

```

1  /**
2   * [Typography]
3   *
4   * Description:
5   + A JavaScript object to automatically set the typography of your webpage based on viewing device dimensions.
6   + Does 3 things: Sets ideal font size, sets ideal line lengths and leading, and applies orphan control.
7   * Version: 4.0
8   * By: Kevin Montgomery
9
10  *
11  * What's New?
12  + No longer requires JQuery. Added orphan control.
13  + Uses class properties instead of inline styles.
14  + Can be applied globally as a body class, or as individual element classes.
15  *
16  */
17
18  var typography_obj = new function() {
19
20    /* [Parameters] */
21    this.parameters = {
22
23      /* [Field of view] */
24      field_of_view : 30, // About 30 degree fov monitor viewing.
25
26      /* [Fovial vision] */
27      fovial_vision : 1.5, // fovial vision, in degrees, for 4-5 letters. Usually 1-2 degrees.
28
29      /* [Placeholder for field of view ratio] */
30      field_of_view_ratio : "",
31
32      /* [Placeholder for diameter of 1 lowercase letter] */
33      fovial_vision_pixels : "",
34      letter_diameter_pixels : "",
35
36      /* [Ratio placeholders] */
37      font_height_ratio : "",
38      golden_ratio : 1.618
39    };
40
41    /* [User Configuration Detection] */
42    this.user_configuration = {
43
44      /* [Capture screen resolution, use average for diameter] */
45      screen_width : screen.width,
46      screen_height : screen.height,
47      screen_avg : "",
48      screen_min : "",
49
50      /* [Establish parameters for users font size] */
51      font_height_pixels : "",
52      ex_height_pixels : ""
53    };
54
55    /* [Outputs] */
56    this.outputs = {
57
58      /* [Values that will be used in the resulting CSS] */
59      font_height_pixels : "",
60      font_height_em : "",
61      ex_height_diameter_pixels : "",
62      line_height_pixels : "",
63      line_height_em : "",
64      line_space_before_pixels : "",
65      line_space_before_em : ""
66    };
67

```

```

68     line_space_after_pixels : "",
69     line_space_after_em : "",
70     line_length_em : "",
71     line_length_pixels : ""
72
73 };
74
75 /* [Initialize the object] */
76 this.init = function() {
77
78     /* [Create element to evaluate users font size] */
79     var user_font_height_obj = document.createElement("div");
80     user_font_height_obj.setAttribute("id", "usr_font_height_obj");
81     user_font_height_obj.setAttribute("style", "line-height: 100%, padding: 0, margin: 0");
82     user_font_height_obj.innerHTML = "&nbsp;";
83     document.getElementsByTagName("body")[0].insertBefore(user_font_height_obj, document.getElementsByTagName("body")[0].
childNodes[0]);
84
85     /* [Calculate users font size] */
86     typography_obj.user_configuration.font_height_pixels = Math.min(
87         Math.min(user_font_height_obj.scrollHeight, user_font_height_obj.offsetHeight),
88         Math.min(user_font_height_obj.offsetHeight, user_font_height_obj.offsetHeight),
89         Math.min(user_font_height_obj.clientHeight, user_font_height_obj.clientHeight)
90     );
91
92     /* [Reduce element font size to calculate 1ex] */
93     user_font_height_obj.setAttribute("style", "font-size: 1ex;");
94     typography_obj.user_configuration.ex_height_pixels = Math.min(
95         Math.min(user_font_height_obj.scrollHeight, user_font_height_obj.offsetHeight),
96         Math.min(user_font_height_obj.offsetHeight, user_font_height_obj.offsetHeight),
97         Math.min(user_font_height_obj.clientHeight, user_font_height_obj.clientHeight)
98     );
99
100    /* [Remove font testing object] */
101    document.getElementsByTagName("body")[0].removeChild(user_font_height_obj);
102
103    /* [User configuration calculations] */
104    typography_obj.user_configuration.screen_avg = (typography_obj.user_configuration.screen_width + typography_obj.user_conf
uration.screen_height) / 2;
105    typography_obj.user_configuration.screen_min = Math.min(typography_obj.user_configuration.screen_width, typography_obj.user
_configuration.screen_height);
106
107    /* [Parameter calculations] */
108
109    /**
110     * Don't forget, we need to compare LOWERCASE letters, not the entire font height.
111     */
112    typography_obj.parameters.font_height_ratio = Math.round((typography_obj.user_configuration.ex_height_pixels / typography_o
bj.user_configuration.font_height_pixels) * 1000) / 1000;
113    typography_obj.parameters.field_of_view_ratio = typography_obj.parameters.foveal_vision / typography_obj.parameters.field_of_
view; // Calculate ratio of foveal vision to field of view
114    typography_obj.parameters.foveal_vision_pixels = Math.round(typography_obj.user_configuration.screen_avg * typography_obj.p
arameters.field_of_view_ratio); // Foveal diameter in pixels, for 4 letters
115    typography_obj.parameters.letter_diameter_pixels = Math.round((typography_obj.parameters.foveal_vision_pixels / 4.5) / typogra
phy_obj.parameters.font_height_ratio); // Diameter of 1 uppercase letter
116
117    /* [Outputs calculations] */
118    typography_obj.outputs.font_height_pixels = Math.max(typography_obj.parameters.letter_diameter_pixels, typography_obj.user_
configuration.font_height_pixels); // Apply the larger of the users setting, or the new setting.
119    typography_obj.outputs.font_height_em = Math.round((typography_obj.outputs.font_height_pixels / typography_obj.user_configur
ation.font_height_pixels) * 1000) / 1000; // Calculate height using em measure.
120    typography_obj.outputs.ex_height_diameter_pixels = Math.round(typography_obj.outputs.font_height_pixels * typography_obj.pa
rameters.font_height_ratio);
121
122    typography_obj.outputs.line_height_pixels = Math.round(typography_obj.outputs.font_height_pixels + typography_obj.outputs.ex
_height_diameter_pixels);
123    typography_obj.outputs.line_height_em = Math.round((typography_obj.outputs.line_height_pixels / typography_obj.outputs.font_h

