To do: Receive a grade

Overview

In this project, you will demonstrate your proficiency with HTML, CSS, JavaScript, jQuery, Bootstrap, and REST APIs by building a dynamic user-interface consuming a REST API.

Your assignment is to build a fully functional vending machine web-page that embodies the attached wireframe layout and meets the general and functional requirements.

Resources

Project wireframes: Wireframes.pdf

Project web service: http://vending.us-east-1.elasticbeanstalk.com

General Requirements

Your assignment is to create an HTML page (with appropriate CSS and JS pages) that matches the functionality outlined in both the functional & file requirements below as well as the layout shown in the wireframes linked above above.

It is highly recommended that you use Postman to explore the API before you begin coding.

Layout - Match The Wireframes!

- Make your layout responsive, matching the wireframes for a variety of widths.
- The loaded items should take up the left side of the page, in rows of three items.
- Three forms (money, business, change) should take up the right side and collect user input.

Functionality - Make it Work!

- **Displaying Items** - all items should be populated using a REST API request and created dynamically from that call.
- **Adding Money** - money should be incremented by interacting with the 'Add $' buttons in the first form.
- **Purchasing Items** - after adding money and selecting an item, this should submit a purchase to the REST API and display the response results.
- **Change Return** - will allow you to abandon a transaction, resetting the funds input and returning their value in coinage.

The Vending Machine Web Service

You were supplied with the link to the Vending Machine REST Web Service above. This service is similar to the ones that you used for code-alongs and other exercises in this module. As this is a live web-service, the external hosting will keep it running, and it requires no manual starting by you.

To verify that you're able to interact with the web service, open Postman and send a GET request to `http://vending.us-east-1.elasticbeanstalk.com/items`. If your request is properly formatted and can connect to the service, your response should contain a JSON array for a list of items in the vending machine.

Initial State

Upon initial load, your page must display as shown in the initial state.

- The heading of the page should be Vending Machine with a horizontal rule beneath it
- The main content area should be divided into two sections, approximately 2:1 ratio.

    - The loaded items should take up the left two-thirds of the screen.
        - The items should be displayed in bordered elements of three items per row.
        - Each item will display its index on the page numbered 1 to N, the product's name, its price (properly formatted), and the quantity available.
        - *NOTE: The total number of elements may **not** be evenly divisible by three.*
    - Three stacked forms, separated by horizontal rules, should take up the right third of the screen.
        - The first form will collect the money, allowing the addition of funds as well as its display. It should have an input field labeled, "Total $ In" and below 4 buttons labeled "Add Dollar", "Add Quarter", "Add Dime", and "Add Nickel".
        - The second form will identify the selected item, display messages, and conduct business. It should be comprised of two input fields labeled "Messages" and "Item:", respectively, followed by a single "Make Purchase" button.
        - The third form will display and dispense change. It should contain a single input item with a "Change" label followed by a "Change Return" button.
        - *NOTE: The input fields on all three forms should be **read-only** and cannot be typed into. They should only be modified by interacting with elements and buttons, and via associated code in home.js.*
- Upon start-up all input fields should be empty, except the "Total $ In" which should display "0.00" and the list of items will be populated by making an "items" request using the REST API specification below and dynamically creating the appropriate elements.

Using the Vending Machine

Adding Money

Users should add money to the vending machine by pressing one of the four 'Add *' buttons in the form on the right side. Each button should update the total input field by the appropriate amount to keep a running total.

Selecting an Item

The user can select an item to be purchased by clicking on the desired item element within the item list on the left side of the page. This should then update the middle form to display the ID of the selected item.

Purchasing an Item

A user may attempt to make a purchase at any time by clicking the "Make Purchase" button. If pressed, and no item selection has been made, the "Messages" input should display "Please make a selection". When the "Make Purchase" button is pressed and there is an item selected, the purchase request is made using the "money" API request using the specification below.

If the purchase request is sent and succeeds, the machine should refresh the item stock, selection, and money, as well as display "Thank You!!" and any change returned by the web service. However, if the purchase request instead returns in an error, the error message should be displayed in the "Messages" input without changing the selection or "Total $ In". In addition to the error message display, the items list should also be refreshed to show current stock and quantities.

Returning Change

When the "Change Return" button is pressed, the amount of money displayed in "Total $ In" should be returned in a correct number of quarters, dimes, nickels, and pennies and the "Total $ In" should be zeroed. If the "Total $ In" is zero, then the "Change" input field should be cleared.

The format of the returned change will be "yy coin(s)" where yy is the number of a specific coin and coin is quarter, dime, nickel, or penny. If multiple coin types are needed, they will be separated by a comma and space.

For example: $0.82 change would be: "3 quarters, 1 nickel, 2 pennies".

After a successful purchase, selecting any item, depositing money, or pressing "Change Return" should clear the input fields before processing.

Vend Item – No Errors

No errors occur if:

- The total in is greater than or equal to the item price.
- The item is in stock (quantity > 0).

When this occurs, the message 'Thank You!!!' should be displayed in the messages input. The JSON result will contain the change due to the user, if any, which should be displayed in the change input. Pressing the change return button will clear the inputs, update the item list, and prepare for the next purchase.

Vend Item – No Inventory

If the user selects an item that has no quantity left, an HTTP 422 status code will be returned with a body containing a message to display. This message should be displayed in the messages input on the middle-right form.

Vend Item – Insufficient Funds

If the user selects an item and has not entered enough money to purchase it, an HTTP 422 status code will be returned with a body containing a message to display. This message should be displayed in the messages input on the middle-right form.

