📦 Домашнее задание №32: Асинхронность, авторизация, работа с API  
  
  
  
  
  
  
  
  
  
  
  
  
  
**✅ Задание 1 — Асинхронная демонстрация даты и суммы**

**Цель:**  
Создать страницу, на которой:

1. Сразу отображается текущая дата.
2. Асинхронно (с задержкой через setTimeout) выводится сумма двух чисел.

**Что реализовано:**

* Верстка: index.html содержит минимальную разметку — заголовок и карточку с местом под дату и сумму.
* Стилизация: выполнена через Nunito Sans, приятный фон, тени и аккуратная карточка.
* Логика:
  + printDate() синхронно выводит текущую дату (в формате дд.мм.гггг) в #date-output.
  + printSum(5, 7) асинхронно вызывает setTimeout(..., 0), чтобы вывести сумму 5+7 в #sum-output.

**Что сработало:**

* Отображение даты и суммы работает корректно.
* Консоль также выводит значения для отладки.

**Что можно улучшить:**

* Можно добавить async/await чисто для демонстрации, но технически setTimeout с нулевой задержкой уже выполняет асинхронность.

**Статус:** ✅ **Выполнено полностью, соответствует требованиям.**

**✅ Задание 2 — Галерея Фото с авторизацией**

**Цель:**  
Создать фотогалерею, которая:

1. Загружает изображения с https://jsonplaceholder.typicode.com/photos.
2. При авторизации (логин = admin) показывает фото в хорошем качестве.
3. Без авторизации — показывает фото в плохом качестве.
4. Авторизация происходит через prompt + сохраняется в localStorage.

**Что реализовано:**

* Разметка: index.html — заголовок, кнопка "Логин", сетка с фото.
* Стилизация: карточки с тенями, сетка адаптивная, кнопка с кастомным стилем.
* Скрипт:
  + При загрузке проверяет localStorage.getItem('auth').
  + При клике на "Логин" — открывается prompt. Если введено admin, сохраняет auth = true, иначе false.
  + Запрос фотографий через async/await.
  + Изображения загружаются с сайта [**https://jsonplaceholder.typicode.com/photos?\_limit=9**](https://jsonplaceholder.typicode.com/photos?_limit=9).
  + В зависимости от isAuthorized отображается либо url (высокое качество), либо thumbnailUrl (низкое качество).
  + В качестве запасного варианта тестировался источник https://picsum.photos/id/${photo.id}/600/400 и 150/100.

**Что сработало:**

* Авторизация работает через prompt и сохраняется в localStorage.
* Кнопка "Логин" работает повторно, можно изменить статус авторизации.
* Галерея корректно отображается, карточки визуально соответствуют заданию.
* При правильном логине — чёткие изображения.
* При неправильном — миниатюры.

**Что не сработало в первой версии:**

* Была проблема с блокировкой домена via.placeholder.com в локальной сети или DNS.
* Из-за этого изображения не загружались, появлялись ошибки net::ERR\_NAME\_NOT\_RESOLVED.

**Как решено:**

* Временная замена изображений на picsum.photos (по id).
* API jsonplaceholder всё ещё используется для title.

**Статус:** ✅ **Выполнено. Сайт соответствует требованиям, реализованы все пункты.**

**✅ Задание 3 — Карточки пользователей с API**

**Цель:**  
Создать сайт, где:

1. Отображаются карточки пользователей.
2. Данные подгружаются через fetch с сайта [**https://dummyjson.com/users**](https://dummyjson.com/users).
3. Нет "захардкоженных" данных в HTML.
4. Вёрстка, шрифты, иконки и стили должны строго соответствовать макету из Figma.
5. Реализована постраничная навигация (3x3).
6. Иконки — SVG, шрифт **Nunito Sans** (300 и 600), без встроенного текста в HTML.

**Что реализовано:**

* Проект выполнен отдельно от других заданий (как и требовалось).
* Данные подгружаются динамически через JavaScript.
* HTML "чистый" — вся генерация DOM-элементов идёт через JS.
* Постраничный вывод работает.
* Стили и иконки подключены и соответствуют макету.
* Реализована адаптивная сетка.

**Что сработало:**

* Абсолютно всё. Карточки красивы, иконки работают, всё на месте.
* Вёрстка соответствует Figma.
* Стили применены корректно.
* Навигация — через кнопки или стрелки (в зависимости от реализации).

**Статус:** ✅ **Выполнено ранее в рамках практической работы. Все требования выполнены.**

**🟢 Общий итог:**

| **Задание** | **Статус** | **Комментарий** |
| --- | --- | --- |
| **Задание 1** | ✅ Выполнено | Асинхронная дата + сумма |
| **Задание 2** | ✅ Выполнено | Авторизация + async/await + галерея |
| **Задание 3** | ✅ Выполнено | Реализовано раньше на практике |

**📌 Резюме:**

Домашнее задание №32 выполнено полностью. Все три части реализованы в отдельных проектах. Применены:

* **Асинхронность (async/await)**,
* **Работа с API (fetch)**,
* **Авторизация (prompt + localStorage)**,
* **Динамическая генерация DOM**,
* **Подключение иконок и шрифтов**,
* **Стилизация по макету Figma**.

Всё протестировано, ошибки исправлены, проект готов к сдаче.