**Вежба 1:**

**Eдноставен прикажувач на производи и скeнeр нa баркодoви**

* **Тежина: 4/10**

**Преглед:** Замислете сценарио каде курир за испорака треба да собере производи од различни продавници. Вашата задача е да напишете модул за апликација што ќе прикажува информации за производите, генерира баркодови и наоѓа најблиска продавница за секој производ. Генерирањето на баркод се базира врз конвертирањето на идентификациониот број на производот во бинарен облик и визуелно прикажување со црни и бели вертикални ленти.

**Барања:**

1. **Добивање на информации за производ и продавница:**
   * Информациите за сите производи се зачувани во датотеката **scans.json**.
   * Информациите за сите продавници се зачувани во датотеката **shops.json**.
2. **Информации за производ:**
   * Прикажување на детали за секој производ, вклучувајќи име, идентификационен број, цена, количина и големина.
   * Прикажувањето на производите е во обратен редослед, т.е. прв се прикажува последниот производ од низата со производи, па претпоследниот итн..
3. **Генерирање на баркод:**
   * Конвертирање на идентификациониот број на производ во бинарен облик.
   * Генерирање на баркод претстава каде што секој бинарен знак се прикажува како вертикална лента (7px широка).
   * Користење на црна за '0' и бела за '1'. Висината на лентите не е специфицирана.
4. **Процес на скиенирање:**
   * Симулирајте процесот на скенирање со позиционирање со покажувачот над баркодот.
   * Процесот на скенирање трае 1.3 секунди.
   * Ако покажувачот остане над баркодот во текот на целиот период, скенирањето е успешно. По успешно скенирање, во периодот додека се чека вчитувањето на следниот производ, се оневозможува скенирање на веќе успешно скенираниот баркод.
   * Ефектот на скенирање се визуелизира со намалување на непроѕирноста (opacity) во текот на 1.3 секунди.
   * Ако покажувачот излезе од областа на баркодот, скенирањето се откажува.
5. **Достапност во продавница:**
   * Проверка на достапноста на предметот во различни продавници.
   * Предметот е достапен ако продавницата го поседува и има количина поголема или еднаква на потребната количина.
   * Ако повеќе продавници го имаат истиот производ, прикажете ги информациите за најблиската продавница според растојанието.
   * **Забелешка: Проверката на достапноста во продавниците се извршува САМО по успешно скенирање.**
6. **Слики на предмети:**
   * Секој производ има придружена слика во папката **images**.
   * Името на сликата се конструира на следниот начин:   
     "{име на производ}- {последните 5 цифри од ид-то}.jpg".
   * Пример:   
     Производ: Retro Red Dress, со ID: 123456789.   
     Името на сликата е: dress-56789.jpg
7. **Пораки:**
   * По успешно скенирање, прикажете информации за најблиската продавница каде што предметот е достапен.
     + Форматот на пораката треба да биде: "Име на продавница, Улица на продавница, Број на улица, Skopje".
   * Ако предметот не е достапен во ниту една продавница, прикажете порака: "The item is NOT available anywhere
8. **Автоматско прикажување на следен производ:**
   * По успешно скенирање и прикажување на информации за продавницата, почекајте 3 секунди.
   * Автоматски преминете на следниот производ и повторете го процесот.
9. **Циклично изминување на производи**:
   * Доколку сите производи од **scans.json** датотеката се процесираат, процесот започнува од почеток, т.е. циклична имплементација.

**Насоки за имплементација:**

* Во прилог ви е даден почетниот HTML и CSS. Потребно е да ја напишете само JavaScript имплементацијата.

**Примерен Сценарио:**

1. Добивање на информации за производи и продавници од **scans.json** и **shops.json**.
2. Прикажување на детали за Производ 1, генерирање и прикажување на баркод, и прикажување на слика на предметот.
3. Позиционирање над баркодот за симулирање на скиенирање (1.3 секунди).
4. Ако скенирањето е успешно, наоѓање на најблиската продавница со производот, прикажување на информации за продавницата.
   * Пример: "Продавница ABC, Главна Улица, 123, Скопје."
5. После 3 секунди, автоматски преминување на следниот производ и повторување на процесот.

Имплементирајте ја апликацијата според вашето разбирање и стил, обезбедувајќи дека податоците се добиваат од JSON датотеките.

Пример слики:

A screenshot of a computer

Description automatically generated

Слика 1. Успешно прикажани информации за производ и успешно изгенериран баркод.

A screenshot of a computer

Description automatically generated

Слика 2. Избледување на баркодот при негово скенирање.

A screenshot of a computer

Description automatically generated

Слика 3. Успешно скниран баркод и прикажување на информации за неговата достапност.

A screenshot of a gym shorts

Description automatically generated

Слика 4. После 3 секунди, успешно вчитан следен производ.

**Exercise 1:**

**Build a Simplified Barcode Scanner for Shipping Courier**

* **Difficulty Level: 4/10**

**Overview:** Imagine a scenario where a shipping courier needs to collect items from various shops. Your task is to build a module for an application that displays item information, generates item barcodes, and finds the closest shop for each item. The barcode generation is based on converting the item ID to binary and visually representing it with black and white vertical bars.

**Requirements:**

1. **Fetch Item and Shop Information:**
   * Fetch item information from the **scans.json** file.
   * Fetch shop information from the **shops.json** file.
2. **Item Information:**
   * Display the details of each item, including item name, ID, price, amount, and size.
3. **Barcode Generation:**
   * Convert the item ID to binary.
   * Generate a barcode representation where each binary digit is displayed as a vertical bar (7px wide).
   * Use black for '0' and white for '1'. The height of the bars is not specified.
4. **Scanning Process:**
   * Simulate the scanning process by hovering over the barcode.
   * The scanning process takes 1.3 seconds.
   * If the mouse remains over the barcode for the entire duration, the scanning is successful.
   * The scanning effect is visualized by decreasing opacity over the 1.3 seconds.
   * If the mouse leaves the barcode area, the scanning is canceled.
5. **Shop Availability:**
   * Check the availability of the item in various shops.
   * An item is considered available if the shop has it and has an amount greater than or equal to the required amount.
   * If multiple shops have the same item, display information from the closest shop based on distance.
   * **Note: Shop availability check is performed after a successful scan.**
6. **Item Pictures:**
   * Every item has an associated picture in the **images** folder.
   * The picture name is constructed as follows: "itemName-last 5 digits of ID.jpg".
7. **Messages:**
   * After a successful scan, display information about the closest shop where the item is available.
     + The message format should be: "Shop Name, Shop Street Name, Street Number, Skopje".
   * If an item is not available in any shop, show the message: "The item is NOT available anywhere."
8. **Automatic Item Display:**
   * After a successful scan and displaying shop information, wait for 3 seconds.
   * Automatically move to the next item and repeat the process.
9. **Cyclic iteration of products:**

* If all products from the scans.json file are processed, the process starts from the beginning, ie. cyclic implementation.

**Implementation Guidelines:**

The initial HTML and CSS are provided. You only need to write the JavaScript implementation.

**Example Scenario:**

1. Fetch item and shop information from **scans.json** and **shops.json**.
2. Display Item 1 details, generate and display the barcode, and show the item picture.
3. Hover over the barcode to simulate scanning (1.3 seconds).
4. If successful, find the closest shop with the item, display shop information, and wait for 3 seconds.
   * Example: "ABC Store, Main Street, 123, Skopje."
5. Automatically move to the next item and repeat the process.