```
chart = {
  // Display Options
  let width = 930;
  let height = width;
  let bold = true;
  let black = false;
  let shadow = true;
  let multicolor = true;
  let hexcolor = "#0099cc";

  const format = d3.format(",d")

  const pack = data => d3.pack()
    .size([width, height])
    .padding(3)
    (d3.hierarchy(data)
     .sum(d => d.size)
     .sort((a, b) => b.value - a.value))

  const root = pack(data);
  let focus = root;
  let view;

  let fontsize = d3.scaleOrdinal()
    .domain([1,3])
    .range([24,16])

  function setColorScheme(multi){
    if (multi) {
      let color = d3.scaleOrdinal()
        .range(d3.schemeCategory10)
      return color;
    }
  }

  let color = setColorScheme(multicolor);

  function setCircleColor(obj) {
    let depth = obj.depth;
    while (obj.depth > 1) {
      obj = obj.parent;
    }
    let newcolor = multicolor ? d3.hsl(color(obj.data.name)) : d3.hsl(hexcolor);
    newcolor.l += depth == 1 ? 0 : depth * .1;
    return newcolor;
  }
```

```
 function setStrokeColor(obj) {
   let depth = obj.depth;
   while (obj.depth > 1) {
     obj = obj.parent;
   }
   let strokecolor = multicolor ? d3.hsl(color(obj.data.name)) : d3.hsl(hexcolor);
   return strokecolor;
 }

 const svg = d3.select(DOM.svg(width, height))
     .attr("viewBox", `-${width / 2} -${height / 2} ${width} ${height}`)
     .style("display", "block")
     .style("margin", "0 -14px")
     .style("width", "calc(100% + 28px)")
     .style("height", "auto")
     .style("background", "white")
     .style("cursor", "pointer")
     .on("click", () => zoom(root));

 const node = svg.append("g")
   .selectAll("circle")
   .data(root.descendants().slice(1))
   .enter().append("circle")
     .attr("fill", setCircleColor)
     .attr("stroke", setStrokeColor)
     .attr("pointer-events", d => !d.children ? "none" : null)
     .on("mouseover", function() { d3.select(this).attr("stroke", d => d.depth == 1 ? "black" :
"white"); })
     .on("mouseout", function() { d3.select(this).attr("stroke", setStrokeColor); })
     .on("click", d => focus !== d && (zoom(d), d3.event.stopPropagation()));

 const label = svg.append("g")
     .style("fill", function() {
       return black ? "black" : "white";
     })
     .style("text-shadow", function(){
       if (shadow) {
         return black ? "2px 2px 0px white" : "2px 2px 0px black";
       } else {
         return "none";
       }
     })
     .attr("pointer-events", "none")
     .attr("text-anchor", "middle")
   .selectAll("text")
   .data(root.descendants())
   .enter().append("text")
```

```
      .style("fill-opacity", d => d.parent === root ? 1 : 0)
      .style("display", d => d.parent === root ? "inline" : "none")
      .style("font", d => fontsize(d.depth) + "px sans-serif")
      .style("font-weight", function() {
        return bold ? "bold" : "normal";
      })
      .text(d => d.data.name);

  zoomTo([root.x, root.y, root.r * 2]);

  function zoomTo(v) {
    const k = width / v[2];

    view = v;

    label.attr("transform", d => `translate(${(d.x - v[0]) * k},${(d.y - v[1]) * k +
fontsize(d.depth)/4})`);
    node.attr("transform", d => `translate(${(d.x - v[0]) * k},${(d.y - v[1]) * k})`);
    node.attr("r", d => d.r * k);
  }

  function zoom(d) {
    const focus0 = focus;

    focus = d;

    const transition = svg.transition()
        .duration(d3.event.altKey ? 7500 : 750)
        .tween("zoom", d => {
          const i = d3.interpolateZoom(view, [focus.x, focus.y, focus.r * 2]);
          return t => zoomTo(i(t));
        });

    label
      .filter(function(d) { return d.parent === focus || this.style.display === "inline"; })
      .transition(transition)
        .style("fill-opacity", d => d.parent === focus ? 1 : 0)
        .on("start", function(d) { if (d.parent === focus) this.style.display = "inline"; })
        .on("end", function(d) { if (d.parent !== focus) this.style.display = "none"; });
  }

  return svg.node();
}
```