Vend Item – Invalid Item

If the user selects an item that is not stocked, an HTTP 422 status code will be returned with a body containing a message to display. This message should be displayed in the messages input on the middle-right form.

Vending Machine Web Service API

Below are **examples** of the available API calls and the data they return. Item ids are not guaranteed to be sequential or start at 1. They will, however, be unique. There is no guarantee that 9 items will be returned.

Retrieving All Items

HTTP Method:	GET
URL:	http://vending.us-east-1.elasticbeanstalk.com/items
Path Variable:	None
Request Body:	None

Response Body:	**JSON array of Items:**

```
[
  {
    "id": 1,
    "name": "Snickers",
    "price": 1.5,
    "quantity": 10
  },
  {
    "id": 42,
    "name": "M&M's",
    "price": 1.25,
    "quantity": 8
  },
  {
    "id": 33,
    "name": "Almond Joy",
    "price": 1.25,
    "quantity": 11
  },
  {
    "id": 74,
    "name": "Milky Way",
    "price": 1.65,
    "quantity": 3
  },
  {
    "id": 5,
    "name": "Payday",
    "price": 1.75,
    "quantity": 2
  },
  {
    "id": 16,
    "name": "Reese's",
    "price": 1.5,
    "quantity": 5
  },
  {
    "id": 87,
    "name": "Pringles",
    "price": 2.35,
    "quantity": 4
  },
  {
    "id": 82,
    "name": "Cheez-Its",
    "price": 2,
    "quantity": 6
  },
  {
    "id": 9,
    "name": "Doritos",
    "price": 1.95,
    "quantity": 7
```

```
  }
]
```

Vending an Item

HTTP Method:	POST
URL:	http://vending.us-east-1.elasticbeanstalk.com /money/{amount}/item/{id}
Path Variable:	{amount} – Amount of money deposited for the purchase
Path Variable:	{id} – ID of item to be vended
Request Body:	None
Response Body (no errors):	**JSON representation of Change object:** `{ "quarters": 1, "dimes": 0, "nickels": 0, "pennies": 0 }`
Response Body (no inventory):	**HTTP Status: 422 Unprocessable Entity** `{ "message": "SOLD OUT!!!" }`
Response Body (insufficient funds):	**HTTP Status: 422 Unprocessable Entity** `{ "message": "Please deposit: <amount short>" }`
Response Body (invalid item):	**HTTP Status: 422 Unprocessable Entity** `{ "message": "Invalid item selected" }`

File Structure

The file structure of your submitted project should match the following:

- Your project folder should contain a static html page called **home.html** in the root folder.
- A sub-folder named "css" should contain any custom CSS.
  - Your custom CSS should be in a file named named **home.css**.
- A sub-folder named "js" should contain all custom JavaScript.
  - Your custom JS file should be named **home.js**.
  - All JavaScript in **home.js** should use jQuery for your dynamic DOM manipulation. This includes accessing, altering, creating, and removing elements and event handlers.
  - Use correct industry standards for style and naming of variables, functions, classes, and ids.
  - Comment your JavaScript code appropriately.

Submitting your Assessment

When you are satisfied that all of the objectives are completed take the following actions:

1. Submit your files using the instructions provided by your instructor.
2. If you are attending the Guild online, schedule a time with a staff member to review your code. If you are attending the Guild in person, your code will be reviewed during the weekly code review.
3. Be prepared to answer questions about your code and thought processes.

Add submission

Submission status

Submission status	No submissions have been made yet
Grading status	Not graded

Grading criteria

Page Layout: The page uses HTML and CSS to implement the wireframe layout.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
Bootstrap: The page uses Bootstrap appropriately.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
API Methods and REST: The apprentice can discuss each Vending Machine API method and describe how it's related to REST.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
JavaScript: Browser events (clicks) execute JavaScript methods.	Meets Expectations ***10 points***	Needs Improvement ***5 points***	No Credit ***0 points***
GET: The application successfully executes a GET request to retrieve vending machine items with jQuery.ajax.	Meets Expectations ***10 points***	Needs Improvement ***5 points***	No Credit ***0 points***
JSON: The application handles JSON appropriately in both success and failure cases for the Vending Machine API.	Meets Expectations ***10 points***	Needs Improvement ***5 points***	No Credit ***0 points***

Dynamic HTML: Generates dynamic HTML and inserts it into the DOM based on the data returned from the Vending Machine API.	Meets Expectations ***10 points***	Needs Improvement ***5 points***	No Credit ***0 points***
Money: The application tracks money in the browser.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
Input Capture: Clicking an item copies its ID into the item text input.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
Invalid Selection: The application cannot vend an item if an item hasn't been selected.	Meets Expectations ***5 points***	Needs Improvement ***3 points***	No Credit ***0 points***
Vending Items: The application successfully executes a POST request to vend an item with jQuery.ajax.	Meets Expectations ***10 points***	Needs Improvement ***5 points***	No Credit ***0 points***
Successful Vend: The application handles successful vend requests appropriately. On success, change is returned, the item vended has its quantity reduced by one, and an appropriate message is displayed.	Meets Expectations ***10 points***	Needs Improvement ***5 points***	No Credit ***0 points***

	Vend Errors: The application handles errors in vend requests appropriately. On error, it retrieves the error status and message and displays it.	Meets Expectations ***10 points***	Needs Improvement ***5 points***	No Credit ***0 points***
Last modified	-			
Submission comments	▸ Comments (0)			

◀ Previous activity

Jump to...