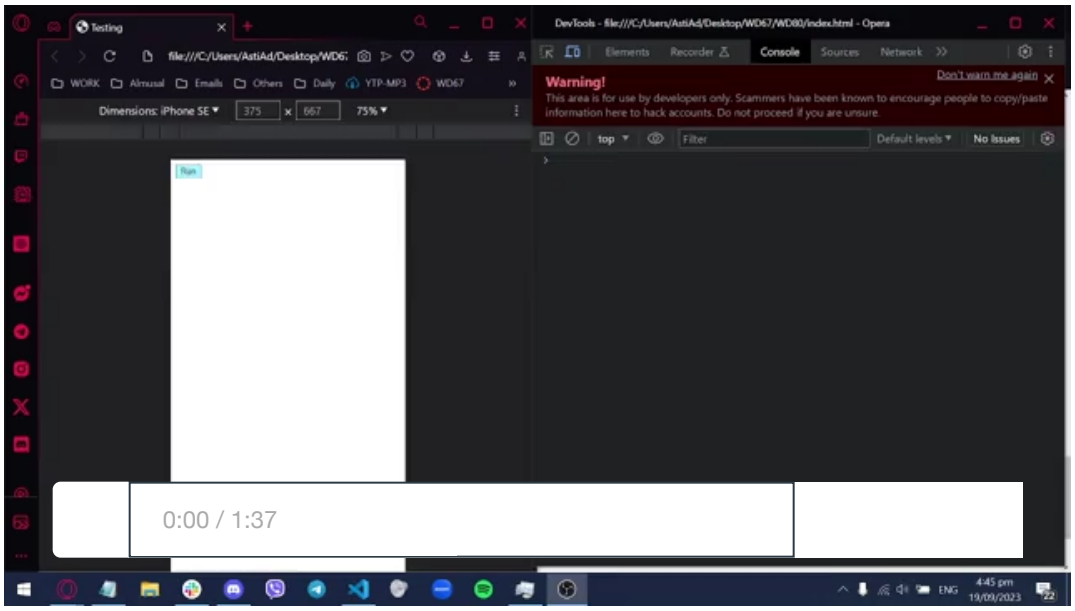
ACT #40 New Virtual Store

New Attempt

Due No Due Date **Points** 80 **Submitting** a file upload **File Types** js

SOFT DEADLINE: 12/06/2023 3:00 PM

HARD DEADLINE: 12/06/2023 3:05 PM



Instruction

Create a program that recreates common store transactions.

Tools

JavaScript, HTML, Visual Studio Code

Description

- **Follow the Submit Your Work steps. (https://kodego.instructure.com/courses/379/pages/7-dot-1-essential-javascript-functions?module_item_id=17474)**
- Name the file **fruit_store_v2.js**
- Write a script that does the following:
  - Give your store a name, and output it using `alert();`

- Create functions for the following store operations.
  - `addItem`
    - This function would ask the user the name of the item they would like to sell.
    - It would then ask for the sell price of that item.
    - This new item would have a stock of 0.
    - The name and price of the item would then be displayed through the console in the format `[item name] added as item for sale. Each [item name] sells for [price of item]. Stock set to 0.`
  - `restock`
    - This function would ask the user for an item.
    - It would then ask for the amount of stock they would like to add to that item.
    - The program would add the specified amount of stock to the item.
    - The name and new stock of item would be displayed through the console in the format `[added stock] stock has been added to [item name]. New stock: [new total stock of item].`
  - `checkPrice`
    - This function would ask for a name of an item.
    - It would then ask for the amount of that item to check the price of.
    - The name and price of the item (based on the specified amount, i.e. price of item * amount of item) would be displayed through the console in the format `[item amount] of [item name] is worth [total price].`
  - `sell`
    - This function would remove stock of an item.
    - When purchase is made, the function would output through the console the name of the item sold, the price sold for, and the new stock after the items sold is deducted.
- Set up a loop that would repeatedly ask the user which function to run, until the user stops the program.
  - Use numbers to designate the functions. The prompt would be something like this:
    - (1) Add new item (2) Restock (3) Check price (4) Sell (5) Stop
  - You may try something like this this:

```
    let i = 0
    while (i == 0){
      let choice = Number(prompt("(1) Add new item (2) Restock (3) Check price (4) Sell (5) Stop"));
      if (choice == 1){
        addItem();
      }else if (choice == 2){
        restock();
      }else if (choice == 3){
        checkPrice();
```

```
      }else if (choice == 4){
        sell();
      }else if (choice == 5){
        i = 1;
      }
    }
  }
```

- Submit your JavaScript file only here.

