18. Основные глобальные объекты Node.js и их назначение

1. global:

- Глобальный объект, который доступен во всех модулях. Позволяет хранить и получать глобальные переменные.

2. process:

- Объект, представляющий текущий процесс Node.js. Содержит информацию о процессе, такие как аргументы командной строки, переменные окружения и методы для управления процессом.

3. Buffer:

- Глобальный объект, используемый для работы с бинарными данными. Позволяет хранить и манипулировать необработанными байтовыми данными.

4. require:

- Функция для импорта модулей. Позволяет загружать модули как встроенные, так и пользовательские.

5. module:

- Объект, представляющий текущий модуль. Позволяет экспортировать функции и переменные из модуля.

6. \_\_dirname:

- Строка, содержащая путь к директории текущего модуля.

7. \_\_filename:

- Строка, содержащая полный путь к текущему файлу.

19. Асинхронная функция

Асинхронная функция — это функция, которая выполняется асинхронно, позволяя выполнять другие операции, не блокируя основной поток выполнения. В Node.js асинхронные функции часто используются для работы с I/O-операциями, такими как чтение файлов или запросы к базе данных. Асинхронные функции возвращают Promise, что позволяет использовать await для ожидания их завершения, упрощая обработку асинхронного кода.

20. Стандартные системные потоки

Стандартные системные потоки — это интерфейсы, через которые процесс может взаимодействовать с внешними данными. В Node.js есть три основных стандартных потока:

1. stdin:

- Стандартный поток ввода, через который программа может получать данные от пользователя.

2. stdout:

- Стандартный поток вывода, через который программа может отправлять данные пользователю.

3. stderr:

- Стандартный поток ошибок, используемый для вывода ошибок и предупреждений.

Эти потоки позволяют строить интерактивные приложения и обрабатывать сообщения о состоянии программы.

21. Функции process.nextTick и setImmediate

- process.nextTick(callback):

- Запланирует выполнение функции callback в конце текущей операции, перед тем как Node.js продолжит выполнение других операций в цикле событий. Это позволяет завершить текущую операцию, а затем выполнить указанную функцию.

- setImmediate(callback):

- Запланирует выполнение функции callback в следующем цикле событий, после завершения всех текущих операций. Это полезно для выполнения кода после всех I/O-операций.

Разница между process.nextTick и setImmediate:

- Очередь выполнения: process.nextTick добавляет коллбек в очередь "next tick", которая выполняется перед любыми другими I/O-операциями. setImmediate добавляет коллбек в очередь I/O, которая выполняется после завершения текущего цикла событий.

- Приоритет: process.nextTick имеет более высокий приоритет и выполняется раньше, чем setImmediate.