# Практическая работа GitHub №3. Репозиторий с конспектом

## Подготовка:

### Задание 1. Выполните задание «[Опорный конспект по GitHub ч.1](#_2g7ci1pqyty2)»

### Задание 2. Совместно с преподавателем составьте mindmap по теме «Основы JS»

## Практическая работа:

### Задание 1. Создание репозитория для практической работы.

Следуйте инструкции:

1. Создание нового репозитория на GitHub:
   1. Перейдите на страницу GitHub (<https://github.com/>) и войдите в свой аккаунт.
   2. В правом верхнем углу нажмите на "+" и выберите "New repository".
   3. Заполните информацию о репозитории:
      1. Название: «summary-<surname>-<group-number>»
      2. Описание:

« Student's abstract by <name> <surname>, Group No. <group-number>, on the subject "Applied Programming.Hexlet College, Academic Year 2023-2024.»

* + 1. Переведите описание репозитория на русский язык и продублируйте ниже англоязычной версии.
    2. Нажмите "Create repository".

1. Инициализация (создание) локального репозитория:
   1. Откройте терминал на своем устройстве.
   2. Создайте директорию (папку) с названием практики с помощью следующей команды: mkdir summary-<surname>-<group-number>
   3. Перейдите в каталог проекта с помощью проекта: cd путь/к/вашему/проекту.
   4. Выполните команду git init, чтобы инициализировать новый локальный репозиторий.
2. Добавление файлов и создание коммита:
   1. Создайте новый файл или внесите изменения в существующий: touch test.js
   2. Откройте файл в VSCode и отредактируйте: файл должен содержать функцию вывода с текстом «test finished».
   3. Запустите файл в консоли с помощью команды node.
   4. Выполните git add -A, чтобы добавить все изменения в индекс.
   5. Выполните git commit -m "First commit" для создания первого коммита.
3. В директории проекта создайте папку images и поместите в нее изображения из архива images.zip.
4. Связывание с удаленным репозиторием на GitHub:
   1. Скопируйте SSH вашего репозитория на GitHub.
   2. Выполните git remote add origin <SSH вашего репозитория>, чтобы связать локальный репозиторий с удаленным на GitHub.
5. Отправка изменений на GitHub:
   1. Выполните git push, чтобы отправить ваши изменения на GitHub.
   2. Если терминал предлагает вам выполнить другую конфигурацию команды git push, следуйте инструкции в терминале.
   3. Зайдите в репозиторий на GitHub и проверьте, что репозитории синхронизировались.

### 

### Задание 2. Создание главной страницы конспекта.

1. Создайте в директории проекта файл index.html и откройте его в VSCode.
2. Сверстайте макет по следующему ТЗ:
   1. Язык страницы – русский, кодировка UTF-8. Название страницы «Contents».
   2. Шапка сайта должна содержать: логотип (файл logo.png), Фамилия Имя студента, Номер группы, Имя преподавателя.
   3. В теле страницы:
      1. Заголовок 1 уровня с текстом «Конспект по предмету «Прикладное программирование»
      2. Заголовок 2 уровня с текстом «Оглавление»
      3. Обертки с разделами конспекта: Основы HTML, Основы CSS, Основы JS, GitHub, Linux. Название раздела – заголовок 3 уровня, в котором содержится ссылка на страницу с разделом, например <h3><a href=”basic-css.html>Основы CSS</a></h3>.

Добавьте в обертки изображения, соответствующие названию раздела, дополнительно можно добавить краткое описание раздела.

* 1. Футер содержит дату создания данной страницы и ссылку на сайт Хекслет.

1. Создайте в директории проекта файл main-style.css и откройте его в VSCode.
2. Задайте макету стили по следующему ТЗ:
   1. Вы можете добавлять свои стили к нижеуказанным.
   2. Выберите 2 цветовые гаммы по схеме «Триада» и «Контрастная триада». Одна схема используется для стилизации страницы, вторая – для стилизации блоков.
   3. Создайте в директории проекта папку fonts и скачайте в нее шрифты Roboto, Roboto Mono, Montserrat, SFMono. Подключите шрифт к файлу стилей.
   4. Стили body:
      1. Сбросьте внешние отступы;
      2. Размер шрифта равен 18px, насыщенность 400, межстрочный интервал 1.5;
      3. Цвет текста #212529;
   5. Стили шапки:
      1. Размер логотипа 50px/50px;
      2. Шрифт Roboto;
      3. Размер шрифта равен 0.8 от родительского;
   6. Стили заголовков:
      1. Шрифт Montserrat;
      2. Размеры шрифтов: для заголовка 1 уровня 1.5 от родителя, для заголовка 2 уровня равен родителю;
      3. Выравнивание по левому краю;
      4. Внутренние отступы 6px по вертикали и 10px по горизонтали;
   7. Стили обертки для блоков с разделами:
      1. Высота 1640px;
      2. Ширина 1180px;
      3. Блоки размещены в 3 колонки;
   8. Стили блоков с разделами:
      1. Шрифт повторяет шрифт заголовка;
      2. Размер шрифта заголовка блока равен родителю;
      3. Цвета фонов блоков соответствуют цветам схемы «Контрастная триада»;
      4. Ссылки внутри заголовков не имеют подчеркивание, меняют цвет на контрастный для цвета блока при наведении.
      5. Блоки имеют закругление 24px:
      6. При наведении у блока появляется тень;
      7. Высота блока 580px, ширина 530px;
      8. Размер изображений внутри блока составляет не более 20% от его высоты;
      9. Если вы используете описание блока, укажите для него нормальную насыщенность.
   9. Стили футера:
      1. Футер находится в нижней части экрана;
      2. Дата размещается в правом углу футера;
      3. Размер шрифта 0.7 от родителя;
      4. Ссылка на Хекслет размещается по центру футера, имеет иконку и текст «Хекслет».