```

```

eight_pixels) * 1000) / 1000;
124
125 typography_obj.outputs.line_space_before_pixels = typography_obj.outputs.ex_height_diameter_pixels;
126 typography_obj.outputs.line_space_before_em = Math.round((typography_obj.outputs.line_space_before_pixels / typography_obj.
j.outputs.font_height_pixels) * 1000) / 1000;
127 typography_obj.outputs.line_space_after_pixels = typography_obj.outputs.ex_height_diameter_pixels;
128 typography_obj.outputs.line_space_after_em = Math.round((typography_obj.outputs.line_space_after_pixels / typography_obj.ou
tputs.font_height_pixels) * 1000) / 1000;
129 typography_obj.outputs.line_length_em = Math.max(39, typography_obj.outputs.font_height_pixels * 2); // Calculate ideal line len
gth, compare the 'alphabet-and-a-half rule' to the 'points-times-two' rule
130 typography_obj.outputs.line_length_pixels = typography_obj.outputs.line_length_em * typography_obj.outputs.font_height_pixels;
131
132 /*
133  * Apply style properties to "typography" class
134  * Locate all elements with applicable "typography" class and adjust them.
135  */
136
137 typography_obj.quotes_control();
138 typography_obj.orphans_control();
139 typography_obj.lines_control();
140
141 };
142
143 this.lines_control = function() {
144
145     var typography_css = document.createElement('style');
146     typography_css.setAttribute("type", "text/css");
147
148     var typography_body_size = "html.typography { "+
149     "font-size: "+typography_obj.outputs.font_height_em+"em; "+
150     "} ";
151
152     var typography_margins = "html .typography.lines, "+
153     "html.typography.lines h1, "+
154     "html.typography.lines h2, "+
155     "html.typography.lines h3, "+
156     "html.typography.lines h4, "+
157     "html.typography.lines h5, "+
158     "html.typography.lines h6, "+
159     "html.typography.lines p, " +
160     "html.typography.lines td, " +
161     "html.typography.lines li " +
162     "{ "+
163     "line-height: "+typography_obj.outputs.line_height_em+"em; "+
164     "max-width: "+typography_obj.outputs.line_length_em+"rem; "+
165     "margin-top: "+typography_obj.outputs.line_space_before_em+"rem; "+
166     "margin-bottom: "+typography_obj.outputs.line_space_after_em+"rem; "+
167     "margin-left: auto; "+
168     "margin-right: auto; "+
169     "} ";
170
171     if (typography_css.styleSheet) {
172         typography_css.styleSheet.cssText = typography_body_size + typography_margins;
173     } else {
174         typography_css.appendChild (document.createTextNode(typography_body_size));
175         typography_css.appendChild(document.createTextNode(typography_margins));
176     }
177
178     document.getElementsByTagName ("head")[0].appendChild(typography_css);
179
180 };
181
182 this.orphans_control = function() {
183
184     /* [Locate elements for typesetter] */
185     var typesetter_elements = document.querySelectorAll("html .typography.orphans, "+
186     "html.typography.orphans h1, "+

