

```

1  var JSReflow = JSReflow || {};
2
3  JSReflow.ps = 11;
4  JSReflow.vs = 15;
5  JSReflow.pt_per_px = 0.75; // 12pt = 16px
6  JSReflow.top_margin = 2 * JSReflow.vs;
7  JSReflow.bot_margin = JSReflow.top_margin / 2;
8  JSReflow.min_gutter = 15;
9  JSReflow.scale = 1;
10 JSReflow.multi_column = true;
11 JSReflow.current_paragraph = 0;
12 JSReflow.current_line = 0;
13 JSReflow.min_badness_index = 0;
14 JSReflow.cols_per_page = 1;
15 JSReflow.pageWidth = 0;
16 JSReflow.pageHeight = 0;
17
18
19 JSReflow.float_types = Object.freeze({
20     NONE: 0,
21     DUMB: 1,
22     MSWORD: 2,
23     SIMPLEQUEUE: 3
24 });
25
26
27 JSReflow.float_type = JSReflow.float_types.SIMPLEQUEUE;
28
29
30 JSReflow.getDiv = function (cssclass, width, height, text) {
31     "use strict";
32     return '<div class="' + cssclass + '" style="width: ' + width + 'pt; height: ' + height + 'pt; font-size: ' + JSReflow.ps * JSReflow.scale + 'pt">' + text + '</div>\n';
33 };
34
35 JSReflow.getLine = function (words) {
36     "use strict";
37     var text, length, offset, word, i, pos;
38     text = "<div class=\"rel\">";
39     length = words.length;
40     offset = null;
41     word = null;
42     pos = 0;
43     for (i = 0; i < length; i++) {
44         offset = JSReflow.deltasdict[words[i][0]];
45         word = words[i][1];
46         text += "<div class=\"inner\" style=\"left: " + (pos + offset) * JSReflow.scale + "pt;\>" + JSReflow.dictionary[word][0] + "</div>";
47         pos = pos + JSReflow.dictionary[word][1] + offset;
48     }
49     text += "</div>";
50     return JSReflow.getDiv("line", 0, 0, text);
51 };
52
53 JSReflow.getFloat = function (width, height, text) {
54     "use strict";
55     return JSReflow.getDiv("floatable", width * JSReflow.scale, height * JSReflow.scale, text);
56 };
57
58 JSReflow.outputDiv = function (cssclass, x, y, text) {
59     "use strict";
60     $('#main').append('<div class="' + cssclass + '" style="position:absolute; top: ' + y + 'pt; left: ' + x + 'pt">' + text + '</div>');
61 };
62
63 JSReflow.nextPos = function () {
64     "use strict";
65     // Increments JSReflow.curr_row and JSReflow.curr_col to next available empty space (possibly the same empty space)
66     var c, r;
67     for (c = JSReflow.curr_col; true; c++) {

```

```

68     for (r = JSReflow.curr_row; r < JSReflow.num_rows; r++) {
69         if (Array.isArray(JSReflow.columns[c])) {
70             if (JSReflow.columns[c][r] === undefined) {
71                 JSReflow.curr_col = c;
72                 JSReflow.curr_row = r;
73                 return;
74             }
75         } else {
76             JSReflow.columns[c] = [];
77             JSReflow.curr_col = c;
78             JSReflow.curr_row = 0;
79             return;
80         }
81     }
82     JSReflow.curr_row = 0;
83 }
84 };
85
86 JSReflow.checkSpace = function (nlines, span) {
87     "use strict";
88     var c, r, cols_left_on_page;
89
90     cols_left_on_page = JSReflow.cols_per_page - (JSReflow.curr_col % JSRefl
91 ow.cols_per_page);
92     if (cols_left_on_page < span) { // don't span page boundaries
93         return false;
94     }
95     if (nlines > JSReflow.num_rows && JSReflow.curr_row === 0) { //too big,
96 so put it at the top of a column
97         return true;
98     }
99     if (JSReflow.curr_row + nlines > JSReflow.num_rows) { // too far down
100         return false;
101     }
102
103     // Now check that there is enough contiguous space in both directions
104     for (r = JSReflow.curr_row; r < JSReflow.curr_row + nlines; r++) {
105         for (c = JSReflow.curr_col; c < JSReflow.curr_col + span; c++) {
106             if (!Array.isArray(JSReflow.columns[c])) {
107                 JSReflow.columns[c] = [];
108             }
109             if (JSReflow.columns[c][r] !== undefined) {
110                 //console.log("not undefined:", c, r);
111                 return false;
112             }
113         }
114     }
115     return true;
116 };
117
118 JSReflow.placeFloat = function (w, h, nlines, span, text) {
119     "use strict";
120     // Reserve space for the figure, and place it
121
122     // How many lines do we need to reserve?
