```
// creates all the svg elements necessary for the details on demand
// for files and folders
function drawDetailsOnDemand(svg){

    // width and height of pie SVG
    dodWidth = dodWindowSize;
    dodHeight = document.getElementById(svg.attr("id")).getBoundingClientRect().height;
    var h=dodHeight;// shortcut for height
    var xOffset = 10;// translates x of gDod
    var columnWidth = 150;
    currYPercent = 0;
    yPercentSpacing = .03;

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

    // g for all svg elements
    gDodMain = svg.append("g").attr("id", "gDetails").attr("transform","translate("+xOffset+
    ",20)");
    gDodFolder = gDodMain.append("g").attr("id", "gDetailsFolder");

    // FOLDER DETAILS
    gDod = gDodFolder;

    // DOD Title: Folder Name
    makeTextElement(dodWidth/2-xOffset, 0, 1, "dodTitle").text("Folder Details:");
    dodFolderTitle = makeTextElement(dodWidth/2-xOffset, h*currYPercent, 2, "dodTitle");

    // Full Path Name
    makeTextElement(0,h*currYPercent,1.5,"dodHeading").text("Path");
    dodFolderPathSelect = createSelectElement(0, h*currYPercent, 2.5, "dodPathSelect", []);

    // One Level Deep
    makeTextElement(0,h*currYPercent,1.5,"dodHeading").text("One Level Deep");
    dodFolderOneFiles = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Number Files:  ");
    dodFolderOneFolders = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Number Folders:  ");
    dodFolderOneRecentFile = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Most Recent File:  ");
    dodFolderOneRecentFolder = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue");
    makeTextElement(0,h*currYPercent,2,"dodAttrName").text("Most Recent Folder:  ");

    // All Levels
    makeTextElement(0,h*currYPercent,1.5,"dodHeading").text("All Levels");
    dodFolderAllFiles = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Number Files:  ");
    dodFolderAllFolders = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Number Folders:  ");
    dodFolderAllSize = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Size:  ");
    dodFolderAllDeepLevel = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Deepest Level:  ");
```

```
    //dodAllDeepName = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue");
    dodFolderAllDeepNameSelect = createSelectElement(columnWidth, h*currYPercent, 0,
    "dodAllDeepNameSelect", []);
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Deepest Folder:  ");

    // FILE DETAILS
    gDodFile = gDodMain.append("g").attr("id", "gDetailsFiles");
    gDod = gDodFile;
    currYPercent = 0;

    // DOD Title: File Name
    makeTextElement(dodWidth/2-xOffset, 0, 1, "dodTitle").text("File Details:");
    dodFileTitle = makeTextElement(dodWidth/2-xOffset, h*currYPercent, 2, "dodTitle").text(
    "ASDF");

    // File Path
    makeTextElement(0,h*currYPercent,1.5,"dodHeading").text("Path");
    dodFilePathSelect = createSelectElement(0, h*currYPercent, 2.5, "dodPathSelect", []);

    // File Details
    makeTextElement(0,h*currYPercent,1.5,"dodHeading").text("Details");
    dodFileSize = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue").text("asdf");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("File Size:  ");
    dodFileParent = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue").text("asdf");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Parent Folder:  ");
    dodFileLevels = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue").text("asdf");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("Levels Deep:  ");
    dodFileType = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue").text("asdf");
    makeTextElement(0,h*currYPercent,1,"dodAttrName").text("File Type:  ");
    dodFileCategory = makeTextElement(columnWidth,h*currYPercent,0,"dodAttrValue").text("asdf");
    makeTextElement(0,h*currYPercent,2,"dodAttrName").text("File Category:  ");

    makeTextElement(0,h*currYPercent,1.5,"dodHeading").text("Last Modified");
    dodFileModTime = makeTextElement(0,h*currYPercent,0,"dodAttrValue").text("asdf");

    // Legend
    gLegend = svg.append("g").attr("id", "gLegend").attr("transform","translate(0,"+dodHeight*
    .73+")");
    addBoundingRect(gLegend);
    currYPercent = yPercentSpacing;
    makeTextElementLegend(xOffset,h*currYPercent,1.5,"dodHeading").text("File Type Legend");
    var boxThick = 10;
    var alt = true;
    for (var category in catToColor){
        alt = !alt;
        var x = alt*(dodWidth/2);
        gLegend.append("rect")
            .attr("stroke", "black")
            .attr("stroke-width", "1")
            .attr("x", x+xOffset)
            .attr("y", h*currYPercent-boxThick*3/4)
            .attr("width", boxThick)
            .attr("height", boxThick)
```

```
            .attr("fill", catToColor[category]);
        makeTextElementLegend(x+boxThick*1.5+xOffset,h*currYPercent,1*alt,"dodLegendCat").text(
        category);
    }

    refreshFolderDetailsOnDemand(curr);
    //refreshFileDetailsOnDemand(curr.fileChildren[0]);
}

// creates a text element in gDod
function makeTextElement(x, y, numYSpaces, className){
    var text = gDod.append("text")
        .attr("x", x)
        .attr("y", y)
        .attr("class", className);
    currYPercent += yPercentSpacing*numYSpaces;
    return text;
}

function makeTextElementLegend(x, y, numYSpaces, className){
    var text = gLegend.append("text")
        .attr("x", x)
        .attr("y", y)
        .attr("class", className);
    currYPercent += yPercentSpacing*numYSpaces;
    return text;
}


