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:

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

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

Hexlet College, Academic Year 2023-2024.»

        1.3.3. Переведите описание репозитория на русский язык и продублируйте ниже англоязычной версии.
        1.3.4. Нажмите "Create repository".
2. Инициализация (создание) локального репозитория:
    2.1. Откройте терминал на своем устройстве.
    2.2. Создайте директорию (папку) с названием практики с помощью следующей команды: mkdir summary-<surname>-<group-number>

2.3. Перейдите в каталог проекта с помощью проекта: cd путь/к/вашему/проекту.

2.4. Выполните команду git init, чтобы инициализировать новый локальный репозиторий.

3. Добавление файлов и создание коммита:

3.1. Создайте новый файл или внесите изменения в существующий: touch test.js

3.2. Откройте файл в VSCode и отредактируйте: файл должен содержать функцию вывода с текстом «test finished».

3.3. Запустите файл в консоли с помощью команды **node test.js**

3.4. Выполните git add -A, чтобы добавить все изменения в индекс.

3.5. Выполните git commit -m "First commit" для создания первого коммита.

4. В директории проекта создайте папку images и поместите в нее изображения из архива images.zip.

5. Связывание с удаленным репозиторием на GitHub:

5.1. Скопируйте SSH вашего репозитория на GitHub.

5.2. Выполните git remote add origin <SSH вашего репозитория>, чтобы связать локальный репозиторий с удаленным на GitHub.

6. Отправка изменений на GitHub:

6.1. Выполните git push, чтобы отправить ваши изменения на GitHub.

6.2. Если терминал предлагает вам выполнить другую конфигурацию команды git push, следуйте инструкции в терминале.

6.3. 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:
   2.1. Язык страницы – русский, кодировка UTF-8. Название страницы «Contents».
   2.2. Шапка сайта должна содержать: логотип (файл logo.png), Фамилия Имя студента, Номер группы, Имя преподавателя.
   2.3. В теле страницы:
      2.3.1. Заголовок 1 уровня с текстом «Конспект по предмету «Прикладное программирование»
      2.3.2. Заголовок 2 уровня с текстом «Оглавление»
      2.3.3. 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>Основы CSS</h3>.

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

   2.4. Футер содержит дату создания данной страницы и ссылку на сайт Хекслет.
3. Создайте в директории проекта файл main-style.css и откройте его в VSCode.
4. Задайте макету стили по следующему ТЗ:
   4.0. Вы можете добавлять свои стили к нижеуказанным.
   4.1. Выберите 2 цветовые гаммы по схеме «Триада» и «Контрастная триада». Одна схема используется для стилизации страницы, вторая – для стилизации блоков.
   4.2. Создайте в директории проекта папку fonts и скачайте в нее шрифты Roboto, Roboto Mono, Montserrat, SFMono. Подключите шрифт к файлу стилей.
   4.3. Стили body:
      4.3.1. Сбросьте внешние отступы;
      4.3.2. Размер шрифта равен 18px, насыщенность 400, межстрочный интервал 1.5;
      4.3.3. Цвет текста #212529;

4.4. Стили шапки:

4.4.1. Размер логотипа 50px/50px;

4.4.2. Шрифт Roboto;

4.4.3. Размер шрифта равен 0.8 от родительского;

4.5. Стили заголовков:

4.5.1. Шрифт Montserrat;

4.5.2. Размеры шрифтов: для заголовка 1 уровня 1.5 от родителя, для заголовка 2 уровня равен родителю;

4.5.3. Выравнивание по левому краю;

4.5.4. Внутренние отступы 6px по вертикали и 10px по горизонтали;

4.6. Стили обертки для блоков с разделами:

4.6.1. Высота 1180px;

4.6.2. Ширина 1640px;

4.6.3. Блоки размещены в 3 колонки;

4.7. Стили блоков с разделами:

4.7.1. Шрифт повторяет шрифт заголовка;

4.7.2. Размер шрифта заголовка блока равен родителю;

4.7.3. Цвета фонов блоков соответствуют цветам схемы «Контрастная триада»;

4.7.4. Ссылки внутри заголовков не имеют подчеркивание, меняют цвет на контрастный для цвета блока при наведении.

4.7.5. Блоки имеют закругление 24px:

4.7.6. При наведении у блока появляется тень;

4.7.7. Высота блока 580px, ширина 530px;

4.7.8. Размер изображений внутри блока составляет не более 20% от его высоты;

4.7.9. Если вы используете описание блока, укажите для него нормальную насыщенность.

4.8. Стили футера:

4.8.1. Футер находится в нижней части экрана;

4.8.2. Дата размещается в правом углу футера;

4.8.3. Размер шрифта 0.7 от родителя;

4.8.4. 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», 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:

   6.1. Используйте шрифт Roboto;

   6.2. Установите нормальную насыщенность для основного текста;

   6.3. Размер шрифта равен родительскому (см. стили файла index.html);

   6.4. Межстрочный интервал 1.7;

   6.5. Сохраняйте выделение текста жирным начертанием согласно заданию «Опорный конспект по GitHub ч.1»;

   6.6. Укажите внешний отступ для элемента страницы снизу 16px, остальные сбросьте;

   6.7. Все фрагменты кода/команд терминала выделены в отдельный блок:

      6.7.1. Фон блока имеет цвет #6d757d;

      6.7.2. Внутренние отступы 16px;

      6.7.3. Укажите внешний отступ снизу 16px, остальные сбросьте;

      6.7.4. Шрифт SFMono;

      6.7.5. Размер шрифта 0.85 от родителя;

      6.7.6. Цвет текста внутри блока:

6.7.6.1. Цвет команд: #24292E;

6.7.6.2. Цвет аргументов команд: #009999;

6.7.6.3. Цвет комментариев: #999988.

6.7.7. 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. Добавьте заголовок раздела “Шпаргалка”.
6. Воссоздайте схему mindmap по теме «Основы JS». Схема на вашей странице не обязана в точности повторять схему по ссылке. Это может быть таблица или схема другого вида. Для желающих повторить именно в виде схемы, можно посмотреть тут несложный вариант верстки.
7. 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 разработке**. 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 (подключение)** на **удаленном (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**.

- 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**
    2. Нажмите Enter, чтобы принять стандартное расположение файла.
    3. Введите пароль или нажмите Enter, чтобы оставить его пустым.
    4. Выполните **cat ./.ssh/id_rsa.pub**, чтобы скопировать (ПКМ) 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 (зеленая кнопка Code).
    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 (например, вы склонировали репозиторий Николая).
    4. Выполните **git remote set-url origin <SSH-link>.**
    5. Выполните **git push**.

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

- **Шаги:**
    1. Выполните **git add -A/ git add - -all**, 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. * VSCode - cmd+Shift+P - “shell” - для Mac
    2. В терминале выполните **code <имя-файла(папки)>.** для открытия текущего каталога в VSCode.

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

- **Шаги:**
    1. Введите **node 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 <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.

2. **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.
3. **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.
4. **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.
5. **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.