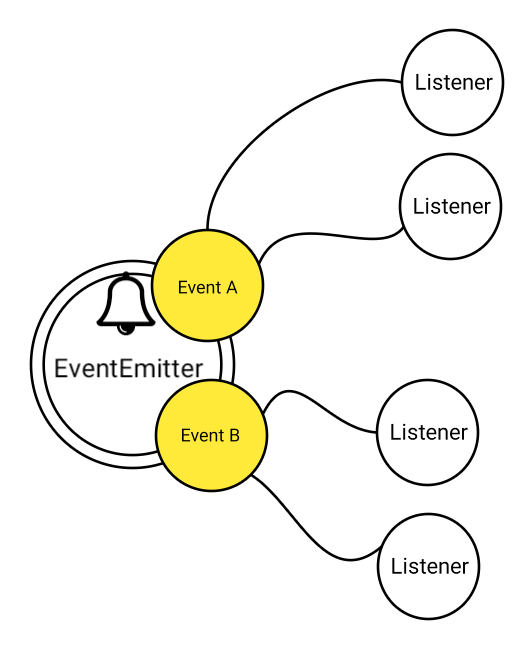
Лекция 04

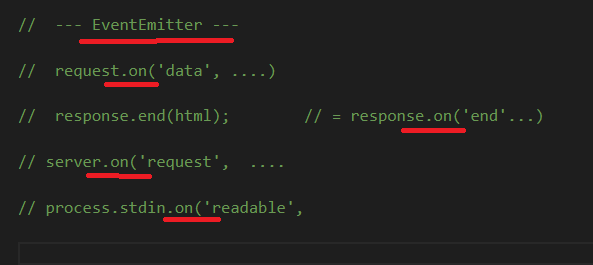
ПСКП, ПОИТ-3

**КЛАСС EventEmitter**

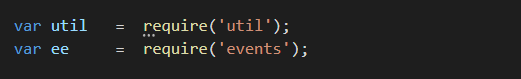
1. **EventEmitter:** JS-класс, предоставляющий функциональность для асинхронной обработки событий в **Node.js.** Событие в программном объекте – это процесс перехода объекта из одного состояние в другое. При этом, об этом переходе могут быть извещены другие объекты. У события есть **издатель** (или генератор)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



1. **EventEmitter:** применяется в базовых объектах **Node.js.**

****

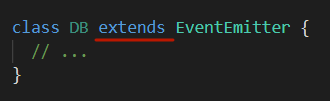
1. **EventEmitter:** необходимовключения двух модулей: ***events*** и ***util***.

****

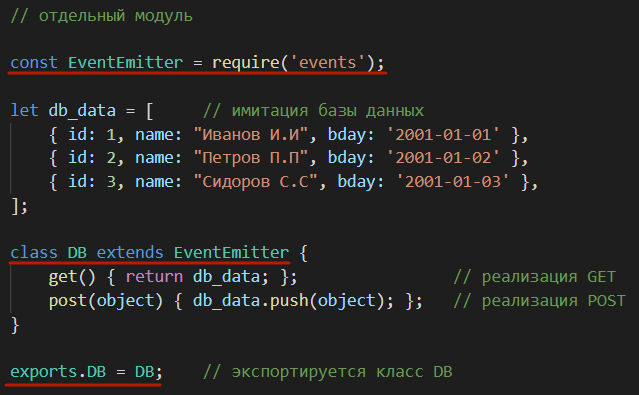
1. **EventEmitter:** как правило, применяется в качестве базового для пользовательского объекта. Производный от **EventEmitter** объект может быть создан с помощью функции **inherits** модуля **utils** (устарело).

