**Конспект по Node.js**

**ЛР-1**

require — подключение модулей в Node.js

require('http');

//Для использования HTTP-сервера и клиента

-Для работы с сервером и протоколом http в Node.js используется модуль http.

-Чтобы создать сервер, следует вызвать метод http.createServer()

-чтобы сервер мог прослушивать и обрабатывать входящие подключения, у объекта сервера необходимо вызвать метод listen(), в который в качестве параметра передается номер порта, по которому запускается сервер.

Для обработки подключений в метод createServer можно передать специальную функцию:



**Эта функция принимает два параметра:**

request: хранит информацию о запросе

response: управляет отправкой ответа

**Request**

Параметр request позволяет получить информацию о запросе и представляет объект ***http.IncomingMessage.*** Отметим некоторые основные свойства этого объекта:

**headers:** возвращает заголовки запроса

**method**: тип запроса (GET, POST, DELETE, PUT)

**url:** представляет запрошенный адрес



**Response**

Параметр response управляет отправкой ответа и представляет объект http.ServerResponse. Среди его функциональности можно выделить следующие методы:

statusCode: устанавливает статусный код ответа

statusMessage: устанавливает сообщение, отправляемое вместе со статусным кодом

setHeader(name, value): добавляет в ответ один заголовок

write: пишет в поток ответа некоторое содержимое

writeHead: добавляет в ответ статусный код и набор заголовков

end: сигнализирует серверу, что заголовки и тело ответа установлены, в итоге ответ отсылается клиента. Данный метод должен вызываться в каждом запросе.



Метод getHeaders()-*возвращает мелкую копию текущих исходящих заголовков.*

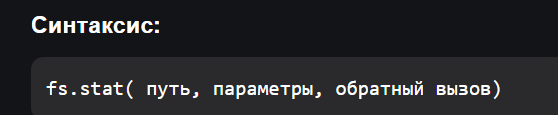
Лр 2 теорка

 fs модуль позволяет взаимодействовать с файловой системой способом, смоделированным на основе стандартных функций POSIX

fs- обеспечивает файловый ввод-вывод

**Метод fs.readFileSync()** представляет собой встроенный интерфейс прикладного программирования модуля fs, который используется для чтения файла и возврата его содержимого.

**Метод fs.stat()** используется для возврата информации о данном файле или каталоге. Он возвращает **fs.Объект Stat**, который имеет несколько свойств и методов для получения подробной информации о файле или каталоге.



**Параметры:** Этот метод принимает три параметра, как указано выше и описано ниже:

* **путь:**Он содержит путь к файлу или каталогу, который необходимо проверить. Это может быть строка, буфер или URL.
* **параметры:** это объект, который можно использовать для указания необязательных параметров, которые будут влиять на результат. Он имеет один необязательный параметр:
  + **bigint :** это логическое значение, которое указывает, являются ли числовые значения, возвращаемые в объекте fs.Stats, bigint . Значение по умолчанию равно false.
* **обратный вызов:** это функция, которая будет вызываться при выполнении метода.
  + **ошибка:** это ошибка, которая будет выдана, если метод
  + **статистика:** это объект статистики, который содержит подробную информацию о пути к файлу.

Оператор == сравнивает на равенство, а вот === — на идентичность. Плюс оператора === состоит в том, что он не приводит два значения к одному типу. Именно из-за этого он обычно и используется.

XMLHttpRequest – это встроенный в браузер объект, который даёт возможность делать HTTP-запросы к серверу без перезагрузки страницы.

**responseText**

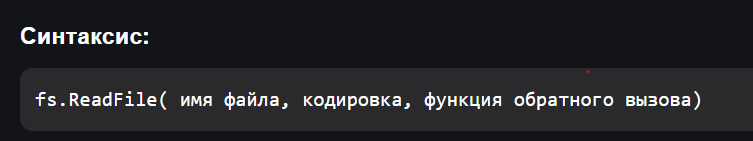
Свойство, доступное только для чтения[XMLHttpRequest](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest)responseText, возвращает текст, полученный с сервера после отправки запроса.

