Porting D3JS to kotlin multiplaform

Gaëtan Zoritchak
@gz_k

Pierre Mariac
@imtam5

The idea

The idea

D3.js

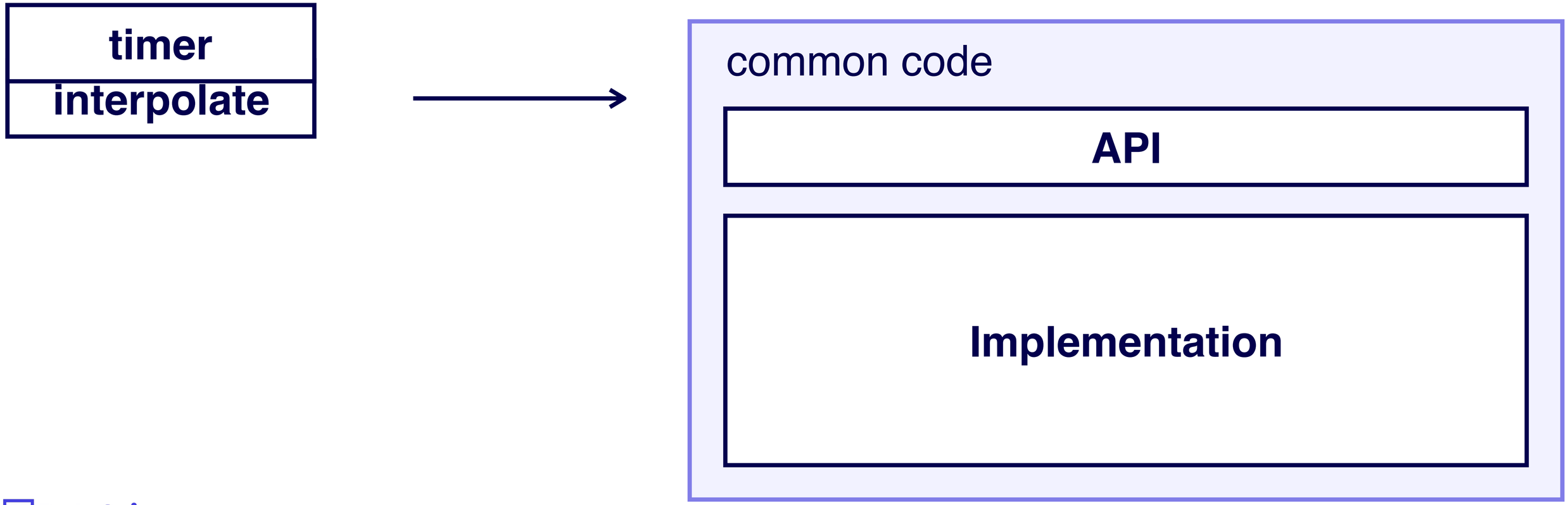
shape **force** **…**

data2viz

JS **JFx** **Android**

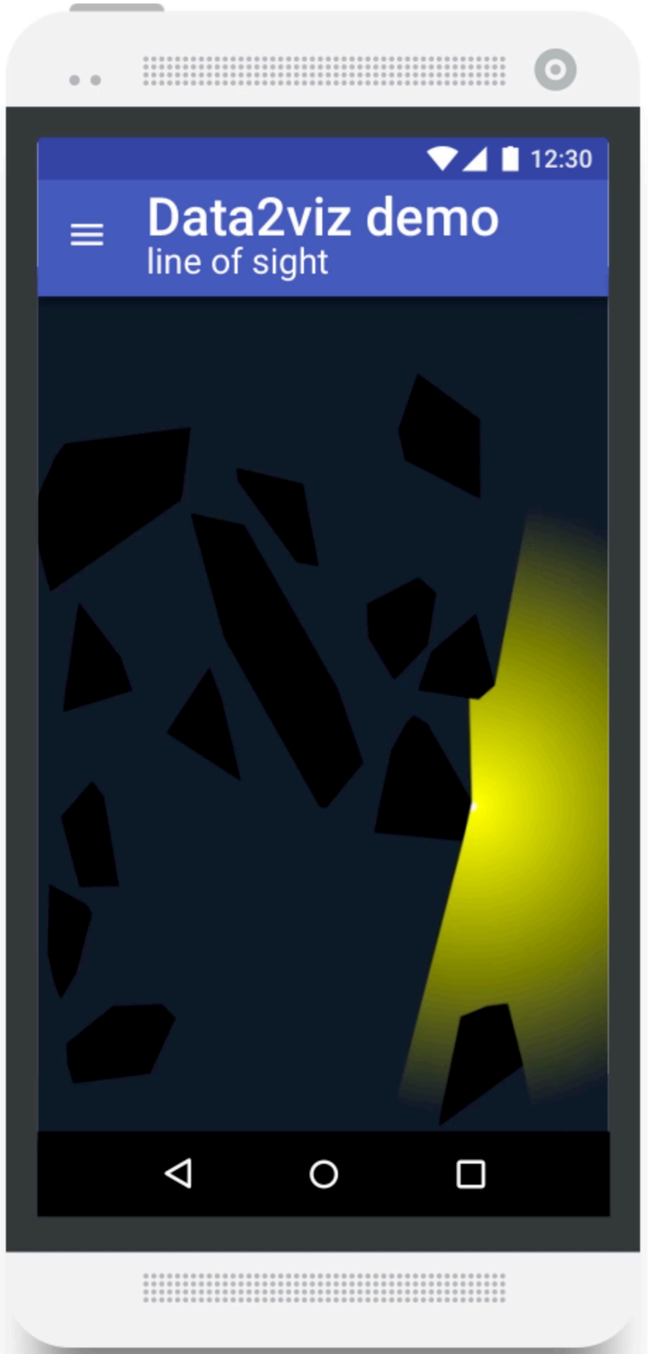
isomorphic, open source

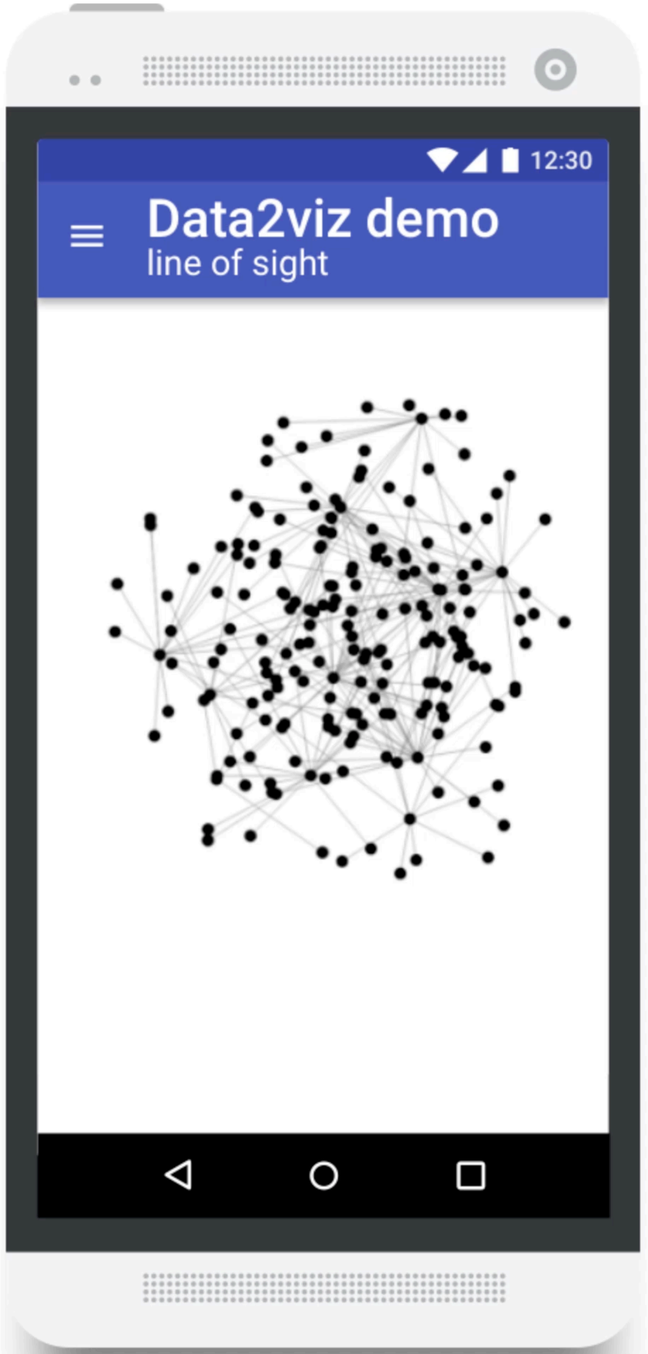
The idea



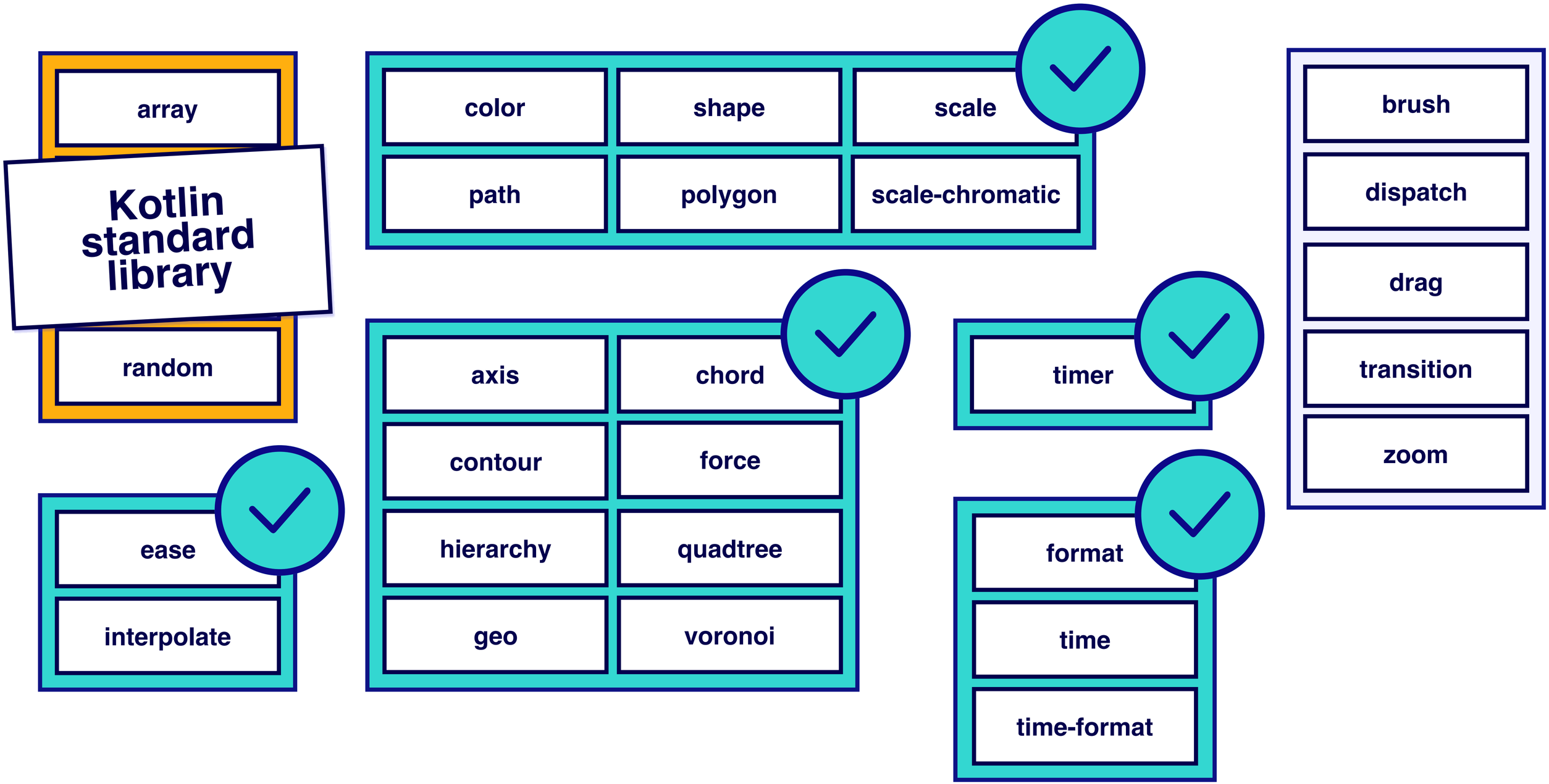
What is a dataviz library?


12:30
Data2viz demo
chord


12:30
Data2viz demo
line of sight


12:30
Data2viz demo
line of sight


12:30
Data2viz demo
world


array
Kotlin standard library
random
color
shape
scale
path
polygon
scale-chromatic
brush
dispatch
drag
transition
zoom
axis
chord
contour
force
hierarchy
quadtree
geo
voronoi
timer
ease
interpolate
format
time
time-format

How can Kotlin improve D3 API ?