123     // Let's say if it's less than n.5, no need for an extra space
124
125     var i, j;
126
127     for (i = JSReflow.curr_row; i < JSReflow.curr_row + nlines; i++) {
128         for (j = JSReflow.curr_col; j < JSReflow.curr_col + span; j++) {
129             if (!Array.isArray(JSReflow.columns[j])) {
130                 JSReflow.columns[j] = [];
131             }
132             JSReflow.columns[j][i] = "";
133         }
134     }
135
136     JSReflow.columns[JSReflow.curr_col][JSReflow.curr_row] = JSReflow.getFlo
137 at(w, h, text);

```

```

138     if (JSReflow.curr_row > 0) { // blank out any lines above
139         for (i = JSReflow.curr_col + 1; i < JSReflow.curr_col + span; i++) {
140             JSReflow.columns[i][JSReflow.curr_row - 1] = "";
141         }
142     }
143 };
144
145 JSReflow.paginate = function (offset) {
146     "use strict";
147     var start, end, p, l;
148     if (offset + JSReflow.curr_page > JSReflow.pages) {
149         $('#next').attr("disabled", true);
150     } else if (offset + JSReflow.curr_page < 1) {
151         $('#prev').attr("disabled", true);
152     } else {
153         JSReflow.curr_page += offset;
154
155         if (JSReflow.curr_page > 1) {
156             $('#prev').attr("disabled", false);
157         }
158         if (JSReflow.curr_page < JSReflow.pages) {
159             $('#next').attr("disabled", false);
160         }
161
162         if (JSReflow.curr_page >= JSReflow.pages) {
163             $('#next').attr("disabled", true);
164         } else if (JSReflow.curr_page <= 1) {
165             $('#prev').attr("disabled", true);
166         }
167
168         $('#pageno').html(JSReflow.curr_page);
169
170         $('#main').html("");
171
172         // Now we need to output the contents of the JSReflow.columns array
173         start = (JSReflow.curr_page - 1) * JSReflow.cols_per_page;
174         end = JSReflow.curr_page * JSReflow.cols_per_page;
175
176         for (p = start; p < JSReflow.columns.length && p < end; p++) {
177             if (!Array.isArray(JSReflow.columns[p])) {
178                 continue;
179             }
180
181             for (l = 0; l < JSReflow.columns[p].length; l++) {
182                 if (JSReflow.columns[p][l] === undefined || JSReflow.columns
183 [p][l] === "") {
184                     continue;
185                 }
186
187                 JSReflow.outputDiv("container", ((JSReflow.col_sep - JSReflow
w.galley_width * JSReflow.scale) / 2 + (p - start) * JSReflow.col_sep), (JSR
eflow.top_margin + 1 * JSReflow.vs) * JSReflow.scale, JSReflow.columns[p][l]
);
188             }
189         }
190         //console.log(JSReflow.columns);
191     }
192
193     JSReflow.computePagination();
194 };
195
196 JSReflow.computePagination = function () {
197     "use strict";
198     var p, l, w, h, scale, span, nlines, currcol, currrow, prev_col;
199
200     for (p = JSReflow.current_paragraph; p < JSReflow.paragraph_tree.length;
p++) {
201
202         if (Array.isArray(JSReflow.paragraph_tree[p])) {
203             // We assume it's a normal paragraph
204             if (JSReflow.paragraph_tree[p][JSReflow.min_badness_index] === u
ndefined) {

```

```

205         continue; // hack due to extra empty paragraph generated by
the Java/C combo
206     }
207     for (l = JSReflow.current_line; l < JSReflow.paragraph_tree[p][JSReflow.min_badness_index].length; l++) {
208         prev_col = JSReflow.curr_col;
209         JSReflow.nextPos();
210         if (Math.floor(JSReflow.curr_col / JSReflow.cols_per_page) >
Math.floor(prev_col / JSReflow.cols_per_page)) {
211             // Reached a page boundary
212             JSReflow.current_paragraph = p;
213             JSReflow.current_line = l;
214             //console.log(JSReflow.current_paragraph, JSReflow.curre
nt_line)
215             return;
216         }
217         JSReflow.columns[JSReflow.curr_col][JSReflow.curr_row] = JSReflow.getLine(JSReflow.paragraph_tree[p][JSReflow.min_badness_index][l]);
218     }
219     JSReflow.nextPos();
220     if (JSReflow.curr_row !== 0) { // Suppress blank line at top of
column
221         JSReflow.columns[JSReflow.curr_col][JSReflow.curr_row] = "";
222     }
223     } else {
224         // It must be a Float
225         if (JSReflow.float_type === JSReflow.float_types.NONE) {
226             continue;
227         }
228
229         // Scale width to fit column (and height proportionately)
