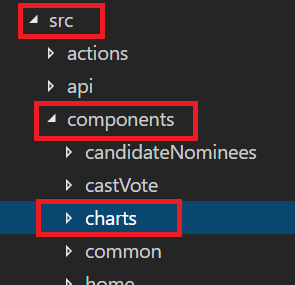
**D3 Pie Chart implementation steps**

1. Install the following D3 libraries d3, d3-array,d3-scale, d3-shape
   1. npm install d3
   2. npm install d3-scale
   3. npm install d3-array
   4. npm install d3-shape
2. Create a folder **charts** under **src/components**



1. Create 2 JavaScript files under **charts** folder
   1. **PieChart.js**
   2. **BarChart.js**
2. Add the following styles under styles.css

.myPieGraph .toolTip {

font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;

position: absolute;

display: none;

width: auto;

height: auto;

background: none repeat scroll 0 0 white;

border: 0 none;

border-radius: 8px 8px 8px 8px;

box-shadow: -3px 3px 15px #888888;

color: black;

font: 12px sans-serif;

padding: 5px;

text-align: center;

}

.myPieGraph text {

font: 28px sans-serif;

}

.myPieGraph text {

font-family: sans-serif;

font-size: 28px;

}

.myPieGraph .textForIE11 {

font-family: sans-serif;

font-size: 15px;

}

.myPieGraph .graphIE {

width : 960;

height: 200;

}

1. Add the following code in PieChart.js

import React, { Component } from 'react';

import PropTypes from 'prop-types';

import { scaleLinear, scaleOrdinal } from 'd3-scale';

import { max } from 'd3-array';

import { schemeCategory10, select, pie, arc, event } from 'd3';

class PieChart extends Component {

constructor(props, context) {

super(props, context);

this.createBarChart = this.createBarChart.bind(this);

}

componentDidMount() {

this.createBarChart();

}

componentDidUpdate() {

this.createBarChart();

}

createBarChart() {

let sales = this.props.data;

let pieHISD = pie()

.value(function (d) { return d.keyValue; });

let slices = pieHISD(sales);

let arcHISD = null;

let arcOver = null;

arcHISD = arc()

.innerRadius(0)

.outerRadius(100);

arcOver = arc()

.innerRadius(0)

.outerRadius(100 + 5);

let color = scaleOrdinal(["red","blue","green"]);

const node = this.node;

let div = select(node).append("div").attr("class", "toolTip");

let isIE11 = !!window.MSInputMethodContext && !!document.documentMode;

let g = null;

let svg = select(node)

.append('svg');

if (isIE11) {

arcHISD = arc()

.innerRadius(0)

.outerRadius(90);

arcOver = arc()

.innerRadius(0)

.outerRadius(90 + 10);

select(node).select('svg')

.attr("width", "100%")

.attr("height", "250px");

g = svg.append('g')

.attr('transform', 'translate(200,100)');

}

else {

arcHISD = arc()

.innerRadius(0)

.outerRadius(180);

arcOver = arc()

.innerRadius(0)

.outerRadius(180 + 10);

select(node).select('svg')

.attr("viewBox", "0 0 960 500")

.attr("preserveAspectRatio", "xMidYMid meet");

g = svg.append('g')

.attr('transform', 'translate(400,200)');

}

let arcGraph = g.selectAll('path.slice')

.data(slices)

.enter();

arcGraph.append('path')

.attr('class', 'slice')

.attr('d', arcHISD)

.attr('fill', function (d) {

return color(d.data.key);

})

.on("mouseover", function (d) {

div.style("left", 100 + "px");

div.style("top", 10 + "px");

div.style("display", "inline-block");

div.html((d.data.key) + "<br>" + (d.value) + "%");

select(this).transition()

.duration(1000)

.attr("d", arcOver);

})

.on("mouseout", function (d) {

div.style("display", "none");

select(this).transition()

.duration(1000)

.attr("d", arcHISD);

});

if (isIE11) {

arcGraph.append("text")

.attr("transform", function (d) { return "translate(" + arcHISD.centroid(d) + ")"; })

.attr("class", "textForIE11")

.attr("dy", "0.35em")

.text(function (d) { return d.data.keyValue; });

svg.append('g')

.attr('class', 'legend')

.selectAll('text')

.data(slices)

.enter()

.append('text')

.text(function (d) { return '• ' + d.data.key; })

.attr("class", "textForIE11")

.attr('fill', function (d) { return color(d.data.key); })

.attr('y', function (d, i) { return 30 \* (i + 1); });

}

else {

arcGraph.append("text")

.attr("transform", function (d) { return "translate(" + arcHISD.centroid(d) + ")"; })

.attr("dy", "0.35em")

.text(function (d) { return d.data.keyValue; });

svg.append('g')

.attr('class', 'legend')

.selectAll('text')

.data(slices)

.enter()

.append('text')

.text(function (d) { return '• ' + d.data.key; })

.attr('fill', function (d) { return color(d.data.key); })

.attr('y', function (d, i) { return 30 \* (i + 1); });

}

}

render() {

return (<div>

<div ref={node => this.node = node} className="myPieGraph"></div>

</div>

);

}

}

PieChart.propTypes = {

data: PropTypes.array

};

export default PieChart;

1. Add the following PieChart control in your code where you want to display the pie chart. This control should have an attribute **data** which accepts a json in the below following format.

<PieChart data={[

{ key: 'High School', keyValue: 20 },

{ key: 'Middle School', keyValue: 35 },

{ key: 'Elementory School', keyValue: 45 }

]} />

1. Will get the below output .