Better code structure

```
d3.select("#viz")
  .selectAll("g")
  .data(temperatures)
  .enter().append("g")
    .append("circle")
      .attr("cx", 10)
      .attr("r", function(d) {
        return radiusScale(d.celsius);
      })
      .attr("style", "fill: #FFAA00;")
```

```
viz {
  temperatures.forEach {
    group {
      circle {
        x = 10.0
        radius = radiusScale(it.celsius)
        style.fill = 0xFFAA00.color
      }
    }
  }
}
```

Type everything

```
d3.select("#viz")
  .selectAll("g")
  .data(temperatures)
  .enter().append("g")
    .append("circle")
      .attr("cx", 10)
      .attr("r", function(d) {
        return radiusScale(d.celsius);
      })
      .attr("style", "fill: #FFAA00;")
```

```
viz {
  temperatures.forEach {
    group {
      circle {
        x = 10.0
        radius = radiusScale(it.celsius)
        style.fill = 0xFFAA00.color
      }
    }
  }
}
```

DSLs

```
d3.select("#viz")
  .selectAll("g")
  .data(temperatures)
  .enter().append("g")
    .attr("transform", function(d, i) {
      return "translate(0," + yScale(i) + ")";
    })
  .append("circle")
    .attr("cx", 10)
    .attr("r", function(d) {
      return radiusScale(d.celsius);
    })
    .attr("style", "fill: #FFAA00;")
```

```
viz {
  temperatures.forEachIndexed { index, temp ->
    group {
      transform {
        translate(.0, yScale(index))
      }
      circle {
        x = 10.0
        radius = radiusScale(temp.celsius)
        style.fill = 0xFFAA00.color
      }
    }
  }
}
```

Bootstrapping

```
fun main(args: Array<String>) {
  viz {
    temperatures.forEachIndexed { index, temp ->
      group {
        transform {
          translate(.0, yScale(index))
        }
        circle {
          x = 10.0
          radius = radiusScale(temp.celsius)
          style.fill = 0xFFAA00.color
        }
      }
    }
  }
}
```

Bootstrapping

Pure common code
available on any platform

```
fun main(args: Array<String>) {
  viz {
    temperatures.forEachIndexed { index, temp ->
      group {
        transform {
          translate(.0, yScale(index))
        }
        circle {
          x = 10.0
          radius = radiusScale(temp.celsius)
          style.fill = 0xFFAA00.color
        }
      }
    }
  }
}
```

Bootstrapping

Pure common code
available on any platform

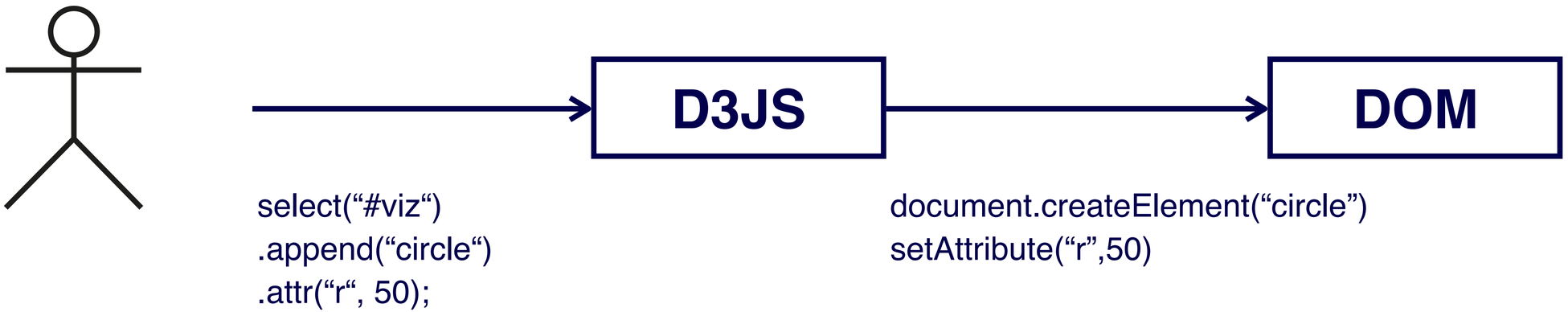
```
fun main(args: Array<String>) {
  viz {
    temperatures.forEachIndexed { index, temp ->
      group {
        transform {
          translate(.0, yScale(index))
        }
        circle {
          x = 10.0
          radius = radiusScale(temp.celsius)
          style.fill = 0xFFAA00.color
        }
      }
    }
  }.bindRendererOn("myCanvas")
}
```