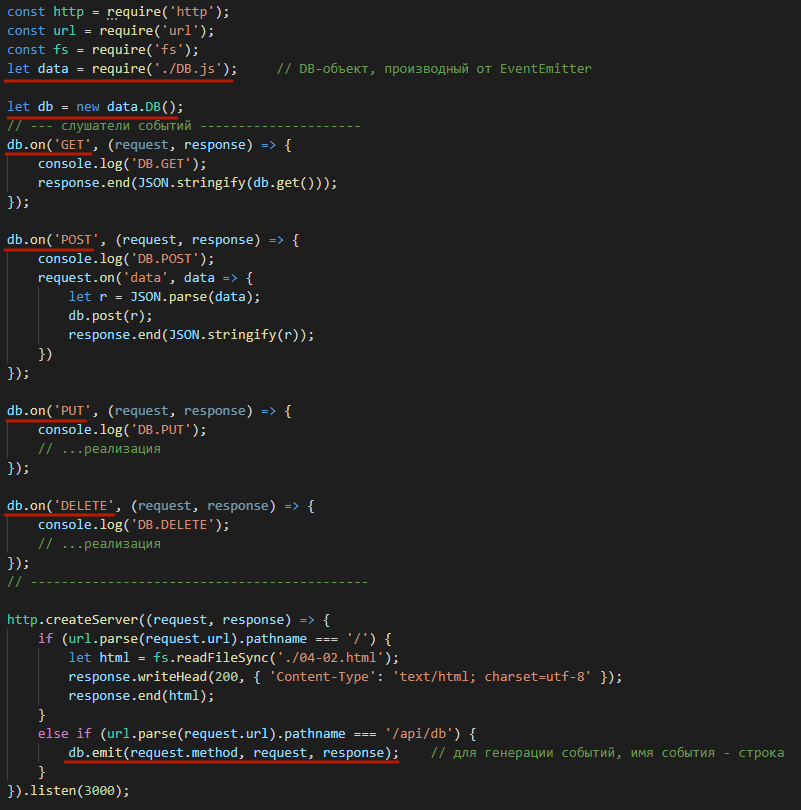
****

1. **EventEmitter:** для наследования можно использовать ключевое слово **extends** (ES6).

