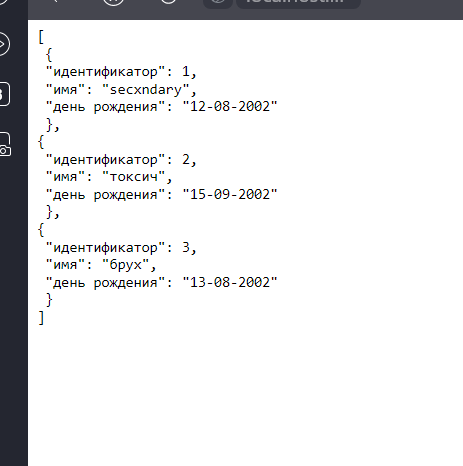
Лабораторная работа 04

ПвИ

**Задание 01**

1. Разработайте серверное приложение **04-01**, которое принимает HTTP-запросы **GET**, **POST**, **PUT**, **DELETE** на [**http://localhost:5000/api/db**](http://localhost:5000/api/db).



1. Назначение запросов.

|  |  |
| --- | --- |
| **GET** | Получить все строки таблицы базы данных (БД). В теле ответа возвращается массив строк в json-формате. |
| **POST** | Добавить новую строку в таблицу БД; строка передается серверу в json-формате.  В теле ответа возвращается измененная строка в json-формате. |
| **PUT** | Изменить существующую строку таблицы БД; строка передается серверу в json-формате. |
| **DELETE** | Удалить существующую строку (по id) в таблице БД; серверу передается id (в query-строке). В теле ответа возвращается удаленная строка в json-формате.  Пример: [**http://localhost:5000/api/db?id=25**](http://localhost:5000/api/db?id=25) |

1. Для работы с БД в приложении должен быть разработан специальный объект **DB**, который должен располагаться в отдельном модуле.
2. Объект **DB** должен уметь генерировать и ***асинхронно*** обрабатывать следующие события

|  |  |
| --- | --- |
| **GET** | Получить все строки таблицы базы данных (БД). |
| **POST** | Добавить новую строку в таблицу БД. |
| **PUT** | Изменить существующую строку таблицы БД. |
| **DELETE** | Удалить существующую строку в таблице БД. |

1. Для выполнения операций с БД объект **DB** должен обеспечивать следующие функции

|  |  |
| --- | --- |
| **select** | Получить все строки таблицы базы данных (БД). |
| **insert** | Добавить новую строку в таблицу БД. |
| **update** | Изменить существующую строку таблицы БД. |
| **delete** | Удалить существующую строку в таблице БД. |

1. Строка таблицы содержит:

- идентификатор (**id**);

- имя (**name**);

- дата рождения (**bday**).

1. Для имитации таблицы БД используйте JS-массив, а для имитации операций с БД соответствующие операции с массивом.
2. Проверьте работоспособность приложения с помощью **POSTMAN**.

**Задание 02**

1. Разработайте серверное приложение **04-02**, использовав код приложения **04-01**.
2. Дополнительно к функциональности, унаследованной от **04-01,** приложение **04-02** должно отвечать на GET-запросы к [**http://localhost:5000/**](http://localhost:5000/)**.**
3. В ответ на GET-запрос к [**http://localhost:5000/**](http://localhost:5000/)сервер пересылает клиенту HTML-страницу, которая должна располагаться в той же директории, что и js-файл приложения**.**
4. HTML-страница должна обеспечивать ввод необходимых данных, отправку всех типов запросов к серверу и обработку полученных результатов с применением функции **fetch**.
5. Проверьте работоспособность приложения с помощью браузера.

**Задание 03.**Ответьте на следующие вопросы.

1. Дайте пояснение понятию «событие программного объекта».
2. Какой встроенный механизм используется в **Node.js** для генерации и обработки событий. Поясните принцип его работы.

14. Дайте пояснение понятию «событие программного объекта».

