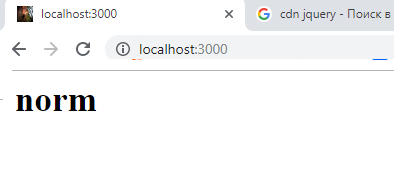
Лабораторная работа 03

ПСКП

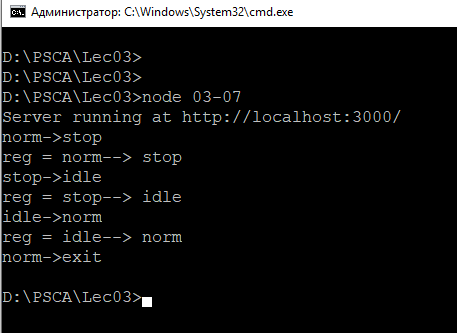
ПОИТ-3

**Задание 01**

1. Разработайте серверное приложение **03-01**, которое на запрос [**http://localhost:5000**](http://localhost:5000)возвращает страницу, отражающую состояние приложения (см. рис.).



1. Приложение может находиться в четырех состояниях: **norm**, **stop**, **test**, **idle**.
2. Состояние приложение переключается с помощью стандартного системного ввода, который назначен на консоль. Консоль в приглашении (prompt) указывает текущее состояние приложения.
3. Пользователь может ввести новое состояние (**norm**, **stop**, **test**, **idle**). При корректном вводе состояния осуществляется переключение состояния приложения.
4. При ошибочном вводе режима, ошибочная введенная последовательность символов просто отображается, но переключение режима не осуществляется.
5. Допускается ввод состояния exit, которое приводит к завершению приложения.
6. См. рис.



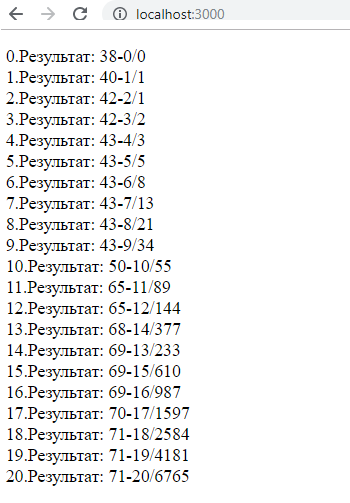
ПОИТ-3

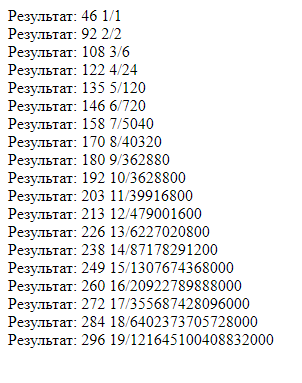
**Задание 02**

1. Разработайте серверное приложение **03-02**, которое на GET-запрос вида [**http://localhost:5000**](http://localhost:5000)**/fact?k=3** возвращает ответ, в теле которого содержится сообщение в json-формате вида {k:3, fact:6}, где **k** – полученное в качестве параметра значение, а **fact** – значение факториала.
2. Для расчета факториала используйте рекурсивный алгоритм.
3. Проверьте работоспособность приложения с помощью **POSTMAN**.

**Задание 03**

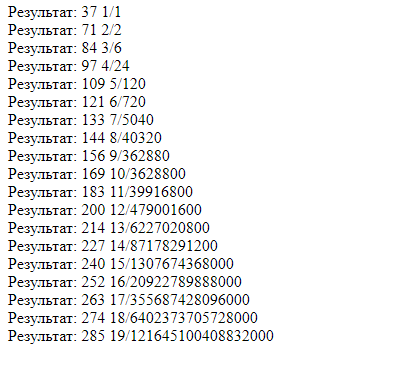
1. Доработайте приложение **03-02** таким образом, чтобы на GET-запрос приложение отправляло HTML-страницу, содержимое которой формировалось бы с помощью JS.
2. JS в цикле x = 1,…,20 с помощью функции fetch делает GET-запросы к [**http://localhost:5000/fact?k=x**](http://localhost:5000/fact?k=x) и содержимое ответа выводит в окно браузера, примерно так, как это представлено на следующем рисунке.



1. Результаты вычислений должны иметь следующий вид **t-k/fac**,где t – количество миллисекунд прошедшее с момента начала работы цикла запросов, k – параметр пересылаемый серверу, **fac** факториал k.
2. Выполните приложение запишите общую продолжительность всего цикла запросов.
3. Запустите приложение поочередно еще в двух вкладках браузера и запишите продолжительность всего цикла запросов для каждой вкладки.
4. Запустите приложение одновременно в трех вкладках браузера и запишите продолжительность всего цикла запросов для каждой вкладки.
5. 