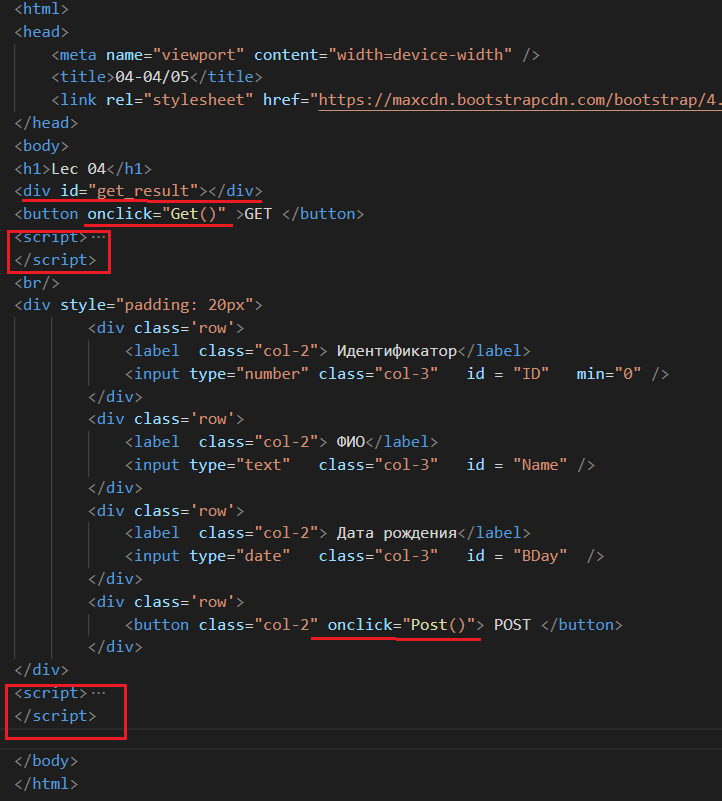


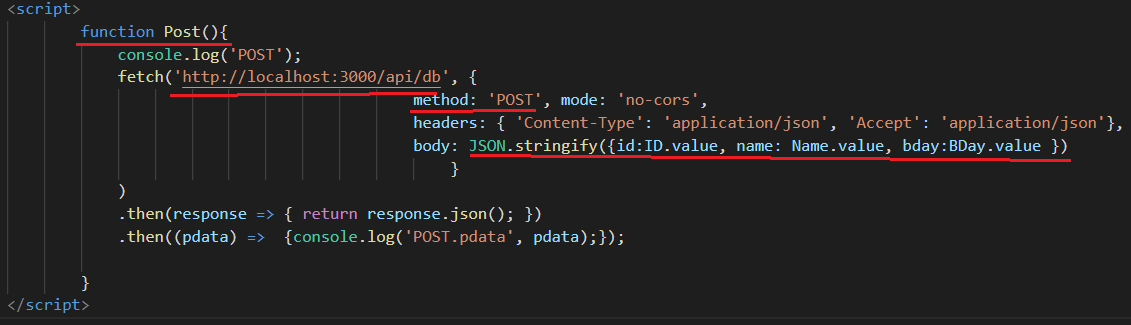
1. **EventEmitter:** производный от **EventEmitter** объект приобретает функциональность, позволяющую генерировать и прослушивать события.
2. **EventEmitter:** для генерации событий предназначена функция **emit,** адля прослушивания функция **on.**
3. **EventEmitter:** примердля выполнения лабораторной работы

