```
// runs on website startup
function init(){

    var content = d3.select("#content");

    // size of pie and dod svg windows
    pieWindowSize = 200;
    dodWindowSize = 300;


    // top svg element
    var svg = content.append("svg")
        .attr("width", "100%")
        .attr("height", "100%")
        .attr("id", "svgTop");

    // svg for tree map
    svgTree = svg.append("svg")
        .attr("id", "svgTree")
    addBoundingRect(svgTree);

    // svg for pie chart
    var svgPie = svg.append("svg")
        .attr("y", (window.innerHeight-pieWindowSize))
        .attr("height", pieWindowSize)
        .attr("id", "svgPie");
    addBoundingRect(svgPie);
    svgPie.attr("width", document.getElementById(svgPie.attr("id")).getBBox().height);

    // svg for back and forward history buttons
    var svgHistory = svg.append("svg")
        .attr("width", "150")
        .attr("height", "75");
    historyFillColorOn = "#00dd00";// color when back or forward arrow is active
    historyFillColorOff = "#222222";// color when back or forward arrow is disabled
    addBoundingRect(svgHistory);


    // svg for details on demand
    var dodX = window.innerWidth-document.getElementById(svgPie.attr("id")).
    getBoundingClientRect().width;
    var svgDod = svg.append("svg")
        .attr("x", window.innerWidth-dodWindowSize)
        .attr("id", "svgDod");
    addBoundingRect(svgDod);

    // mapping of file category to category color
    catToColor = {
        "Data/Text":"#a6cee3",
        "Document":"#1f78b4",
        "Presentation":"#b2df8a",
        "Spreadsheet":"#33a02c",
        "Layout":"#fb9a99",
```

```
        "Picture":"#e31a1c",
        "Program":"#fdbf6f",
        "Video":"#ff7f00",
        "Executable":"#cab2d6",
        "Compressed":"#6a3d9a",
        "Audio":"#ffff66",
        "Disk/System":"#b15928",
        "Other":"#999999"
    };

    // mapping of file type to file category
    typeToCat = {
        "txt":"Data/Text",
        "csv":"Data/Text",
        "tsv":"Data/Text",
        "xml":"Data/Text",
        "dat":"Data/Text",
        "tar":"Data/Text",

        "doc":"Document",
        "docx":"Document",
        "wpd":"Document",
        "wps":"Document",

        "ppt":"Presentation",
        "pptx":"Presentation",
        "pps":"Presentation",
        "key":"Presentation",

        "xls":"Spreadsheet",
        "xlsx":"Spreadsheet",
        "ods":"Spreadsheet",

        "pdf":"Layout",

        "jpg":"Picture",
        "png":"Picture",
        "gif":"Picture",
        "bmp":"Picture",
        "tif":"Picture",

        "c":"Program",
        "class":"Program",
        "cmd":"Program",
        "cpp":"Program",
        "cs":"Program",
        "dtd":"Program",
        "fla":"Program",
        "h":"Program",
        "java":"Program",
        "javac":"Program",
        "m":"Program",
        "pl":"Program",
```

```
"py":"Program",
"pyc":"Program",
"pyw":"Program",
"m":"Program",
"sh":"Program",
"sln":"Program",
"vcxproj":"Program",
"xcodeproj":"Program",
"asp":"Program",
"aspx":"Program",
"cer":"Program",
"cfm":"Program",
"csr":"Program",
"css":"Program",
"htm":"Program",
"html":"Program",
"js":"Program",
"jsp":"Program",
"php":"Program",
"rss":"Program",
"xhtml":"Program",
"url":"Program",


"mpg":"Video",
"flv":"Video",
"mp4":"Video",
"avi":"Video",
"mov":"Video",
"wmv":"Video",
"mkv":"Video",
"mts":"Video",
"ogv":"Video",

"apk":"Executable",
"app":"Executable",
"bat":"Executable",
"cgi":"Executable",
"com":"Executable",
"exe":"Executable",
"gadget":"Executable",
"jar":"Executable",
"pif":"Executable",
"fb":"Executable",
"wsf":"Executable",

"7z":"Compressed",
"cbr":"Compressed",
"deb":"Compressed",
"gz":"Compressed",
"pkg":"Compressed",
"rar":"Compressed",
"rpm":"Compressed",
```

```
        "sitz":"Compressed",