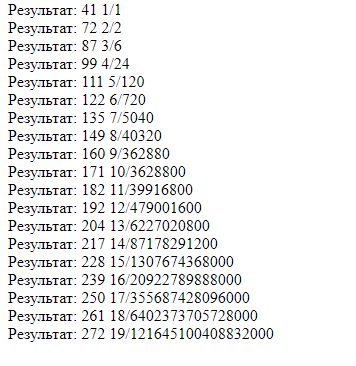
**Задание 04**

1. Разработайте приложение **03-04** на основе приложения **03-02,** но функцию для вычисления факториала реализуйте асинхронной с помощью механизма **process.nextTick**.
2. Выполните аналогичные заданию 3 замеры.



**Задание 05**

1. Разработайте приложение **03-05** на основе приложения **03-02,** но функцию для вычисления факториала реализуйте асинхронной с помощью механизма **setImmediate**.
2. Выполните аналогичные заданию 3 замеры.



**Задание 06.Ответьте на следующие вопросы**

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

**global, process, buffer**

1. ***хранит var-данные на уровне модуля***.
2. **process:** информация о среде выполнения **process: stdin, stdout, stderr,…**
3. класс **Buffer** – предназначен для работы с двоичными данными: набором октетов.
4. Поясните понятие «асинхронная функция».
5. Поясните понятие стандартные «системные потоки».

### [process.stdin](http://nodejs.org/docs/latest/api/process.html#process_process_stdin)

Поток на чтение содержит стандартный системный поток ввода для вашей программы.

По умолчанию он находится в режиме паузы, но после первого вызова .resume() он начнет исполняться в  
[следующем системном тике](http://nodejs.org/docs/latest/api/process.html#process_process_nexttick_callback).

Если process.stdin указывает на терминал (проверяется вызовом  
[tty.isatty()](http://nodejs.org/docs/latest/api/tty.html#tty_tty_isatty_fd)), тогда входящие данные будут буферизироваться построчно. Вы можете выключить построчную буферизацию вызвав process.stdin.setRawMode(true). Однако, имейте ввиду что в этом случае обработчики системных нажатий (таких как^C и ^D) будут удалены.

### [process.stdout](http://nodejs.org/api/process.html#process_process_stdout)

Поток на запись, содержащий стандартный системный вывод для вашей программы. Посылайте туда данные, если вам нужно передать их в stdout.

### [process.stderr](http://nodejs.org/api/process.html#process_process_stderr)

Поток на запись, содержащий стандартный системный вывод ошибок для вашей программы. Посылайте туда данные, если вам нужно передать их в stderr.

## [child\_process.spawn()](https://nodejs.org/api/child_process.html)

Данная функция запускает процесс, и возвращает объект содержащий stderr/stdin/stdout потоки данного процесса.

### [fs.createReadStream()](https://nodejs.org/api/fs.html#fs_fs_createreadstream_path_options)

Поток на чтение, содержащий указанный файл. Используйте, если вам надо прочесть большой файл без больших затрат ресурсов.

### [fs.createWriteStream()](https://nodejs.org/api/fs.html#fs_fs_createwritestream_path_options)

Поток на запись, позволяющий сохранить переданные данные в файл.

## net

### [net.connect()](http://nodejs.org/docs/latest/api/net.html#net_net_connect_options_connectionlistener)

Данная функция вернет дуплексный поток, который позволяет подключиться к удаленному хосту по протоколу tcp.

[net.createServer()](https://nodejs.org/api/net.html#net_net_createserver_options_connectionlistener)

Создает сервер для обработки входящих соединений. Параметром передается функция обратного вызова (callback), которая вызывается при создании соединения, и содержит поток на запись.

### [http.request()](https://nodejs.org/api/http.html#http_http_request_options_callback)

Создает поток на чтение, позволяющий сделать запрос к веб-серверу и вернуть результат.

### [http.createServer()](https://nodejs.org/api/http.html#http_http_createserver_requestlistener)

Создает сервер для обработки входящих веб-запросов. Параметром передается функция обратного вызова (callback), которая вызывается при создании соединения, и содержит поток на запись.

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

process.nextTick() срабатывает сразу на той же фазе

setImmediate() срабатывает на следующей итерации или «тике» цикла событий