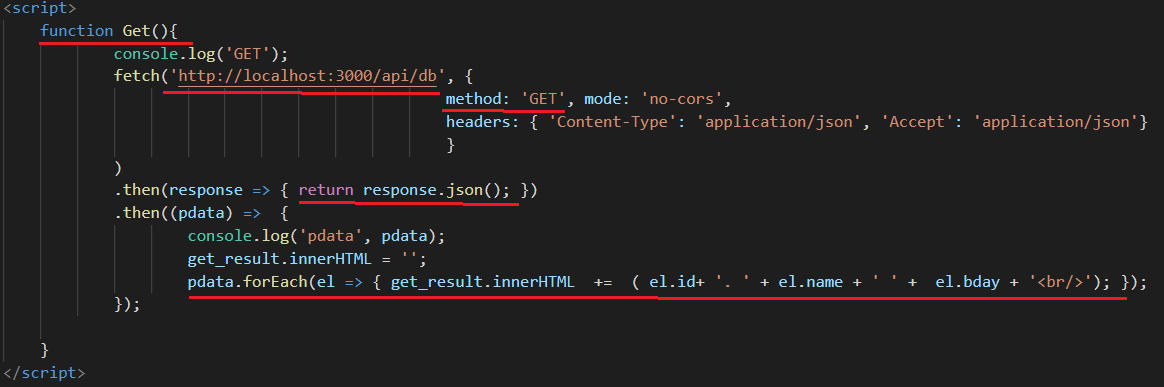


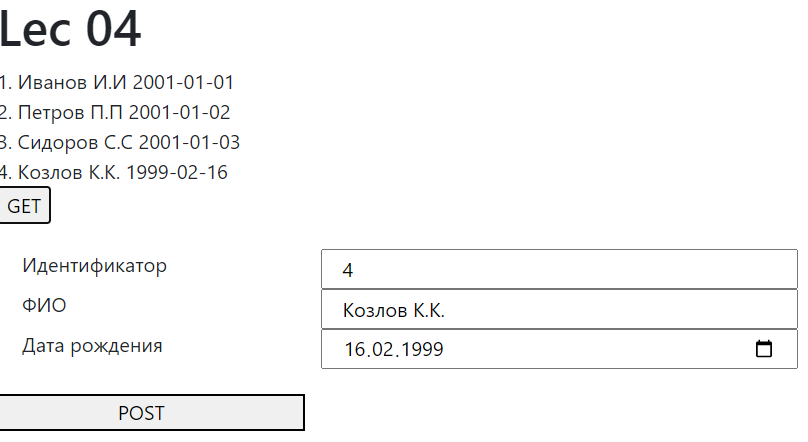


1. **Клиент:** пример для выполнения лабораторной работы

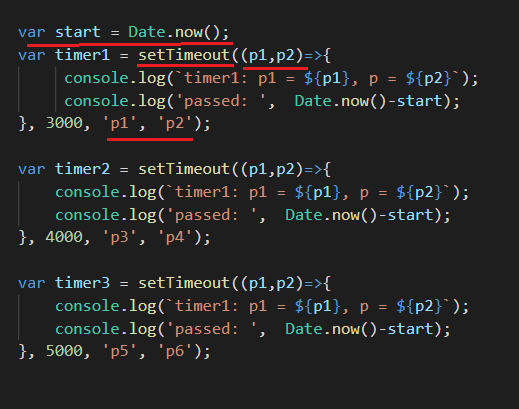
****

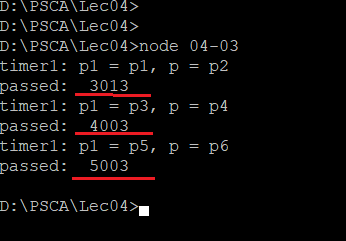
****

****

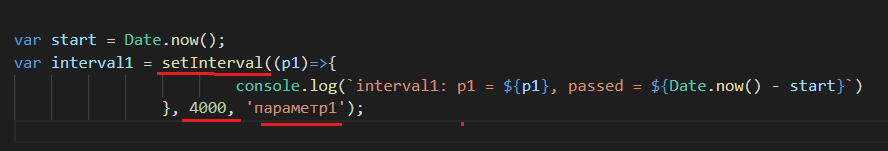


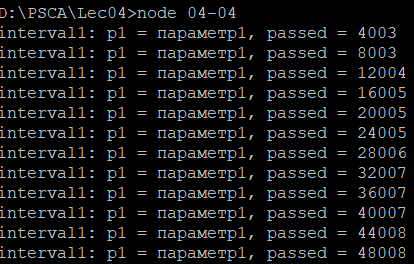
1. **EventEmitter:** методы **once** (вызов слушателя только один раз), **removeAllListeners** (удаление всех слушателей или всех для указанного события), **removeListener** (удаление указанного слушателя для некоторого события).
2. **Таймер:** механизм, позволяющий генерировать событие или выполнить некоторое действие через заданный промежуток времени.
3. **NODEJS: setTimeout()**, **setInterval()**; реализованы библиотекой **libuv.**
4. **setTimeout():** выполняется только один раз через некоторый промежуток времени.

****

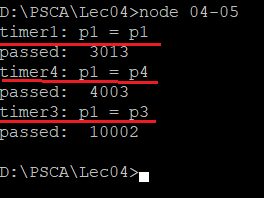
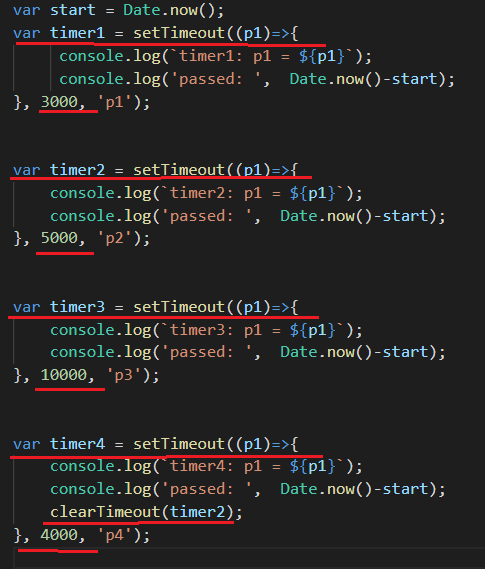
****

1. **setInterval():** выполняется регулярно через некоторый промежуток времени.

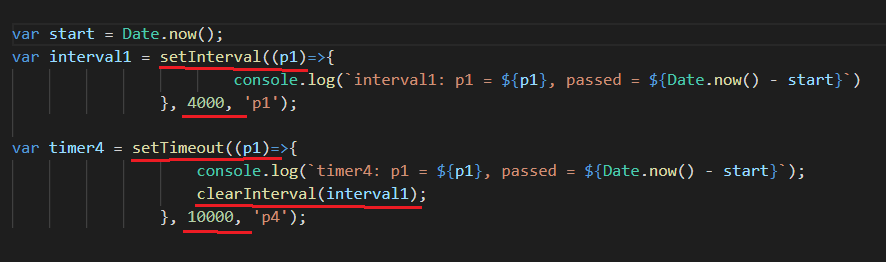
****

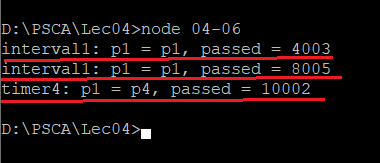
****

1. **clearTimeout():** останавливает таймер, созданный с помощью setTimeout(). Параметр – ID таймера, который необходимо отменить.

