```
/**
* @fileOverview Products JavaScript Function Library.
* @author <a href="https://github.com/richardhenyash">Richard Ash</a>
* @version 1.1.1
*/
/*jshint esversion: 6 */
/* globals $, buildConfirmModal, buildInformationModal */

/* Set product rating stars for all divs with the class product-rating-stars */
/* Note - reads the product rating from the product-rating data attribute */
$('.product-rating-stars').each(function () {
    let productRating = $(this).data("product-rating");
    if ((productRating == "None") || (productRating == null) || (productRating == ""))
        $(this).html("<i class=rating-text>Not Rated</i>");
    else {
        let productRatingRounded = (Math.round(productRating));
        let stars = $(this).children();
        stars.each(function(si) {
            if (productRatingRounded > si) {
                $(this).addClass("fg-yellow");
            }
        });
    }
});

/* On click event handlers added to product review edit stars */
/* Note, updates the hidden form input rating with the selected rating */
$('.product-review-edit-stars').each(function () {
    let currentRating = $(this).data("product-rating");
    let stars = $(this).children();
    stars.each(function(si) {
        if (currentRating > si) {
            $(this).addClass("fg-yellow");
        }
        $(this).click(function() {
            let starId = $(this).attr('id');
            let rating = parseInt(starId.slice(-1));
            colourStars(rating, "#star-");
        });
    });
});

// On click event handler added to product image link to build modal dialog
$("#productInformationImgLink").click(function() {
    (buildInformationModal("#productInformationImgLink", "information-modal-title", "#prod
});

// On click event handler added to product information button to build modal dialog
$("#productInformationBtn").click(function() {
    (buildInformationModal("#productInformationBtn", "information-modal-title", "#product-
});

// On click event handler added to size information button to build modal dialog
$("#sizeInformationBtn").click(function() {
    (buildInformationModal("#sizeInformationBtn", "information-modal-title", "#size-inform
});

// On click event handler added to type information button to build modal dialog
$("#typeInformationBtn").click(function() {
    (buildInformationModal("#typeInformationBtn", "information-modal-title", "#type-inform
});

// On click event handler added to create plan button to build modal dialog
$("#createPlanBtn").click(function() {
    (buildInformationModal("#createPlanBtn", "information-modal-title", null, "#informatio
});


// On click event handler added to product delete button to build delete confirmation moda
$("#productDeleteBtn").click(function() {
    (buildConfirmModal("#productDeleteBtn", "#confirmModal"));
});
```

CONFIGURE

Metrics

There are 23 functions in this file.

Function with the largest signature take 6 arguments, while the median is 0.

Largest function has 16 statements in it, while the median is 2.

