realkinetic.com | @real_kinetic

REALKINETIC

YOUR POTENTIAL REALIZED

```
const PointTag = Symbol('Point')
const Point = (x, y) => {
   if (typeof x !== 'number') throw TypeError('x must be a Number')
   if (typeof y !== 'number') throw TypeError('y must be a Number')
   return { x, y, tag: PointTag }
}

const CircleTag = Symbol('Circle')
const RectangleTag = Symbol('Rectangle')
const Shape = {
   Circle: (center, radius) => {
      if (center.tag !== PointTag) throw TypeError('center must be a Point')
      if (typeof radius !== 'number') throw TypeError('radius must be a Number')
      return { center, radius, tag: CircleTag }
   },
   Rectangle: (corner1, corner2) => {
      if (corner1.tag !== PointTag) throw TypeError('corner1 must be a Point')
      if (corner2.tag !== PointTag) throw TypeError('corner2 must be a Point')
      return { corner1, corner2, tag: RectangleTag }
   }
}
```

```
const circ1 = Shape.Circle(Point(2, 3), 6.5)
const circ2 = Shape.Circle(Point(5, 1), 3)

const rect1 = Shape.Rectangle(Point(1.5, 9), Point(7, 7))
const rect2 = Shape.Rectangle(Point(0,   3), Point(3, 0))

console.log('Is circ1 a circle?', circ1.tag === CircleTag) // true
console.log('Is circ2 a circle?', circ2.tag === CircleTag) // true
console.log('Is rect1 a rectangle?', rect1.tag === RectangleTag) // true
console.log('Is rect2 a rectangle?', rect2.tag === RectangleTag) // true

const rect3 = Shape.Rectangle(Point(1, 2), 9) // ERROR: corner2 must be a Point
```

```
const PointTag = Symbol('Point')
const Point = (x, y) => {
   if (typeof x !== 'number') throw TypeError('x must be a Number')
   if (typeof y !== 'number') throw TypeError('y must be a Number')
   return { x, y, tag: PointTag }
}

const CircleTag = Symbol('Circle')
const RectangleTag = Symbol('Rectangle')
const Shape = {
   Circle: (center, radius) => {
      if (center.tag !== PointTag) throw TypeError('center must be a Point')
      if (typeof radius !== 'number') throw TypeError('radius must be a Number')
      return { center, radius, tag: CircleTag }
   },
   Rectangle: (corner1, corner2) => {
      if (corner1.tag !== PointTag) throw TypeError('corner1 must be a Point')
      if (corner2.tag !== PointTag) throw TypeError('corner2 must be a Point')
      return { corner1, corner2, tag: RectangleTag }
   }
}
```