COMP 2406 B - Fall 2022

Assignment #1

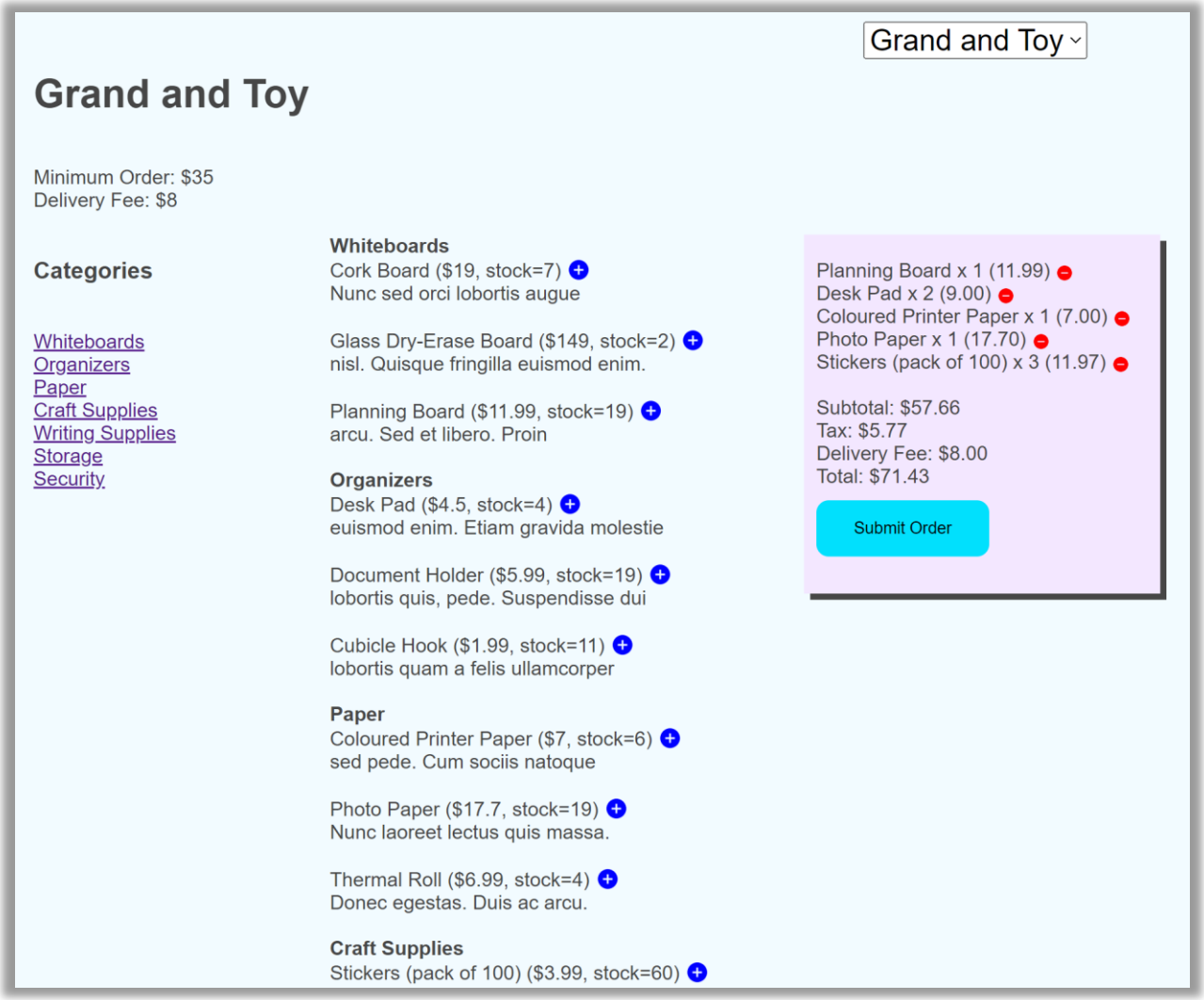
Client-side programming with JavaScript

Due Wednesday, October 5, 23:59. No late assignments will be accepted.

