

IS322 Project #1 – Basic JS and Responsive Design

In this assignment you will be tasked with creating a simple E-Commerce site with a basic set of features. A landing page to advertise your site and some of your top selling products, a product list where the user can view all your products with filters and sorting, and a product page for viewing information on a specific product (only required for 3 person groups)

The **instructions** below give a step-by-step guide for creating each of the pages, as well as provide visual examples. You can however not follow them and just make sure your pages match all the **requirements**. Bootstrap CSS is allowed, but not any of it's JavaScript Components. Anything above and beyond the original requirements will be considered for extra credit.

This assignment only requires HTML, CSS, and JavaScript so it won't be required to be fully functional, only using mock data to display products. There is no back-end component required for this assignment.

General Requirements

1. All code must be in GitHub, and the website should be publicly accessible from at least one of your group members AFS web spaces. Both links will be required.
2. All pages should be responsive, and can be easily viewed on the following resolutions, you do not need to specifically code for these resolutions, these are just the ones I will be testing:
 - a. 1366px wide (Large Desktop)
 - b. 1024px wide (Small Desktop)
 - c. 768px wide (Tablet)
 - d. 411px wide (Phone)
3. Not using Bootstrap at all and doing all the responsive work yourself will reward 20 points of extra credit on this project.

Part 1: Landing Page (Required to Teams of 2+)

Description

Landing pages are typically the first page the user sees when they go to your site, it's your index.html, your welcoming mat. As such it should give your potential customers a reason to stay, with information about your company/store as well as some examples of products and services you offer.

Requirements

1. Some kind of Primary Navigation that allows your users to navigate to the product list and any other potential pages.
2. A Carousel at the top of the page that displays various products or sales you have going on. Should change automatically every 3-5 seconds to display another. Should contain at least 3 slides.
3. A blurb about your company on the bottom of the page (can use Lorem Ipsum text generator)
4. A footer containing all of your fake companies' information (Phone Number, Address, links to social media)
 - a. Horizontally on Desktop
 - b. Vertically on Mobile

