**Практическая работа GitHub №3. Репозиторий с конспектом**

**Подготовка:**

**Задание 1. Выполните задание «Опорный конспект по GitHub ч.1»**

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

**Практическая работа:**

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

Следуйте инструкции:

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

«Student's abstract by <name> <surname>, Group No. <group-number>, on the subject "Applied Programming."

Hexlet College, Academic Year 2023-2024.»

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

**Задание 2. Создание главной страницы конспекта.**

* Создайте в директории проекта файл index.html и откройте его в VSCode.
* Сверстайте макет по следующему ТЗ:
* Язык страницы – русский, кодировка UTF-8. Название страницы «Contents».
* Шапка сайта должна содержать: логотип (файл logo.png), Фамилия Имя студента, Номер группы, Имя преподавателя.
* В теле страницы:
* Заголовок 1 уровня с текстом «Конспект по предмету «Прикладное программирование»
* Заголовок 2 уровня с текстом «Оглавление»
* 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.

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

**Задание 3. Создание страницы Основы GitHub**

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

**Задание 4. Создание страницы Основы JS**

* Создайте в директории проекта файл basic-js.html и откройте его в VSCode.
* Создайте шапку и футер аналогичный главной странице.
* В тело страницы добавьте кнопку «На главную», которая ведет к файлу index.html.
* Добавьте заголовок страницы, соответствующий названию раздела.
* Воссоздайте схему mindmap по теме «Основы JS».
* 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 **команде, самостоятельно, параллельно** и отслеживать **изменения, 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 (подключения)** на **удалённом облачном** 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 (обновлений)**.

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

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

* **Шаги:**
* Скопируйте **SSH** репозитория с GitHub (зелёная кнопка Code).
* В терминале выполните **git clone <SSH репозитория>**.

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

* **Шаги:**
* Создайте пустой репозиторий на GitHub.
* Скопируйте его **SSH-ссылку**.
* Войдите в директорию склонированного репозитория в терминале (например, вы склонировали репозиторий Николая).
* Выполните **git remove set-url origin <SSH-link>.**
* Выполните **git push**.

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

* **Шаги:**
* Выполните **git add -A/ git add - -all**, чтобы добавить все изменения.
* Выполните **git commit -m "Описание коммита"**.
* Выполните **git push**, чтобы отправить изменения на GitHub.

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

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

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

* **Шаги:**
* В терминале выполните **code <имя-файла(папки)> .** для открытия текущего каталога в VSCode.

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

* **Шаги:**
* Введите **node имя\_файла.js**, чтобы выполнить программу Node.js в терминале.

**8. Как создать ветку:**

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

**Часто задаваемые вопросы:**

* **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.
* **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.
* **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.
* **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.
* **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.