### Задание 3. Создание страницы Основы GitHub

1. Создайте в директории проекта файл basic-github.html и откройте его в VSCode.
2. Создайте шапку и футер аналогичный главной странице.
3. В тело страницы добавьте кнопку «На главную», которая ведет к файлу index.html.
4. Добавьте заголовок страницы, соответствующий названию раздела.
5. Оформите текст, полученный в результате выполнения задания [«Опорный конспект по GitHub ч.1»](#_2g7ci1pqyty2), в качестве макета страницы. Данная страница будет пополняться в будущем по мере изучения работы с GitHub, вы всегда можете использовать ее как опорный конспект;
6. Стили страницы:
   1. Используйте шрифт Roboto;
   2. Установите нормальную насыщенность для основного текста;
   3. Размер шрифта равен родительскому (см. стили файла index.html);
   4. Межстрочный интервал 1.7;
   5. Сохраняйте выделение текста жирным начертанием согласно заданию «Опорный конспект по GitHub ч.1»;
   6. Укажите внешний отступ для элемента страницы снизу 16px, остальные сбросьте;
   7. Все фрагменты кода/команд терминала выделены в отдельный блок:
      1. Фон блока имеет цвет #6d757d;
      2. Внутренние отступы 16px;
      3. Укажите внешний отступ снизу 16px, остальные сбросьте;
      4. Шрифт SFMono;
      5. Размер шрифта 0.85 от родителя;
      6. Цвет текста внутри блока:
         1. Цвет команд: #24292E;
         2. Цвет аргументов команд: #009999;
         3. Цвет комментариев: #999988.
      7. Выделите начало каждого логического блока (это может быть стиль первой буквы, маркер, выделение цветом и т.д.).

### Задание 4. Создание страницы Основы JS

1. Создайте в директории проекта файл basic-js.html и откройте его в VSCode.
2. Создайте шапку и футер аналогичный главной странице.
3. В тело страницы добавьте кнопку «На главную», которая ведет к файлу index.html.
4. Добавьте заголовок страницы, соответствующий названию раздела.
5. Воссоздайте схему mindmap по теме «Основы JS».
6. Используйте стили из заданий выше. Помните о цветовых схемах.

## Задание «Опорный конспект по GitHub ч.1»

### Составьте понятное вам определение исходя из аналогии:

**GitHub:**

- Объяснение: GitHub - это платформа \_хостинга, хранения и совместной разработки \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Здесь разработчики могут хранить свои \_\_\_проекты\_\_\_\_\_, работать над ними в \_комманде, парраленльно, самостоятельно\_\_\_\_\_\_\_\_\_\_ и отслеживать \_\_обновления\_\_\_\_\_\_\_.

- Аналогия: GitHub, как облачное хранилище проектов, подобен социальной сети для разработчиков, где они вместе строят и совершенствуют свои творения.

**SSH-ключ:**

- Объяснение: SSH-ключ - это способ безопасной аутентификации\_\_\_\_\_\_\_ на \_\_\_\_\_облачном/удаленном\_\_\_\_\_ сервере. Он используется для подключения к GitHub, например, без постоянного ввода пароля.

- Аналогия: SSH-ключ, как сканер отпечатка пальца, позволяет вам разблокировать устройство (подключаться к серверу) без необходимости запоминать пароль.

**Репозиторий:**

- Объяснение: Репозиторий - это место, где \_\_хранится код проекта\_\_\_\_\_\_\_\_, вместе со всей его историей изменений\_\_\_\_\_\_\_\_\_\_.

- Аналогия: Репозиторий, как папка с проектом, подобен архиву, где хранятся все чертежи, планы и изменения к вашему главному творению.

**Коммит:**

- Объяснение: Коммит - это сохранение изменений в коде проекта. Он создает точку в истории, которую можно \_\_сохранить\_\_\_\_\_\_ и \_вернуться\_\_\_\_\_\_\_ к ней.

- Аналогия: Коммит, как сохранение в игре, фиксирует текущее состояние, чтобы в случае чего можно было вернуться к нему. **Ветка:**

- Объяснение: Ветка - это отдельная линия разработки, которая может быть создана для работы над новой \_версией/функцией\_\_\_\_\_\_\_\_\_или \_исправлением\_\_\_\_\_\_\_\_\_, не затрагивая \_\_основной\_\_\_\_\_\_\_\_ код проекта.

