PY 5

This is a Prove Yourself (PY). It allows you to showcase all of the topics that you have learned so far. You can use any of the previous projects (website and curriculum) that you have completed to help complete this PY.

The Sensei can give hints and minimal help because the goal of a PY is to have the student showcase their own unique solution to the problem. JS 13 – 17 is a good reference point for this PY.

==

1. CREATE A STOREFRONT

Context: Amazon and Ebay have both heard that you are a talented software engineer. They are battling over your talents because they are in need of your service.

Situation: A store front needs to be created as an example that will be shown to the investors. A store front is an e-commerce module that is used to perform digital transactions. This includes a shopping cart, digital payments, and everything in between.

Problem: The store front must be created using Javascript, array manipulation, and object oriented programming (OOP). They know that you are talented with the 3 topics from above and are competing for your service.

Solution: Create a store front that satisfies the following requirements

HTML

0. Create 5 paragraph tags and give them the id as follows
    a. clothingCount
    b. foodCount
    c. bookCount
    d. computerCount
    e. gameCount

1. create 5 buttons and do the following

    a. give the button an id of “**buyClothing**” and attach an **onclick** event. When the button is clicked on, the JS function “**decClothingQt()**” is called.

    b. give the button an id of “**buyFood**” and attach an **onclick** event. When the button is clicked on, the JS function “**decFoodQt()**” is called.

    c. give the button an id of “**buyBook**” and attach an **onclick** event. When the button is clicked on, the JS function “**decBookQt()**” is called.

    d. give the button an id of “**buyComputer**” and attach an **onclick** event. When the button is clicked on, the JS function “**decComputerQt()**” is called.

    e. give the button an id of “**buyGame**” and attach an **onclick** event. When the button is clicked on, the JS function “**decGameQt()**” is called.

CSS
Style as follows

```
p#clothingCount
{

                        width   : 50px;
                        height  : 50px;

                        background-color: #ffffff;
}



p#foodCount
{
                        width   : 50px;
                        height  : 50px;

                        background-color: #ffffff;

}



p#bookCount
{
                        width   : 50px;
```

```
                    height  : 50px;

                    background-color: #ffffff;

}



p#computerCount
{
                    width   : 50px;
                    height  : 50px;

                    background-color: #ffffff;

}


p#gameCount
{
                    width   : 50px;
                    height  : 50px;

                    background-color: #ffffff;

}
```

Javascript Part 1

0. create a global variable called **currBal** and load the data 5000 into it.

1. Create an array called **catalogArr** and make it an empty array

2. Write the function definition for `decFoodQt()`. Inside the function body, do the following:

    a. Create a digital key called **i** and load the data 0 into it.

b. use a "**for**" loop. The start is 0, the end is `catalogArr.length`, the jump is by 1

   i. inside the body of the "for" loop, do the following

      1. use the digital key **i** and to select ONLY A SINGLE array position
      2. access the private attribute iCat and check if it is equal to "food"

      if **true**, then call the private function **update_iRepo();**

   ii. link JS with the HTML element whose id is "`foodCount`". Next, use **.innerHTML** to show the inventory of food items. The right side of the assignment operator should call the private function `get_iRepo()`;

   iii. link JS with the HTML element whose id is "`foodCount`". Next, change the style of the backgroundColor. The right side of the assignment operator should call the private function `get_repo_color()`;

Javascript Part 2

1. write a class definition called `CatalogItem`

   1.1. a constructor is used to initialize the private attributes

2. the `CatalogItem` has the following private attributes
   2.1. iNum
   2.2. iTitle
   2.3. iPrice
   2.4. iRepo → item repository. This number indicates how much are left in stock. By default, there are 5 left in stock
   2.5. iCat → item category, like clothing, games, books, food, and computer to match the HTML from above

3. the `CatalogItem` has the following private functions

   3.1. **get_repo_color();** This function definition uses the private variable iRepo. It will check if there is enough in stock to be purchased.

      3.1.1. If **there is 0 items in stock**, then return the data "**#ff0000**"

      3.1.2. **else if there is 4 items or less**, then return the data "**#ffa500**"

      3.1.3. **else**, then return the data "**#909090**"

   3.2. **get_purchase_state();** This function definition checks if the user's balance (currBal) is equal to or greater than the price of the item. The body of the function definition does the following:

3.2.1. if the user's balance is equal to OR greater than the price of the item, then return the data 1

else, return the data 0

3.3. **update_iRepo();** this function decrements the private variable iRepo by 1.

3.4. **get_iRepo();** this function returns the private variable iRepo.

4. Create five objects, one for each category of
   a. clothingCount
   b. foodCount
   c. bookCount
   d. computerCount
   e. gameCount

5. After making an object, “**push**” it into the array **catalogArr**.