1. JavaScript — однопоточный, и это значит, что он выполняет код в одном потоке. Но при этом умеет работать с асинхронными задачами: setTimeout, fetch, Promise.

2. Эти задачи не блокируют основной поток. Вместо этого они отправляются в Web APIs, и когда завершаются — ставятся в очередь колбэков. 3. Тут вступает в работу event loop — он постоянно проверяет, пуст ли call stack, и если да — забирает задачу из очереди и исполняет.

4. Это позволяет JavaScript не зависать, не ждать окончания таймеров и сетевых запросов, а продолжать выполнять код. То есть, event loop — это механизм, который делает асинхронность возможной, не ломая однопоточность.

Джуны пытаются вызубрить: "там стек, очередь, цикл... эм... microtasks..." Мидлы понимают: \* что уходит в Web API (и почему console.log идёт раньше setTimeout) \* как Promise и setTimeout попадают в разные очереди \* чем macrotask отличается от microtask \* почему async/await — это не магия, а синтаксический сахар