The most complex function has a cyclomatic complexity value of 7 while the median is 1.

```
73
74  // On click event handler added to review delete buttons to build delete confirmation moda
75  $('.reviewDeleteBtn').each(function () {
76      let btnId = "#" + $(this).attr("id");
77      $(btnId).click(function() {
78          (buildConfirmModal(btnId, "#confirmModal"));
79      });
80  });
81
82  // On click event handler added to minus button to decrease product quantity and update pr
83  $("#product-quantity-minus-btn").click(function() {
84      (incrementQuantity("#product-quantity", "#product-quantity-minus-btn", "#product-quant
85      (setPriceBasedOnSizeAndQuantity("#product-size", "#product-quantity", "#product-price-
86  });
87
88  // On click event handler added to plus button to decrease product quantity and update pri
89  $("#product-quantity-plus-btn").click(function() {
90      (incrementQuantity("#product-quantity", "#product-quantity-minus-btn", "#product-quant
91      (setPriceBasedOnSizeAndQuantity("#product-size", "#product-quantity", "#product-price-
92  });
93
94  // On change event handler added to size selector to update price
95  $("#product-size").change(function() {
96      (setPriceBasedOnSizeAndQuantity("#product-size", "#product-quantity", "#product-price-
97  });
98
99  // On change event handler added to size selector to update price
100 $('#id_size').change(function() {
101     setPriceBasedOnSize("#id_size :selected", "#product-price-dict", "#id_price");
102 });
103
104 // On change event handler added to image in custom clearable file input
105 // Displays new file name
106 $('#new_image').change(function() {
107     let file = $('#new_image')[0].files[0];
108     $('#filename_image').text(`Image will be set to: ${file.name}`);
109 });
110
111 // On change event handler added to alternate image in custom clearable file input
112 // Displays new file name
113 $('#new_image_alt').change(function() {
114     let file = $('#new_image_alt')[0].files[0];
115     $('#filename_image_alt').text(`Image will be set to: ${file.name}`);
116 });
117
118 /**
119 * [Function to add yellow colour class to stars]
120 * @return {[rating]}                    [rating, integer]
121 */
122 function colourStars(rating, starIdPrefix) {
123
124     // Remove yellow class from all rating stars
125     for (let i = 1; i <= 5; i++) {
126         $(starIdPrefix + i).removeClass("fg-yellow");
127     }
128     // Add yellow class to correct rating stars
129     for (let i = 1; i <= rating; i++) {
130         $(starIdPrefix + i).addClass("fg-yellow");
131     }
132     $('input[name=rating]').val(rating);
133     return rating;
134 }
135
136 /**
137 * [Function to increment product quantities given quantityId, positive or negative increme
138 * @return {[newQuantity]}                    [New Quantity, string]
139 */
140 function incrementQuantity(quantityId, btnMinusID, btnPlusID, inc, minValue, maxValue){
141     let currentQuantity = parseInt($(quantityId).val());
142     let newQuantity = currentQuantity;
143     if (Math.sign(inc) == 1) {
144         if ((currentQuantity + inc) <= maxValue) {
145             newQuantity = currentQuantity + inc;
146         }
```

CONFIGURE

Metrics

There are 23 functions in this file.

Function with the largest signature take 6 arguments, while the median is 0.

Largest function has 16 statements in it, while the median is 2.

The most complex function has a cyclomatic complexity value of 7 while the median is 1.

```
147       } else {
148           if ((currentQuantity + inc) >= minValue) {
149               newQuantity = currentQuantity + inc;
150           }
151       }
152       if (newQuantity != currentQuantity) {
153           $(quantityId).val(newQuantity);
154       }
155       if (newQuantity <= minValue) {
156           $(btnMinusID).attr("disabled", true);
157       } else {
158           $(btnMinusID).removeAttr('disabled');
159       }
160       if (newQuantity >= maxValue) {
161           $(btnPlusID).attr("disabled", true);
162       } else {
163           $(btnPlusID).removeAttr('disabled');
164       }
165       return newQuantity;
166   }
167
168   /**
169   * [Function to set price based on size selected]
170   * @return {[priceStr]}                    [price, string]
171   */
172   function setPriceBasedOnSize(sizeId, scriptId, priceId) {
173       let size = $(sizeId).text();
174       let priceDict = JSON.parse($(scriptId).text());
175       let priceStr = priceDict[size];
176       $(priceId).val(priceStr);
177       return priceStr;
178   }
179
180   /**
181   * [Function to set price based on size and quantity selected]
182   * @return {[priceStr]}                    [price, string]
183   */
184   function setPriceBasedOnSizeAndQuantity(sizeId, quantityId, scriptId, priceId) {
185       let size = $(sizeId).val();
186       let quantity = parseInt($(quantityId).val());
187       // Get price object
188       let priceDict = JSON.parse($(scriptId).text());
189       let price = priceDict[size];
190       let priceStr = ("£" + (price * quantity).toFixed(2));
191       $(priceId).text(priceStr);
192       return priceStr;
193   }
194
```

CONFIGURE

Metrics

There are 23 functions in this file.

Function with the largest signature take 6 arguments, while the median is 0.

Largest function has 16 statements in it, while the median is 2.

The most complex function has a cyclomatic complexity value of 7 while the median is 1.