Submit a single zip file called "YourName-a1.zip".

This assignment has 100 marks.

Web Page Sample



Assignment 1 – due Wednesday, October 5, 23:59

Assignment Background

In this assignment, you will develop a web page that allows a user to browse office supplies for several vendors, add items from a vendor to an order, and simulate placing an order. Everything you implement for this assignment will involve client-side programming within the browser. You must implement everything using plain JavaScript and cannot use external libraries/tools such as jQuery and Bootstrap.

Before starting your design for the assignment, you are encouraged to read through the entire specification. It is possible that later problems will inform your design decisions and, by preparing in advance, you can avoid having to significantly refactor your code as you progress through the assignment.

Provided files

The base code for this assignment consists of 5 files, which you can download from Brightspace:

- **order.html**
- **styles.css**
- **client.js**
- **add.png**
- **remove.png**

The provided HTML file **order.html** contains all the elements necessary for the assignment and, in general, should not be changed. The page contains a drop-down list (with the `id="vendor-select"`) for selecting the current vendor. You can either have a default vendor selected, in which case the page contents should be initialized to the supplies of that vendor (see the page sample above) or have a 'Select a vendor...' type of message across the page. The contents of the drop-down list are already dynamically created (inside the **client.js** file) from the provided vendor data. This example demonstrates that you should not hard code the selections (or supply values) in the HTML file.

The JavaScript file **client.js** contains three variables (`vendor01`, `vendor02`, `vendor03`) containing the office supplies data for each of the three included vendors. There is also an array called `vendors`, which contains each of the three vendor objects. Each vendor object has four properties:

1. `name` – the name of the vendor,
2. `min_order` – the minimum order amount required to place an order
3. `delivery_charge` – the delivery charge for this vendor
4. `supplies` – an object containing the office supplies data for this vendor
5. The `supplies` object may have any number of properties, each of which indicates a category within the vendor's supply list. You can add more categories. The value associated with each category property will also be an object. The properties of the category object are **unique IDs** associated with each office supply item. The value associated with each unique supply item ID includes that item's `name`, `description`, `stock`, and `price`. Feel free to add more items or change the existing values. In particular, the `description` field contains random text, - you can change this. In general, you may re-format or edit the data provided in the variables

(`vendor01`, `vendor02`, `vendor03`) in any way you desire with the only exception - do not delete the keys: `name`, `stock`, and `price`.

The Cascading Style Sheets file **styles.css** contains some basic styles for the web page. You need to add more design rules to the file to make your page visually appealing and well-structured.

The two **.png** files can be used to attach click events to your webpage. Feel free to use other images.

Selecting a Vendor (20 marks)

When the selected vendor is changed, the following requirements must be met:

1. If there is a current order with items in it, a prompt should be displayed to the user to confirm whether they want to clear the order or not. This can be implemented using the `confirm()` method https://www.w3schools.com/jsref/met_win_confirm.asp. If the user does not want to clear the order, then no data on the page should change (i.e., it cancels the effect of the change).
2. If the user accepts clearing the current order, or if there is no current order, then the newly selected vendor information should be placed on the page. Any information from the previous vendor should be removed. Any existing order information should be cleared.
3. The vendor's name, minimum order, and delivery fee should be displayed near the top of the page.
4. The supplies and order information should be displayed in a three-column layout below the basic vendor information. The left column must have a list of the categories in the vendor's supply list. These categories should also be links to allow the user to skip directly to that category within the supply list. The middle column should present the supply list of the vendor (see Supply List Requirements below). The right column should present the current order summary (see Order Summary Requirements below). Notice that the provided HTML file already contains three **div** elements to support the three-column layout.

Supply List Requirements (20 marks)

1. The supply list should be organized, so each category of items is clearly separated.
2. Each item should be clearly separated (e.g., you can use a list) and show the name, description, stock information, and price of the item.
3. The **add.png** image should be shown near each item and clicking this image should allow the user to increase the quantity of that item in the order by 1. You will have to scale the size of the image. You can use an alternate image or element, but it must be clear that it is responsible for increasing the amount of an item in the order.
4. All dollar values should be displayed with two decimal places. This can be done using `toFixed()` method: https://www.w3schools.com/jsref/jsref_tofixed.asp.

