```
// creates all the svg elements necessary for the pie chart
// representing the types of files in the current root folder
function drawPieChart(svg){

    // white background
    var rect = svg.select(".boundingRect").attr("fill", "white");

    // width and height of pie SVG
    pieSVGSize = pieWindowSize;

    // translation function used for pie slices
    transPie = "translate("+(pieSVGSize*.25)+","+(pieSVGSize*.4)+")";

    // arc used for pie slices
    arc = d3.svg.arc()
        .innerRadius(0)
        .outerRadius(pieSVGSize*.2);

    // g element for pie stuff
    gPie = svg.append("g").attr("id", "gPieChart");
    gPieSlices = svg.append("g").attr("id", "gPieSlices");

    // Pie Chart Title
    pieTitle = gPie.append("text")
        .attr("x", pieSVGSize/2)
        .attr("y", "10%")
        .attr("id", "pieTitleText")
        .attr("text-anchor", "middle");

    // Pie Chart Details
    defaultPieDetails = ["Mouse over", "pie slice", "for details..."];
    pieDetails = gPie.append("text")
        .attr("x", pieSVGSize*.04)
        .attr("y", pieSVGSize*.75)
        .attr("text-anchor", "left")
        .attr("id", "pieDetailsText");

    // Select boxes
    var buttonX = .5;
    var buttonY = .22;
    var buttonDY = .12;
    pieChartMeasure = createOptionButton("by % files","by % size", "pieChartMeasure", buttonX,
    buttonY);
    pieChartType = createOptionButton("by file type","by category", "pieChartType", buttonX,
    buttonY+buttonDY);
    pieChartDeep = createOptionButton("one level","all levels", "pieChartDeep", buttonX, buttonY
    +buttonDY*2);
    //pieChartSort = createOptionButton("sort value","asdf", "pieChartSort", buttonX,
    buttonY+buttonDY*3);

    refreshPieChart();
}
```

```
// creates a select box with options in the pie chart
function createOptionButton(alt1, alt2, id, xPercent, yPercent){
    var select = gPie.append("g")
        .attr("class", "svgSelect")
        .append("foreignObject")
            .attr("x", pieSVGSize*xPercent)
            .attr("y", pieSVGSize*yPercent)
            .attr("width", 200)
            .attr("height",70)
            .append("xhtml:body")
                .style("font", "14px");
    var selectHTML = "<button id='"+id+"' altText='"+alt2+"' style='width:95px' ";
    selectHTML += "onclick='switchButtonText(this); refreshPieChart();'>"+alt1+"</button>";
    select.html(selectHTML);
    return document.getElementById(id);
}

// when curr folder changes, refreshes the pie chart
function refreshPieChart(){

    var split = curr.name.split("/");// only get folder name
    pieTitle.text(split[split.length-1]);
    gPieSlices.selectAll("path").remove();

    // determines if pie is drawn by number of files vs percent size of files
    var byNumber = pieChartMeasure.textContent==="by % files";
    // determines if pie is drawn by file type or file category
    var byType = pieChartType.textContent==="by file type";
    // determines if folder will be searched one level deep or by all levels
    var byRecursion = pieChartDeep.textContent==="all levels";

    // if all levels deep, then will search entire folder
    // if one level deep, will only look at immmediate files
    currRad = 0;
    var folders = [curr];
    var fileTypes = {};
    var numFiles = 0;
    var sizeFiles = 0;
    while (folders.length>0){
        var currFolder = folders.shift();
        for (var i=0; i<currFolder.fileChildren.length; i++){
            numFiles+=1;
            var type = currFolder.fileChildren[i].type;
            if (!byType) type = typeToCat[type];
            var size = +currFolder.fileChildren[i].size;
            if (fileTypes[type]==null) fileTypes[type] = {};
            if (fileTypes[type]["num"]==null) fileTypes[type]["num"] = 1;
            else fileTypes[type]["num"] += 1;
            if (fileTypes[type]["size"]==null) fileTypes[type]["size"] = size;
            else fileTypes[type]["size"] += size;
            sizeFiles += size;
        }
        if (byRecursion){
            for (var i=0; i<currFolder.folderChildren.length; i++) folders.push(currFolder.
