```javascript
var JSReflow = JSReflow || {};

JSReflow.ps = 11;
JSReflow.vs = 15;
JSReflow.pt_per_px = 0.75; // 12pt = 16px
JSReflow.top_margin = 2 * JSReflow.vs;
JSReflow.bot_margin = JSReflow.top_margin / 2;
JSReflow.min_gutter = 15;
JSReflow.scale = 1;
JSReflow.multi_column = true;
JSReflow.current_paragraph = 0;
JSReflow.current_line = 0;
JSReflow.min_badness_index = 0;
JSReflow.cols_per_page = 1;
JSReflow.pageWidth = 0;
JSReflow.pageHeight = 0;


JSReflow.float_types = Object.freeze({
    NONE: 0,
    DUMB: 1,
    MSWORD: 2,
    SIMPLEQUEUE: 3
});


JSReflow.float_type = JSReflow.float_types.SIMPLEQUEUE;


JSReflow.getDiv = function (cssclass, width, height, text) {
    "use strict";
    return '<div class="' + cssclass + '" style="width: ' + width + 'pt; height: ' + height + 'pt; font-size: ' + JSReflow.ps * JSReflow.scale + 'pt">' + text + '</div>\n';
};

JSReflow.getLine = function (words) {
    "use strict";
    var text, length, offset, word, i, pos;
    text = "<div class=\"rel\">";
    length = words.length;
    offset = null;
    word = null;
    pos = 0;
    for (i = 0; i < length; i++) {
        offset = JSReflow.deltasdict[words[i][0]];
        word = words[i][1];
        text += "<div class=\"inner\" style=\"left: " + (pos + offset) * JSReflow.scale + "pt;\">" + JSReflow.dictionary[word][0] + " </div>";
        pos = pos + JSReflow.dictionary[word][1] + offset;
    }
    text += "</div>";
    return JSReflow.getDiv("line", 0, 0, text);
};

JSReflow.getFloat = function (width, height, text) {
    "use strict";
    return JSReflow.getDiv("floatable", width * JSReflow.scale, height * JSReflow.scale, text);
};

JSReflow.outputDiv = function (cssclass, x, y, text) {
    "use strict";
    $('#main').append('<div class="' + cssclass + '" style="position:absolute; top: ' + y + 'pt; left: ' + x + 'pt">' + text + '</div>');
};

JSReflow.nextPos = function () {
    "use strict";
    // Increments JSReflow.curr_row and JSReflow.curr_col to next available empty space (possibly the same empty space)
    var c, r;
    for (c = JSReflow.curr_col; true; c++) {
        for (r = JSReflow.curr_row; r < JSReflow.num_rows; r++) {
            if (Array.isArray(JSReflow.columns[c])) {
                if (JSReflow.columns[c][r] === undefined) {
                    JSReflow.curr_col = c;
                    JSReflow.curr_row = r;
                    return;
                }
            } else {
                JSReflow.columns[c] = [];
                JSReflow.curr_col = c;
                JSReflow.curr_row = 0;
                return;
            }
        }
        JSReflow.curr_row = 0;
    }
};

JSReflow.checkSpace = function (nlines, span) {
    "use strict";
    var c, r, cols_left_on_page;

    cols_left_on_page = JSReflow.cols_per_page - (JSReflow.curr_col % JSReflow.cols_per_page);
    if (cols_left_on_page < span) { // don't span page boundaries
        return false;
    }

    if (nlines > JSReflow.num_rows && JSReflow.curr_row === 0) { //too big, so put it at the top of a column
        return true;
    }

    if (JSReflow.curr_row + nlines > JSReflow.num_rows) { // too far down
        return false;
    }

    // Now check that there is enough contiguous space in both directions
    for (r = JSReflow.curr_row; r < JSReflow.curr_row + nlines; r++) {
        for (c = JSReflow.curr_col; c < JSReflow.curr_col + span; c++) {
            if (!Array.isArray(JSReflow.columns[c])) {
                JSReflow.columns[c] = [];
            }
            if (JSReflow.columns[c][r] !== undefined) {
                //console.log("not undefined:", c, r);
                return false;
            }
        }
    }
    return true;
};

JSReflow.placeFloat = function (w, h, nlines, span, text) {
    "use strict";
    // Reserve space for the figure, and place it

    // How many lines do we need to reserve?