This extension function is
only available on the
Javascript platform

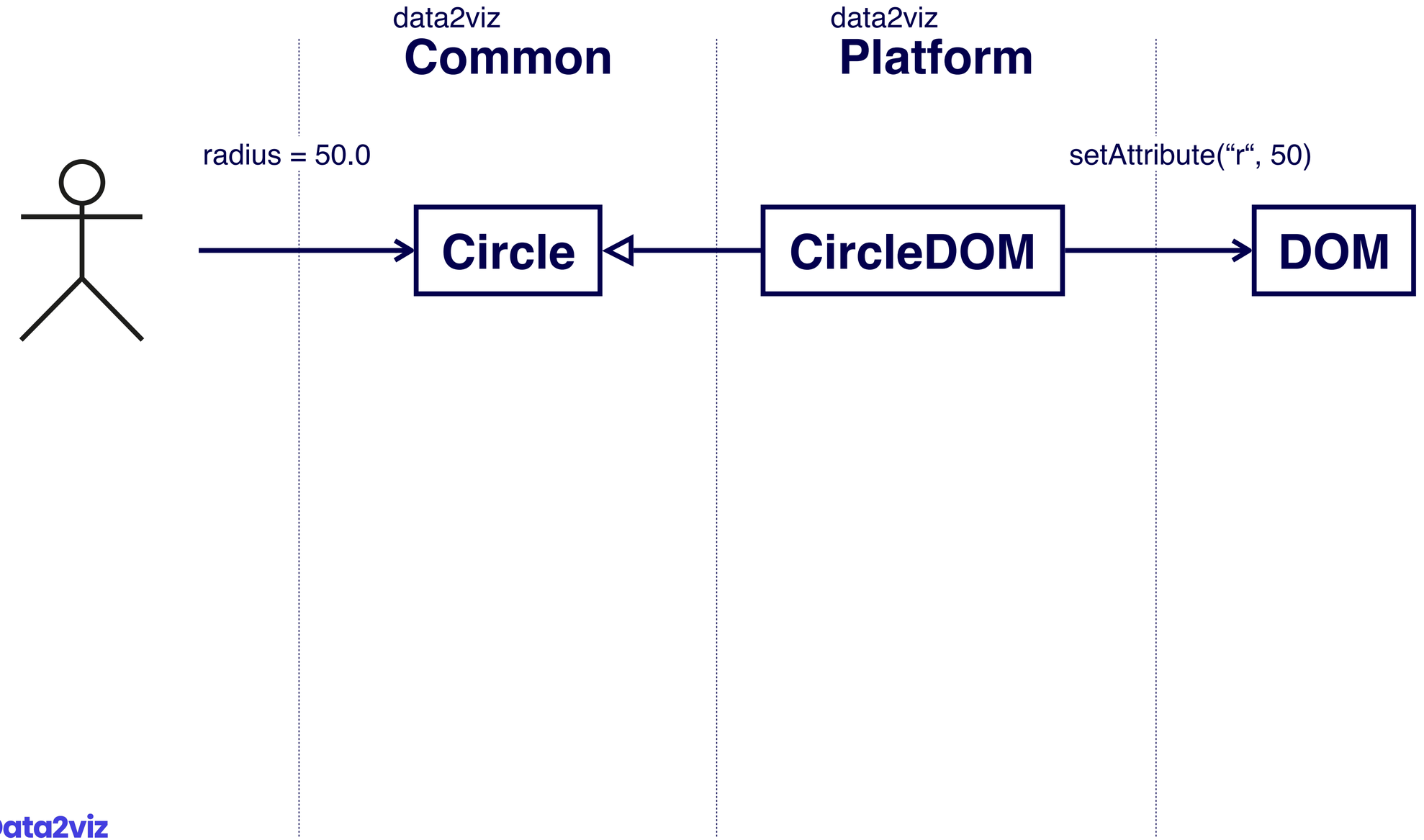
Multiplatform Development

Architecture

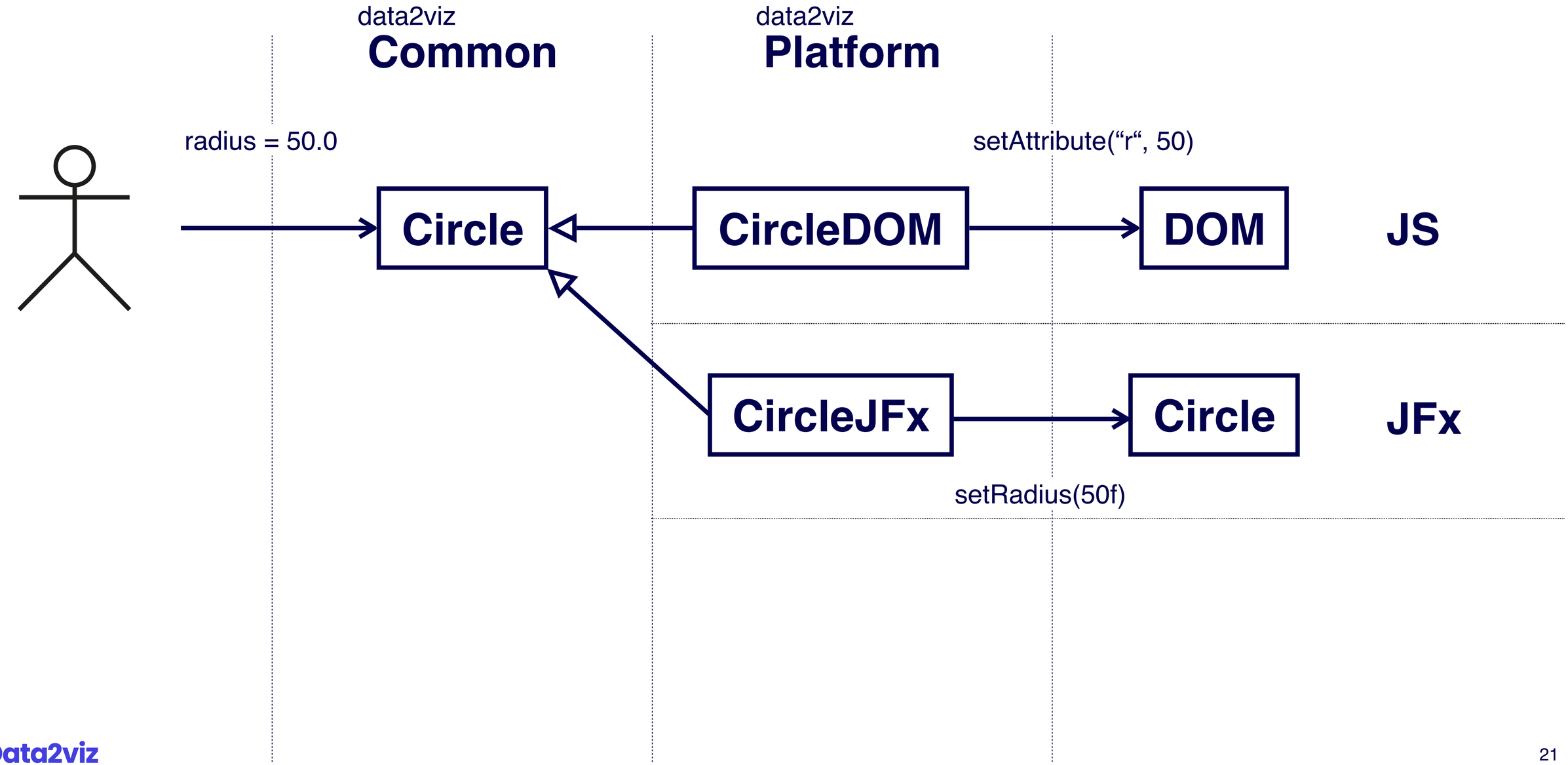
Architecture



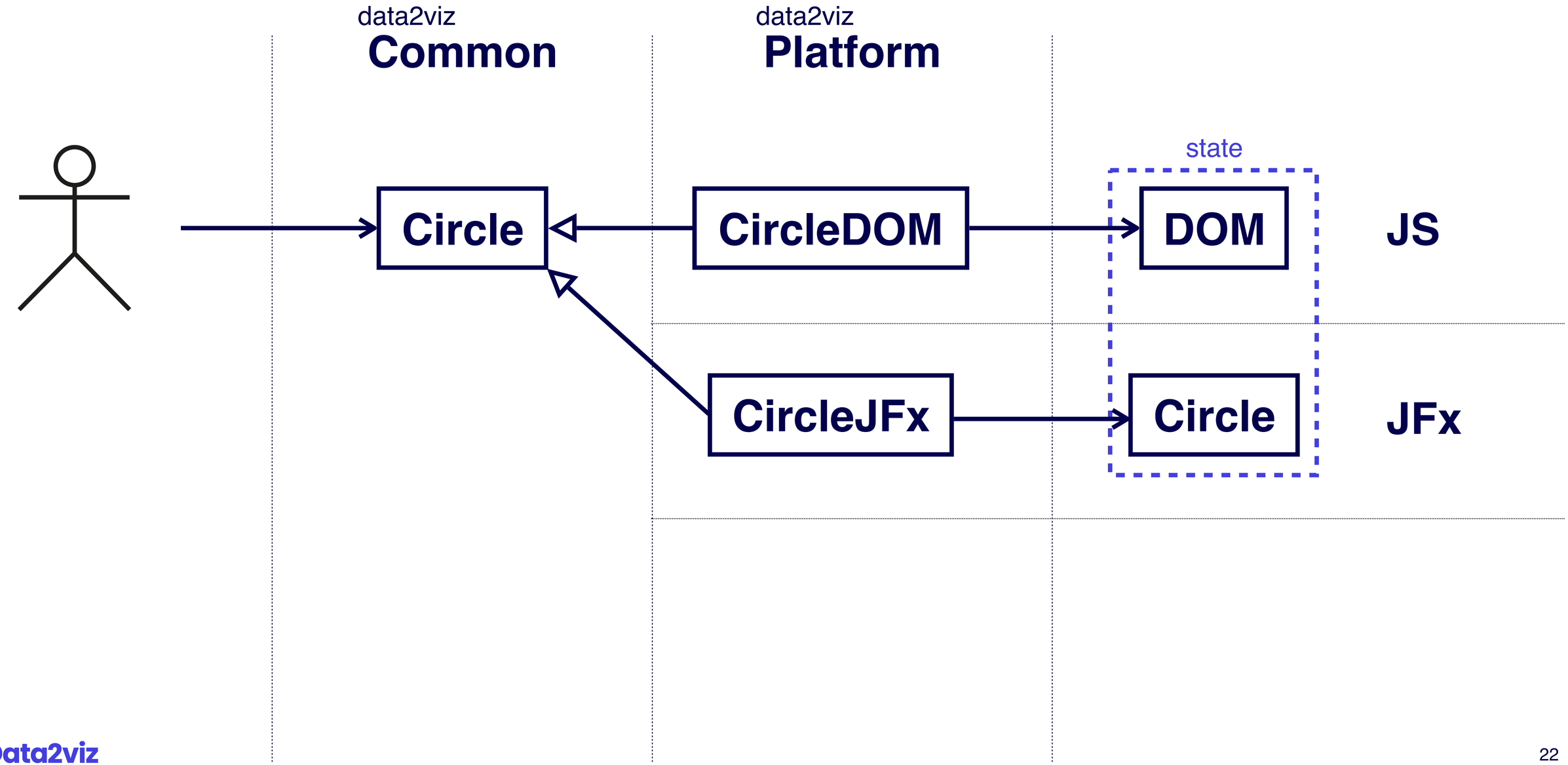
Architecture



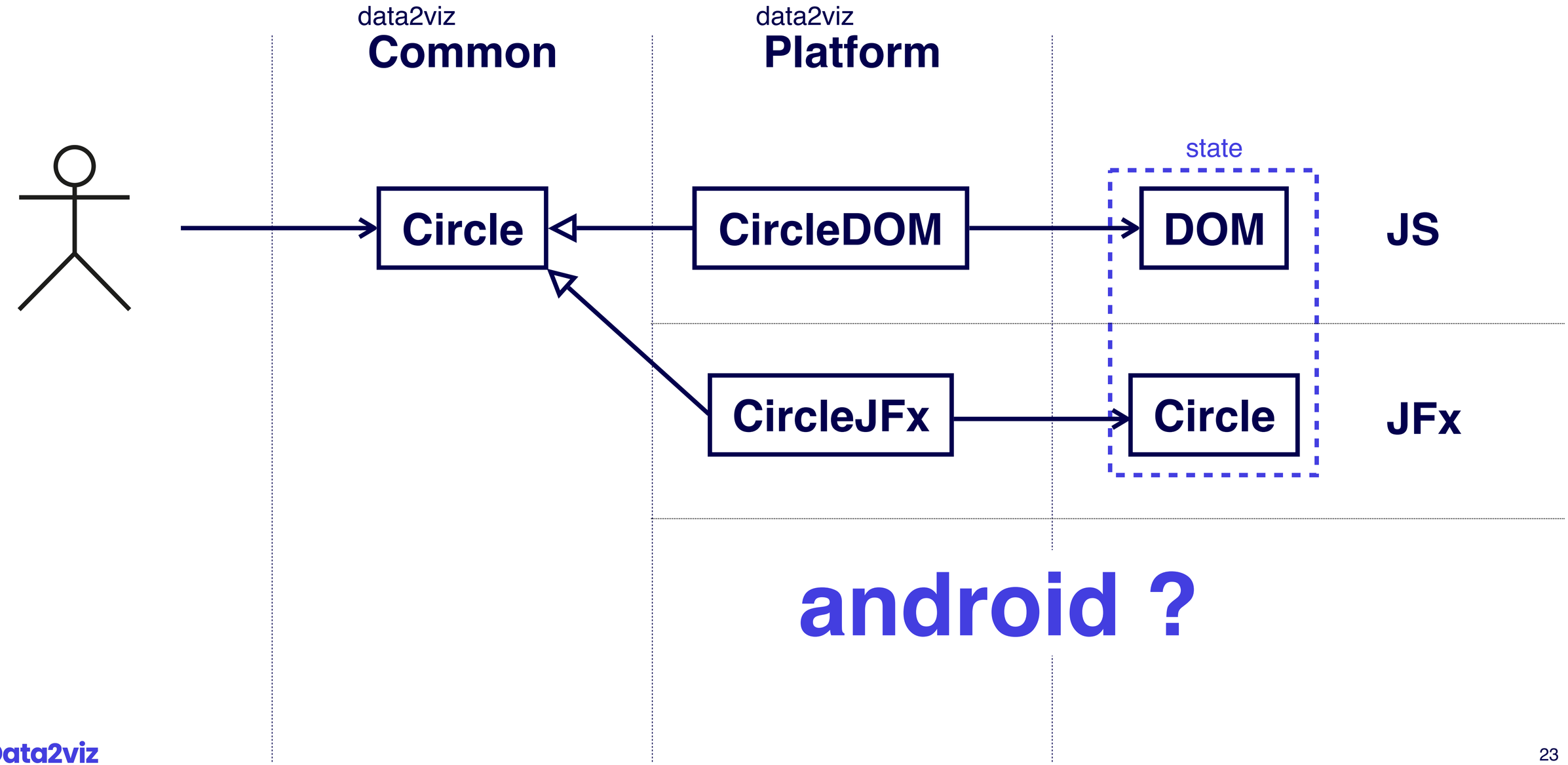
Architecture



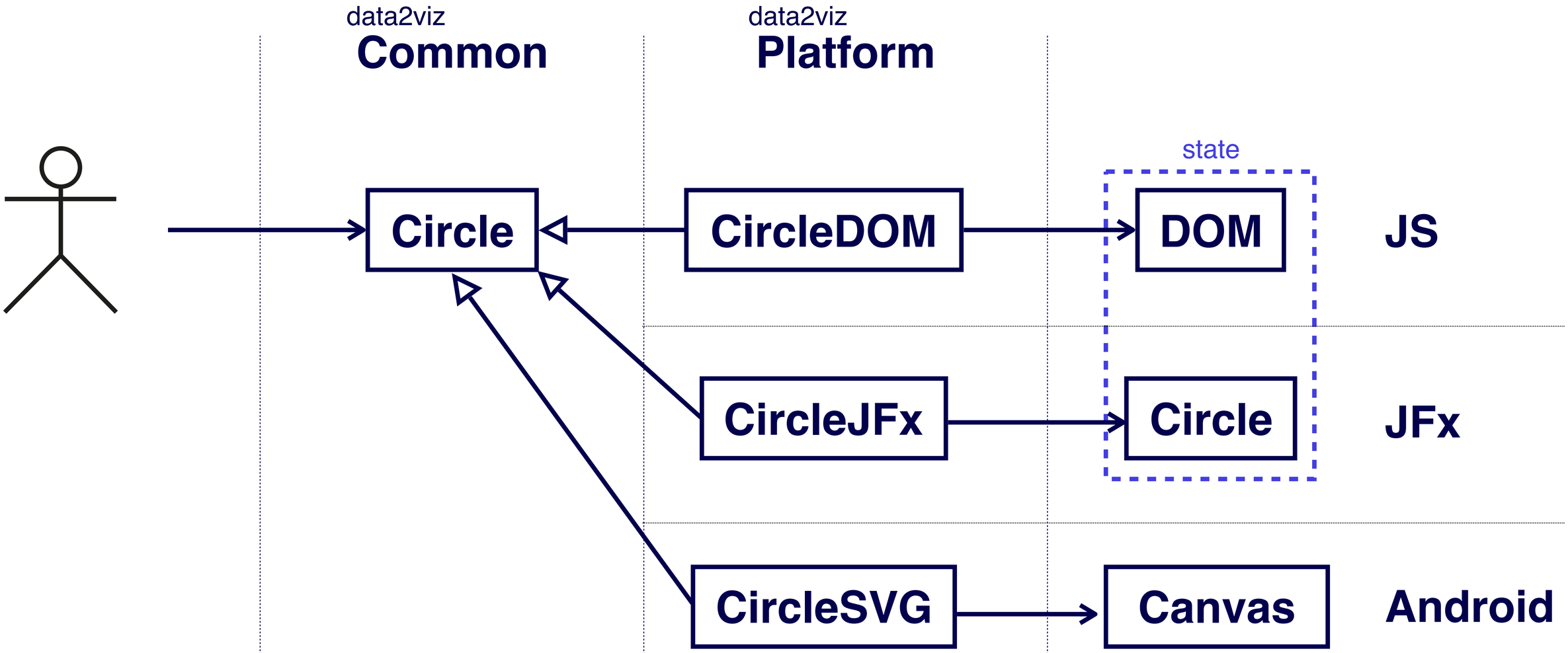
Architecture



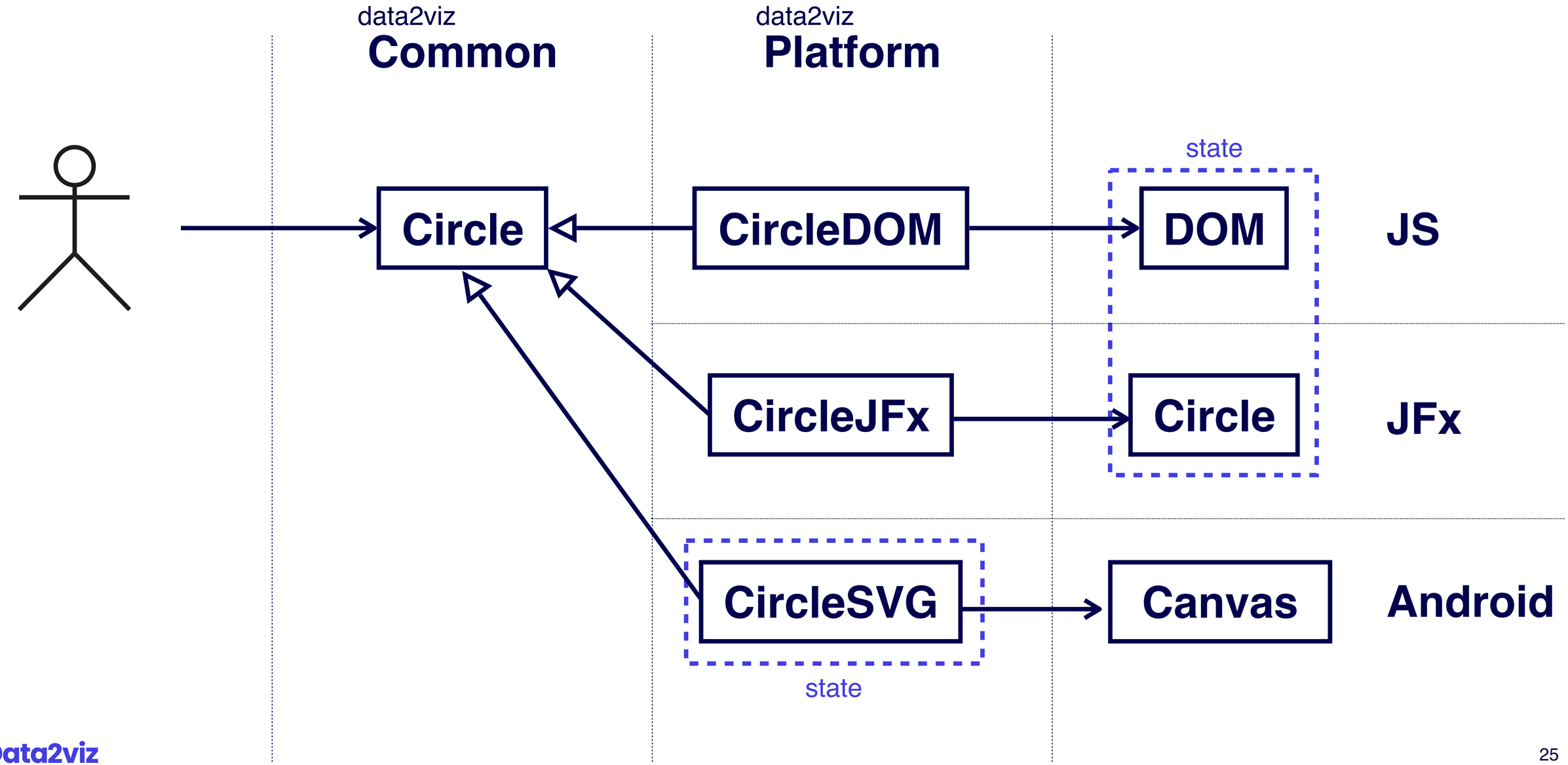
Architecture



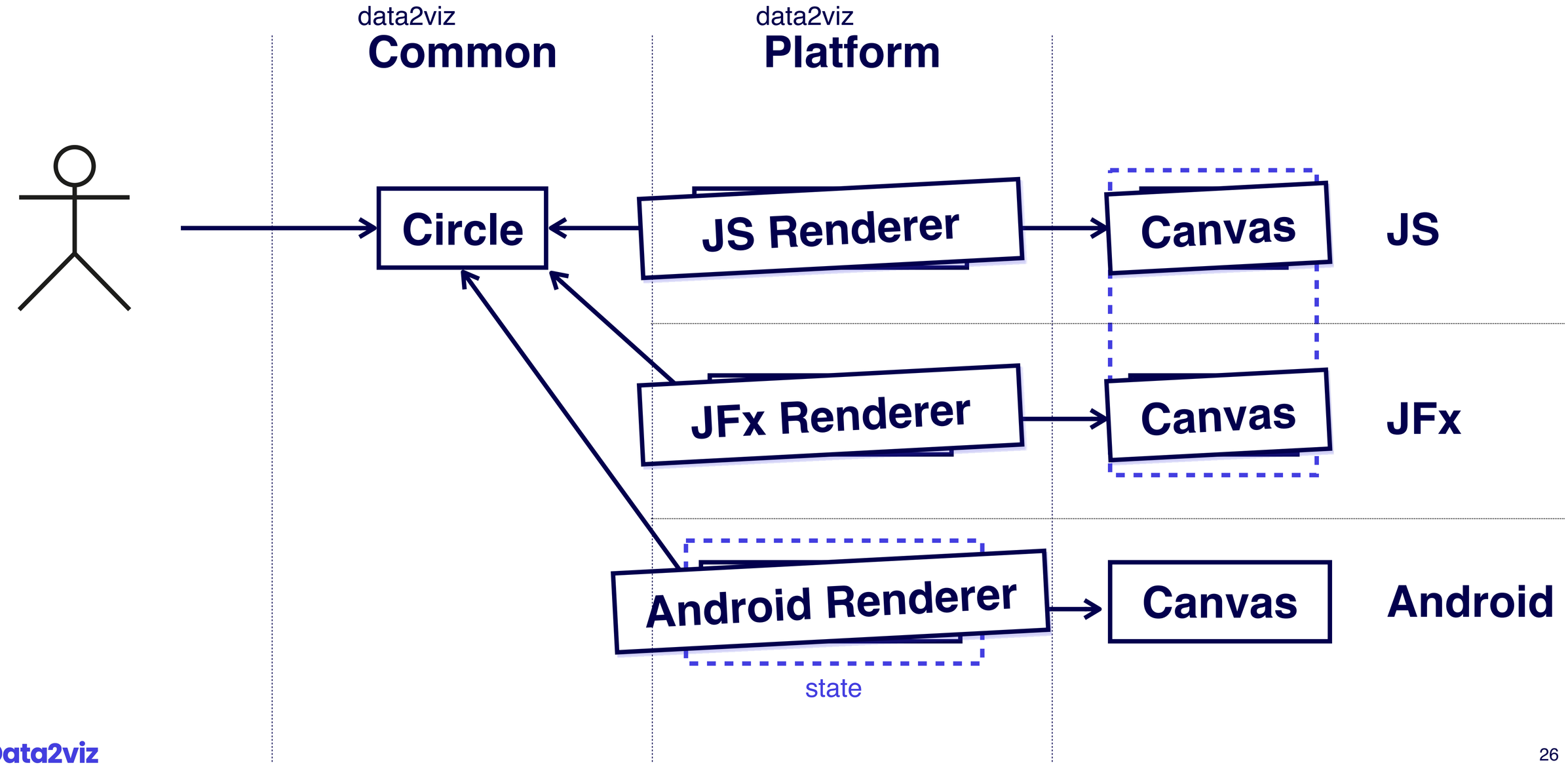
Architecture



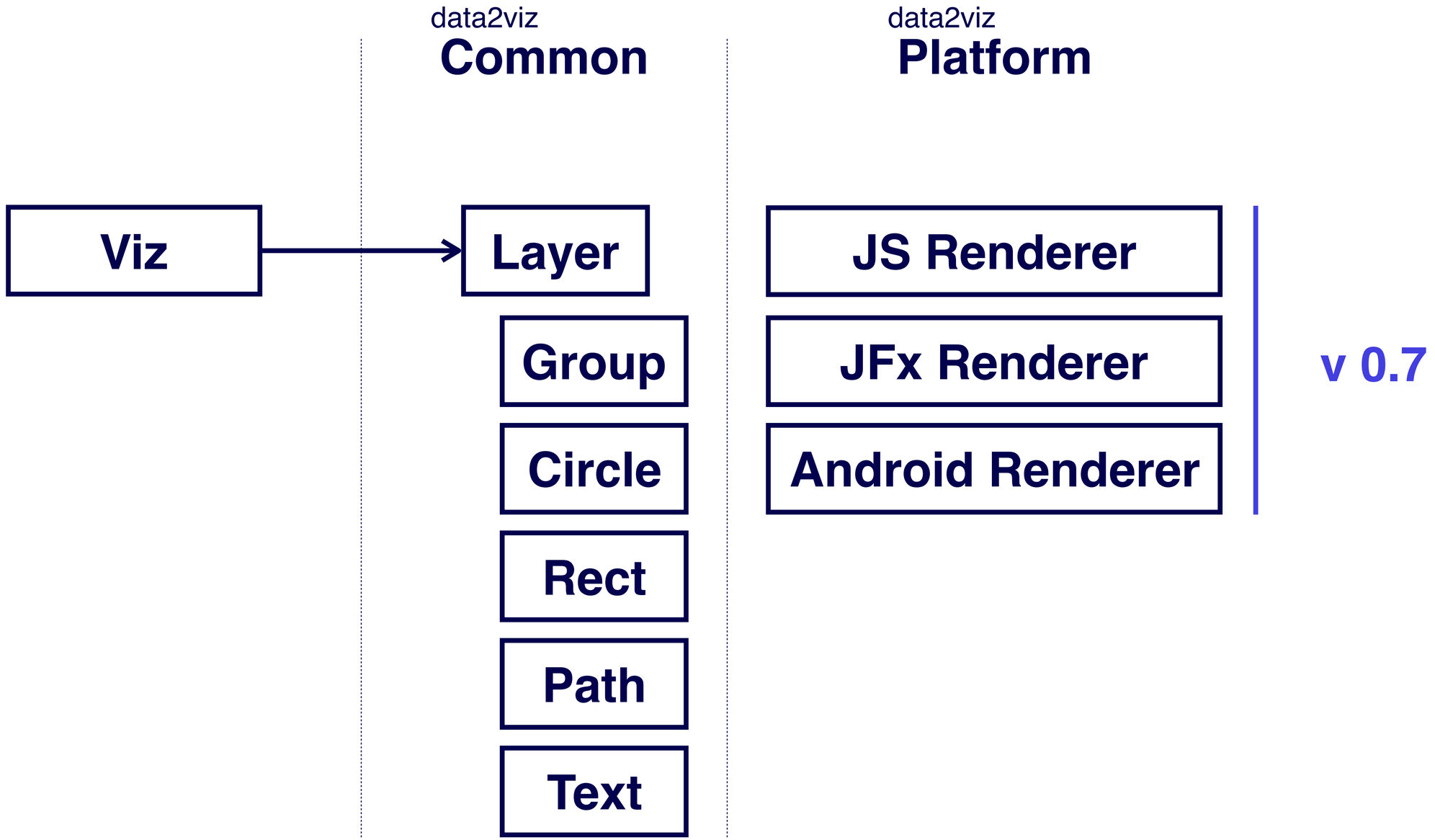
Architecture



Architecture



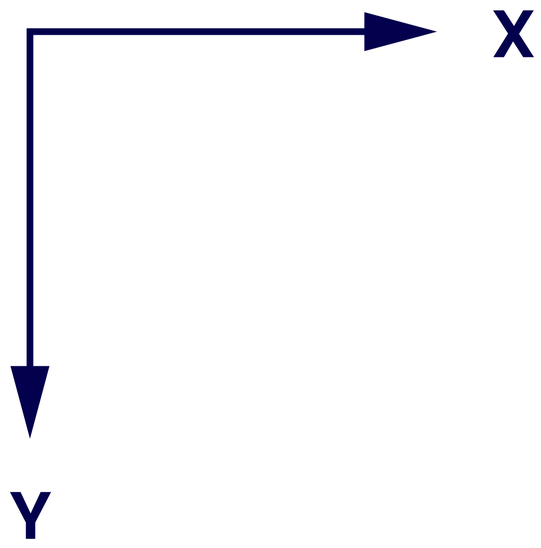
Architecture



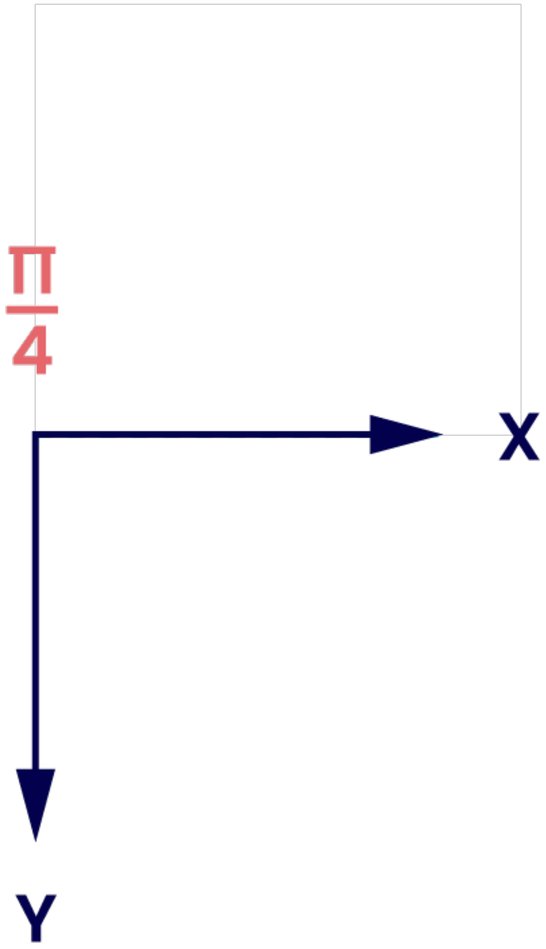
Multiplatform Development

Aligning platform APIs

Aligning platform APIs



Aligning platform APIs

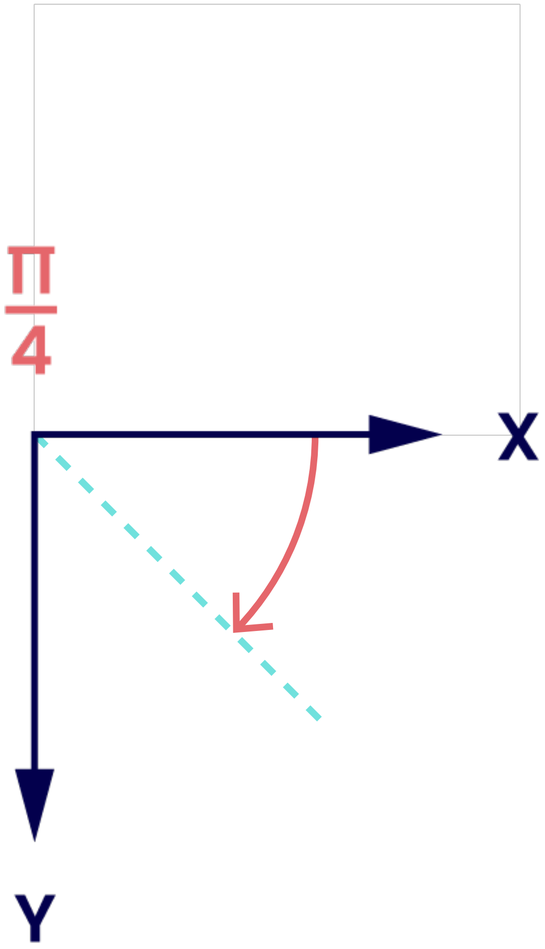


JS

radian

clockwise

Aligning platform APIs

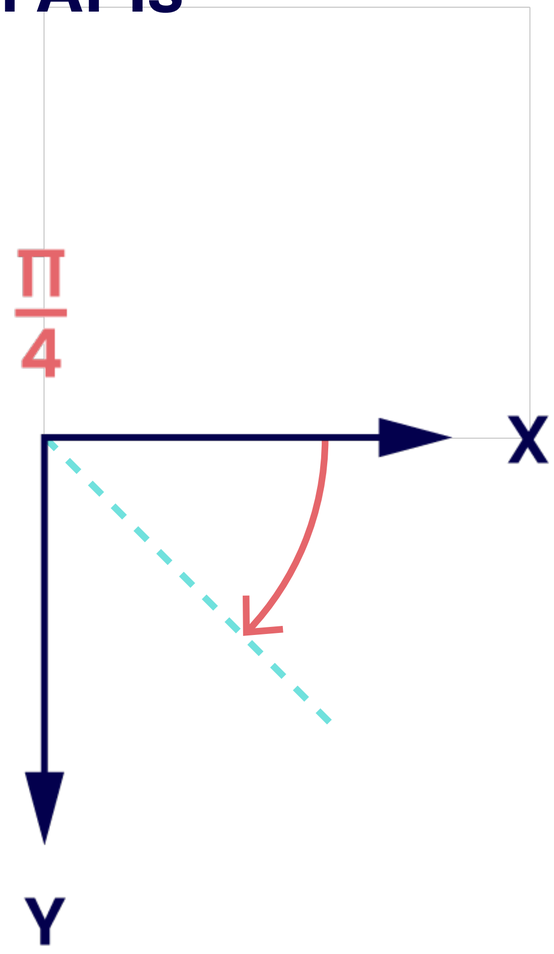


JS

radian

clockwise

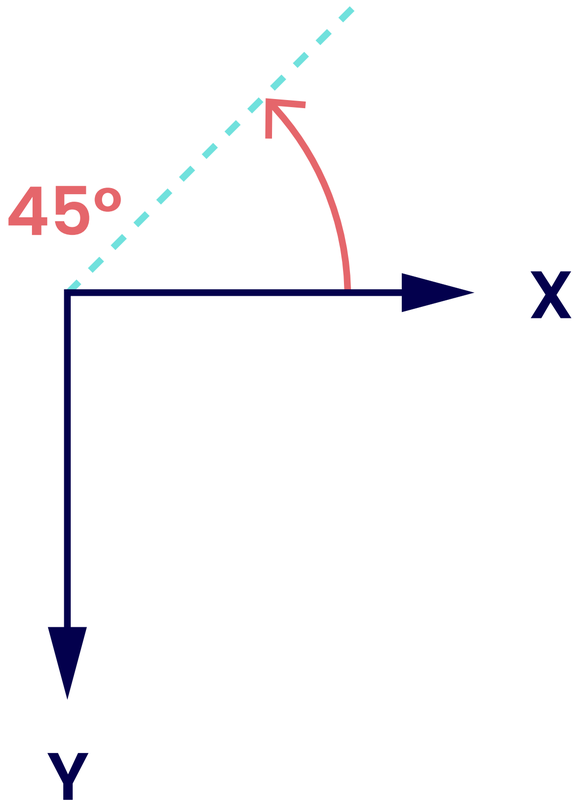
