```
/**
 * [Typography]
 *
 * Description:
 + A JavaScript object to automatically set the typography of your webpage based on viewing device dimensions.
 + Does 3 things: Sets ideal font size, sets ideal line lengths and leading, and applies orphan control.
 * Version: 4.0
 * By: Kevin Montgomery

 *
 * What's New?
 + No longer requires JQuery. Added orphan control.
 + Uses class properties instead of inline styles.
 + Can be applied globally as a body class, or as individual element classes.
 *
 */

var typography_obj = new function() {

  /* [Parameters] */
  this.parameters = {

    /* [Field of view] */
    field_of_view  : 30, // About 30 degree fov monitor viewing.

    /* [Fovial vision] */
    fovial_vision  : 1.5, // fovial vision, in degrees, for 4-5 letters. Usually 1-2 degrees.

    /* [Placeholder for field of view ratio] */
    field_of_view_ratio   : "" ,

    /* [Placeholder for diameter of 1 lowercase letter] */
    fovial_vision_pixels   : "" ,
    letter_diameter_pixels   : "" ,

    /* [Ratio placeholders] */
    font_height_ratio   : "" ,
    golden_ratio  : 1.618

  };

  /* [User Configuration Detection] */
  this.user_configuration = {

    /* [Capture screen resolution, use average for diameter] */
    screen_width  : screen .width,
    screen_height  : screen .height,
    screen_avg : "" ,
    screen_min  : "" ,

    /* [Establish parameters for users font size] */
    font_height_pixels   : "" ,
    ex_height_pixels  : ""

  };

  /* [Outputs] */
  this.outputs = {

    /* [Values that will be used in the resulting CSS] */
    font_height_pixels   : "" ,
    font_height_em  : "" ,
    ex_height_diameter_pixels   : "" ,
    line_height_pixels  : "" ,
    line_height_em  : "" ,
    line_space_before_pixels   : "" ,
    line_space_before_em  : "" ,
    line_space_after_pixels  : "" ,
    line_space_after_em  : "" ,
    line_length_em  : "" ,
    line_length_pixels  : ""

  };

  /* [Initialize the object] */
  this.init = function() {

    /* [Create element to evaluate users font size] */
    var user_font_height_obj = document.createElement("div");
    user_font_height_obj.setAttribute("id", "usr_font_height_obj");
    user_font_height_obj.setAttribute("style", "line-height: 100%, padding: 0, margin: 0");
    user_font_height_obj.innerHTML = "&nbsp;";
    document.getElementsByTagName("body")[0].insertBefore(user_font_height_obj, document.getElementsByTagName("body")[0].
childNodes[0]);

    /* [Calculate users font size] */
    typography_obj.user_configuration.font_height_pixels = Math.min(
      Math.min (user_font_height_obj.scrollHeight, user_font_height_obj.scrollHeight),
      Math.min(user_font_height_obj.offsetHeight, user_font_height_obj.offsetHeight),
      Math.min(user_font_height_obj.clientHeight, user_font_height_obj.clientHeight)
      );

      /* [Reduce element font size to calculate 1ex] */
      user_font_height_obj.setAttribute("style", "font-size: 1ex;");
      typography_obj.user_configuration.ex_height_pixels = Math.min(
      Math.min (user_font_height_obj.scrollHeight, user_font_height_obj.scrollHeight),
      Math.min(user_font_height_obj.offsetHeight, user_font_height_obj.offsetHeight),
      Math.min(user_font_height_obj.clientHeight, user_font_height_obj.clientHeight)
      );

      /* [Remove font testing object] */
      document.getElementsByTagName("body")[0].removeChild(user_font_height_obj);

      /* [User configuration calculations] */
      typography_obj.user_configuration.screen_avg = (typography_obj.user_configuration.screen_width + typography_obj.user_confi
guration.screen_height) / 2;
    typography_obj.user_configuration.screen_min = Math.min(typography_obj.user_configuration.screen_width, typography_obj.user
_configuration.screen_height);

      /* [Parameter calculations] */

      /**
       * Don't forget, we need to compare LOWERCASE letters, not the entire font height.