Свойство innerHTML позволяет считать содержимое элемента в виде HTML-строки или установить новый HTML.

# **.innerHTML**

Читаем и изменяем содержание HTML-элемента.

Метод **fs.ReadFile() - это встроенный метод**, который используется для чтения файла. Этот метод считывает весь файл в буфер.



**Параметры:** метод принимает три параметра, как указано выше и описано ниже:

* **имя файла:** содержит имя файла для чтения или полный путь, если он хранится в другом месте.
* **кодирование:** он содержит кодировку файла. Его значением по умолчанию является **‘utf8’**.
* **функция обратного вызова:** это функция обратного вызова, которая вызывается после чтения файла. Он принимает два параметра:
  + **ошибка:** если произошла какая-либо ошибка.
  + **данные:** содержимое файла.

# jQuery функция $.ajax()

функция **$.ajax()** позволяет выполнить асинхронный AJAX запрос. **AJAX** (от англ. *Asynchronous Javascript and XML* — "асинхронный JavaScript и XML") — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в "фоновом" обмене данными браузера с веб-сервером.

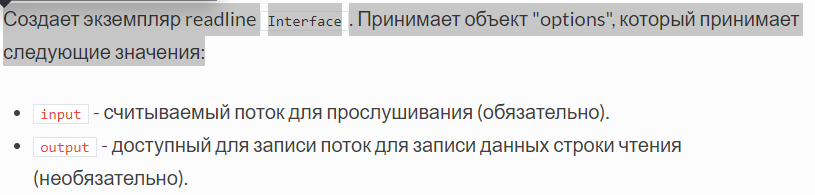
## Значения параметров:

* **method** - указываем метод HTTP *POST*, используемый для запроса.
* **url** - файл, к которому мы обращаемся ("**about.php**"), он содержит следующий PHP код
* **data** - данные, которые будут отправлены на сервер.
* **success** - функции обратного вызова, которые вызываются если AJAX запрос выполнится успешно. Обратите внимание, что если используется несколько функций, то необходимо помещать их в массив. Каждая функция выполняется в свою очередь.
* **statusCode** - объект числовых кодов *HTTP* и функции, которые будут вызываться, когда код ответа сервера имеет соотвествующее значение (определенный код *HTTP*). В нашем случае код 200.

**Лр3а**

node:readline Модуль предоставляет интерфейс для чтения данных из [читаемого](https://nodejs.org/api/stream.html#readable-streams) потока

server.listen

* Это метод создает прослушиватель на указанном порту или пути метод создает прослушиватель на указанном порту или пути.
* readline.createInterface
* 

process Объект предоставляет информацию о текущем Node.js процесс.

process.stdin-Свойство возвращает поток , подключенный к stdin (fd 0).

process.stdout Свойство возвращает поток, подключенный к stdout (fd 1)

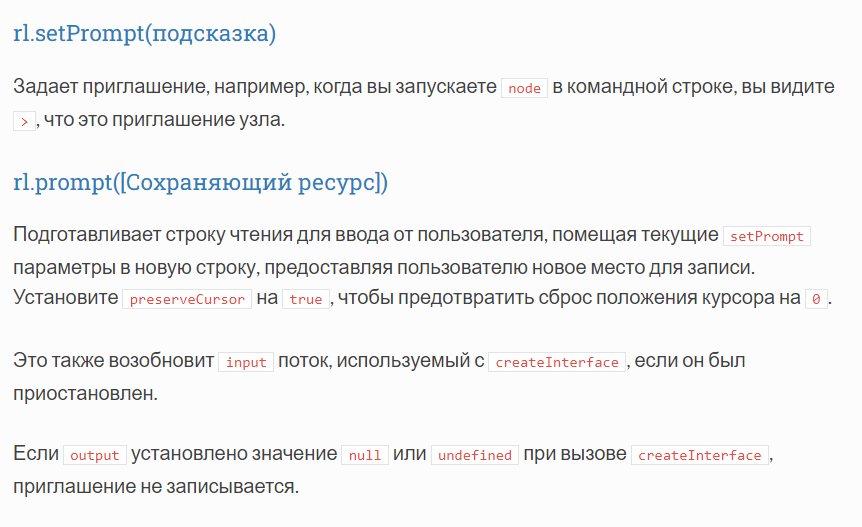


trim();

