        edge;
}
```

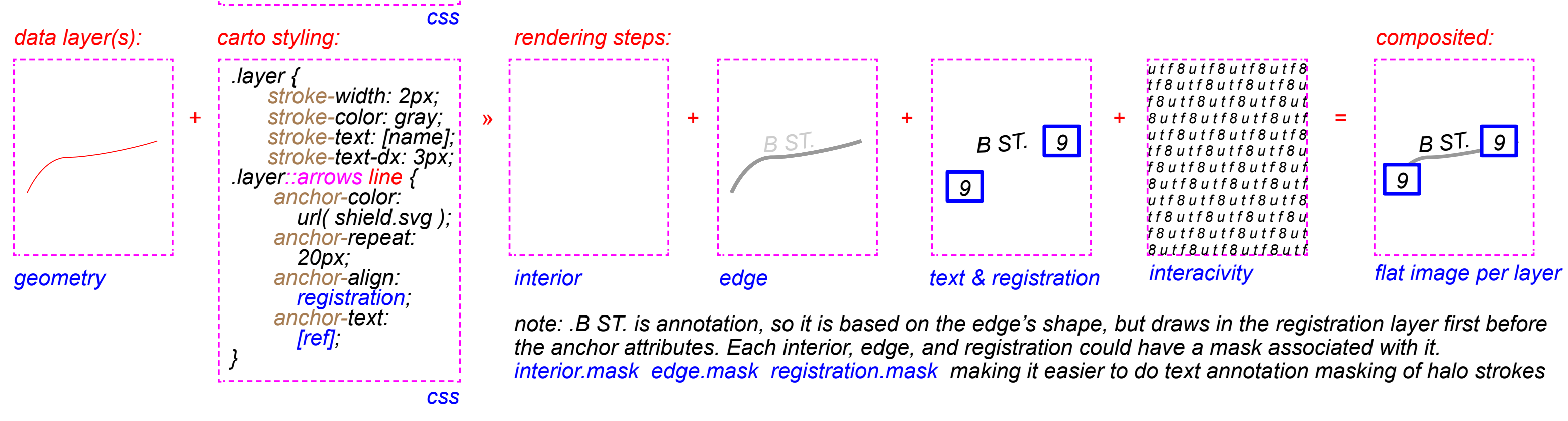


```
.layer {
    stroke-width: 2px;
    stroke-color: gray;
.layer::arrows line {
     anchor-color:
        url( arrow.png );
     anchor-repeat:
        20px;
     anchor-align:
        registration;
}
```

note: .layer::arrows line { is the same as .layer line {

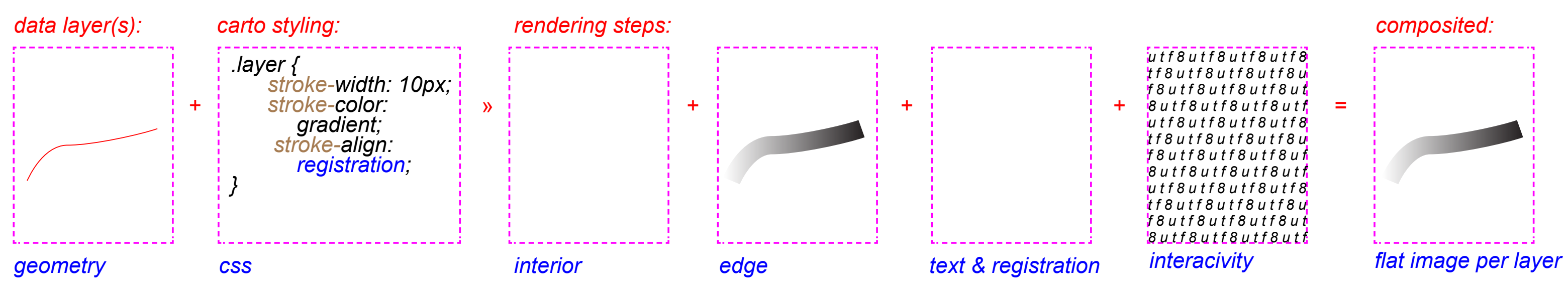