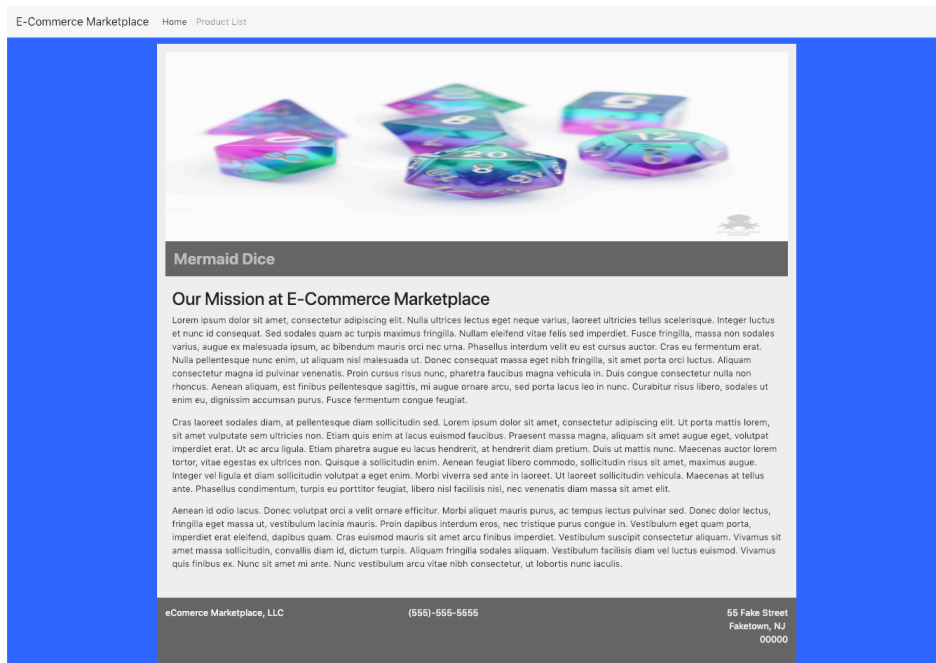
Instructions

1. Create an *index.html* and *landing.js* files and add them to the product.
2. Populate your *index.html* file with the basic HTML page structure. Make sure to include your *landing.js* file at the end of the body tag.
3. Either make or include an existing CSS file within the HEAD tag using the LINK tag. (Also include bootstrap at this point if you so choose)
4. Create a Navigation Menu to link to other pages of your site, you can make it yourself from scratch or use this (<https://getbootstrap.com/docs/4.3/components/navbar/>). Either way just make sure it works well on all tested resolutions. You only need 2 links, so collapsing isn't necessary (will give extra credit for it though)
5. Add a DIV tag at the top of the page containing 3 images. Create an *active* class for the image that sets the DISPLAY to BLOCK, while the default styling for the image will have a CSS DISPLAY property of NONE. This will make it so only image is displayed at a time.
6. In your *landing.js* JavaScript file, create a function that removes the active class from one slide and adds it to another. It should be able to keep track of the current slide with a variable declared outside of the function and be able to loop back to the first slide after it reaches the end.
7. Use the setInterval Method (https://www.w3schools.com/jsref/met_win_setinterval.asp) to call your function at regular intervals. (time is in milliseconds so 1 second = 1000ms)
8. Add Description of company that contains at least a title and 3 paragraphs (can use text generator)

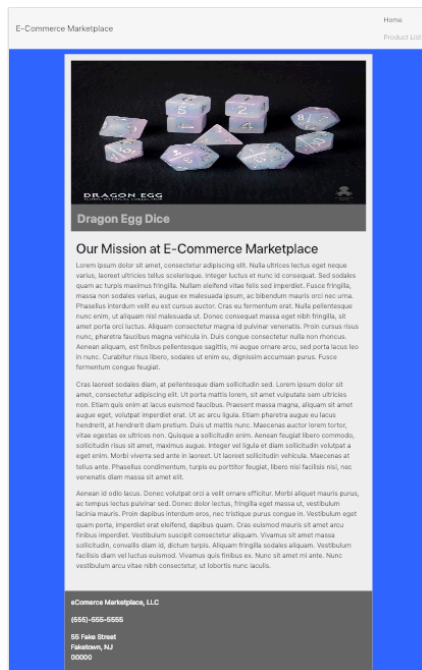
9. Add a footer on the bottom of the page that displays the text horizontally on desktop and vertically on mobile. Can use FLEXBOX, setting the CSS property FLEX-DIRECTION from row to column based on a media query.

Examples

Desktop



Mobile



Part 2: Product List

Description

Your product list is where your customer can search for the exact product that meets their needs. To allow for this you should provide several filters and sorting options to let your users narrow down your vast inventory to the specific product they're looking for (think Amazon).

Requirements

1. A Filterable list of products. Each product item should contain an image, title, price and price. (NO TABLES)
2. A set of filters to narrow down the list. Should have a least 2. Some suggestions **IF GROUP OF 3 PEOPLE YOU'LL NEED 4 FILTERS. ADD META DATA TO YOUR DATABASE TO FILTER BY IF NECESSARY**
 - a. Price Range
 - b. Meta data fields you don't need to display (ex. Category, Screen Size, Department)
3. The ability to sort the list by at least two fields. Recommended sorting options:
 - a. Price (1-10)
 - b. Title (A-Z)
4. List should be able to be viewed comfortably on any of the previously stated resolutions