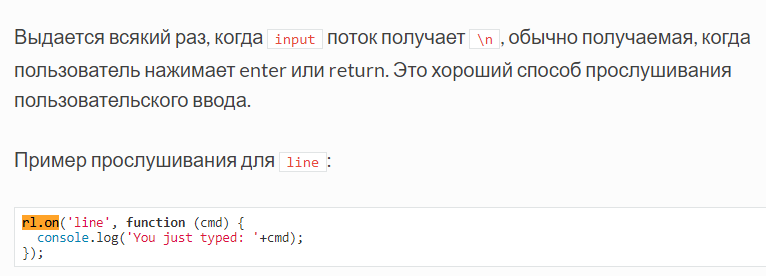


События process:

'exit'-Событие генерируется, когда Node.js процесс завершается



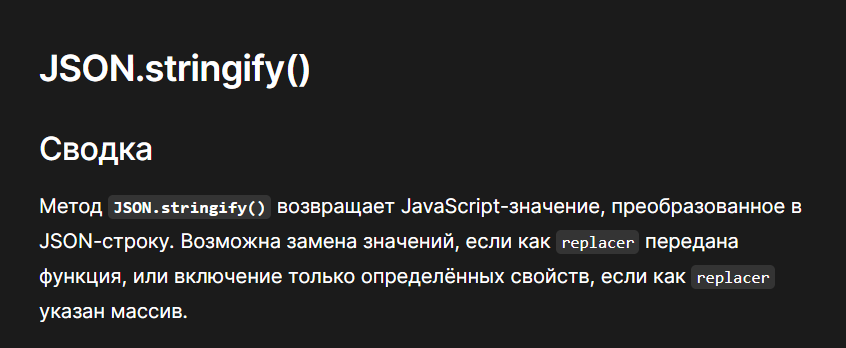
**rl.on**



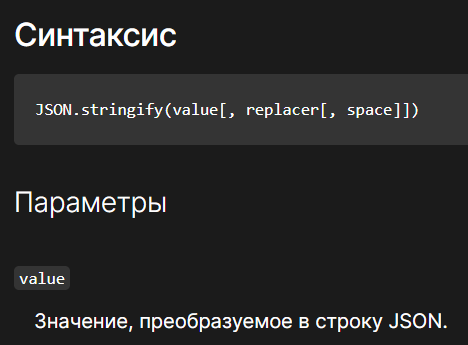
**Рекурсивный алгоритм** – это алгоритм, в описании которого прямо или косвенно содержится обращение к самому себе.

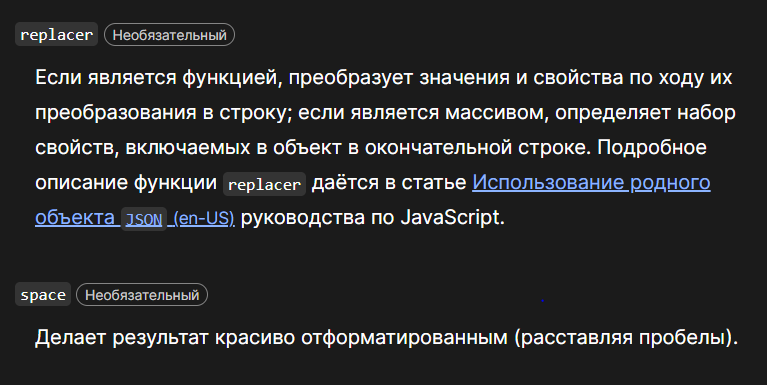
**Рекурсивная функция** (лат. recursio — возвращение) — функция, которая в своей записи содержит вызов себя же.