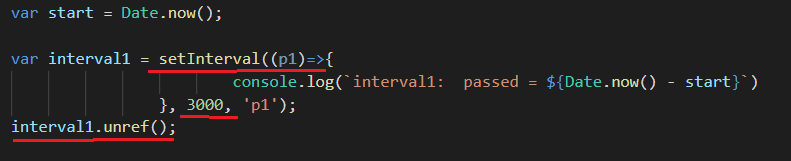
****

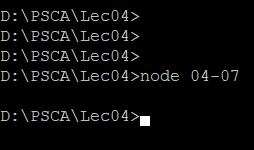
1. **clearInterval():**останавливает таймер, созданный посредством setInterval(). Параметр – ID таймера, который необходимо отменить.

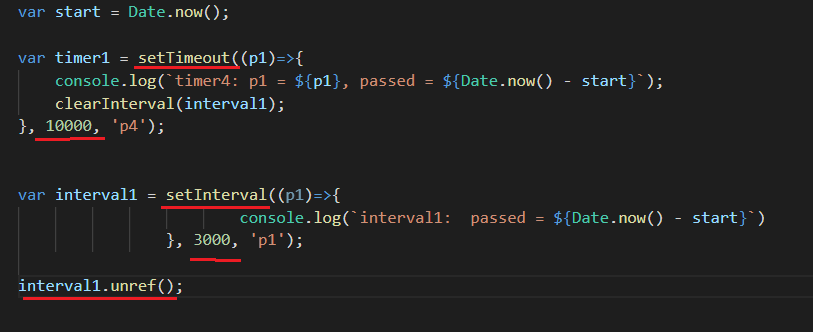
****

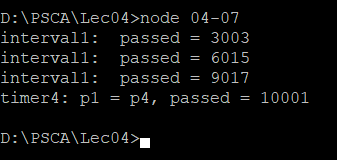
****

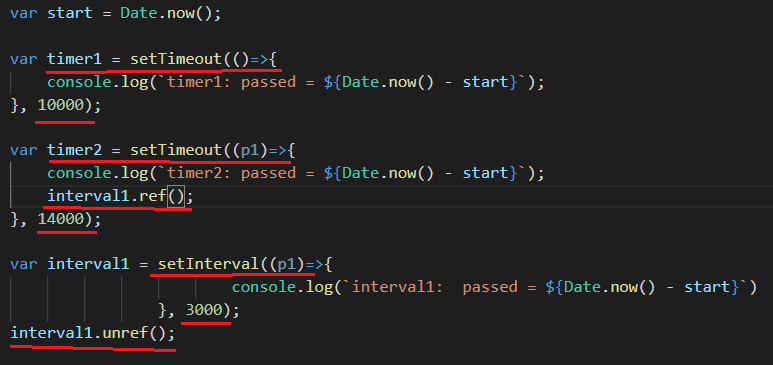
1. **NODEJS:** процессNode.jsработает до тех пор, пока есть события, требующие обработки;если выполнить для таймера **unref,** тособытия**,** генерируемые таймером не будут учитываться при завершении работы Node.js**, ref** –противоположная операция.

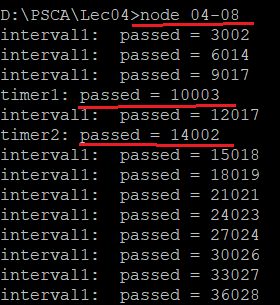
****

****

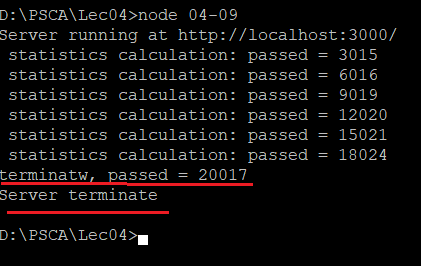
****

****

****

****

****

****

1. **NODEJS:** метод unref() есть не только у таймеров, есть еще у серверов (server.unref()), сетевых сокетов (socket.unref()) и др.