       **/
      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;
      typography_obj.parameters.field_of_view_ratio = typography_obj.parameters.fovial_vision / typography_obj.parameters.field_of_
view; // Calculate ratio of fovial vision to field of view
      typography_obj.parameters.fovial_vision_pixels = Math.round(typography_obj.user_configuration.screen_avg * typography_obj.p
arameters.field_of_view_ratio); // Fovial diameter in pixels, for 4 letters
      typography_obj.parameters.letter_diameter_pixels = Math.round((typography_obj.parameters.fovial_vision_pixels / 4.5) / typogra
phy_obj.parameters.font_height_ratio); // Diameter of 1 uppercase letter

      /* [Outputs calculations] */
      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.
      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.
      typography_obj.outputs.ex_height_diameter_pixels = Math.round(typography_obj.outputs.font_height_pixels * typography_obj.pa
rameters.font_height_ratio);

      typography_obj.outputs.line_height_pixels = Math.round(typography_obj.outputs.font_height_pixels + typography_obj.outputs.ex
_height_diameter_pixels);
    typography_obj.outputs.line_height_em = Math.round((typography_obj.outputs.line_height_pixels / typography_obj.outputs.font_h
eight_pixels) * 1000) / 1000;

      typography_obj.outputs.line_space_before_pixels =  typography_obj.outputs.ex_height_diameter_pixels;
      typography_obj.outputs.line_space_before_em = Math.round((typography_obj.outputs.line_space_before_pixels / typography_obj.outputs.font_height_pixels) * 1000) / 1000;
      typography_obj.outputs.line_space_after_pixels =  typography_obj.outputs.ex_height_diameter_pixels;
      typography_obj.outputs.line_space_after_em = Math.round((typography_obj.outputs.line_space_after_pixels / typography_obj.outputs.font_height_pixels) * 1000) / 1000;
      typography_obj.outputs.line_length_em = Math.max(39, typography_obj.outputs.font_height_pixels * 2); // Calculate ideal line length, compare the 'alphabet-and-a-half rule' to the 'points-times-two' rule
      typography_obj.outputs.line_length_pixels = typography_obj.outputs.line_length_em * typography_obj.outputs.font_height_pixels;

      /*
       * Apply style properties to "typography" class
       * Locate all elements with applicable "typography" class and adjust them.
       */

      typography_obj.quotes_control();
      typography_obj.orphans_control();
    typography_obj.lines_control();

  };

  this.lines_control = function() {

    var typography_css = document.createElement('style');
    typography_css.setAttribute("type", "text/css");

    var typography_body_size = "html.typography { "+
    "font-size: "+typography_obj.outputs.font_height_em+"em; "+
    "} ";

    var typography_margins = "html .typography.lines, "+
         "html.typography.lines h1, "+
         "html.typography.lines h2, "+
         "html.typography.lines h3, "+
         "html.typography.lines h4, "+
         "html.typography.lines h5, "+
         "html.typography.lines h6, "+
         "html.typography.lines p, " +
         "html.typography.lines td, " +
         "html.typography.lines li " +
    "{ "+
    "line-height: "+typography_obj.outputs.line_height_em+"em; "+
    "max-width: "+typography_obj.outputs.line_length_em+"rem; "+
    "margin-top: "+typography_obj.outputs.line_space_before_em+"rem; "+
    "margin-bottom: "+typography_obj.outputs.line_space_after_em+"rem; "+
    "margin-left: auto; "+
    "margin-right: auto; "+
    "} ";

    if (typography_css.styleSheet) {
      typography_css.styleSheet.cssText     = typography_body_size    + typography_margins;
    } else {
      typography_css.appendChild    (document.createTextNode(typography_body_size));
      typography_css.appendChild(document.createTextNode(typography_margins));
    }

    document .getElementsByTagName  ("head")[0].appendChild(typography_css);

  };

  this.orphans_control = function() {

      /* [Locate elements for typesetter] */
      var typesetter_elements = document.querySelectorAll("html .typography.orphans, "+
         "html.typography.orphans h1, "+
         "html.typography.orphans h2, "+
         "html.typography.orphans h3, "+
         "html.typography.orphans h4, "+
         "html.typography.orphans h5, "+
         "html.typography.orphans h6, "+
         "html.typography.orphans p," +
         "html.typography.orphans td," +
         "html.typography.orphans li");

       var punctuation = new Array("!", ".", ",", "?", ":", ";");

      for (var i = 0; i < typesetter_elements.length; i++) {

      /* [Apply orphan control] */
         var line = typesetter_elements[i].innerHTML;
         var word_array = line.split(" ");
         var word_cound = word_array.length;

         for (var ii = 0; ii < word_array.length - 2; ii++) { // Look at each word for punctuation

           var word = word_array[ii];
         var evaluate_punctuation = word.substring(word.length - 1, word.length);

           /*
            * If punctuation is found, apply non-breaking spaces to the preceding and following words.