```
.layer {
    stroke-width: 5px;
    stroke-color:
        url(asphalt.png);
.layer::arrows line {
     anchor-color:
        url( arrow.png );
     anchor-repeat:
        20px;
     anchor-align:
        registration;
}
```

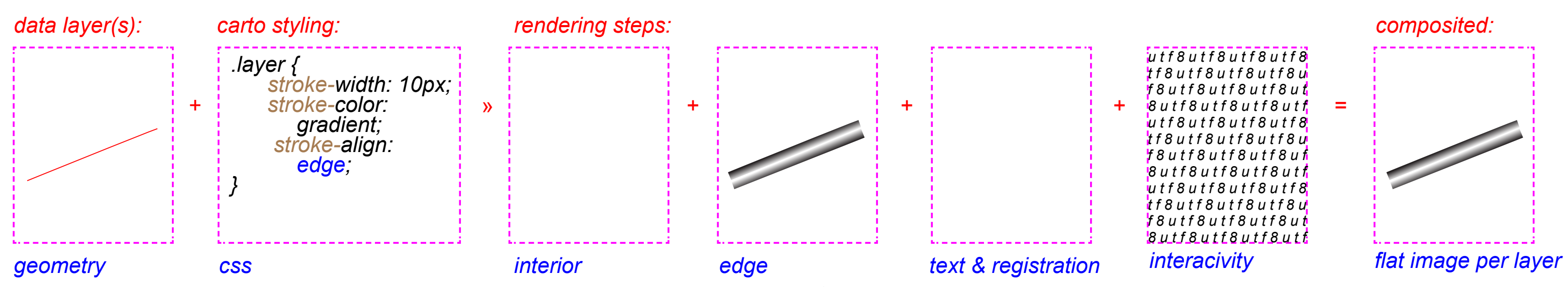


```
.layer {
    stroke-width: 2px;
    stroke-color: gray;
    stroke-text: [name];
    stroke-text-dx: 3px;
.layer::arrows line {
     anchor-color:
        url( shield.svg );
     anchor-repeat:
        20px;
     anchor-align:
        registration;
     anchor-text:
        [ref];
}
```

note: .B ST. is annotation, so it is based on the edge's shape, but draws in the registration layer first before the anchor attributes. Each interior, edge, and registration could have a mask associated with it.
interior.mask edge.mask registration.mask making it easier to do text annotation masking of halo strokes

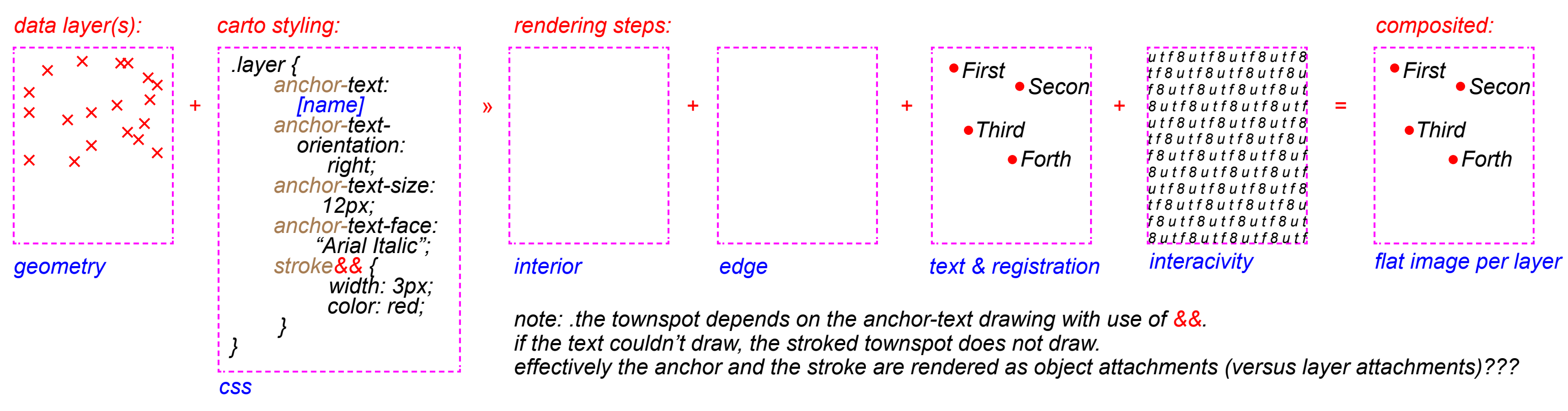


