**Асинхронность**

Как правило, программный код выполняется последовательно, только одна конкретная операция происходит в данный момент времени. Если функция зависит от результата выполнения другой функции, то она должна дождаться пока нужная ей функция не завершит свою работу и не вернёт результат и до тех пор пока это не произойдёт, выполнение программы, по сути, будет остановлено с точки зрения пользователя.

Но при асинхронности код будет читаться так же сверху вниз и при этом следующее действие может не ждать предыдущее действие.

**Event loop**

Event loop отвечает за выполнение кода, сбора и обработки событий и выполнения под-задач из очереди. Благодаря нему происходит асинхронность.

**Пример**

Есть 4 дивов с надписью “GO”, по середине див c надписью “WAIT” с таймером 5 сек.

* В случае Синхронности выйдет:

GO

GO

(Через 5 сек) WAIT

GO

GO

* В случае Асинхронности выйдет:

GO

GO

GO

GO

(Через 5 сек) WAIT

**Что происходит**

|  |  |
| --- | --- |
| Call stack  Сюда попадает все действия  И сразу выполняются все действия. Кроме тех которые “затормаживают” и их отправляет в Web API | Web API  Сюда попадают действия которые “затормаживают”  Ждут того или иного события затем отправляется в очередь (Callback Queue) |
| Callback Queue  Все действия ждет своей очереди и попадают снова в Call stack где уже вызывается | |