// given a list of files or folders, sets the text to the name of the most recent
function setTextRecent(textElement, fileFolderArray){
    if (fileFolderArray.length>0){
        var recent = fileFolderArray[0];
        for (var i=1; i<fileFolderArray.length; i++){
            if (fileFolderArray[i].modTime > recent.modTime) recent = fileFolderArray[i];
        }
        textElement.text(recent.name);
    }
    else textElement.text("");
}

// get every folder in a folder, all levels deep
function getAllFolders(folder){
    var folderQueue = [folder];// folder queue
    var allFolders = [];
    while (folderQueue.length>0){
        var currentFolder = folderQueue.shift();
        allFolders.push(currentFolder);
        // push each folder to the folder queue
        for (var i=0; i<currentFolder.folderChildren.length; i++){
            folderQueue.push(currentFolder.folderChildren[i]);
        }
    }
```

```
    return allFolders;
}

// get every files from each folder in a list
function getAllFiles(folders){
    var allFiles = [];
    for (var i=0; i<folders.length; i++){
        for (var j=0; j<folders[i].fileChildren.length; j++){
            allFiles.push(folders[i].fileChildren[j]);
        }
    }
    return allFiles;
}

// gets the deepest level of folder and the number of levels deep
function getDeepestLevel(folder){
    var folderQueue = [folder];// folder queue
    var deepQueue = [0];
    var deepestValue = 1;
    var deepestName = folder.name;
    while (folderQueue.length>0){
        var currentFolder = folderQueue.shift();
        var currentDeep = deepQueue.shift()+1;
        if (currentDeep>deepestValue){
            deepestValue = currentDeep;
            deepestName = currentFolder.path;
        }
        // push each folder to the folder queue
        for (var i=0; i<currentFolder.folderChildren.length; i++){
            folderQueue.push(currentFolder.folderChildren[i]);
            deepQueue.push(currentDeep);
        }
    }

    if (deepestValue==1){
        deepestName = folder.name;
        var split = deepestName.split("/");// only get folder name
        deepestName = split[split.length-1];
    }
    else deepestName = deepestName.substring(folder.path.length)

    return [deepestValue, deepestName];
}

// Sets all the text values for the details on demand based on the given folder
function refreshFolderDetailsOnDemand(folder){

    gDodFolder.attr("display", null);
    gDodFile.attr("display", "none");

    // Title
    var split = folder.name.split("/");// only get folder name
    dodFolderTitle.text(split[split.length-1]);
```

```
    // Path
    setSelectOptions(dodFolderPathSelect, folder.path.split("/"));// select options

    // One Level Deep
    dodFolderOneFiles.text(folder.fileChildren.length);
    dodFolderOneFolders.text(folder.folderChildren.length);
    setTextRecent(dodFolderOneRecentFile, folder.fileChildren);
    setTextRecent(dodFolderOneRecentFolder, folder.folderChildren);

    // ALL LEVELS
    var dodFolders = getAllFolders(folder);
    var dodFiles = getAllFiles(dodFolders);
    dodFolders.shift();// remove folder from folder list
    var size = +folder.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"};
    var size = Math.round(size*1000)/1000;
    var deep = getDeepestLevel(folder);
    var deepValue = deep[0];
    var deepName = deep[1];
    dodFolderAllFiles.text(dodFiles.length);
    dodFolderAllFolders.text(dodFolders.length);
    dodFolderAllSize.text(size+sizeLabel);
    dodFolderAllDeepLevel.text(deepValue);
    setSelectOptions(dodFolderAllDeepNameSelect, deepName.substring(1).split("/"));// select
    options

}


function refreshFileDetailsOnDemand(file){

    gDodFile.attr("display", null);
    gDodFolder.attr("display", "none");

    // Title
    dodFileTitle.text(file.name);

    // Path
    setSelectOptions(dodFilePathSelect, file.path.split("/"));// select options

    // Size
    var size = +file.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"};
    var size = Math.round(size*1000)/1000;
    dodFileSize.text(size+sizeLabel);
```

```
    // Other Details

    dodFileModTime.text(new Date().toUTCString(+file.moddate));
    var parentName = file.parent.name.split("/");
    parentName = parentName[parentName.length-1];
    dodFileParent.text(parentName);
    var levels = file.path.split("/").length;
    levels -= file.parent.path.split("/").length;
    dodFileLevels.text(levels);
    dodFileType.text(file.type);
    cat = typeToCat[file.type];
    if (cat==null) cat="Other";
    dodFileCategory.text(cat);
}


// creates a select box with options in the pie chart
function createSelectElement(x, y, numYSpaces, id, optionList){
    var select = gDod.append("g")
        .attr("class", "svgSelect")
        .attr("id", id)
        .append("foreignObject")
            .attr("x", x)
            .attr("y", y-yPercentSpacing*dodHeight*.75)
            .attr("width", 400)
            .attr("height",70)
            .append("xhtml:body")
            .style("font", "14px");

    setSelectOptions(select, optionList);
    currYPercent += yPercentSpacing*numYSpaces;
    return select;
}

function setSelectOptions(select, optionList){
    var selectHTML = "<select>";
    for (var i=0; i<optionList.length; i++){
        if (i===0) selectHTML += "<option>"+optionList[i]+"/</option>";
        else if (i==optionList.length-1) selectHTML += "<option disabled>"+optionList[i]+
        "</option>";
        else selectHTML += "<option disabled>"+optionList[i]+"/</option>";

    }
    selectHTML += "</select>";
    select.html(selectHTML);
}
```