            folderChildren[i]);
        }
    }

    defaultPieDetails[3] = "Folder has";
    defaultPieDetails[4] = numFiles+" files";
    if (byRecursion) defaultPieDetails[5] = "all levels deep";
    else defaultPieDetails[5] = "one level deep";

    setPieDetailsText(defaultPieDetails);
    var sliceData = {};
    var idx=10;
    // adds slice for each file type represented
    for (var type in fileTypes){
        // determine color
        if (byType) var category = typeToCat[type];
        else category = type;
        if (category==null || category==undefined || category=="undefined") category = "Other";
        color = catToColor[category];

        // pie slice text details
        var lines = [];
        var num = fileTypes[type]["num"];
        var size = +fileTypes[type]["size"];
        var sizeLabel = " B";
        if (size>1024){size/=1024;sizeLabel=" KB"};
        if (size>1024){size/=1024;sizeLabel=" MB"};
        if (size>1024){size/=1024;sizeLabel=" GB"};
        size = Math.round(size*10)/10;

        // if option selected: by percent number of files
        if (byNumber){
            var perc = fileTypes[type]["num"]/numFiles;
            lines.push("PERCENT OF FILES: "+Math.round(perc*100)+"%");
        }
        // if option selected: by percent size of files
        else{
            var perc = fileTypes[type]["size"]/sizeFiles;
            lines.push("PERCENT OF SIZE: "+Math.round(perc*100)+"%");
        }

        // if option selected: by file type
        if (byType){
            var typeStr = "TYPE: "+type;
            if (category!=null) typeStr += " ("+category+")";
            lines.push(typeStr);
            lines.push("FILES: "+num);
            lines.push(" ");
            lines.push(" ");
            lines.push("SIZE: "+size+sizeLabel);
        }
        // if option selected: by file category
        else{
            lines.push("CATEGORY:         "+category);
            lines.push("FILES:  "+num);
            lines.push(" ");
            lines.push(" ");
            lines.push("SIZE:  "+size+sizeLabel);
        }

        sliceData[idx.toString()]={"lines":lines, "color":color, "type":type, "perc":perc};
        idx+=1;
    }


    // places slices in order of descending value
    var i=1;
    while (i>0){
        i=0;
        var bestIdx = -1;
        var bestValue = 0;
        for (data in sliceData){
            i+=1
            if (sliceData[data]["perc"]>bestValue){
                bestIdx = data;
                bestValue = sliceData[data]["perc"];
            }
        }
        if (i>0){
            console.log(sliceData[bestIdx]["type"]);
            addSlice(sliceData[bestIdx]["color"], bestValue, sliceData[bestIdx]["type"].replace(
            "/",""), sliceData[bestIdx]["lines"]);
            delete sliceData[bestIdx];
        }
    }

}

// add slice to pie chart
function addSlice(color, percent, id, lines){
    // pie slice
    var rad = percent*Math.PI*2;
    currSlice = gPieSlices.append("path")
        .attr("fill", color)
        .attr("transform", transPie)
        .attr("stroke", "black")
        .attr("stroke-width", "1")
        .attr("id", id)
        .datum({
            startAngle: currRad,
            endAngle: currRad+rad,
            percent: percent,
            id:id,
            lines:lines
        })
        .attr("d",arc)
        .on("mouseover", function(d){
            //currSlice.attr("stroke-width", "1");
            var slice = document.getElementById(d.id);
            var parent = slice.parentNode;
            for (var i=0; i<parent.children.length; i++){
                if (parent.children[i]!==slice) parent.appendChild(parent.children[i]);
            }
            currSlice = d3.select("#"+d.id);
            currSlice.attr("stroke-width", "3");
            setPieDetailsText(d.lines);
            treeBrushing(d.id);
        })
        .on("mouseout", function(d){
            currSlice.attr("stroke-width", "1");
            setPieDetailsText(defaultPieDetails);
            treeBrushing(null);

        });

    currRad += rad;
};

function treeBrushing(type){
    var isBrushing = type!=null;

    treeNodes.each(function(d){
        var isFolder = d.size==null;
        if (!isFolder){

            if (!isBrushing){
                d.element.attr("opacity", null);
                d.text.attr("display", "none");
            }
            else if (d.element.attr("type")!==type && typeToCat[d.element.attr("type")]!==type
            && (type!="undefined" || typeToCat[d.element.attr("type")]!=undefined)){
                d.element.attr("opacity", "0");
            }
            else{
                d.text.attr("display", null);
            }

        }
    });
}


// resets the details for the pie chart given an array of strings
function setPieDetailsText(lines){
    pieDetails.html("");
    for (var i=0; i<lines.length; i++){
        var dy = 0;
        var x = +pieDetails.attr("x");
        if (i>0) dy = 20;
        if (i==3) dy *= -2;
        if (i>=3) x += pieSVGSize*.52;
        pieDetails.append("tspan")
            .attr("x", x)
            .attr("dy", dy)
            .text(lines[i]);
    }
}

// given a button element, switch the button value with the alttext attribute
function switchButtonText(button){
    var curr=button.textContent;
    button.textContent=button.attributes.altText.value;
    button.attributes.altText.value = curr;
}
```