**setTimeout** позволяет вызвать функцию **один раз** через определённый интервал времени.

**setInterval** позволяет вызывать функцию **регулярно**, повторяя вызов через определённый интервал времени.

Node.js работает до тех пор, пока есть события, требующие обработки; если выполнить для таймера **unref**, то события, генерируемые таймером, не будут учитываться при завершении работы Node.js, ref – противоположная операция.

В отличие от браузерного JavaScript, здесь timer – это объект, и метод timer.unref указывает libUV, что этот таймер является второстепенным. То есть, его не следует учитывать при проверке внутренних watcher на завершение процесса.

Есть еще метод **ref**. Он является противоположным unref. То есть, если я сделал timer.unref, потом     передумал, то могу вызвать метод ref. В практике он используется очень редко.

1. setTimeout

позволяет вызвать функцию один раз через определённый интервал времени.

2. setInterval

позволяет вызывать функцию регулярно, повторяя вызов через определённый интервал времени.

3. ref

противоположный unref

4. unref

timer является второстепенным, не нужно проверять, не влияет на завершение процесса.

1. Дайте пояснению понятию «событие программного объекта».

Событие — это именованный делегат, при вызове которого, будут запущены все подписавшиеся на момент вызова события методы заданной сигнатуры. (Событие - это ситуация, при возникновении которой, произойдут некоторые действия).

2. Объясните механизм генерации и обработки событий в C#.

События объявляются в классе с помощью ключевого слова event, после которого указывается тип делегата, который представляет событие

3. Поясните как самостоятельно реализовать механизм генерации и обработки событий на JS или С++.

4. Какой встроенный механизм используется в Node.js для генерации и обработки событий. Поясните принцип его работы.

EventEmitter: JS-класс, предоставляющий функциональность для асинхронной обработки событий в Node.js. Событие в программном объекте – это процесс перехода объекта из одного состояние в другое. При этом, об этом переходе могут быть извещены другие объекты. У события есть издатель (или генератор) события и могут быть подписчики (или обработчики) события.

EventEmitter: как правило, применяется в качестве базового для пользовательского объекта. Производный от EventEmitter объект может быть создан с помощью функции inherits модуля utils.

EventEmitter: производный от EventEmitter объект приобретает функциональность, позволяющую генерировать и прослушивать события.

EventEmitter: для генерации событий предназначена функция emit, а для прослушивания функция on.