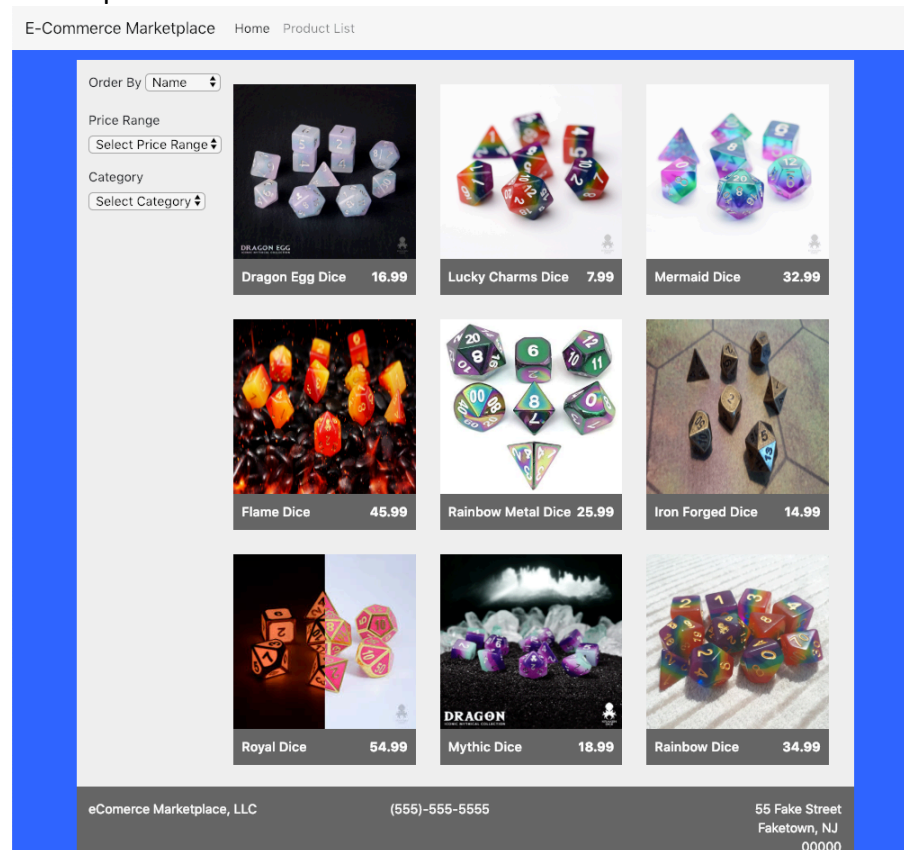
Instructions

1. For the JavaScript portion of this assignment, the *AdvancedJS* repository from week 3 provides a lot of examples. Just as the *Media Queries* repository from week 2 will provide examples of CSS.
2. Create a *productList.html* and *productList.js* file and include them in your product.
3. Populate your *productList.html* file with the basic HTML page structure. Make sure to include your *productList.js* file at the end of the body tag.
4. Either make or include an existing CSS file within the HEAD tag using the LINK tag. (Also include bootstrap at this point if you so choose)
5. Create an area to contain your filters and add the relevant form inputs. Dropdowns or text inputs tend to work best.
6. Add a div to you page with an ID, this will eventually be populated with your list data, but should be empty for the time being.
7. In the *productList.js* create a mockDatabase of JavaScript objects to render into your HTML.
8. Create a *renderList* function similar to that of *AdvancedJS* and to render out your mockDatabase into your HTML. Some things to consider:
 - a. Make sure you clear out innerHTML first.
 - b. Don't use the mockDB variable directly, pass in a parameter. As we'll be rendering different lists later.
9. Hook up a sorting function via an event listener similar to that of *AdvancedJS*. Some things to keep in mind are strings and numbers need different strategies for sorting.

10. Create 2 filters for your data based off the examples from Advanced JS. By listening to the change event on your DropDowns or Text Inputs you can get the value of it and use the `Array.filter` function to pass a new data set to your `renderList` function.
11. Use media queries to position your filters on the side of the page for Desktop views, and the top of the page for mobile views. Examples of this can be found in the *Media Queries Repository* from week 2.
12. Make sure your list always looks good on all screen resolutions I will be test, use the *Media Queries Repository* and the following examples for insight

Examples

Desktop



Tablet

E-Commerce Marketplace

[Home](#)
[Product List](#)

Order By:


Name

Price Range:


Select Price Range

Category:


Select Category




Dragon Egg Dice16.99




Lucky Charms Dice7.99




Mermaid Dice32.99



Flame Dice45.99



Rainbow Metal Dice25.99



Iron Forged Dice14.99

Mobile

E-Commerce Marketplace

[Home](#)
[Product List](#)

Order By:


Name

Price Range:


Select Price Range

Category:


Select Category




Dragon Egg Dice16.99




Lucky Charms Dice7.99




Mermaid Dice32.99




Flame Dice45.99



Rainbow Metal Dice25.99



Iron Forged Dice14.99



Royal Dice54.99

Part 3: Product Details (Only required for 3-person teams)

Description

After the user has sorted through your list and picked a product they might want to buy, they should be able to drill down and find out more information about it. That's where the product details page comes in handy, it gives more information than the list view and also allows the user to focus on a single product as opposed to slipping they're focus.