            */

           if (ii > 0 && punctuation.indexOf(evaluate_punctuation) > 0) {
             var preceding_words =  new Array(word_array[ii - 1], word);
             var preceding_words_join = preceding_words.join("&nbsp;");
             word_array.splice(ii - 1, 2, preceding_words_join);
             ii--;

             var following_words =  new Array(word_array[ii + 1], word_array[ii + 2]);
             var following_words_join = following_words.join("&nbsp;");
             word_array.splice(ii + 1, 2, following_words_join);

           }

         }

         var last_words =  word_array.splice(-2);
         var last_words_join = last_words.join("&nbsp;");
      word_array.push(last_words_join);

      line = word_array.join(" ");
      typesetter_elements[i].innerHTML = line;

    }

  };

  this.quotes_control = function() {

      /* [Locate elements for typesetter] */
      var typesetter_elements = document.querySelectorAll("html .typography.quotes, "+
         "html.typography.quotes h1, "+
         "html.typography.quotes h2, "+
         "html.typography.quotes h3, "+
         "html.typography.quotes h4, "+
         "html.typography.quotes h5, "+
         "html.typography.quotes h6, "+
         "html.typography.quotes p," +
         "html.typography.quotes td," +
         "html.typography.quotes li");

      var punctuation = new Array("!", ".", ",", "?", ":", ";"); // Need to scan for quotes placed before punctuation
```

```

      for (var i = 0; i < typesetter_elements.length; i++) {

      /* [Apply orphan control] */
        var line = typesetter_elements[i].innerHTML;
        var word_array = line.split(" ");
        var word_cound = word_array.length;

        for (var ii = 0; ii < word_array.length - 2; ii++) { // Look at each word for punctuation

          var word = word_array[ii];

          var has_punctuation = "";
          var evaluate_punctuation = word.substring(word.length - 1, word.length);

          /* [If punctuation is found] */
          if (ii > 0 && punctuation.indexOf(evaluate_punctuation) > 0) {
            var evaluate_close_quotes = word.substring(word.length - 2, word.length - 1);
            has_punctuation = evaluate_punctuation;
          } else {
            var evaluate_close_quotes = word.substring(word.length - 1, word.length);
          }

          var evaluate_open_quotes = word.substring(0,1);
        var evaluate_apostrophes = word.substring(1, word.length - 1);

        /* [Apostrophes] */
        if (evaluate_apostrophes.indexOf("'")) {
          word  = word. replace ("'","&rsquo;");
          word_array[ii] = word;
        }

        /* [Double Quotes] */
        if (evaluate_open_quotes == '"' && evaluate_close_quotes == '"') {
          if (has_punctuation.length > 0) {
            word  = "&ldquo;"   + word. substring  (1, word.length - 2) + "&rdquo;" + has_punctuation;
          } else {
            word  = "&ldquo;"   + word. substring  (1, word.length - 1) + "&rdquo;";
          }
          word_array  [ii] = word;
        }

        /* [Single Quotes] */
        else if (evaluate_open_quotes == "'" && evaluate_close_quotes == "'") {
          if (has_punctuation.length > 0) {
            word  = "&lsquo;"   + word. substring  (1, word.length - 2) + "&rsquo;" + has_punctuation;
          } else {
            word  = "&lsquo;"   + word. substring  (1, word.length - 1) + "&rsquo;";
          }
          word_array  [ii] = word;
        }

        }

      line  = word_array. join (" ");
      typesetter_elements[i].innerHTML = line;

    }

  };

};
```