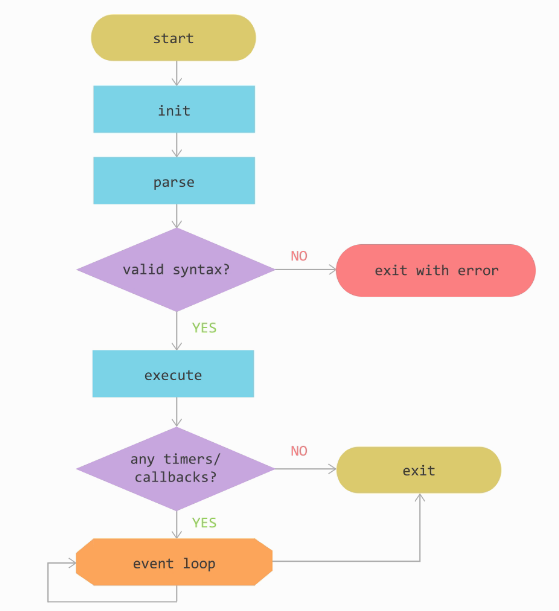
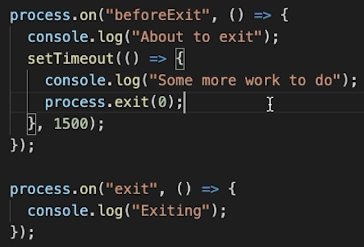
Жизненный цикл процесса Node.js

Укрупненно: запускается процесс node -> парсится синтаксис, затем выполняется код. Если в процессе парсинга выявлена ошибка синтаксиса, то процесс останавливается и выбрасывается соответствующая ошибка.

Если в скрипте есть асинхронный код (таймеры, интервалы, промисы), то в дело вступает event loop. Цикл запускается бесконечно и проверяет наступление выполнения асинхронного кода (завершился таймер, пришел ответ от сервера и т.д.). Если что-то из этого произошло, то выполняются все подписанные на эти события колбэки. Когда асинхронные процедуры закончились, то цикл завершается и скрипт закачивается.

Перед завершением процесса выполняется событие “exit”. В этот момент мы можем выполнить некий синхронный код.

Асинхронные действия после этого не выполнятся, так как процесс вышел из event loop.

Для асинхронного кода есть событие «beforeExit».

Это все ситуации успешной отработки инструкции и естественного завершения процесса.

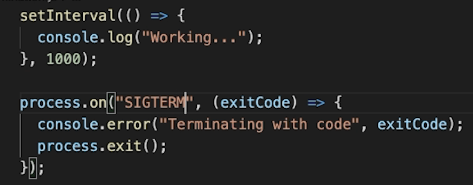
Есть особые случаи завершения процесса:

- логическая ошибка - код работает как нужно, но програма не может продолжить работу (не получен нужный конфигурационный параметр)

- Uncaught Exeption - возникает когда синхронный код выбрасывает exeption, который мы не обработали. В этом случае процесс завершается.

- Unhandled Redjected – ошибка реджекнутого промиса, которую мы не обработали.

- system signal – сигнал завершения. Например убийство процесса или завершение его через менеджер задач.



Коды выхода из программы.

Код 0 – успешное завершение. Любой положительный код означает ошибку.

При обработке system signal обязательно следует вызывать process.exit(), иначе процесс никогда не завершится.

Во всех остальных перечисленных ситуациях необходимо не отлавливать и продолжать работу программы, а вывести наибольшее количество информации по ошибке и завершить процесс.