    // Let's say if it's less than n.5,  no need for an extra space

    var i, j;

    for (i = JSReflow.curr_row; i < JSReflow.curr_row + nlines; i++) {
        for (j = JSReflow.curr_col; j < JSReflow.curr_col + span; j++) {
            if (!Array.isArray(JSReflow.columns[j])) {
                JSReflow.columns[j] = [];
            }
            JSReflow.columns[j][i] = "";
        }
    }

    JSReflow.columns[JSReflow.curr_col][JSReflow.curr_row] = JSReflow.getFloat(w, h, text);

    if (JSReflow.curr_row > 0) { // blank out any lines above
        for (i = JSReflow.curr_col + 1; i < JSReflow.curr_col + span; i++) {
            JSReflow.columns[i][JSReflow.curr_row - 1] = "";
        }
    }
};

JSReflow.paginate = function (offset) {
    "use strict";
    var start, end, p, l;
    if (offset + JSReflow.curr_page > JSReflow.pages) {
        $('#next').attr("disabled", true);
    } else if (offset + JSReflow.curr_page < 1) {
        $('#prev').attr("disabled", true);
    } else {
        JSReflow.curr_page += offset;

        if (JSReflow.curr_page > 1) {
            $('#prev').attr("disabled", false);
        }
        if (JSReflow.curr_page < JSReflow.pages) {
            $('#next').attr("disabled", false);
        }

        if (JSReflow.curr_page >= JSReflow.pages) {
            $('#next').attr("disabled", true);
        } else if (JSReflow.curr_page <= 1) {
            $('#prev').attr("disabled", true);
        }


        $('#pageno').html(JSReflow.curr_page);

        $('#main').html("");

        // Now we need to output the contents of the JSReflow.columns array
        start = (JSReflow.curr_page - 1) * JSReflow.cols_per_page;
        end = JSReflow.curr_page * JSReflow.cols_per_page;

        for (p = start; p < JSReflow.columns.length && p < end; p++) {
            if (!Array.isArray(JSReflow.columns[p])) {
                continue;
            }

            for (l = 0; l < JSReflow.columns[p].length; l++) {
                if (JSReflow.columns[p][l] === undefined || JSReflow.columns[p][l] === "") {
                    continue;
                }

                JSReflow.outputDiv("container", ((JSReflow.col_sep - JSReflow.galley_width * JSReflow.scale) / 2 + (p - start) * JSReflow.col_sep), (JSReflow.top_margin + l * JSReflow.vs) * JSReflow.scale, JSReflow.columns[p][l]);
            }
        }
        //console.log(JSReflow.columns);
    }

    JSReflow.computePagination();
};

JSReflow.computePagination = function () {
    "use strict";
    var  p, l, w, h, scale, span, nlines, currcol, currrow, prev_col;

    for (p = JSReflow.current_paragraph; p < JSReflow.paragraph_tree.length; p++) {

        if (Array.isArray(JSReflow.paragraph_tree[p])) {
            // We assume it's a normal paragraph
            if (JSReflow.paragraph_tree[p][JSReflow.min_badness_index] === undefined) {
                continue; // hack due to extra empty paragraph generated by the Java/C combo
            }
            for (l = JSReflow.current_line; l < JSReflow.paragraph_tree[p][JSReflow.min_badness_index].length; l++) {
                prev_col = JSReflow.curr_col;
                JSReflow.nextPos();
                if (Math.floor(JSReflow.curr_col / JSReflow.cols_per_page) > Math.floor(prev_col / JSReflow.cols_per_page)) {
                    // Reached a page boundary
                    JSReflow.current_paragraph = p;
                    JSReflow.current_line = l;
                    //console.log(JSReflow.current_paragraph, JSReflow.current_line)
                    return;
                }
                JSReflow.columns[JSReflow.curr_col][JSReflow.curr_row] = JSReflow.getLine(JSReflow.paragraph_tree[p][JSReflow.min_badness_index][l]);
            }
            JSReflow.nextPos();
            if (JSReflow.curr_row !== 0) { // Suppress blank line at top of column
                JSReflow.columns[JSReflow.curr_col][JSReflow.curr_row] = "";
            }
        } else {
            // It must be a Float
            if (JSReflow.float_type === JSReflow.float_types.NONE) {
                continue;
            }

            // Scale width to fit column (and height proportionately)
            w = JSReflow.paragraph_tree[p].w * JSReflow.scale;
            h = JSReflow.paragraph_tree[p].h * JSReflow.scale;

            // Have a look at the width and see if it could span any columns
            span = Math.round(w / JSReflow.galley_width);

            if (span > JSReflow.cols_per_page) {
                span = JSReflow.cols_per_page;
            }

            if (span < 1 || !JSReflow.multi_column) {
                span = 1;