- Аналогия: Ветка, как отдельный поток в разговоре, позволяет вам обсуждать и вносить изменения, не прерывая основной диалог.

### Заполните пропуски в инструкциях:

**1. Как сгенерировать SSH-ключ и добавить к себе на GitHub:**

* **Шаги:**
  1. Откройте терминал и выполните **\_** **ssh-keygen -t rsa -b 4096 papka/.ssh/** **id\_rsa.pub \_\_\_\_\_\_\_\_\_\_\_**
  2. Нажмите Enter, чтобы принять стандартное расположение файла.
  3. Введите пароль или нажмите Enter, чтобы оставить его пустым.
  4. Выполните **\_\_\_ papka/.ssh/** **id\_rsa.pub \_\_\_\_\_\_\_\_\_\_\_\_**, чтобы скопировать открытый ключ.
  5. Перейдите в настройки **GitHub -> SSH and GPG keys -> New SSH key -> вставьте ключ**.

**2. Как добавить к себе чей-то репозиторий:**

* **Шаги:**
  1. Скопируйте **URL** репозитория с GitHub.
  2. В терминале выполните **\_\_git clone\_\_\_\_\_\_\_ <URL репозитория>**.

**3. Как добавить на свой GitHub склонированный репозиторий:**

* **Шаги:**
  1. Создайте пустой репозиторий на GitHub.
  2. Скопируйте его \_ssh\_\_-ссылку.
  3. Войдите в директорию склонированного репозитория в терминале.
  4. Выполните **\_\_** **git remote set-url origin <ссылка на ваш репозиторий> \_\_\_\_\_\_\_\_\_\_\_.**
  5. Выполните **git push**.

**4. Как сохранить изменения на GitHub (сделать коммит):**

* **Шаги:**
  1. Выполните **\_git add -A / git add\_\_\_\_\_\_\_\_\_\_**, чтобы добавить все изменения.
  2. Выполните **\_\_git commit -m\_\_\_\_\_\_\_\_\_ "Описание коммита"**.
  3. Выполните **\_git push\_\_\_\_\_\_\_\_\_\_**, чтобы отправить изменения на GitHub.

**5. Как работать со своим репозиторием с другого устройства:**

* **Шаги:**
  1. Склонируйте свой репозиторий с GitHub на другое устройство: **\_\_git clone\_\_\_\_\_\_\_ <URL вашего репозитория>**.
  2. Вносите изменения, коммитите и отправляйте их обратно на GitHub.

**6. Как открыть VSCode из терминала Linux:**

* **Шаги:**
  1. В терминале выполните **\_\_code <filename>\_\_\_\_ .** для открытия текущего каталога в VSCode.

**7. Как запустить программу в терминале (через Node.js):**

* **Шаги:**
  1. Введите **\_node\_filename.js\_\_\_\_\_\_\_\_ имя\_файла.js**, чтобы выполнить программу Node.js в терминале.

**8. Как создать ветку:**

* **Шаги:**
  1. Выполните **\_\_git branch\_\_\_\_\_\_\_\_ новая\_ветка**, чтобы создать новую ветку.
  2. Выполните **\_\_git checkout -b <new branch>\_\_\_\_\_\_\_\_ новая\_ветка**, чтобы переключиться на новую ветку.
  3. Или используйте **git checkout -b новая\_ветка** для создания и переключения на новую ветку сразу.

### Часто задаваемые вопросы:

1. **В каком случае нужно генерировать новый SSH-ключ:**
   * Когда вы впервые начинаете работу с GitHub.
   * При использовании нового компьютера или сервера для работы с репозиториями.
   * Когда старый ключ утерян или скомпрометирован.

1.1 **Могу ли я использовать один и тот же SSH ключ на разных устройствах:**

* Да, вы можете использовать один и тот же SSH ключ на разных устройствах. Это удобно при работе с несколькими устройствами.

1. **В каких случаях надо использовать git config user.name (user.email) <аргументы>:**
   * При первой настройке Git на новом устройстве.
   * При изменении вашего имени или электронной почты.
   * Когда вам нужно использовать разные имена или адреса электронной почты для разных проектов.
2. **В каком случае нужно создавать ветку и для чего она нужна:**
   * Новая функциональность: создайте ветку для работы над новой функцией, чтобы не затрагивать основной код.
   * Исправление ошибок: создайте ветку для исправления ошибок, чтобы избежать конфликтов с основной разработкой.
   * Разработка параллельных версий: создайте ветку для разработки и тестирования новых идей без влияния на стабильный код.
3. **Что писать в комментарии для git commit:**
   * Краткое, но информативное описание изменений, чтобы понять суть коммита.
   * Укажите, какие задачи решает этот коммит.
   * Следуйте принципу "Как" и "Почему" для более полного объяснения.
4. **Что такое Pull Request и зачем он нужен:**
   * Pull Request (или Merge Request) - это запрос на внесение изменений из вашей ветки в основную ветку проекта.
   * Используется для обсуждения и рецензии ваших изменений перед их объединением с основным кодом.
   * Позволяет совершать изменения согласованно и избегать конфликтов при слиянии кода.