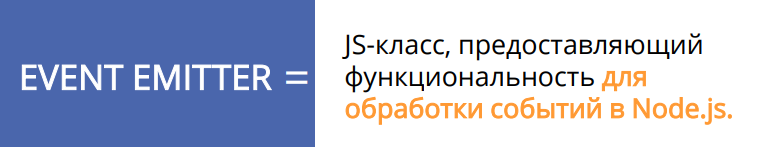
Это процесс перехода объекта из одного состояния в другое, при этом, другие программные объекты могут быть извещены о том, что этот переход произошел. У каждого события есть издатель (генератор, инициатор) и подписчик (обработчик, слушатель).

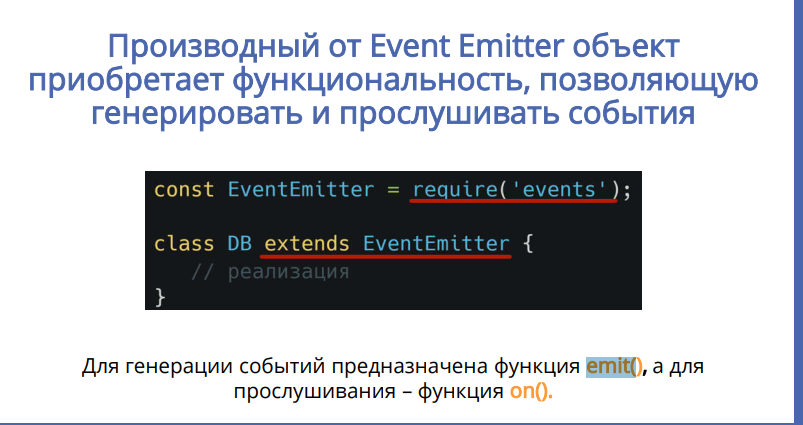
15. Какой встроенный механизм используется в **Node.js** для генерации и обработки событий. Поясните принцип его работы.

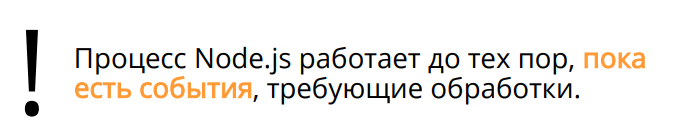
Для этого используется стандартный модуль EventEmitter

Необходимо подключение двух модулей: events и util. Если наследоваться от EventEmitter’a через class extends или inherits. Для генерации события используется emit, а для прослушивания и обработки – on.

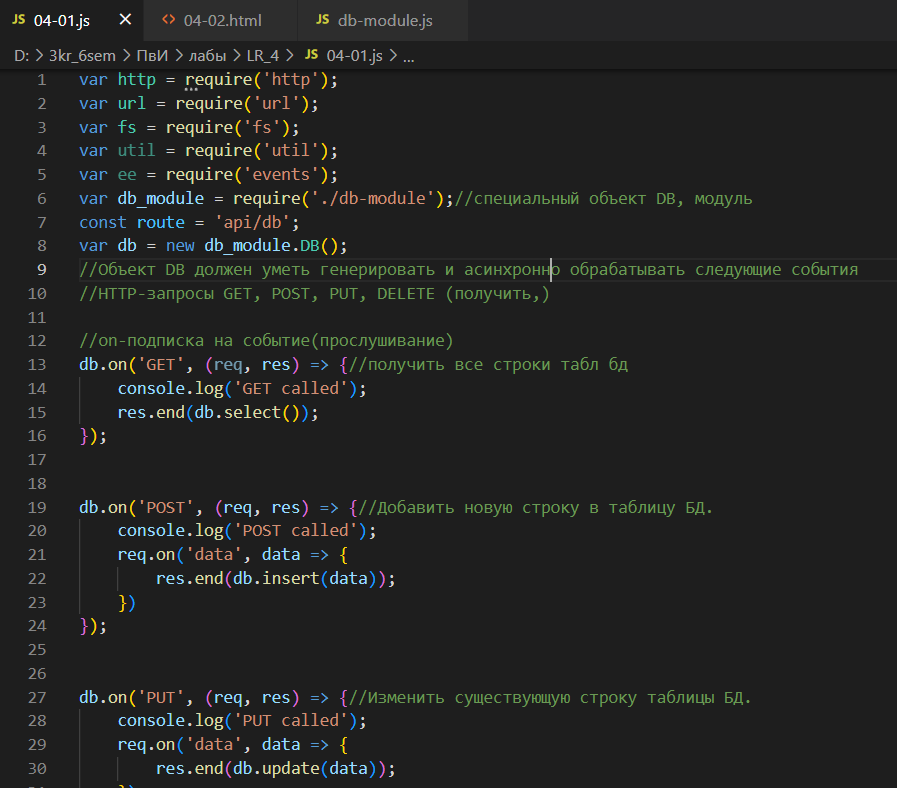
Все объекты, которые генерируют события, представляют экземпляры класса EventEmitter.