            }

            scale = (JSReflow.col_sep * span) - JSReflow.col_sep + JSReflow.galley_width * JSReflow.scale;
            scale = scale / w / JSReflow.scale;//(span * JSReflow.galley_width + (span - 1) * (JSReflow.col_sep - JSReflow.galley_width)) / w;

            w *= scale;
            h *= scale;

            // Is it too big to fit? If so scale down the height
            if (h > JSReflow.pageHeight) {
                scale = JSReflow.pageHeight / h;
                w *= scale;
                h *= scale;
            }

            nlines = Math.round(h / JSReflow.vs) + 1;

            if (JSReflow.float_type === JSReflow.float_types.SIMPLEQUEUE) {
                // Don't actually need a queue in gridlayout
                JSReflow.nextPos();
                currcol = JSReflow.curr_col;
                currrow = JSReflow.curr_row;

                // check if it'll actually fit in the column, and if not, move to a new one
                while (!JSReflow.checkSpace(nlines, span)) {
                    JSReflow.curr_col++;
                    JSReflow.curr_row = 0;
                    JSReflow.nextPos();
                }

                JSReflow.placeFloat(w, h, nlines, span, "<div>" + JSReflow.paragraph_tree[p].d + "</div>");

                JSReflow.curr_col = currcol;
                JSReflow.curr_row = currrow;

            } else if (JSReflow.float_type === JSReflow.float_types.DUMB) {

                // Just stick the figure out wherever
                JSReflow.nextPos();

                JSReflow.placeFloat(w, h, nlines, span, "<div>" + JSReflow.paragraph_tree[p].d + "</div>");

            } else if (JSReflow.float_type === JSReflow.float_types.MSWORD) {
                JSReflow.nextPos();
                // check if it'll actually fit in the column, and if not, move to a new one
                while (!JSReflow.checkSpace(nlines, span)) {
                    JSReflow.curr_col++;
                    JSReflow.curr_row = 0;
                    JSReflow.nextPos();
                }

                JSReflow.placeFloat(w, h, nlines, span, "<div>" + JSReflow.paragraph_tree[p].d + "</div>");

            }
            /*else {
             // Don't output anything.
             // console.log(JSReflow.float_type);
             }*/
        }
        JSReflow.current_line = 0;
        JSReflow.current_paragraph++;
    }
    //console.log(JSReflow.paragraph_tree);

};

JSReflow.pageInit = function () {
    "use strict";
    var i, nc, ex_ws, rq_ws, dropdown, badness, name;

    JSReflow.current_paragraph = 0;
    JSReflow.current_line = 0;

    JSReflow.curr_row = 0;
    JSReflow.curr_col = 0;


    $('body').css('font-size', (JSReflow.ps * JSReflow.scale) + 'pt');

    JSReflow.pageWidth = window.innerWidth * JSReflow.pt_per_px;
    badness = [];
    JSReflow.min_badness_index = 0;

    JSReflow.pageHeight = window.innerHeight * JSReflow.pt_per_px - (JSReflow.bot_margin + JSReflow.top_margin) * JSReflow.scale;

    for (i = 0; i < JSReflow.galley_widths.length; i++) {
        nc = Math.floor(JSReflow.pageWidth / ((JSReflow.galley_widths[i] + JSReflow.min_gutter) * JSReflow.scale));
        ex_ws = JSReflow.pageWidth % ((JSReflow.galley_widths[i] + JSReflow.min_gutter) * JSReflow.scale);
        rq_ws = nc * JSReflow.min_gutter * JSReflow.scale;

        badness.push((ex_ws /*+ rq_ws*/ + 100) * Math.sqrt(nc));
        if (nc === 0) {
            badness[i] = 1000 * i;
        }

        if (badness[i] <= badness[JSReflow.min_badness_index]) {
            JSReflow.min_badness_index = i;
            JSReflow.cols_per_page = nc;
        }
    }

    JSReflow.galley_width = JSReflow.galley_widths[JSReflow.min_badness_index];

    JSReflow.col_sep = Math.floor(JSReflow.pageWidth / JSReflow.cols_per_page);

    /*
     Previously, we (and I quote) "just shove[d] lines onto screen" -- Now we need to populate the JSReflow.columns array and then output the contents of that to the screen.

     So, what goes into the JSReflow.columns array?
     - There must be one element for every column in the document
     - These can be .push()ed on the fly
     - These are themselves arrays with one element per multiple of the main leading (JSReflow.vs in this case) that will fit in the column
     - If an element is undefined -- we assume it's available
     - Otherwise the element will be a string containing the html to output (possibly the empty string)
     - the html should specify its own dimensions, but not its position (this will be calculated by its position in the JSReflow.columns array)

     Yep, that sounds plausible.