Aligning platform APIs



JS

radian

clockwise

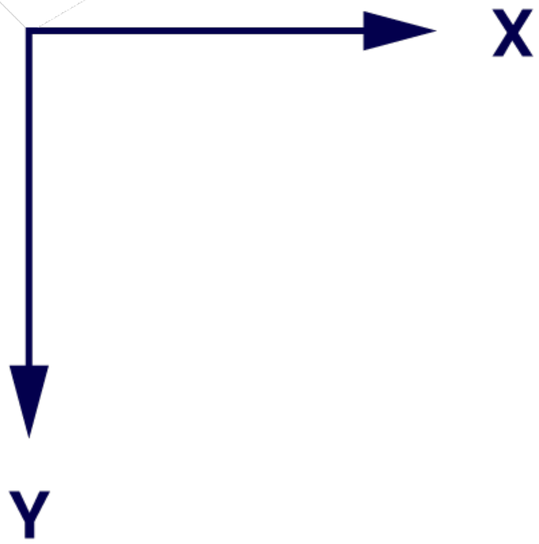


JFx

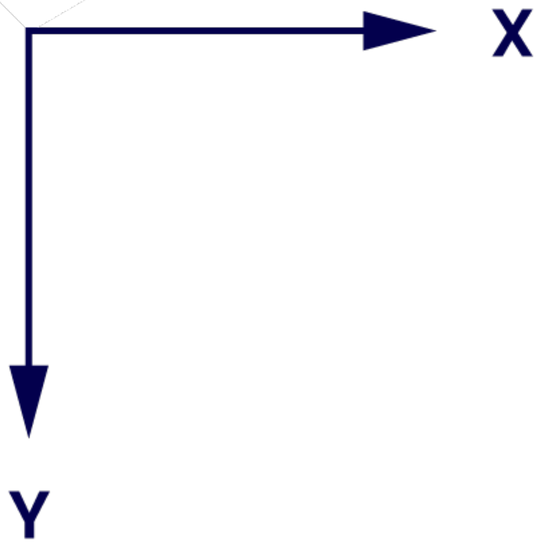
degrees

counterclockwise

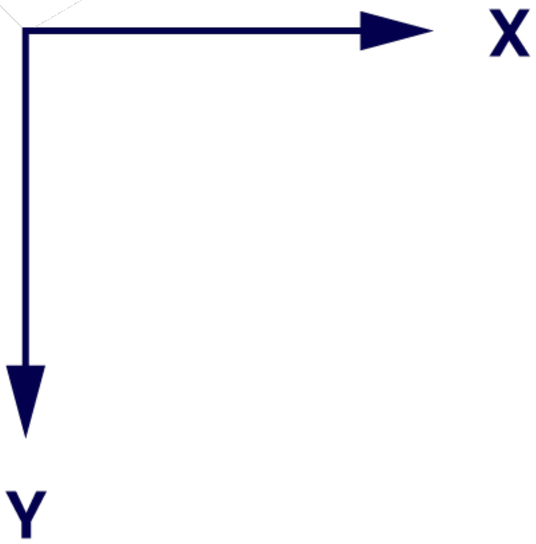
Aligning platform APIs



JS

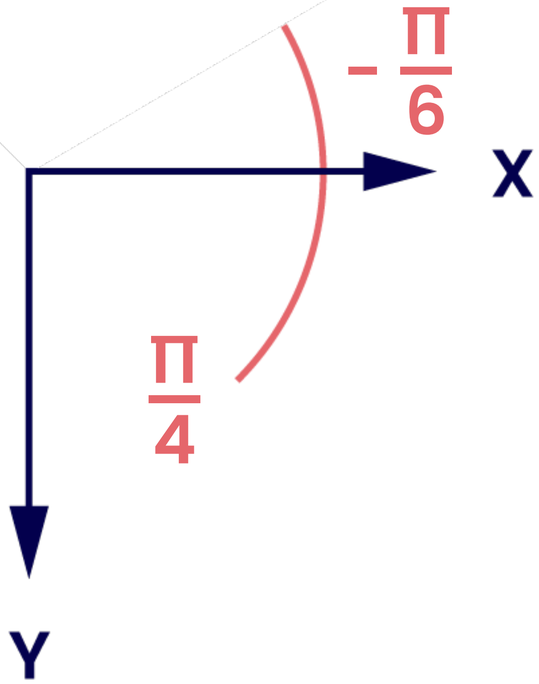


JFx

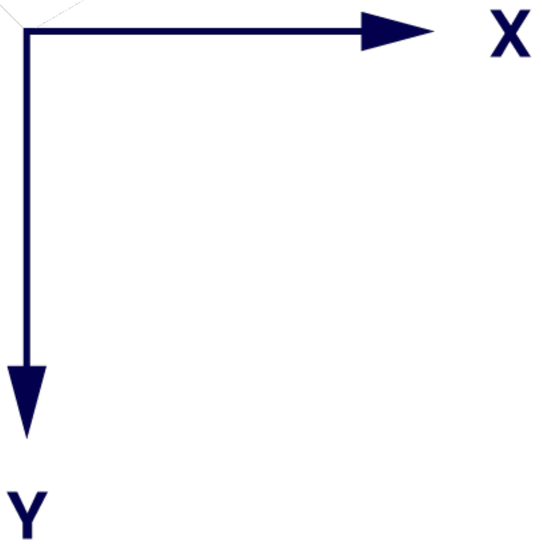


Android

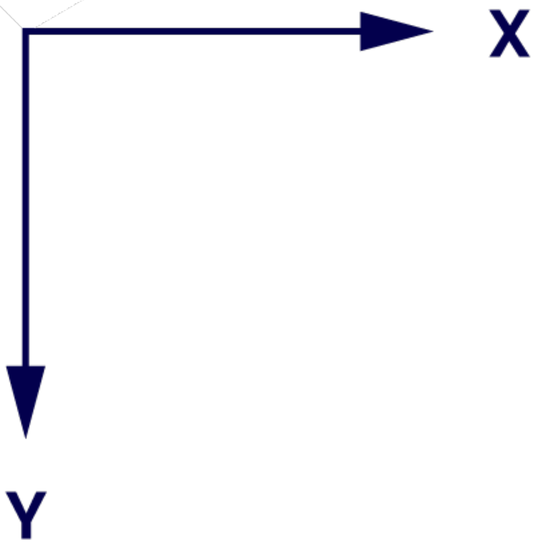
Aligning platform APIs



JS



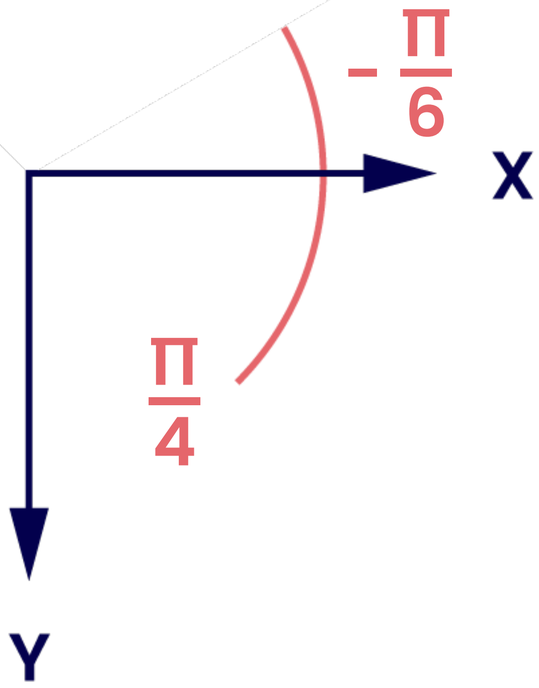
JFx



Android

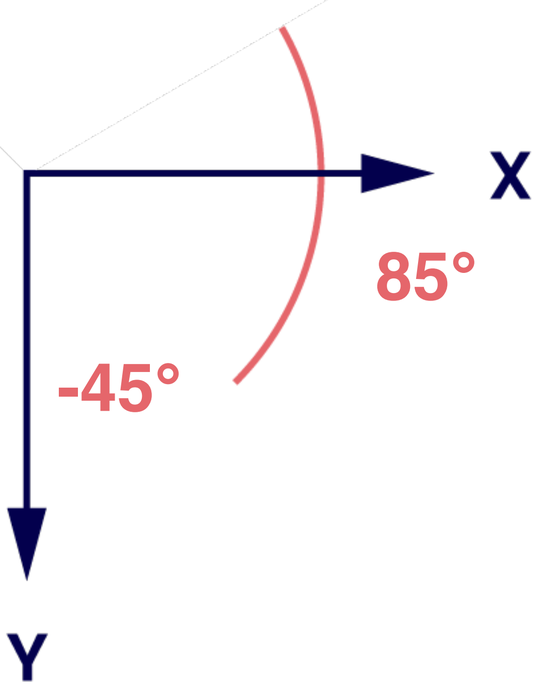
```
context.arc(
  x,
  y,
  r,
  startAngle,
  endAngle,
  counterclockwise);
```

Aligning platform APIs



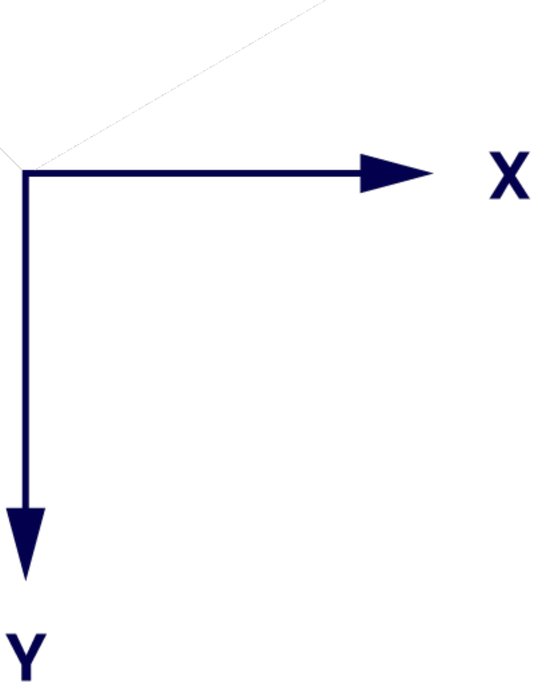
JS

```
context.arc(
  x,
  y,
  r,
  startAngle,
  endAngle,
  counterclockwise);
```



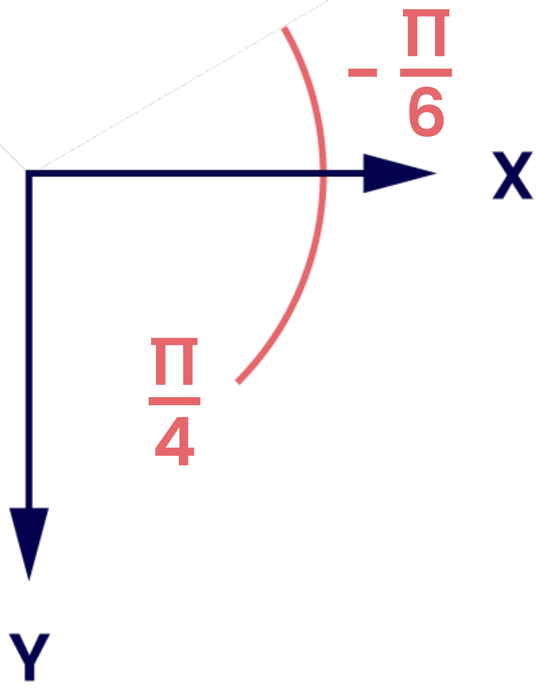
JFx

```
gc.arc(
  x,
  y,
  rx,
  ry,
  startAngle,
  sweepAngle)
```



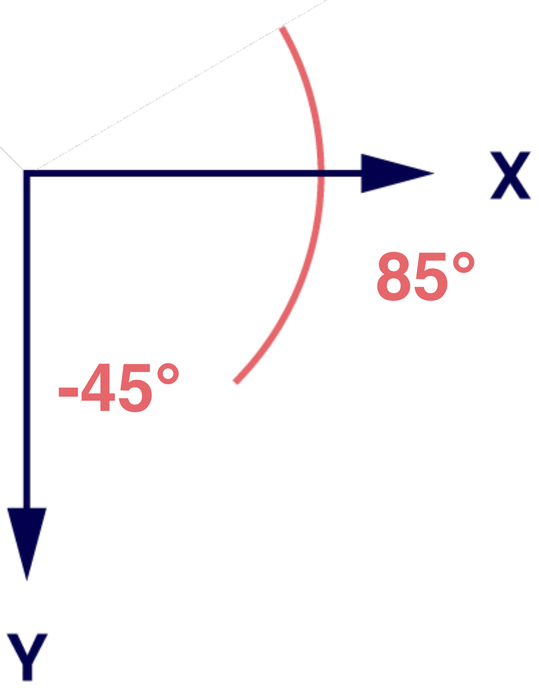
Android

Aligning platform APIs