```

```

187 "html.typography.orphans h2, "+
188 "html.typography.orphans h3, "+
189 "html.typography.orphans h4, "+
190 "html.typography.orphans h5, "+
191 "html.typography.orphans h6, "+
192 "html.typography.orphans p," +
193 "html.typography.orphans td," +
194 "html.typography.orphans li");
195
196 var punctuation = new Array("!", ".", ",", "?", ":", ";");
197
198 for (var i = 0; i < typesetter_elements.length; i++) {
199
200 /* [Apply orphan control] */
201 var line = typesetter_elements[i].innerHTML;
202 var word_array = line.split(" ");
203 var word_cound = word_array.length;
204
205 for (var ii = 0; ii < word_array.length - 2; ii++) { // Look at each word for punctuation
206
207     var word = word_array[ii];
208     var evaluate_punctuation = word.substring(word.length - 1, word.length);
209
210     /*
211      * If punctuation is found, apply non-breaking spaces to the preceding and following words.
212      */
213
214     if (ii > 0 && punctuation.indexOf(evaluate_punctuation) > 0) {
215         var preceding_words = new Array(word_array[ii - 1], word);
216         var preceding_words_join = preceding_words.join("&nbsp;");
217         word_array.splice(ii - 1, 2, preceding_words_join);
218         ii--;
219
220         var following_words = new Array(word_array[ii + 1], word_array[ii + 2]);
221         var following_words_join = following_words.join("&nbsp;");
222         word_array.splice(ii + 1, 2, following_words_join);
223
224     }
225
226 }
227
228 var last_words = word_array.splice(-2);
229 var last_words_join = last_words.join("&nbsp;");
230 word_array.push(last_words_join);
231
232 line = word_array.join(" ");
233 typesetter_elements[i].innerHTML = line;
234
235 }
236
237 };
238
239 this.quotes_control = function() {
240
241 /* [Locate elements for typesetter] */
242 var typesetter_elements = document.querySelectorAll("html .typography.quotes, "+
243 "html.typography.quotes h1, "+
244 "html.typography.quotes h2, "+
245 "html.typography.quotes h3, "+
246 "html.typography.quotes h4, "+
247 "html.typography.quotes h5, "+
248 "html.typography.quotes h6, "+
249 "html.typography.quotes p," +
250 "html.typography.quotes td," +
251 "html.typography.quotes li");
252
253 var punctuation = new Array("!", ".", ",", "?", ":", ";"); // Need to scan for quotes placed before punctuation

```

```

254 for (var i = 0; i < typesetter_elements.length; i++) {
255
256
257 /* [Apply orphan control] */
258 var line = typesetter_elements[i].innerHTML;
259 var word_array = line.split(" ");
260 var word_cound = word_array.length;
261
262 for (var ii = 0; ii < word_array.length - 2; ii++) { // Look at each word for punctuation
263
264     var word = word_array[ii];
265
266     var has_punctuation = "";
267     var evaluate_punctuation = word.substring(word.length - 1, word.length);
268
269     /* [If punctuation is found] */
270     if (ii > 0 && punctuation.indexOf(evaluate_punctuation) > 0) {
271         var evaluate_close_quotes = word.substring(word.length - 2, word.length - 1);
272         has_punctuation = evaluate_punctuation;
273     } else {
274         var evaluate_close_quotes = word.substring(word.length - 1, word.length);
275     }
276
277     var evaluate_open_quotes = word.substring(0, 1);
278     var evaluate_apostrophes = word.substring(1, word.length - 1);
279
280     /* [Apostrophes] */
281     if (evaluate_apostrophes.indexOf("'")) {
282         word = word.replace("'", "&rsquo;");
283         word_array[ii] = word;
284     }
285
286     /* [Double Quotes] */
287     if (evaluate_open_quotes == "" && evaluate_close_quotes == "") {
288         if (has_punctuation.length > 0) {
289             word = "&ldquo;" + word.substring(1, word.length - 2) + "&rdquo;" + has_punctuation;
290         } else {
291             word = "&ldquo;" + word.substring(1, word.length - 1) + "&rdquo;";
292         }
293         word_array [ii] = word;
294     }
295
296     /* [Single Quotes] */
297     else if (evaluate_open_quotes == "" && evaluate_close_quotes == "") {
298         if (has_punctuation.length > 0) {
299             word = "&lsquo;" + word.substring(1, word.length - 2) + "&rsquo;" + has_punctuation;
300         } else {
301             word = "&lsquo;" + word.substring(1, word.length - 1) + "&rsquo;";
302         }
303         word_array [ii] = word;
304     }
305
306 }
307
308 line = word_array.join(" ");
309 typesetter_elements[i].innerHTML = line;
310
311 }
312
313 };
314
315 };

```