Федеральное государственное образовательное бюджетное учреждение высшего образования
«Финансовый университет при Правительстве Российской Федерации»
(Финансовый университет)

Колледж информатики и программирования

ПМ.08 Разработка кода информационных систем

Группа: 3ИСИП-622

УТВЕРЖДАЮ

Председатель предметно-цикловой комиссии информационных систем и программирования

_____ Т.Г. Аксёнова

«____» _____2025 г.

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №12

Анализ функциональности веб-каталога товаров

Преподаватель

_____Р. Р. Абзалимов

Исполнитель

_____Л. Д. Слепцов

Оценка: _____

«____» _____2025 г.

Москва

2025

Цель работы

Создание интерактивного веб-каталога товаров с функциями фильтрации, поиска, пагинации и работы с корзиной покупок.

Ход работы

1. Что такое LocalStorage в контексте веб-разработки и для чего он используется в данном проекте?

LocalStorage — это встроенный в браузер механизм для хранения данных на стороне клиента. Данные сохраняются даже после закрытия браузера и могут быть использованы при повторном посещении сайта. В данном проекте LocalStorage используется для:

- Хранения состояния корзины (cart).
- Хранения списка избранных товаров (favorites).

Преимущества:

- Данные сохраняются между сессиями.
- Удобно для хранения небольших объемов данных (например, корзины или избранного).

2. Какие преимущества дает использование асинхронного подхода при загрузке данных в данном проекте?

Асинхронный подход (с использованием async/await) позволяет:

- Не блокировать основной поток выполнения кода (UI остается отзывчивым).
- Упростить обработку ошибок с помощью try/catch.
- Улучшить производительность, так как загрузка данных происходит в фоновом режиме.

В данном проекте асинхронная загрузка данных из cart.json позволяет:

- Загружать данные без блокировки интерфейса.
- Обрабатывать ошибки загрузки (например, если файл недоступен).

3. Какие методы применяются для фильтрации массивов в JavaScript, и какой из них используется в функции getFilteredProducts()?

Для фильтрации массивов в JavaScript используется метод filter(). В функции getFilteredProducts() он применяется для отбора товаров, соответствующих заданным критериям:

- Фильтрация по категории.
- Фильтрация по тексту поиска.
- Фильтрация по цене.
- Фильтрация по наличию на складе.
- Фильтрация по специальным предложениям (скидки, новинки, хиты продаж).

Пример:

```
return products.filter(product => {
    return categoryMatch && searchMatch && priceMatch && stockMatch && discountMatch && newMatch && bestSellerMatch;
});
```

4. Почему в проекте используется шаблонный литерал (template literal) для формирования HTML-разметки, и какие у этого подхода есть преимущества?

Шаблонные литералы (обратные кавычки `) используются для:

- Удобного вставки переменных и выражений с помощью ${}.

- Многострочного текста без необходимости конкатенации строк.

Преимущества:

- Упрощает создание динамической HTML-разметки.
- Повышает читаемость кода.
- Позволяет легко вставлять переменные и выражения.

Пример:

```
const html = `<div>${product.name}</div>`;
```

5. Как реализована логика динамического обновления текста кнопки "Добавить в корзину"/"В корзине"?

Логика реализована следующим образом:

- При нажатии на кнопку "Добавить в корзину" товар добавляется в объект cart, который хранится в LocalStorage.
- При отрисовке товаров проверяется, есть ли товар в корзине (cart[product.id]).
- Если товар есть в корзине, текст кнопки меняется на "В корзине".

Пример:

```
<button class="cart-button" onclick="addToCart(${product.id})"
${!product.inStock ? 'disabled' : ''}>
   ${cart[product.id] ? 'В корзине' : 'Добавить в корзину'}
</button>
```

6. Что такое делегирование событий, и как этот подход можно было бы применить в данном проекте?

Делегирование событий — это подход, при котором обработчик события добавляется на родительский элемент, а не на каждый дочерний элемент. Это полезно для:

- Уменьшения количества обработчиков.
- Упрощения работы с динамически добавляемыми элементами.

В данном проекте делегирование можно было бы применить для:

- Обработки кликов на кнопках "Добавить в корзину" или "Избранное".
- Обработки кликов на элементах пагинации.

Пример:

```
document.getElementById('products').addEventListener('click', (event) => {
   if (event.target.classList.contains('cart-button')) {
      const productId = event.target.dataset.productId;
      addToCart(productId);
   }
});
```

7. В чем разница между display: grid и display: flex, и почему для отображения товаров выбран именно grid?