230         w = JSReflow.paragraph_tree[p].w * JSReflow.scale;
231         h = JSReflow.paragraph_tree[p].h * JSReflow.scale;
232
233         // Have a look at the width and see if it could span any columns
234         span = Math.round(w / JSReflow.galley_width);
235
236         if (span > JSReflow.cols_per_page) {
237             span = JSReflow.cols_per_page;
238         }
239
240         if (span < 1 || !JSReflow.multi_column) {
241             span = 1;
242         }
243
244         scale = (JSReflow.col_sep * span) - JSReflow.col_sep + JSReflow.galley_width * JSReflow.scale;
245         scale = scale / w / JSReflow.scale; // (span * JSReflow.galley_width + (span - 1) * (JSReflow.col_sep - JSReflow.galley_width)) / w;
246
247         w *= scale;
248         h *= scale;
249
250         // Is it too big to fit? If so scale down the height
251         if (h > JSReflow.pageHeight) {
252             scale = JSReflow.pageHeight / h;
253             w *= scale;
254             h *= scale;
255         }
256
257         nlines = Math.round(h / JSReflow.vs) + 1;
258
259         if (JSReflow.float_type === JSReflow.float_types.SIMPLEQUEUE) {
260             // Don't actually need a queue in gridlayout
261             JSReflow.nextPos();
262             curr_col = JSReflow.curr_col;
263             curr_row = JSReflow.curr_row;
264
265             // check if it'll actually fit in the column, and if not, move to a new one
266             while (!JSReflow.checkSpace(nlines, span)) {
267                 JSReflow.curr_col++;
268                 JSReflow.curr_row = 0;

```

```

269         JSReflow.nextPos();
270     }
271
272     JSReflow.placeFloat(w, h, nlines, span, "<div>" + JSReflow.p
aragraph_tree[p].d + "</div>");
273
274     JSReflow.curr_col = currcol;
275     JSReflow.curr_row = currrow;
276
277     } else if (JSReflow.float_type === JSReflow.float_types.DUMB) {
278
279         // Just stick the figure out wherever
280         JSReflow.nextPos();
281
282         JSReflow.placeFloat(w, h, nlines, span, "<div>" + JSReflow.p
aragraph_tree[p].d + "</div>");
283
284     } else if (JSReflow.float_type === JSReflow.float_types.MSWORD)
{
285         JSReflow.nextPos();
286         // check if it'll actually fit in the column, and if not, mo
ve to a new one
287         while (!JSReflow.checkSpace(nlines, span)) {
288             JSReflow.curr_col++;
289             JSReflow.curr_row = 0;
290             JSReflow.nextPos();
291         }
292
293         JSReflow.placeFloat(w, h, nlines, span, "<div>" + JSReflow.p
aragraph_tree[p].d + "</div>");
294
295     }
296     /*else {
297         // Don't output anything.
298         // console.log(JSReflow.float_type);
299     }*/
300 }
301 JSReflow.current_line = 0;
302 JSReflow.current_paragraph++;
303 }
304 //console.log(JSReflow.paragraph_tree);
305
306 };
307
308 JSReflow.pageInit = function () {
309     "use strict";
310     var i, nc, ex_ws, rq_ws, dropdown, badness, name;
311
312     JSReflow.current_paragraph = 0;
313     JSReflow.current_line = 0;
314
315     JSReflow.curr_row = 0;
316     JSReflow.curr_col = 0;
317
318
319     $('body').css('font-size', (JSReflow.ps * JSReflow.scale) + 'pt');
320
321     JSReflow.pageWidth = window.innerWidth * JSReflow.pt_per_px;
322     badness = [];
323     JSReflow.min_badness_index = 0;
324
325     JSReflow.pageHeight = window.innerHeight * JSReflow.pt_per_px - (JSReflo
w.bot_margin + JSReflow.top_margin) * JSReflow.scale;
326
327     for (i = 0; i < JSReflow.galley_widths.length; i++) {
328         nc = Math.floor(JSReflow.pageWidth / ((JSReflow.galley_widths[i] + J
SReflow.min_gutter) * JSReflow.scale));
329         ex_ws = JSReflow.pageWidth % ((JSReflow.galley_widths[i] + JSReflow.
min_gutter) * JSReflow.scale);
330         rq_ws = nc * JSReflow.min_gutter * JSReflow.scale;
331
332         badness.push((ex_ws /*+ rq_ws*/ + 100) * Math.sqrt(nc));
333         if (nc === 0) {

```

```

334         badness[i] = 1000 * i;
335     }
336
337     if (badness[i] <= badness[JSReflow.min_badness_index]) {
338         JSReflow.min_badness_index = i;
339         JSReflow.cols_per_page = nc;
340     }
341 }
342
343 JSReflow.galley_width = JSReflow.galley_widths[JSReflow.min_badness_index];
344
345 JSReflow.col_sep = Math.floor(JSReflow.pageWidth / JSReflow.cols_per_page);
346
347 /*
348     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.