        "zip":"Compressed",
        "zipx":"Compressed",

        "mp3":"Audio",
        "wav":"Audio",
        "aif":"Audio",
        "m4a":"Audio",
        "ogg":"Audio",
        "flac":"Audio",
        "wma":"Audio",

        "bin":"Disk/System",
        "cue":"Disk/System",
        "dmg":"Disk/System",
        "iso":"Disk/System",
        "mdf":"Disk/System",
        "toast":"Disk/System",
        "vcd":"Disk/System",
        "cab":"Disk/System",
        "cpl":"Disk/System",
        "cur":"Disk/System",
        "deskthemepack":"Disk/System",
        "dll":"Disk/System",
        "dmp":"Disk/System",
        "drv":"Disk/System",
        "icns":"Disk/System",
        "ico":"Disk/System",
        "lnk":"Disk/System",
        "sys":"Disk/System"
    };


    // reads each line in the results tsv file
    d3.tsv("FileFinder/results.tsv", function(data){

        // puts each file/folder as an array object in files
        var temp = data.map(function(d){
            return{
                name:d.name,
                size:d.size,
                isFolder:d.isfolder==="Y",
                parent:+d.parent,
                modTime:+d.moddate,
                modTimeString: new Date().toUTCString(+d.moddate)
            }
        });

        files = [];
        // iterate over files/folders in array
        for (var i=0; i<temp.length; i++){
            // folder
            if (temp[i].isFolder) files.push(new Folder(temp[i], files, i));
```

```
            // file
            else files.push(new File(temp[i], files, i));
        }

        // root is always the root of the folder tree
        root = files[0];
        console.log(files);

        // curr changes based on user interaction
        curr = root;
        currHistory = [curr];
        currHistoryIdx = 0;

        // console.log the path name of each file and folder
        var folders = [root];// folder queue
        while (folders.length>0){
            var currentFolder = folders.shift();

            // do stuff to current folder
            console.log(currentFolder.path);

            // do something to each file child in the current folder
            for (var i=0; i<currentFolder.fileChildren.length; i++){
                // do stuff to file
                console.log(currentFolder.fileChildren[i].path+", "+currentFolder.fileChildren[i
                ].type);
            }

            // push each folder to the folder queue
            for (var i=0; i<currentFolder.folderChildren.length; i++){
                folders.push(currentFolder.folderChildren[i]);
            }

        }


        // draws tree map
        drawTreeMap(svgTree);

        // draws dod
        drawDetailsOnDemand(svgDod);

        // draws pie chart
        drawPieChart(svgPie);

        drawHistoryArrow(svgHistory);

    });

}

// add faint grey rectangle to given svg
function addBoundingRect(svgElement){
```

```
    svgElement.append("rect")
        .attr("width","100%")
        .attr("height","100%")
        .attr("fill", "none")
        .attr("class", "boundingRect")
        .attr("stroke-width", "1px")
        .attr("stroke", "#aaaaaa");
}

// Contains all data for a file
function File(obj, files, fid){
    this.name = obj.name;
    this.size = obj.size;
    this.modTime = obj.modTime;
    this.parent = files[obj.parent];
    this.parent.fileChildren.push(this);
    this.path = getPathName(this);
    var dotSplit = obj.name.split(".");
    if (dotSplit.length>1) this.type = dotSplit[dotSplit.length-1].toLocaleLowerCase();
    else this.type = "";
    this.fid = fid;
}

// Contains all data for a folder
function Folder(obj, files, fid){
    this.name = obj.name;
    this.size = obj.size;
    this.modTime = obj.modTime;
    if (obj.parent>=0){