     */

    JSReflow.columns = [];

    // How many lines? We have (window.innerHeight - JSReflow.top_margin - JSReflow.bot_margin) total space, and each line is JSReflow.vs high. So:
    JSReflow.num_rows = Math.floor((window.innerHeight * JSReflow.pt_per_px - (JSReflow.top_margin + JSReflow.bot_margin) * JSReflow.scale) / (JSReflow.vs * JSReflow.scale));
    if (JSReflow.num_rows < 1) {
        JSReflow.num_rows = 1;
    }


    JSReflow.pages = 100;//Math.ceil(JSReflow.columns.length / JSReflow.cols_per_page);
    JSReflow.curr_page = 1;

    $('#main').empty(); //clear old stuff

    dropdown = '<select id="dropdown" onchange="JSReflow.updateFloatType(this)">\n';
    for (name in JSReflow.float_types) {
        if (JSReflow.float_types.hasOwnProperty(name)) {
            dropdown += '<option value="' + JSReflow.float_types[name] + '"' + (JSReflow.float_type === JSReflow.float_types[name] ? " selected" : "") + '>' + name + '</option>';
        }
    }
    dropdown += "</select>\n";

    $('#top').html('<div class="title" style="position:absolute; width: 100%; height: ' + (JSReflow.ps + 5) + 'pt; left: 0pt; top: 3pt">Press \'h\' to hide this bar ' + dropdown + (JSReflow.float_type === JSReflow.float_types.NONE ? "none" : (JSReflow.float_type === JSReflow.float_types.DUMB ? "dumb" : (JSReflow.float_type === JSReflow.float_types.MSWORD ? "MS Word" : (JSReflow.float_type === JSReflow.float_types.SIMPLEQUEUE ? "simple queue" : "unknown (" + JSReflow.float_type + ")")))) + ", " + JSReflow.cols_per_page + " cols, " /* + JSReflow.pages + " pages, */ + "badness: " + Math.round(badness[JSReflow.min_badness_index] * 100) / 100 + ' <label><input id="multi" onclick="
JSReflow.toggleMulti();" type="checkbox" ' + (JSReflow.multi_column ? 'check
ed' : '') + ' > multicolumn?</label> <input type="button" id="prev" value="p
rev" onclick="paginate(-1)" disabled> <span id="pageno">' + JSReflow.curr_pa
ge + '</span> <input type="button" id="next" value="next" onclick="JSReflow.
paginate(1)" ' + (JSReflow.pages < 2 ? 'disabled' : '') + '></div>');

    JSReflow.computePagination();
    JSReflow.paginate(0);

};

JSReflow.updateFloatType = function (dd) {
    "use strict";
    JSReflow.float_type = parseInt(dd.options[dd.selectedIndex].value, 10);
    JSReflow.pageInit();
};

JSReflow.toggleMulti = function () {
    "use strict";
    JSReflow.multi_column = !JSReflow.multi_column;
    $('#multi').prop('checked', JSReflow.multi_column);
    JSReflow.pageInit();
};

JSReflow.scaleUp = function (factor) {
    "use strict";
    if (JSReflow.galley_widths[0] + JSReflow.min_gutter < JSReflow.page_widt
h) {
        JSReflow.scale *= factor;
    }
};

JSReflow.scaleDown = function (factor) {
    "use strict";
    JSReflow.scale /= factor;
};


$(window).resize(function () {
    "use strict";
    clearTimeout(JSReflow.resizetimer);
    JSReflow.resizetimer = setTimeout(JSReflow.pageInit, 200);
    JSReflow.pageInit();
});

$(document).ready(function () {
    "use strict";
    JSReflow.pageInit();
    $('#top').hide();
});

$("body").touchwipe({
    wipeLeft: function () {
        "use strict";
        JSReflow.paginate(1);
    },
    wipeRight: function () {
        "use strict";
        JSReflow.paginate(-1);
    },
    wipeUp: function () {
        "use strict";
        JSReflow.scale *= 1.1;
        JSReflow.pageInit();
    },
    wipeDown: function () {
        "use strict";
        JSReflow.scale /= 1.1;
        JSReflow.pageInit();
    },
    min_move_x: 10,
    min_move_y: 10,
    preventDefaultEvents: true
```

```
});

$(document).keydown(function (e) {
    "use strict";
    var key = e.which;

    // stop page from scrolling with arrow keys
    //noinspection FallthroughInSwitchStatementJS
    switch (key) {
    case 37:
    case 38:
    case 39:
    case 40:
        e.preventDefault();
    }
    //alert(key);
    if (key === 37) { // left
        JSReflow.paginate(-1);
    } else if (key === 38) { //up
        JSReflow.scale *= 1.01;
        JSReflow.pageInit();
    } else if (key === 39) { //right
        JSReflow.paginate(1);
    } else if (key === 40) { //down
        JSReflow.scale /= 1.01;
        JSReflow.pageInit();
    } else if (key === 77) { // m ("multicolumn")
        JSReflow.toggleMulti();
    } else if (key === 72) { // h ("hide bar")
        $('#top').toggle();
    }
});
```