Assignment 1 – due Wednesday, October 5, 23:59

Order Summary Requirements (30 marks)

1. For each item currently in the order, the summary should include the item's name, the number of units in the order, and the total price of that item (unit price * number of units).
2. The **remove.png** image should be shown near each item and clicking this image should allow the user to decrease the quantity of that item by 1. If the quantity goes to 0, the item should not show up in the summary. You may use an alternate image or element, but it must be clear that it is responsible for decreasing the amount of an item in the order.
3. Following the list of items, the current subtotal (sum of all item prices) should be displayed, along with the delivery fee, the tax (use a tax rate of 10%) and the current order total.
4. If the current subtotal is greater than or equal to the minimum order amount of the selected vendor, there should be an element that allows the user to submit the order (e.g., a button). When this element is clicked, an **alert** should be displayed (using `alert()` method) confirming that the order has been submitted, and the page should be reset.
5. If the current subtotal is less than the minimum order, a small message should be placed at the bottom of the summary indicating the total amount of dollars that must be added before the order can be placed. For example, if the minimum order is $25 and the current subtotal is $10, the message could say, "You must add $15.00 more to your order before submitting".
6. All dollar values should be displayed with two decimal places.

Out of Stock Requirements (10 marks)

Each supply item has associated stock information. If the user tries to add more items than the vendor has in stock, an **alert** message should be displayed informing that the item is out of stock. No other data on the page should change.

Visual and Structural Page Quality (10 marks)

These marks will be allocated for the overall quality of your page's implementation. This will include factors such as the visual appeal of your page, as well as the **responsiveness** of your page to user actions and changes. For example, how does your page react to changing browser dimensions? Is the information easy to find on your page? Is everything on the page formatted clearly? The provided base code already contains behavioural elements with the ids `left`, `middle`, and `right`. You can use them for aligning the categories and order summary with the current location in the supplies list. You are also free to add additional components or functionality to your page. Include a description of any additions/changes you made within your **README.txt** file.

Code Quality and Documentation (10 marks)

Your code should be well-written, commented, and easy to understand. This includes providing clear documentation explaining the purpose and functionality of pieces of your code. You should use good variable/function names that make your code easier to read. You should do your best to avoid unnecessary computation and ensure that your code runs smoothly throughout the operation. You

should also include a **README.txt** file that explains any design decisions that you made, as well as any additional instructions that may be helpful to the marking TA.

Academic Integrity

Submitting not original work for marks is a serious offence. We use special software to detect plagiarism. The consequences of plagiarism vary from grade penalties to expulsion, based on the severity of the offense.

You may:

- Discuss general approaches with course staff and your classmates,
- Show your web page, but not your code,
- Use a search engine / the internet to look up basic HTML, CSS, and JavaScript syntax.

You may not:

- Send or otherwise share your solution code or code snippets with classmates,
- Use code not written by you,
- Use a search engine / the internet to look up approaches to the assignment,
- Use code from previous iterations of the course, unless it was solely written by you,
- Use the internet to find source code or videos that give solutions to the assignment.

If you ever have any questions about what is or is not allowable regarding academic integrity, please do not hesitate to reach out to course staff. We will be happy to answer. Sometimes it is difficult to determine the exact line, but if you cross it the punishment is severe and out of our hands. Any student caught violating academic integrity, whether intentionally or not, will be reported to the Dean and be penalized. Please see Carleton University's Academic Integrity page.

Submit Your Work

To submit your assignment, you must **zip** all your files and submit them to the Assignment 1 submission on Brightspace. Name your .zip file “YourName-a1.zip”. Make sure you download your .zip file and check its contents after submitting it. If your .zip file is missing files or corrupt, you will lose marks.

Your .zip file should contain all the resources required for your assignment to run. The TA should be able to open the **order.html** page and use your page without any additional setup.