        this.parent = files[obj.parent];
        this.parent.folderChildren.push(this);
    }
    else this.parent = null;
    this.fileChildren = [];
    this.folderChildren = [];
    this.path = getPathName(this);
    this.fid = fid;
}

// Used by File and Folder constructors to construct the path name
function getPathName(obj){
    var path = obj.name;
    curr = obj.parent;
    while (curr!=null){
        path = curr.name+"/"+path;
        curr = curr.parent;
    }
    return path;
}

// always call this when setting the new curr folder
function setCurr(newCurrIdx){
    if (curr != files[newCurrIdx]){
```

```
        curr = files[newCurrIdx];

        // maintain history
        console.log(currHistory);
        while (currHistoryIdx+1 < currHistory.length) currHistory.pop();
        currHistory.push(curr);
        currHistoryIdx += 1;
        resetHistoryArrowColors();
        backArrow.attr("fill", historyFillColorOn);

        resetVis();
    }
}

// resets vis with last curr folder in history
function historyBack(){
    if (currHistoryIdx>0){
        currHistoryIdx -= 1;
        curr = currHistory[currHistoryIdx];
        resetVis();
    }
    if (currHistoryIdx>0) backArrow.attr("fill", historyFillColorOn);
    else backArrow.attr("fill", historyFillColorOff);
    if (currHistoryIdx < currHistory.length-1) forwardArrow.attr("fill", historyFillColorOn);
    else forwardArrow.attr("fill", historyFillColorOff);
    resetHistoryArrowColors();
}

// resets vis with next curr folder in history
function historyForward(){
    if (currHistoryIdx < currHistory.length-1){
        currHistoryIdx += 1;
        curr = currHistory[currHistoryIdx];
        resetVis();
    }
    resetHistoryArrowColors();
}

// checks if back and forward arrow buttons should be active or disabled
function resetHistoryArrowColors(){
    if (currHistoryIdx>0) backArrow.attr("fill", historyFillColorOn);
    else backArrow.attr("fill", historyFillColorOff);
    if (currHistoryIdx < currHistory.length-1) forwardArrow.attr("fill", historyFillColorOn);
    else forwardArrow.attr("fill", historyFillColorOff);
}

// resets the pie chart, dod, and tree
function resetVis(){
    drawTreeMap(svgTree);
    refreshPieChart();
    refreshFolderDetailsOnDemand(curr);
}
```

```
// creates back and forward arrow buttons
function drawHistoryArrow(svg){
    var width = svg.attr("width");
    var height = svg.attr("height");
    var radius = height/3;
    var fontsize = height/2;

    var backArrowG = svg.append("g")
        .on("click", historyBack);
    backArrow = backArrowG.append("circle")
        .attr("cx", width/4)
        .attr("cy", height/2)
        .attr("r", radius)
        .attr("fill", historyFillColorOff)
        .attr("stroke", "black")
        .attr("stroke-width", 3);
    backArrowG.append("text")
        .attr("x", width/4-fontsize/10)
        .attr("y", height/2+fontsize/4)
        .attr("text-anchor", "middle")
        .attr("font-size", fontsize)
        .text("◀");

    var forwardArrowG = svg.append("g")
        .on("click", historyForward);
    forwardArrow = forwardArrowG.append("circle")
        .attr("cx", width*3/4)
        .attr("cy", height/2)
        .attr("r", radius)
        .attr("fill", historyFillColorOff)
        .attr("stroke", "black")
        .attr("stroke-width", 3);
    forwardArrowG.append("text")
        .attr("x", width*3/4+fontsize/10)
        .attr("y", height/2+fontsize/4)
        .attr("text-anchor", "middle")
        .attr("font-size", fontsize)
        .text("▶");

}

// repositions pie and dod svgs
window.onresize = function(){
    d3.select("#svgDod")
        .attr("x", window.innerWidth-dodWindowSize);
    d3.select("#svgPie")
        .attr("y", (window.innerHeight-pieWindowSize))
        .attr("height", pieWindowSize);
};
```