349
350     So, what goes into the JSReflow.columns array?
351     - There must be one element for every column in the document
352     - These can be .push()ed on the fly
353     - These are themselves arrays with one element per multiple of the main leading (JSReflow.vs in this case) that will fit in the column
354     - If an element is undefined -- we assume it's available
355     - Otherwise the element will be a string containing the html to output (possibly the empty string)
356     - the html should specify its own dimensions, but not its position (this will be calculated by its position in the JSReflow.columns array)
357
358     Yep, that sounds plausible.
359
360 */
361
362 JSReflow.columns = [];
363
364 // How many lines? We have (window.innerHeight - JSReflow.top_margin - JSReflow.bot_margin) total space, and each line is JSReflow.vs high. So:
365 JSReflow.num_rows = Math.floor((window.innerHeight * JSReflow.pt_per_px - (JSReflow.top_margin + JSReflow.bot_margin) * JSReflow.scale) / (JSReflow.vs * JSReflow.scale));
366
367 if (JSReflow.num_rows < 1) {
368     JSReflow.num_rows = 1;
369 }
370
371 JSReflow.pages = 100; // Math.ceil(JSReflow.columns.length / JSReflow.cols_per_page);
372 JSReflow.curr_page = 1;
373
374 $('#main').empty(); // clear old stuff
375
376 dropdown = '<select id="dropdown" onchange="JSReflow.updateFloatType(this)">\n';
377 for (name in JSReflow.float_types) {
378     if (JSReflow.float_types.hasOwnProperty(name)) {
379         dropdown += '<option value="' + JSReflow.float_types[name] + '"' + (JSReflow.float_type === JSReflow.float_types[name] ? " selected" : "") + ">" + name + '</option>';
380     }
381 }
382 dropdown += "</select>\n";
383
384 $('#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>');
385
386     JSReflow.computePagination();
387     JSReflow.paginate(0);
388
389 };
390
391 JSReflow.updateFloatType = function (dd) {
392     "use strict";
393     JSReflow.float_type = parseInt(dd.options[dd.selectedIndex].value, 10);
394     JSReflow.pageInit();
395 };
396
397 JSReflow.toggleMulti = function () {
398     "use strict";
399     JSReflow.multi_column = !JSReflow.multi_column;
400     $('#multi').prop('checked', JSReflow.multi_column);
401     JSReflow.pageInit();
402 };
403
404 JSReflow.scaleUp = function (factor) {
405     "use strict";
406     if (JSReflow.galley_widths[0] + JSReflow.min_gutter < JSReflow.page_widt
h) {
407         JSReflow.scale *= factor;
408     }
409 };
410
411 JSReflow.scaleDown = function (factor) {
412     "use strict";
413     JSReflow.scale /= factor;
414 };
415
416
417 $(window).resize(function () {
418     "use strict";
419     clearTimeout(JSReflow.resizetimer);
420     JSReflow.resizetimer = setTimeout(JSReflow.pageInit, 200);
421     JSReflow.pageInit();
422 });
423
424 $(document).ready(function () {
425     "use strict";
426     JSReflow.pageInit();
427     $('#top').hide();
428 });
429
430 $("body").touchwipe({
431     wipeLeft: function () {
432         "use strict";
433         JSReflow.paginate(1);
434     },
435     wipeRight: function () {
436         "use strict";
437         JSReflow.paginate(-1);
438     },
439     wipeUp: function () {
440         "use strict";
441         JSReflow.scale *= 1.1;
442         JSReflow.pageInit();
443     },
444     wipeDown: function () {
445         "use strict";
446         JSReflow.scale /= 1.1;
447         JSReflow.pageInit();
448     },
449     min_move_x: 10,
450     min_move_y: 10,
451     preventDefaultEvents: true

```

```
452 });
453
454 $(document).keydown(function (e) {
455     "use strict";
456     var key = e.which;
457
458     // stop page from scrolling with arrow keys
459     //noinspection FallthroughInSwitchStatementJS
460     switch (key) {
461         case 37:
462         case 38:
463         case 39:
464         case 40:
465             e.preventDefault();
466     }
467     //alert(key);
468     if (key === 37) { // left
469         JSReflow.paginate(-1);
470     } else if (key === 38) { //up
471         JSReflow.scale *= 1.01;
472         JSReflow.pageInit();
473     } else if (key === 39) { //right
474         JSReflow.paginate(1);
475     } else if (key === 40) { //down
476         JSReflow.scale /= 1.01;
477         JSReflow.pageInit();
478     } else if (key === 77) { // m ("multicolumn")
479         JSReflow.toggleMulti();
480     } else if (key === 72) { // h ("hide bar")
481         $('#top').toggle();
482     }
483 });
```