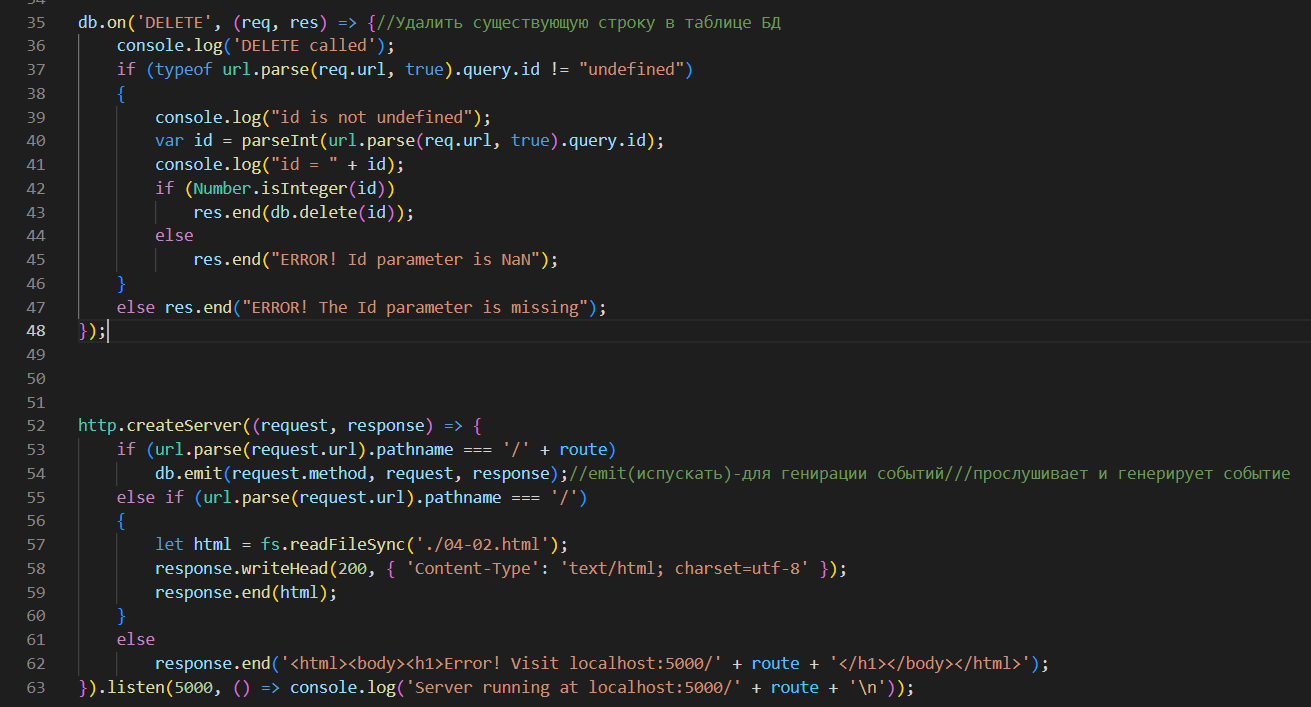






1.





Модуль

