Задание: Анимированная карточка товара.

Цель: Научиться создавать UI-компоненты с помощью классов в JavaScript, управлять DOM, событиями и анимациями, правильно разделяя публичные и приватные свойства и методы.

Навыки для закрепления:

- создание классов и приватных полей;
- построение DOM-элементов через createElement;
- работа с событиями (mouseenter, mouseleave, click);
- управление состоянием компонента через классы и CSSанимации;
- организация кода: разграничение публичного и приватного API класса.

Базовый функционал:

- На странице отображается сетка карточек товаров.
- Каждая карточка — это отдельный экземпляр класса ProductCard.
- При наведении мыши карточка должна анимированно подниматься (hover-эффект).

Класс ProductCard.

Приватные поля:

- #data — объект с данными о товаре (title, price, image).
- #element — DOM-элемент карточки.

Приватные методы:

- #attachEvents() — навешивает обработчики событий (mouseenter, mouseleave, click).
- #handleMouseEnter() — добавляет CSS-класс для анимации.
- #handleMouseLeave() — убирает CSS-класс.

Публичные методы:

- constructor(data, parentElement) — принимает объект с данными и контейнер, куда будет добавляться карточка.
- render() — создаёт DOM-структуру карточки (с <img>, <h3>, <p>), вставляет её в контейнер.

```
class ProductCard {
// приватные поля
  #data;
  #element;

  // публичный конструктор   constructor(data, parentElement) {
this.#data = data;               // сохраняем данные о товаре
    this.parentElement = parentElement; // сохраняем контейнер (публично)
}

  // публичный метод: рендер карточки
render() {
    // TODO: создать div.product-card
    // TODO: добавить внутрь img, h3, p
    // TODO: сохранить div в #element
    // TODO: вызвать #attachEvents()
    // TODO: добавить карточку в parentElement
  }

  // приватный метод: навешивание событий
  #attachEvents() {
    // TODO: слушатели mouseenter и mouseleave
    // TODO: при наведении вызвать this.#handleMouseEnter()
    // TODO: при уходе вызвать this.#handleMouseLeave()
  }

  // приватный метод: обработка наведения
  #handleMouseEnter() {
    // TODO: добавить CSS-класс (например, 'hovered')
  }

  // приватный метод: обработка ухода мыши
  #handleMouseLeave() {
    // TODO: убрать CSS-класс
  }
}
```

Интерфейс страницы:

- В HTML создать контейнер .
- Карточки должны отображаться в виде сетки (через CSS display: grid или flex).
- Внутри карточки обязательно должны быть:
    - <img> — изображение товара; ○ <h3> — название товара; ○ <p> — цена товара.

Пример данных:

```
const products = [
    { title: "Наушники X100", price: "49900 ₸", image:
"https://m.mediaamazon.com/images/I/61-0dR5fsdL._AC_UY654_QL65_.jpg" },
    { title: "Игровая мышь GX200", price: "29900 ₸", image:
"https://compress.ru/img/post/2015/03/30/genius-gx-gaming-gila.jpg" },
    { title: "Клавиатура ProKey", price: "39900 ₸", image:
"https://extremecomp.ru/media/img/330000/337182_v01_b.jpg" } ];
```

Дополнительные задания:

- Кнопка "Добавить в корзину":
    - добавить кнопку внутри карточки; ○ при клике карточка диспатчит кастомное событие 'addToCart', в detail которого передаётся товар;
    - на уровне документа (document.addEventListener('addToCart', ...)) — вывести сообщение в консоль.

- Улучшенный UI:
    - плавная анимация при наведении (например, transition:
      transform 0.3s ease;);
    - выделение активной карточки (подсветка рамки или тень).

Требования к коду:

- Каждая карточка должна быть отдельным экземпляром класса.

- Все данные о товаре должны храниться в приватном поле #data.
- Ссылку на DOM-элемент хранить в приватном поле #element.
- Все обработчики событий навешивать только внутри #attachEvents().
- Внешний код должен управлять карточкой только через публичные методы (constructor, render).

Итоговое поведение:

- На странице выводится несколько карточек товаров.
- При наведении они красиво «поднимаются».
- При клике по кнопке «Добавить в корзину» в консоль выводится объект товара.