1. **Перечислите основные свойства глобальные объекты Node.js и поясните их предназначение.**

Node.js предоставляет специальный объект global, который предоставляет доступ к глобальным, то есть доступным из каждого модуля приложения, переменным и функциям. Примерным аналогом данного объекта в javascript для браузера является объект window.

1. **global: *хранит var-данные на уровне модуля***.
2. **process:** информация о среде выполнения
3. **buffer:** класс Buffer – предназначен для работы с двоичными данными: набором октетов.
4. **Поясните понятие «асинхронная функция»**

Асинхронная функция – это функция, после вызова которой JavaScript приложение продолжает работать, потому что функция сразу выполняет возврат.

Асинхронные функции не являются заменой промисов. Они работают сообща. Асинхронная функция ожидает (await) исполнения промиса и всегда возвращает промис.Промис, возвращаемый асинхронной функцией, будет разрешен (resolve) с тем значением, которое вернет функция.

1. **Поясните понятие стандартные «системные потоки».**

Поток – байтовая последовательность, передаваемая в процессе ввода-вывода.

Стандартные потоки ввода-вывода в [системах](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) типа [UNIX](https://ru.wikipedia.org/wiki/UNIX) (и некоторых других) — [потоки](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) [процесса](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), имеющие номер (дескриптор), зарезервированный для выполнения некоторых «стандартных» функций

Stdin, stdout, stderr, stdprn, stdaux.

1. **Поясните назначение функций process.nextTick, setImmediate, поясните в чем разница.**

setImmediate() и setTimeout() похожи, но ведут себя по-разному в том, когда они вызываются.

* setImmediate() предназначен для выполнения сценария после завершения текущей фазы опроса.
* setTimeout() планирует запуск сценария после истечения минимального порога в миллисекундах.

Оглядываясь назад на нашу диаграмму, каждый раз, когда вы вызываете process.nextTick() на данной фазе, все коллбэки, переданные процессу process.nextTick(), будут разрешаться до того, как цикл событий продолжится. Это может создать некоторые плохие ситуации, потому что это позволяет «замораживать» ваш ввод/вывод, делая рекурсивные вызовы process.nextTick(), что не даёт циклу событий достичь фазы опроса.

У нас есть два вызова, которые похожи для пользователей, но их имена сбивают с толку.

* process.nextTick() срабатывает сразу на той же фазе
* setImmediate() срабатывает на следующей итерации или «тике» цикла событий

4

1. Дайте пояснению понятию «событие программного объекта».

ситуация, при возникновении которой, произойдет действие или несколько действий.

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. Объясните механизм генерации и обработки событий в C#.

события (event) основаны на [делегатах](http://msdn.microsoft.com/ru-ru/library/ms173171.aspx) (delegate), а делегат, говоря очень простым языком — «переменная, хранящая ссылку на метод».

 1. Определите условие возникновения события и методы которые должны сработать.

 2. Определите сигнатуру этих методов и создайте делегат на основе этой сигнатуры.

 3. Создайте общедоступное событие на основе этого делегата и вызовите, когда условие сработает.

 4. Обязательно (где-угодно) подпишитесь на это событие теми методами, которые должны сработать и сигнатуры которых подходят к делегату.

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

let event = new Event(type[, options]);

elem.dispatchEvent(event) Запуск события

с помощью on() обработка

access

delegate

return-type

delegate-type-identifier

(

[ parameters ]

)

public delegate void MyDel(int i);

DelInst(9);

DelInst->Invoke(10);

**Параметры**

*имеет*  
используемых Доступность делегата, который может быть **public** (по умолчанию) или **private** . Прототип функции также может быть дополнен **const** **volatile** ключевыми словами или.

*Тип возвращаемого значения*  
Тип возвращаемого значения прототипа функции.

*delegate-type-identifier*  
Имя объявленного типа делегата.

*parameters*  
Типы и идентификаторы прототипа функции (необязательно).

**Remarks**

Используйте параметр *delegate-type-identifier* для объявления события с таким же прототипом, что и у делегата. Дополнительные сведения см. в статье [Delegates (C++/CX)](https://docs.microsoft.com/ru-ru/cpp/cppcx/delegates-c-cx?view=msvc-160) (Делегаты (C++/CX)).

1. 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.**

1. **EventEmitter:** как правило, применяется в качестве базового для пользовательского объекта. Производный от **EventEmitter** объект может быть создан с помощью функции **inherits** модуля **utils**.
2. **EventEmitter:** производный от **EventEmitter** объект приобретает функциональность, позволяющую генерировать и прослушивать события.
3. **EventEmitter:** для генерации событий предназначена функция **emit,** адля прослушивания функция **on.**