Validation

- Aside from the requirements stated above, the program should also display an error message in the case of...
  - `addItem`
    - The user entering a zero or negative price.
  - `restock`
    - The user entering a zero or negative amount.
    - The user entering an invalid item name (i.e. an item with the name hasn't been added yet previously)
  - `checkPrice`
    - The user entering a zero or negative amount.
    - The user entering an invalid item name (i.e. an item with the name hasn't been added yet previously)
  - `sell`
    - The user entering an invalid item name.
    - The user entering a zero or negative amount to buy.
    - The user tries to buy more than the available stock.
- The program shouldn't stop when it encounters an error. It should go back to the numbered prompt (1) Add new item (2) Restock (3) Check price (4) Sell (5) Stop

Notes and Tips

- Most of your code for this program would be written within the code block of functions.
- You do not have to set a `return` for each function.
- All of the functions except for `addItem()` would use **loops (https://kodego.instructure.com/courses/379/pages/7-dot-5-js-loops)** .

Helper code

```
function runActivity() {
  const fruit_names = [];
  const fruit_prices = [];
  const fruit_stocks = [];
```

```
  function addItem() {
    let fruit_name = prompt("What is the name of the fruit?");
    let fruit_price = Number(prompt("What is the price of the fruit?"));

    if (fruit_price >= 0) {
      fruit_names.push(fruit_name);
      fruit_prices.push(fruit_price);
      //you need to also set this fruit's stock to 0
      //you need to display via the console the name of the fruit, price, and stock
    } else {
      console.error("ERROR: Invalid price! Enter a positive value.");
    }
  }

  function restock() {
    let choice = prompt("What would you like to add stock to?");
    let fruit_found = false;

    //since you're searching for a fruit, you can use a similar logic with the sell() function
  }

  function checkPrice() {
    let choice = prompt("What would you like to check the price of?");
    let fruit_found = false;

    //since you're searching for a fruit, you can use a similar logic with the sell() function
  }

  function sell() {
    let choice_name = prompt("What would you like to buy?");
    let fruit_found = false;

    for (let i = 0; i < fruit_names.length; i++) {
      if (fruit_names[i] == choice_name) {
        fruit_found = true;
        let amount = Number(
          prompt("How many " + choice_name + " would you like buy?")
        );
        if (amount >= 0) {
```

```
          let total = amount * fruit_prices[i];
          //you have to make it so that you don't sell if not enough stock
          //you also have to make it so that stocks are deducted when you buy fruit
          //you also have to display how much the fruit was sold for as well as remaining stock
          //you have to do validation
        } else {
          console.error("ERROR: Can not have negative amount.");
        }
      }
    }

    if (fruit_found == false) {
      console.error("ERROR: Fruit was not found.");
    }
  }

  alert("Daiben's Fruit Store");
  let i = 0;
  while (i == 0) {
    let choice = Number(
      prompt("(1) Add new items (2) Restock (3) Check price (4) Sell (5) Stop")
    );
    switch (choice) {
      case 1:
        addItem();
        break;
      case 2:
        restock();
        break;
      case 3:
        checkPrice();
        break;
      case 4:
        sell();
        break;
      case 5:
        i = 1;
        break;
      default:
        console.error("ERROR: Enter one of the choices.");
```

```
    }
  }
}
```

Test Cases

Input	Result

Prompt order:
2
apple
1
apple
10.5
3
apple
200
3
pineapple
4
pineapple
4
apple
100
2
apple
100
4
apple
5
5
	`Error message`
	`apple added as item for sale. Each apple sells for 10.5. Stock set to 0.`
	`200 of apple is worth 2100`
	`Error message`
	`Error message`
	`Error message`
	`100 stock has been added to apple. New stock: 100`
	`5 of apple sold for 52.5. New stock: 95`

Prompt order:
1
pear
30
1
avocado
60
1
guava
90
	`pear added as item for sale. Each pear sells for 30. Stock set to 0.`
	`avocado added as item for sale. Each avocado sells for 60. Stock set to 0.`
	`guava added as item for sale. Each guava sells for 90. Stock set to 0.`
	`20 stock has been added to pear. New stock: 20`
	`ERROR: Not enough stock of avocado!`
	`9 of pear sold for 270. New stock: 11`

2 pear 20 4 avocado 10 4 pear 9 5	