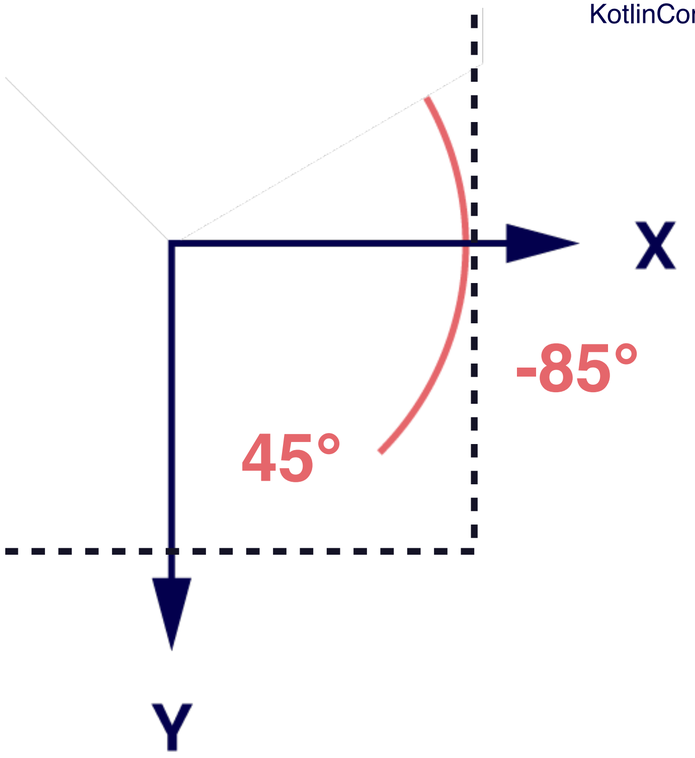
JS

```
context.arc(
  x,
  y,
  r,
  startAngle,
  endAngle,
  counterclockwise);
```



JFx

```
gc.arc(
  x,
  y,
  rx,
  ry,
  startAngle,
  sweepAngle)
```



Android

```
gc.arcTo(
  rectF,
  startAngle,
  sweepAngle)
```

Aligning platform APIs


```
fun arcTo(lastX: Double, lastY: Double, cpX: Double, cpY:Double, x: Double, y: Double, r: Double) {

    with(renderer) {

        val alpha1 = atan2(lastY - cpY, lastX - cpX)
        val alpha2 = atan2(y - cpY, x - cpX)
        val alpha = angle(alpha1 - alpha2)
        val d = r / sin(alpha / 2).absoluteValue
        val cx = cpX + d * cos(alpha1 - alpha / 2)
        val cy = cpY + d * sin(alpha1 - alpha / 2)

        val clockwise = alpha > 0
        val startAngle = alpha1.radToDeg.toFloat() + clockwise.toSign() * 90f
        val sweepAngle =
                if (clockwise)
                    360 - ((180f + alpha.radToDeg.toFloat()) % 360f)
                else
```

