**Package.json** нужен для того, чтобы легко переносить проект, из этого файла установится все, что нужно.

**"dependencies":** это те модули, которые будут установлены с минимальной версией

**express** – сервер развернуть

**body-parser** – расшифровывать данные из формы, которые мы получаем

**ejs** – шаблонизатор

**nodemon** – чтобы не перенагружать сервер отладки

**маршрут** – это адрес который доступен пользователю

**Функции:**

**Use** – обрабатывает любой запрос, всегда работает

**Get** – запрос из строки адреса

**Post** – запрос по кнопке на форме submite?

**Put** – запрос на изменение

**Delete** – запрос на удаление

**Router** – определить дочерние подмаршруты, когда используется сложный путь user/create

**Index.js** – обработка маршрута

**Admin.js** – обработка маршрута

**Article.js** - обработка маршрута

**Смысл admin**: заключается в получении запросов от клиента, их обрабатывать и выдавать ему ответ. Всё что приходит от клиента мы получаем в переменную **req** – это объект, запрос отправляется с помощью объекта **res** (это стандартный параметр, который обрабатывает любой запрос на стороне сервера), они всегда указываются!

**Получение результата**

**Объекты и свойства**

**Request.query** - хранит объект с данными, переданными из строки

запроса вида <http://localhost:3000/about>? Id=3&name=tome (параметры запроса, задается переменная и ей присваивается некоторое значение)

**Request.params** - Это свойства, прикрепленные к URL, то есть

именованные параметры маршрута, которые в маршруте имеют

вид:/user/:userId ( т.е. здесь мы получаем идентификатор некой сущности)

**Request.body** содержит данные формы (name)

**Отправка ответа**

Самый распространенный способ отправки ответа представляет функция

**send().** Можем отправить туда массив, объект, файл, статусные коды, шаблон

**\_\_dirname** получаем абсолютный путь к текущему проекту

**Статические файлы –** это css, html - **express.static(),** указывает на каталог

с файлами.

**EJS**

Управляет представлениями специальный компонент - движок

представлений (view engine), который также называют движок

шаблонов (template engine).

Размещение шаблонов

**Шаблон** – это файл, содержащий разметку HTML и плейсхолдеры

Файл имеет расширение .ejs

Хранится в папке views, которая лежит в корневой папке проекта

**Плейсхолдеры**

Для работы с моделью представления используются выражения

javascript, которые определяются с помощью тега <% выражение

%>.

**Promise** – это специальный объект в JavaScript, который связывает

«создающий» и «потребляющий» коды вместе. «Создающий» код может

выполняться сколько потребуется, чтобы получить результат, а промис

делает результат доступным для кода, который подписан на него, когда

результат готов.

**let promise** = new Promise(function(resolve, reject) {

// функция-исполнитель (executor) });

Её аргументы resolve и reject – это колбэки, которые предоставляет сам JavaScript.

**resolve(value)** — если работа завершилась успешно, с результатом value.

**reject(error)** — если произошла ошибка, error – объект ошибки.

**Потребители**

Объект Promise служит связующим звеном между исполнителем

(«создающим» кодом) и функциями-потребителями, которые

получат либо результат, либо ошибку.

Функции-потребители могут быть зарегистрированы с помощью

методов .then и .catch.

**.then()** выполняется когда промис завершился успешно

**.catch()** выполняется в случае ошибки

**Fetch**

let promise = fetch(url, [options])

**url** – URL для отправки запроса.

**options** – дополнительные параметры: метод, заголовки и так далее.

Без options это простой GET-запрос, скачивающий содержимое по

адресу url. Браузер сразу же начинает запрос и возвращает промис, который

внешний код использует для получения результата.

**Response** предоставляет несколько методов, основанных на

промисах, для доступа к телу ответа в различных форматах:

**response.text()** – читает ответ и возвращает как обычный текст,

**response.json()** – декодирует ответ в формате JSON,

**response.formData()** – возвращает ответ как объект FormData,

**response.blob()** – возвращает объект как Blob (бинарные данные с

типом),

**response.arrayBuffer()** – возвращает ответ как ArrayBuffer

(низкоуровневое представление бинарных данных),

У слова **async** один простой смысл: эта функция всегда возвращает промис. – создает запрос к БД

Ключевое слово **await** заставит интерпретатор JavaScript ждать до тех

пор, пока промис справа от await не выполнится. – получает запрос и его куда-то передает

**MVC** — это шаблон программирования, который позволяет

разделить логику приложения на три части:

**Model (модель).** Получает данные от контроллера, выполняет

необходимые операции и передаёт их в вид.

**View (вид или представление).** Получает данные от модели и выводит их

для пользователя.

**Controller (контроллер).** Обрабатывает действия пользователя, проверяет

полученные данные и передаёт их модели.