Requirements

1. Page should declare a JavaScript Object somewhere before the main script contain the meta data for the page. Nothing should be hardcoded and instead pull it's data from the JavaScript Object. This is so we can eventually replace it with an API call to properly display product data.
2. Page should display all the basic Product information in a stylized way.
 - a. Product Title
 - b. Price and a (fake) button to buy/add to cart
 - c. Image that was displayed on list view.
 - d. The brief description from the Product List
3. Product should have a longer description stored in the JavaScript Object. This description can contain HTML tags (Can use Lorem Ipsum Text generator with some added HTML). When displayed on the page, the HTML tags should be rendered as expected. Each description should contain at least.
 - a. One Header Tag
 - b. One Paragraph Tag
 - c. One Strong Tag (for bold)

Instructions

1. Follow Instructions 1-4 from previous pages. Just instead creating *productPage.html* and *productPage.js*.
2. Create a page that displays all of the information for a product. There are no real requirements other then make it look as good as possible and make sure all the required information is displayed.
3. Within *productPage.js* copy and paste your mock database from the product list. Select a product from the array to set as the main product for this page. Ideally you'd be making a request to an API to get this product, but we don't have that for now.
 - a. For extra credit make it a random product from the array (generate a random number index using `Math.random()`) so I can check that it works for all your products.
4. After getting the product from the array, get reference to and set the innerHTML of all the elements on your page that should normally contain data. Tip, setting an ID for each element you want to populate will make it easier.

Examples

Desktop

Dragon Egg Dice

Product Details

- PLASTIC



16.99

Add to Cart

Dragon Egg Dice

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed commodo vehicula suscipit. Aenean feugiat finibus nisl, mattis tempor tellus pellentesque at. Morbi id aliquam mauris. **Nullam malesuada tellus et elit rutrum interdum. Phasellus pharetra lectus ut tristique mattis.** Mauris ipsum ex, ultricies in malesuada facilisis, varius quis leo. Morbi ante quam, congue et vehicula molestie, blandit quis sem. Duis massa urna, tempus nec dolor et, euismod porta elit. Sed consectetur purus id pulvinar maximus. Phasellus sodales tempus purus, sed egestas nulla porttitor ut.

Maecenas laoreet diam sed dolor iaculis, vulputate blandit leo sodales. Aenean felis ipsum, lacinia sed rutrum a, scelerisque tincidunt dui. Maecenas in nisl et arcu volutpat porta eu sed dui. Duis porttitor aliquam leo, vitae pharetra sem ultricies nec. Cras malesuada purus maximus turpis semper, vitae mollis sapien iaculis. Cras tincidunt, libero ut tincidunt feugiat, justo turpis scelerisque nisl, vitae eleifend leo ipsum sed nisl. Cras sagittis nisi sit amet augue convallis eleifend. Morbi blandit, dolor ac auctor scelerisque, diam enim scelerisque mi, at viverra justo nulla sed enim.

Etiam quis sollicitudin turpis. Maecenas a turpis sollicitudin, iaculis mauris quis, dignissim nisi. Etiam turpis massa, scelerisque et mattis eget, dignissim ac nunc. Nam viverra quam eu sagittis accumsan. Etiam vel blandit enim. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Maecenas ut lobortis erat. Quisque commodo id nisi eget volutpat. Nulla in interdum ligula. Quisque scelerisque dolor sit amet mauris consectetur, sed hendrerit elit faucibus. Proin ullamcorper tincidunt placerat. Pellentesque fringilla dictum luctus. Vestibulum tristique malesuada mauris eu ornare. Fusce sed purus velit.

Mobile

Lucky Charms Dice



7.99

Add to Cart

Product Details

- PLASTIC

Lucky Charms Dice

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris in

Important:

1. Track changes to your code using GitHub
2. Submit your AFS link and GitHub link to the app on Moodle Both Links are required, if you missed one of them, you get 20% off from full credit.
3. 10% off for every day late.
4. You will receive a grade for the project as a whole, and an individual grade for your contributions.