Multiplatform Development

Testing

Multiplatform tests

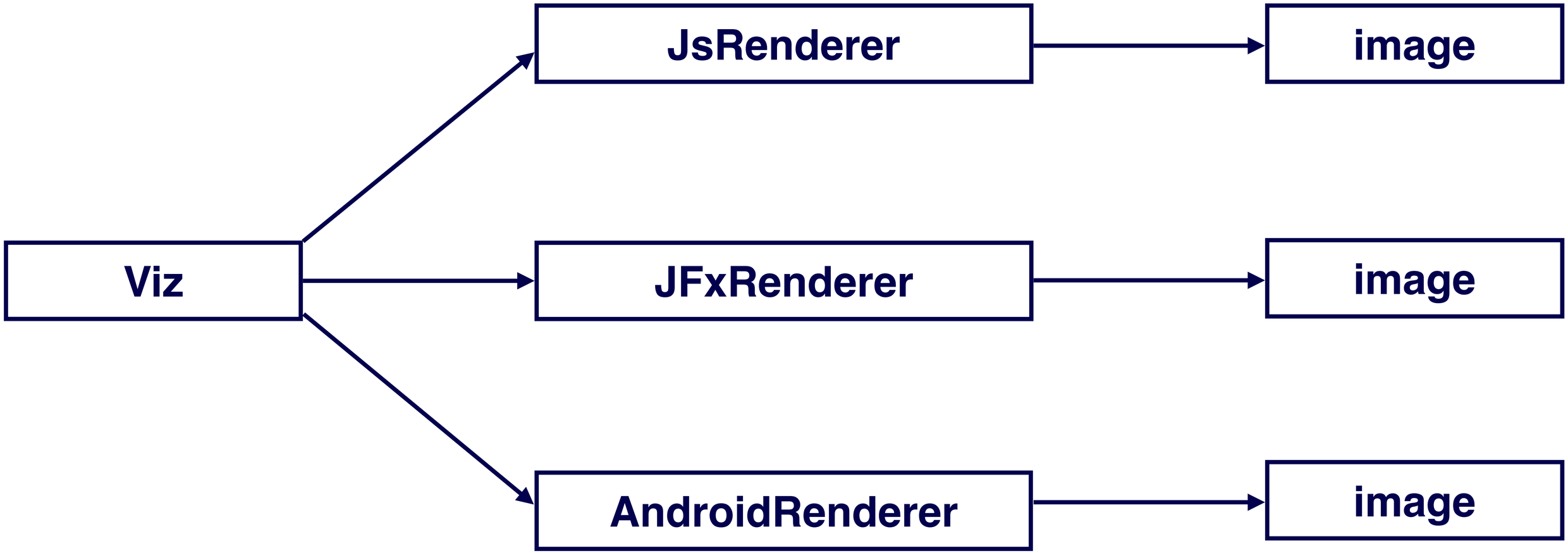
– common code

Multiplatform tests

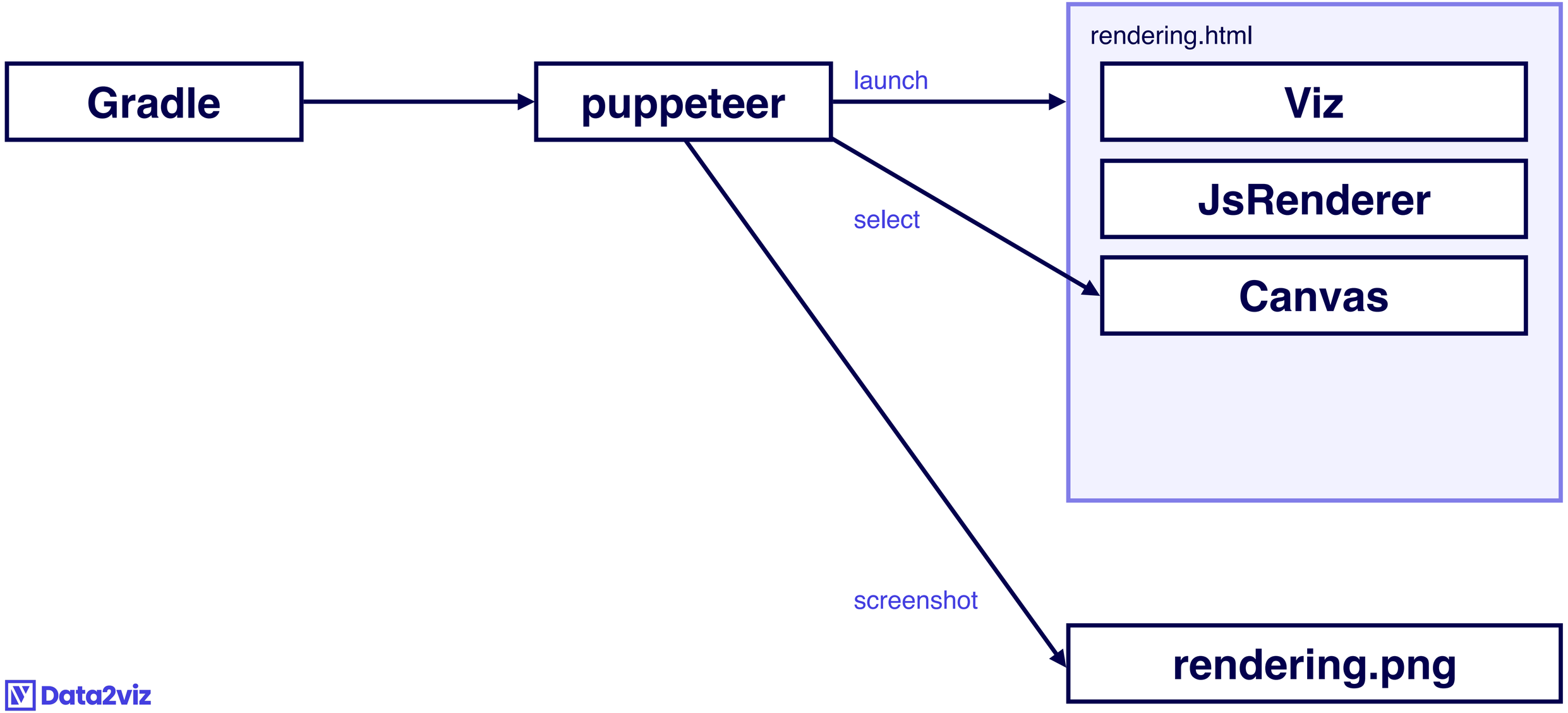
– common code

– specific platform code

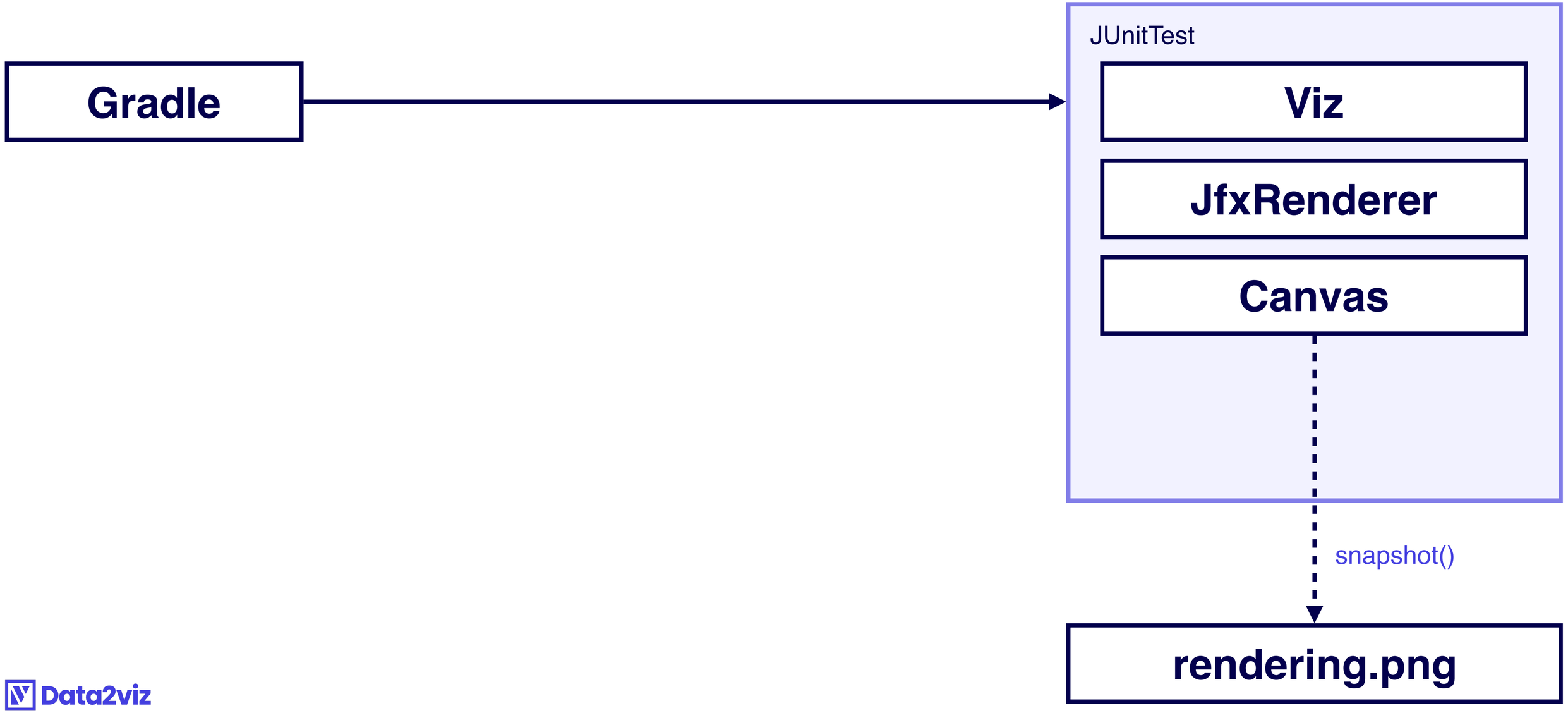
Multiplatform tests



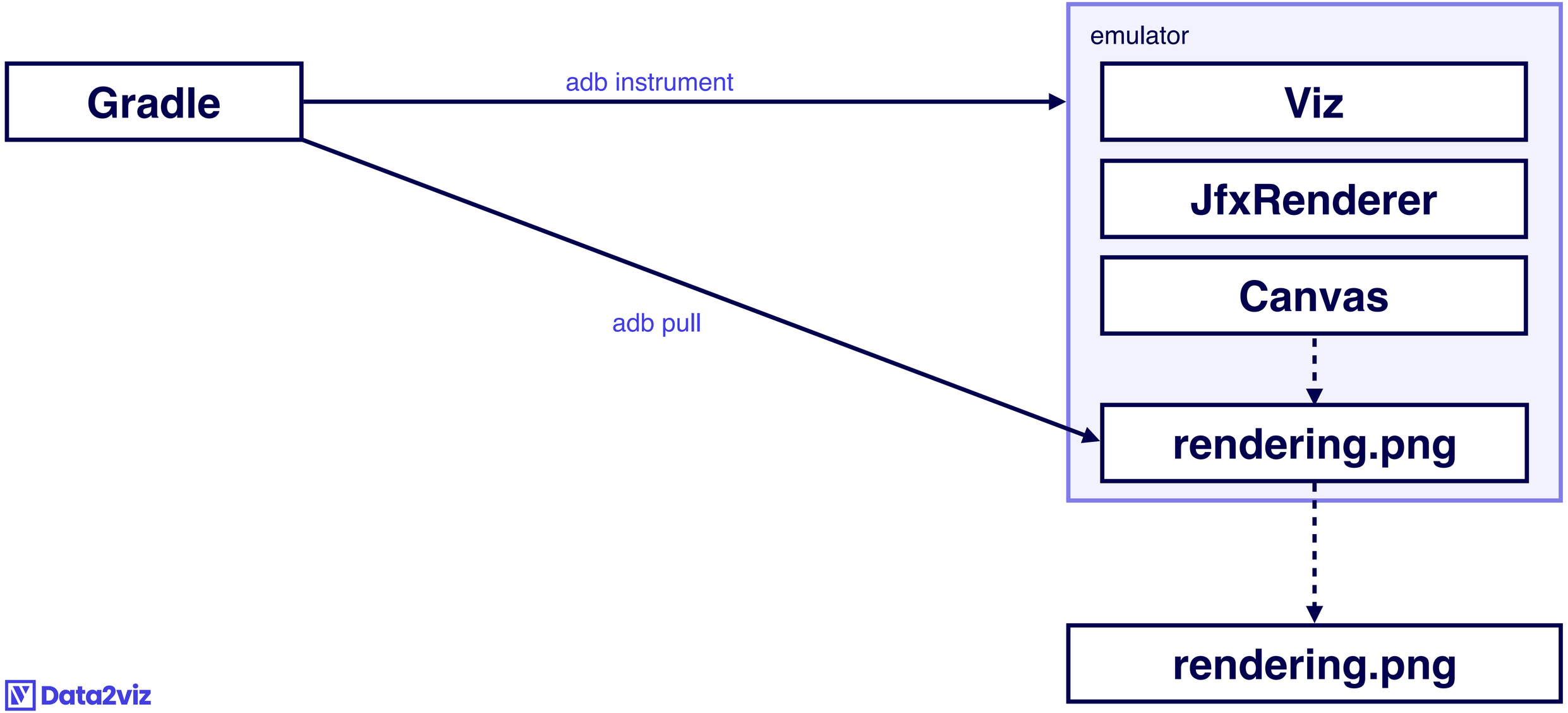
Multiplatform tests JS



Multiplatform tests JFx

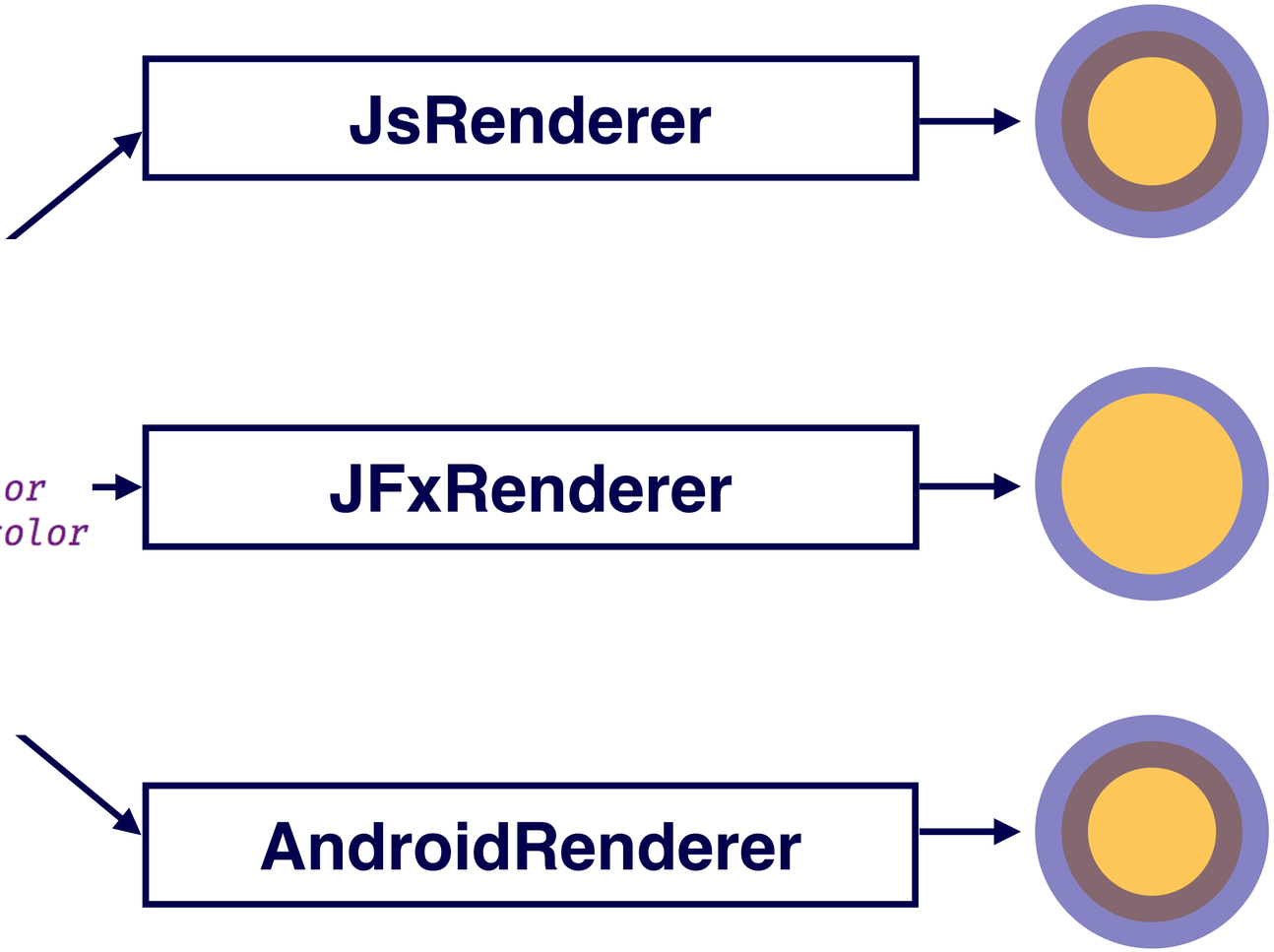


Multiplatform tests Android



Multiplatform tests


```
renderingTest("circle4") {
    circle {
        x = 200.0
        y = 200.0
        radius = 100.0
        style.fill = 0×fdc658.color
        style.stroke = 0×0c0887.color
                .withAlpha(.5f)
        style.strokeWidth = 40.0
    }
},
```



Multiplatform tests


```
io.data2viz.viz.RenderingJfxTest > renderImages PASSED
yarn run v1.9.4
warning package.json: No license field
$ node diff.js
Diffs pixels ref/jfx for arc1-positive-clockwise::0
Diffs pixels ref/jfx for circle1::0
Diffs pixels ref/jfx for circle2::0
Diffs pixels ref/jfx for circle3::3
Diffs pixels ref/jfx for circle4::10643
Diffs pixels ref/jfx for path.rect::0
Diffs pixels ref/jfx for path1::106
Diffs pixels ref/jfx for transform::0
Diffs pixels ref/jfx for visible1::0