- **display: grid**:
  - Позволяет создавать двумерные сетки.
  - Удобен для сложных макетов с строками и столбцами.
  - В данном проекте используется для отображения товаров в виде сетки.
- **display: flex**:
  - Позволяет создавать одномерные макеты (либо строки, либо столбцы).

- Удобен для выравнивания элементов внутри контейнера.

Почему выбран grid?

- Товары отображаются в виде сетки, где важно контролировать как строки, так и столбцы.
- Grid позволяет легко управлять отступами и размерами элементов.

8. Какие методы можно использовать для оптимизации производительности при перерисовке большого количества товаров в методе renderProducts()?

- **Virtual DOM**: Использовать библиотеки, такие как React, для минимизации перерисовок.
- **Ленивая загрузка**: Загружать только видимые товары (например, с помощью Intersection Observer).
- **Мемоизация**: Кэшировать результаты фильтрации и сортировки.
- **Batch rendering**: Отрисовывать товары порциями (например, по 10 штук).

9. Какие подходы существуют для создания пагинации в веб-приложениях, и какой из них реализован в проекте?

Подходы:

- **Классическая пагинация**: Кнопки "Назад", "Вперед", номера страниц.
- **Бесконечная прокрутка**: Загрузка новых данных при прокрутке.
- **Ленивая загрузка**: Загрузка данных по мере необходимости.

В проекте реализована **классическая пагинация**:

- Кнопки "Назад", "Вперед", "Первая", "Последняя".

- Отображение текущей страницы.

10. Как можно улучшить доступность (accessibility) данного веб-приложения для пользователей с ограниченными возможностями?

- Добавить aria-* атрибуты для кнопок и элементов управления.
- Убедиться, что все элементы доступны с клавиатуры.
- Добавить текстовые описания для изображений (alt).
- Использовать семантические теги (<button>, <nav>, <main>).
- Обеспечить достаточный контраст текста и фона.

11. Как реализована фильтрация товаров по категориям в коде?

Фильтрация по категориям реализована через массив selectedCategories, который содержит выбранные категории. В функции getFilteredProducts() проверяется, принадлежит ли товар к одной из выбранных категорий:

```
const categoryMatch = selectedCategories.length === 0 ||
selectedCategories.includes(product.category);
```

12. Каким образом в проекте хранится состояние корзины и избранных товаров?

Состояние корзины и избранных товаров хранится в **LocalStorage**:

- Корзина: localStorage.setItem('cart', JSON.stringify(cart)).
- Избранное: localStorage.setItem('favorites', JSON.stringify(favorites)).

13. Объясните логику работы пагинации в проекте.

Пагинация работает следующим образом:

- Товары делятся на страницы по productsPerPage (6 товаров на страницу).
- При изменении страницы (currentPage) отображаются соответствующие товары.
- Кнопки "Назад", "Вперед", "Первая", "Последняя" обновляют currentPage.

14. Как реализована возможность поиска товаров по тексту?

Поиск реализован через фильтрацию по полям name и description:

```
const searchMatch = !searchQuery ||
product.name.toLowerCase().includes(searchQuery.toLowerCase()) ||
product.description.toLowerCase().includes(searchQuery.toLowerCase());
```

15. Какие свойства CSS используются для создания адаптивной сетки товаров?

- display: grid: Создает сетку.
- grid-template-columns: repeat(auto-fill, minmax(250px, 1fr)): Автоматически подбирает количество столбцов.
- gap: 20px: Отступы между элементами.

16. Каким образом обрабатывается случай, когда в корзине уже есть товар и пользователь нажимает кнопку добавления снова?

При нажатии на кнопку количество товара в корзине увеличивается:

```
cart[productId] = (cart[productId] || 0) + 1;
```

17. Как реализована фильтрация товаров по наличию на складе?

Фильтрация по наличию реализована через проверку поля inStock:

```
const stockMatch = !inStockOnly || product.inStock;
```

18. Почему при изменении фильтров происходит сброс текущей страницы на первую?

Сброс страницы на первую (currentPage = 1) обеспечивает корректное отображение отфильтрованных товаров с начала списка.

19. Какие методы используются для асинхронной загрузки данных?

Используется метод fetch с async/await:

```
const response = await fetch('./cart.json');
const data = await response.json();
```

20. Как взаимодействуют между собой компоненты проекта?

- **Данные**: Загружаются из cart.json и хранятся в переменной products.
- **Фильтрация и сортировка**: Применяются к данным перед отрисовкой.
- **LocalStorage**: Хранит состояние корзины и избранного.
- **UI**: Динамически обновляется на основе данных и состояния.