Sanette Tanaka
Authoring Principles Final Project
Report
Professor Charlie Cullen

Introduction

For my final project for Authoring Principles, I designed and created a dynamic website for a fictional coffee company called Roasted Ground. The website allows for a number of user interactions, such as allowing the user to fill out a form and filter coffee products. In this report, I will go over the various aspects of the website in terms of the assignment requirements. This website was primarily created with HTML, CSS, and JavaScript. The jQuery library and front-end framework Twitter Bootstrap were also used. The site is responsive and is based on a 1,000-pixel, 12-column grid.

Overall Structure

The Roasted Ground website consists of four pages: the home page, an about page, a product page, and a contact page. The home page and about page consist of markup text only. The product page and contact page are both powered by JavaScript files.

The product page has two JavaScript files. The first file, *products.js,* contains an array of coffee product information. The second file, *javascript.js,* contains functions that influence the product page. Similarly, the contact page contains two JavaScript files as well. The first pulls in the jQuery library through an external link. The second file, *contact.js*, contains functions that run the contact page.

Every page makes use of a 1,000-pixel grid, defined in the *onepcssgrid.css* stylesheet. The bulk of the CSS styling is contained in *bootstrap.css* and custom styling can be found in *style.css.*

Documentation of All Requirements

This section runs through the nine requirements of final assignment, with notes regarding extensions of the brief.

1. Inclusion of at least 4 different GUI controls (e.g. text box/dropdown/list/button)

The product page includes four GUI controls that change the display of the coffee products. Two radio buttons allow the user to filter the coffee products by price.[1] The first radio

[1] "CSS: Supported Controls," Mark Otto, Twitter Bootstrap, version 3.3.1 <http://getbootstrap.com/css/#forms-controls>.

button sorts the products by ascending price, and the second button sorts the products by descending price. Both buttons call the function *filter();* when clicked.

The radio buttons are mutually exclusive, since users cannot choose both at the same time. To ensure that only one button would be clicked, I gave each button the same name, as shown below.[2]

```
<div class="radio">
      <form action="" onclick="filter();">
      <input id="priceAscend" type="radio" name="price"
value="priceAscend">Ascending Price</input>
      <!-- Break for legibility -->
      <br/>
      <input id="priceDescend" type="radio" name="price"
value="priceDescend">Descending Price</input>
      </form>
</div>
```

The second GUI control is a dropdown menu that sorts the coffee products by region.[3] Users can select Latin America, Africa, Oceania, or Asia. When the dropdown menu is changed from its default, which is blanked, the function *filter();* is again called.

```
<select id="regionDropdown" class="form-control" onchange="filter()">
      <option value=""></option>
      <option value="latinamerica">Latin America</option>
      <option value="africa">Africa</option>
      <option value="oceania">Oceania</option>
      <option value="asia">Asia</option>
</select>
```

The third GUI control is two checkboxes that sort the coffee product by regular and/or decaf coffees.[4] The checkboxes are not mutually exclusive, so users can select both options to see all coffees. They can also select just one option to see only one type of coffee. Clicking neither option means the display is the default, which shows all coffee products.

```
<div class="checkbox">
      <form onclick="filter();">
            <input id="regularCheckbox" type="checkbox" name="regular"
value="regular">Regular Coffee</input>
            <!-- Break for legibility -->
            <br/>
            <input id="decafCheckbox" type="checkbox" name="decaf"
value="decaf">Decaf Coffee</input>
            <!-- Break for legibility -->
            <br/>
```

---

[2] "How to set that only one radio button can be checked," Stack Overflow, 24 March 2011 <http://stackoverflow.com/questions/5419459/how-to-set-that-only-one-radio-button-can-be-checked>.

[3] "HTML <select> tag," W3Schools, 2014, < http://www.w3schools.com/tags/tag_select.asp>.

[4] Otto, "CSS: Supported Controls."

```
        </form>
    </div>
```

Finally, the fourth GUI control is a button that clears all the options.[5] This control works by calling the function *clearFilters();*, which clears all the GUI options described above.

```
<button class="btn btn-default" type="button"
onclick="clearFilters();">Clear all filters</button>
```

The contact page also includes one button in addition to the HTML form.[6] When clicked, the button calls the function *validateFields();* before sending the message.

```
<button type="button" class="btn btn-default btn-lg"
onclick="validateFields();">Send</button>
```

2. An HTML form containing at least 3 User Input elements (excluding controls)

The contact page contains an HTML form with three user input elements. The first input is of type text, and asks users for their name. When the user clicks off the text field, the input calls the function *validateName();.*

The second input is of type email and asks users for their email address. Similarly, when the user clicks off the text, the input calls the function *validateEmail();.*

The third input is a dropdown list and asks users to select a subject for their message. When users select an input, the input calls the function *validateInput();.*

Finally, the four input is a textarea that asks users to write their message. As soon as users click the textarea, the input calls the function *validateInput();* once more.

3. Proper validation of these input elements using JavaScript

---

[5] "CSS: Buttons," Mark Otto, Twitter Bootstrap, version 3.3.1 <http://getbootstrap.com/css/#buttons>.

[6] Mark Otto, "CSS: Buttons."