Diffs pixels ref/jfx for visible2-layer::0
/Users/gaetan/dev/data2viz/viz/d2v-viz-jfx/diff.js:55
        throw new Error('To much pixel diff');
        ^

Error: To much pixel diff
    at checkPixelErrors (/Users/gaetan/dev/data2viz/viz/d2v-viz-jfx/diff.js:55:15)
```

Current status

Current status

Some statistics

- 14k lines of code
- 12k lines of tests
- more than 95% of common code

Current status

data2viz V 0.7

- android support
- new design with renderers for JS and JFx
- 90% of D3 features now available

Roadmap

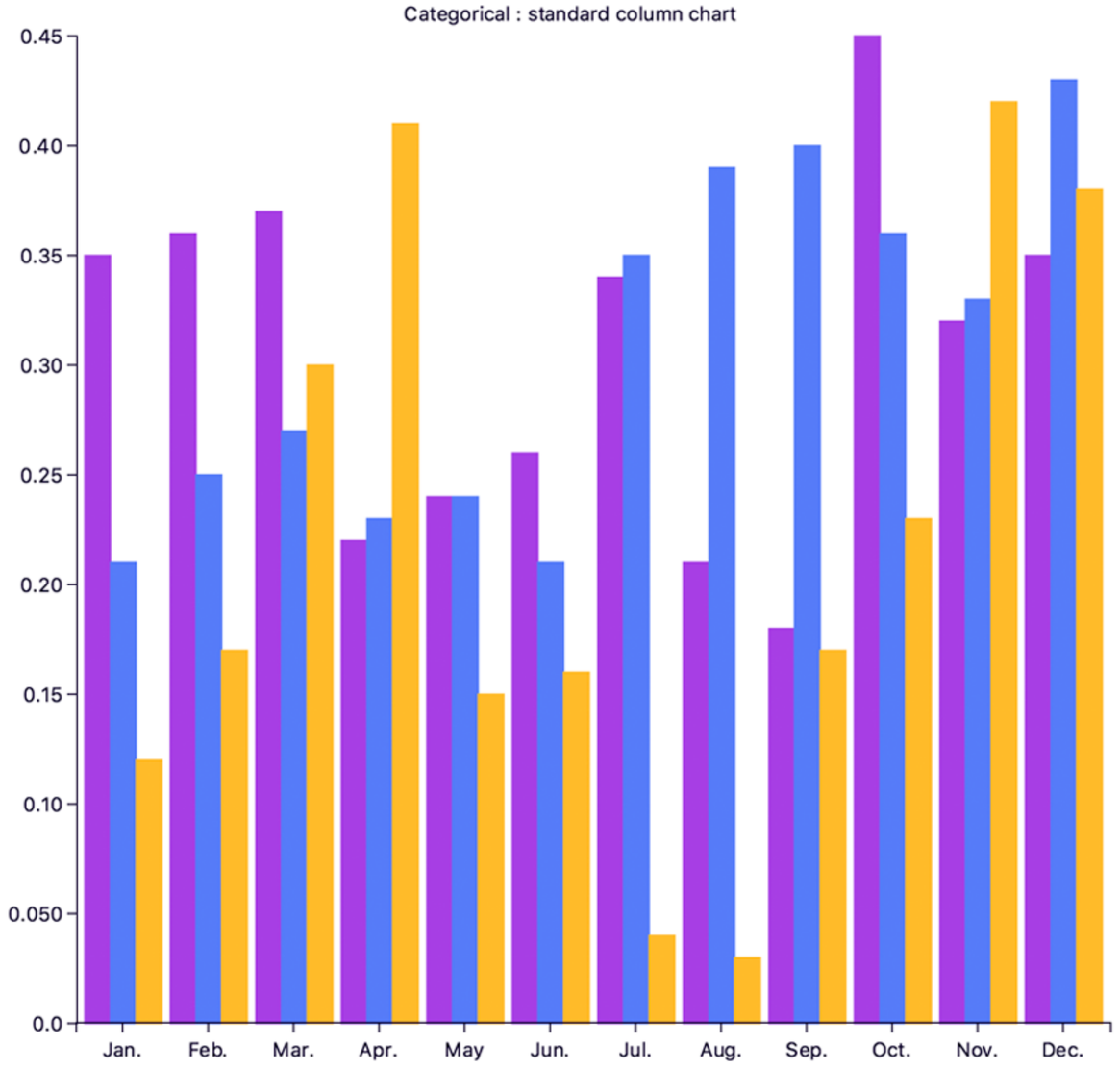
Version v0.8

- events API
- more text features (fonts, multiline, …)
- tutorials with online editable examples

Idea Plugins

charts.kt

charts.kt

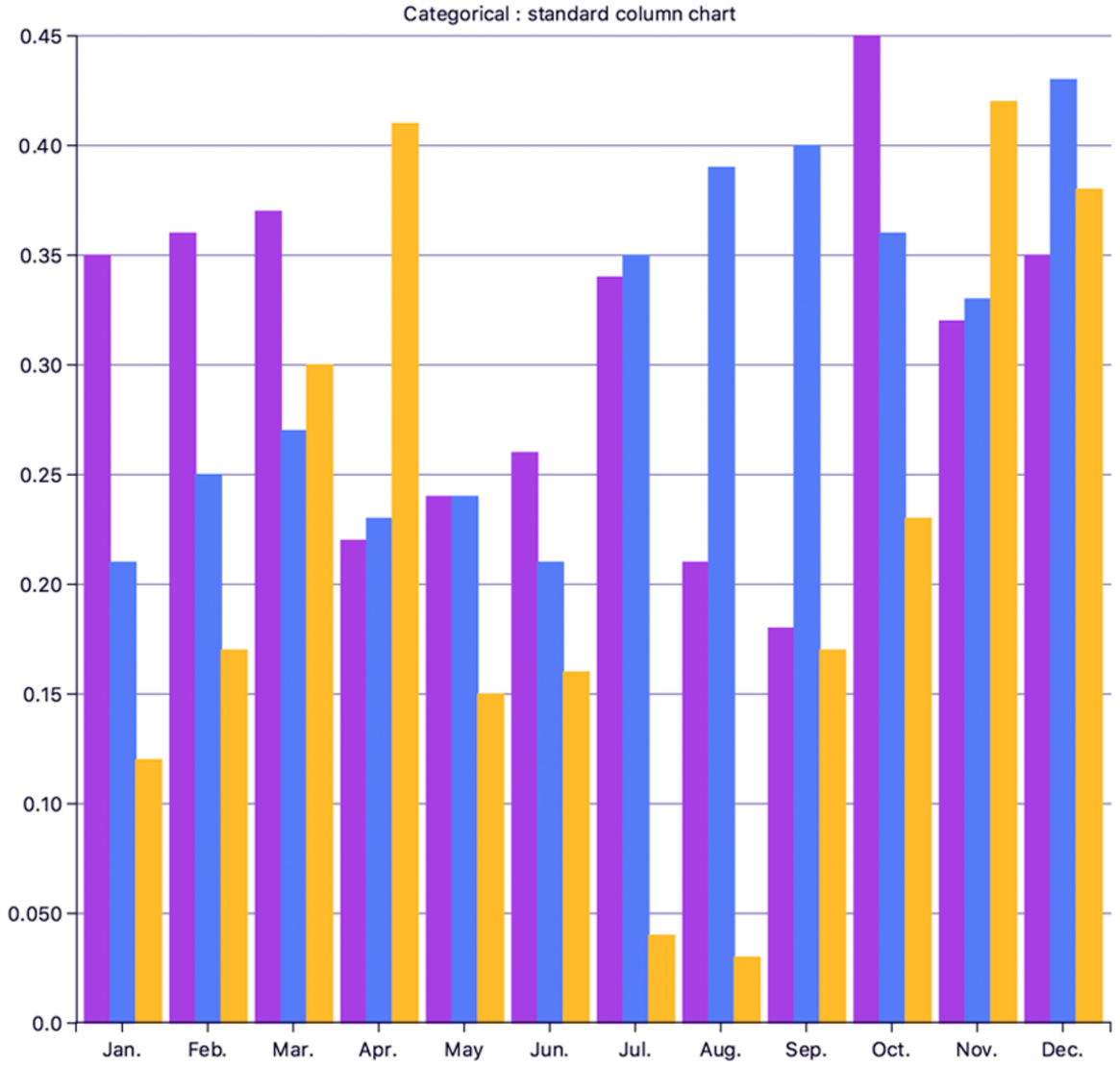



```
viz {
    chartXY<Rainfall> {
        title {
            text = "Categorical: standard column chart"
        }

        xAxis {
            tickFormat = { it.datum.month }
        }

        seriesCategorical {
            data = rainfallByYears
            columnSeries { yAccessor = { it.year(2016) } }
            columnSeries { yAccessor = { it.year(2017) } }
            columnSeries { yAccessor = { it.year(2018) } }
        }
    }
}
```