**Метод fs.readFileSync()** является встроенным интерфейсом прикладного программирования модуля fs, который используется для чтения файла и возврата его содержимого.

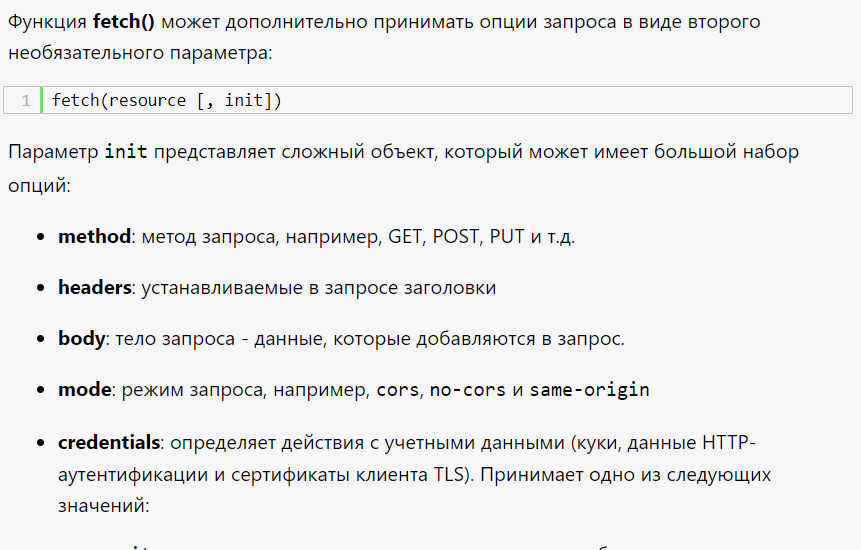
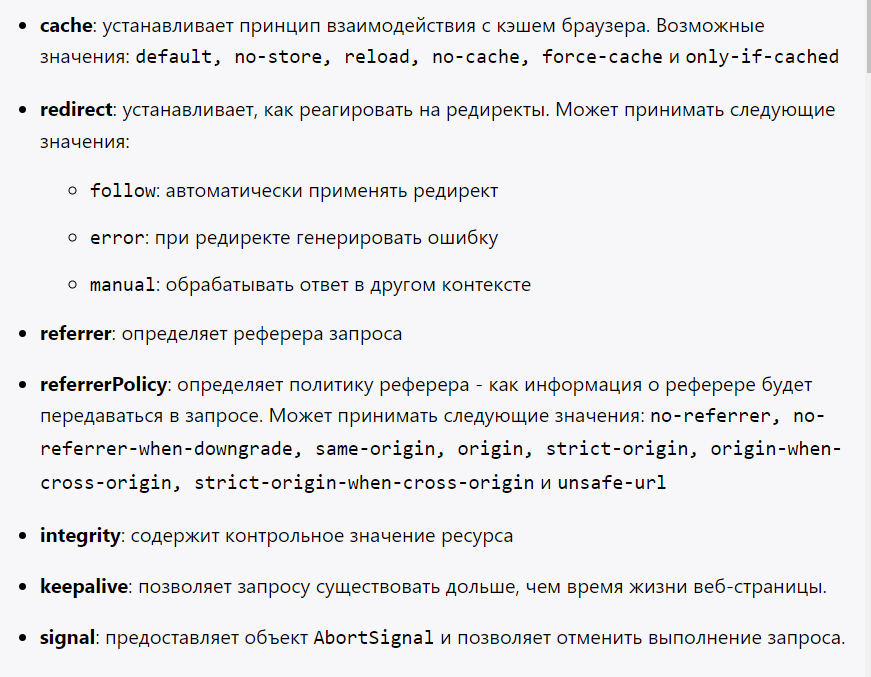


Метод  преобразует значение в представляющую его нотацию JSON





**Date.now()** Статический метод возвращает количество миллисекунд,

**Лр 3б**

Promise((resolve, reject)