```
.layer {
    stroke-width: 10px;
    stroke-color:
        gradient;
    stroke-align:
        registration;
}
```



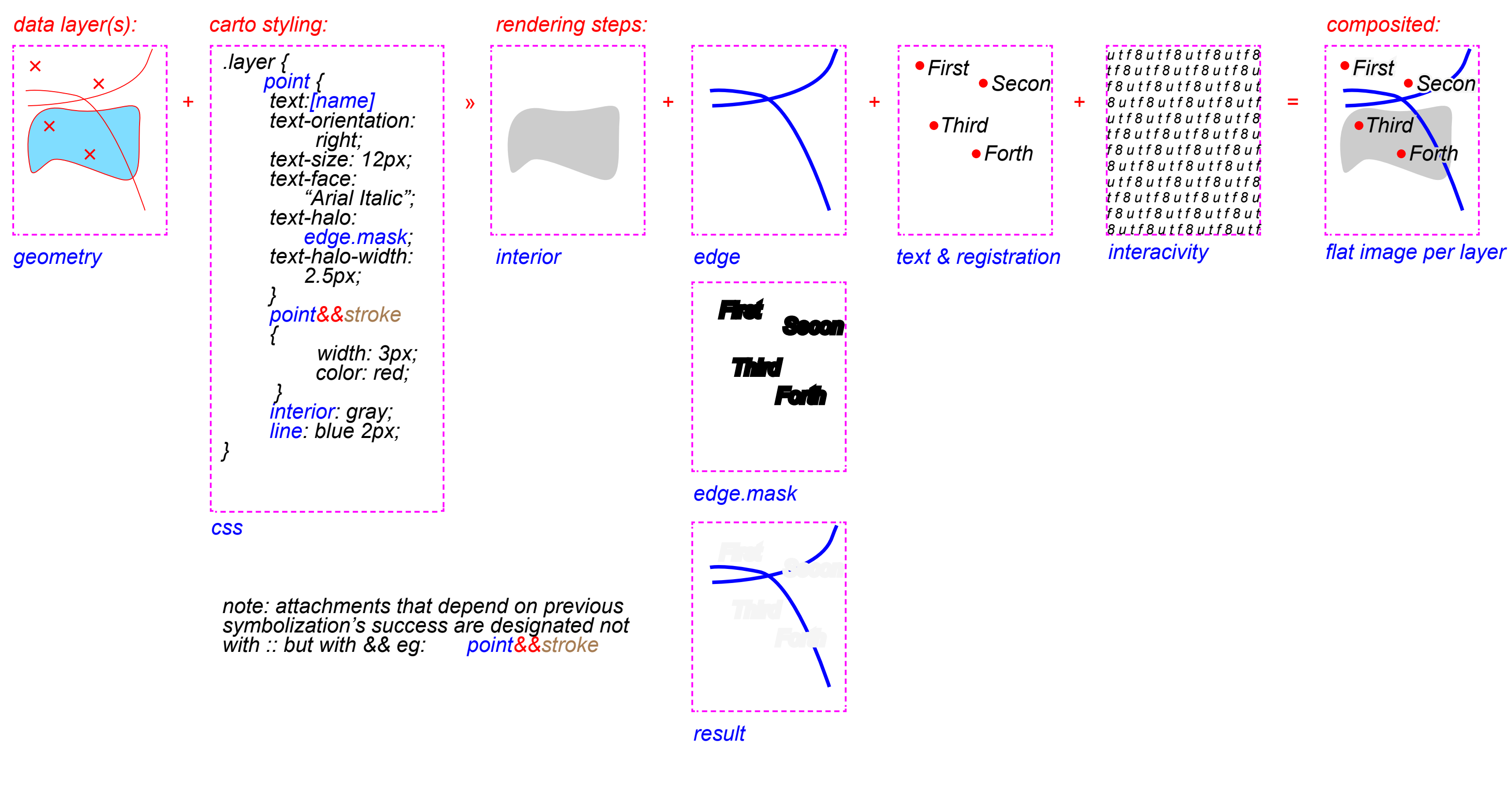
```
.layer {
    stroke-width: 10px;
    stroke-color:
        gradient;
    stroke-align:
        edge;
}
```

align: registration would be front line vertex.start to line vertex.end
align: map would be map's top to bottom



```
.layer {
     anchor-text:
        [name]
     anchor-text-
        orientation:
          right;
     anchor-text-size:
        12px;
     anchor-text-face:
        "Arial Italic";
     stroke&& {
        width: 3px;
        color: red;
     }
}
```

note: .the townspot depends on the anchor-text drawing with use of &&.
if the text couldn't draw, the stroked townspot does not draw.
effectively the anchor and the stroke are rendered as object attachments (versus layer attachments)???



```
.layer {
    point {
     text:[name]
     text-orientation:
          right;
     text-size: 12px;
     text-face:
        "Arial Italic";
     text-halo:
        edge.mask;
     text-halo-width:
        2.5px;
    }
    point&&stroke
    {
        width: 3px;
        color: red;
    }
    interior: gray;
    line: blue 2px;
}
```

note: attachments that depend on previous symbolization's success are designated not with :: but with && eg: point&&stroke