charts.kt



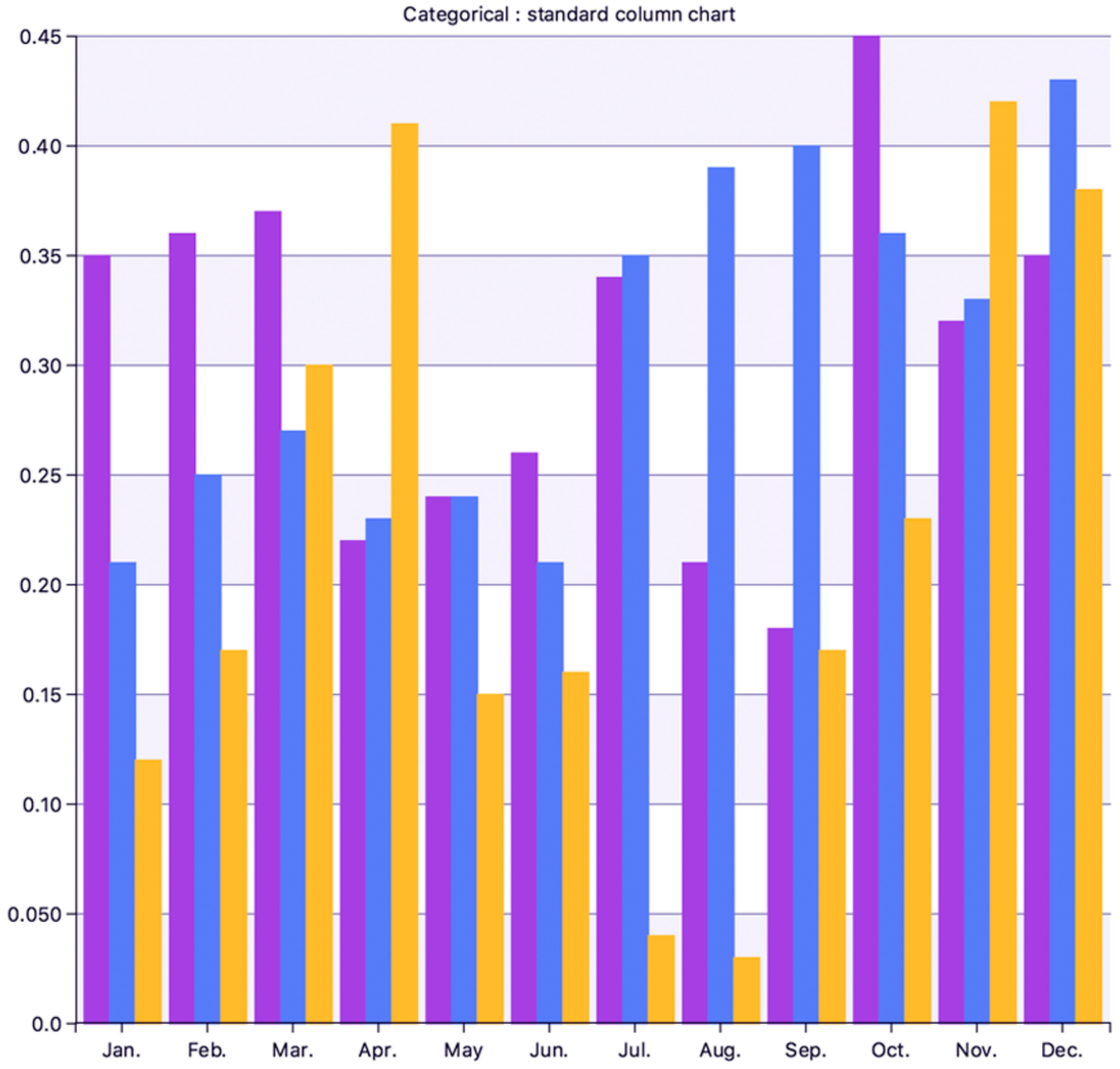

```
viz {
    chartXY<Rainfall> {
        title {
            text = "Categorical: standard column chart"
        }

        xAxis {
            tickFormat = { it.datum.month }
        }

        yAxis {
            gridLines = true
        }

        seriesCategorical {
            data = rainfallByYears
            columnSeries { yAccessor = { it.year(2016) } }
            columnSeries { yAccessor = { it.year(2017) } }
            columnSeries { yAccessor = { it.year(2018) } }
        }
    }
}
```

charts.kt



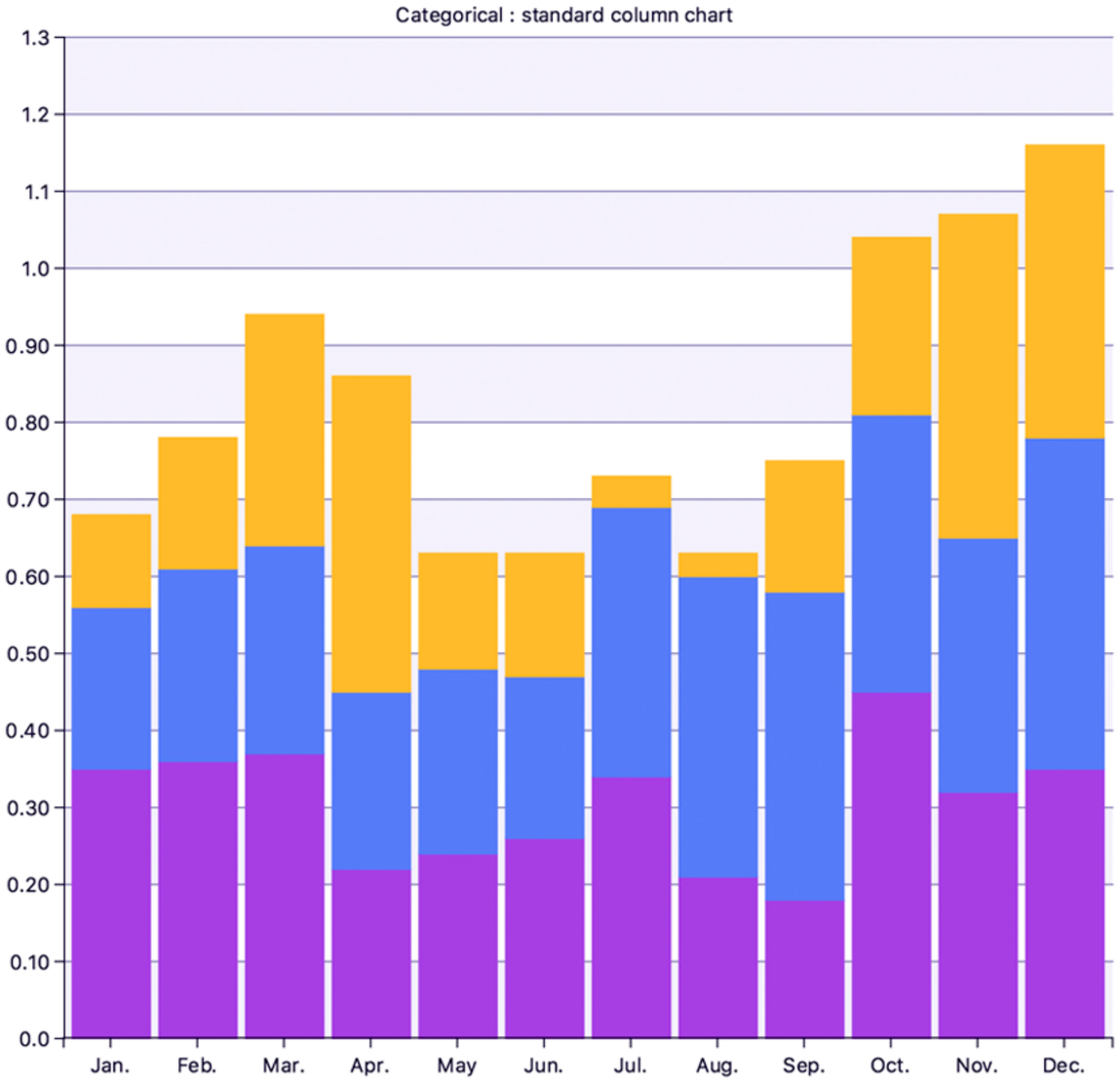

```
viz {
    chartXY<Rainfall> {
        title {
            text = "Categorical: standard column chart"
        }

        xAxis {
            tickFormat = { it.datum.month }
        }

        yAxis {
            gridLines = true
            alternateBackground = true
        }

        seriesCategorical {
            data = rainfallByYears
            columnSeries { yAccessor = { it.year(2016) } }
            columnSeries { yAccessor = { it.year(2017) } }
            columnSeries { yAccessor = { it.year(2018) } }
        }
    }
}
```

charts.kt



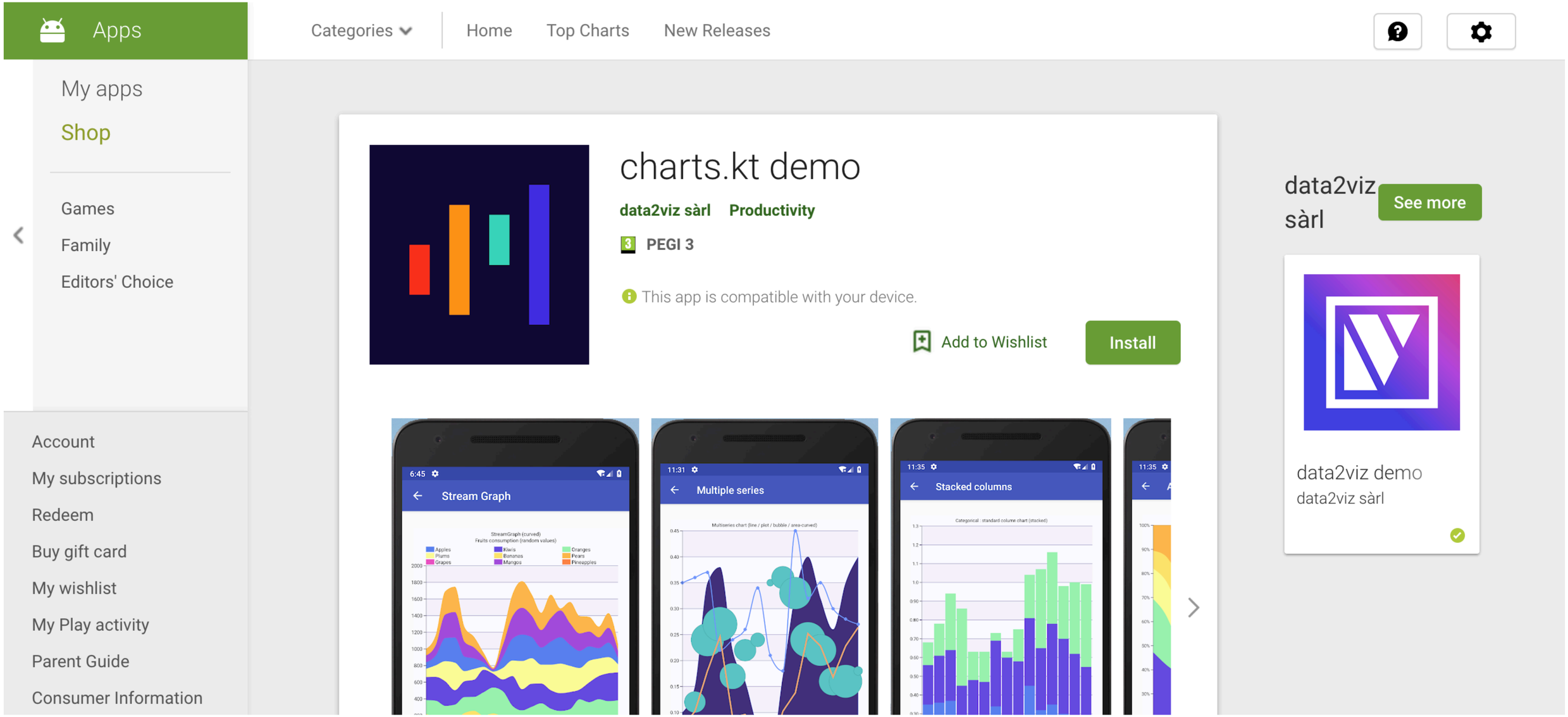

```
viz {
    chartXY<Rainfall> {
        title {
            text = "Categorical: standard column chart"
        }

        xAxis {
            tickFormat = { it.datum.month }
        }

        yAxis {
            gridLines = true
            alternateBackground = true
        }

        seriesCategorical {
            data = rainfallByYears
            stacking = Stacking.Normal
            columnSeries { yAccessor = { it.year(2016) } }
            columnSeries { yAccessor = { it.year(2017) } }
            columnSeries { yAccessor = { it.year(2018) } }
        }
    }
}
```

charts.kt - android demo



charts.kt - early access program

charts.kt

motivations KotlinConf offer road map GET IT ON Google Play

charts.kt

The multiplatform charting library

A powerful yet simple charting API for your Kotlin projects.

charts.kt - early access program

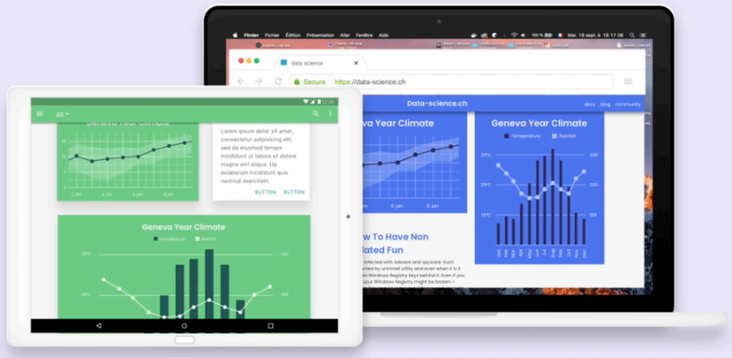
charts.kt

motivations KotlinConf offer road map

GET IT ON Google Play

Customize your charts

charts.kt offers several themes for your needs (light, dark, glow…) but the whole result is totally and easily customisable.



Be productive

Based on Kotlin, **charts.**kt is statically typed so your code editor can helps with completion, allowing you to create your charts in a few minutes.

```
chartXY<TemperatureReading> {
  title {
    text = "Geneva temperatures since 1960"
  }
  xAxis {
    gridLines = true
  }
  seriesContinuous {
    timeLineSeries {
      yAc
        yAccessor          (TemperatureReading) → Double
        yAccessorBaseline  ((TemperatureReading) → …
^↓ and ^↑ will move caret down and up in the editor >>
}
```

Thank you!

Gaëtan Zoritchak
@gz_k

Pierre Mariac
@imtam5

data2viz
@data2viz_io

Github
data2viz/data2viz

Data2viz

data2viz.io

charts.kt

charts-kt.io