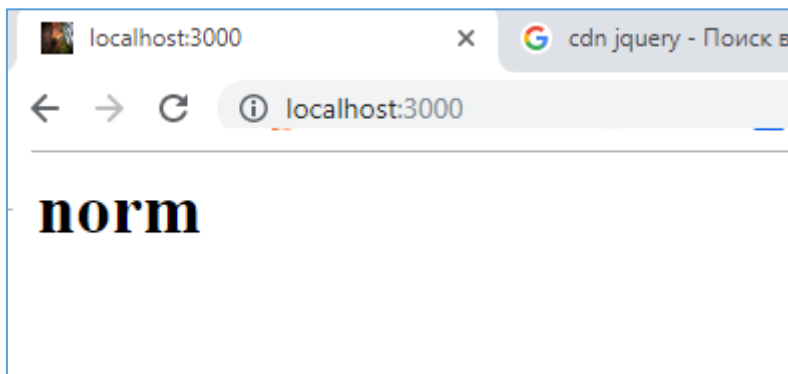


Задание 01

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



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

```
Администратор: C:\Windows\System32\cmd.exe

D:\PSCA\Lec03>
D:\PSCA\Lec03>
D:\PSCA\Lec03>node 03-07
Server running at http://localhost:3000/
norm->stop
reg = norm--> stop
stop->idle
reg = stop--> idle
idle->norm
reg = idle--> norm
norm->exit

D:\PSCA\Lec03>
```

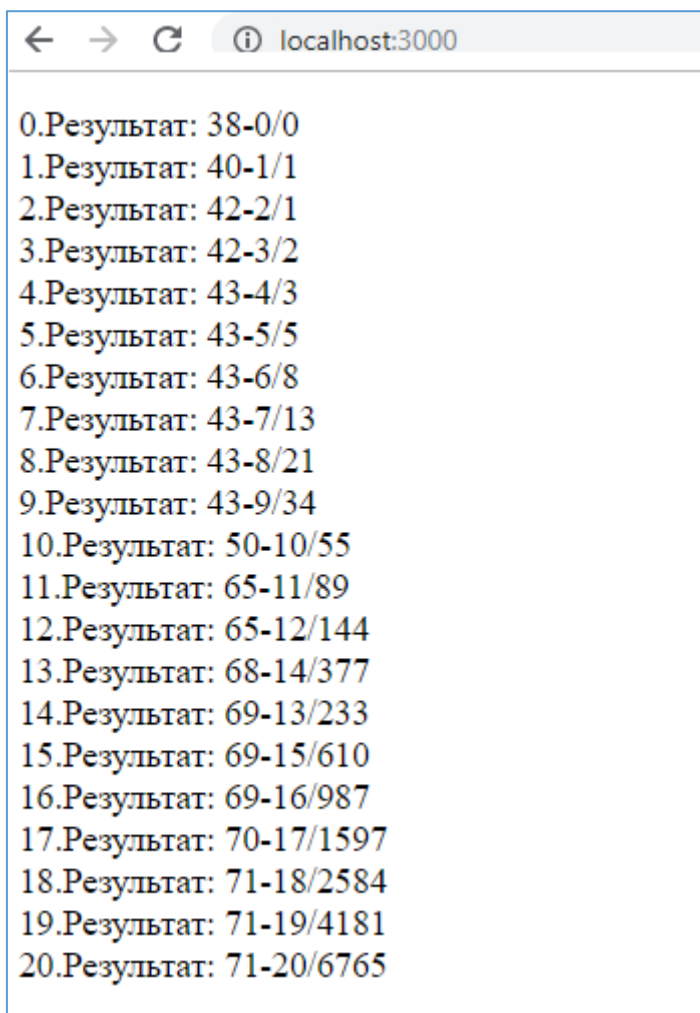
ПОИТ-3

Задание 02

8. Разработайте серверное приложение **03-02**, которое на GET-запрос вида <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.
9. Для расчета факториала используйте рекурсивный алгоритм.
10. Проверьте работоспособность приложения с помощью **POSTMAN**.

Задание 03

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



```
← → ↻ ⓘ localhost:3000
0.Результат: 38-0/0
1.Результат: 40-1/1
2.Результат: 42-2/1
3.Результат: 42-3/2
4.Результат: 43-4/3
5.Результат: 43-5/5
6.Результат: 43-6/8
7.Результат: 43-7/13
8.Результат: 43-8/21
9.Результат: 43-9/34
10.Результат: 50-10/55
11.Результат: 65-11/89
12.Результат: 65-12/144
13.Результат: 68-14/377
14.Результат: 69-13/233
15.Результат: 69-15/610
16.Результат: 69-16/987
17.Результат: 70-17/1597
18.Результат: 71-18/2584
19.Результат: 71-19/4181
20.Результат: 71-20/6765
```

13. 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 .
14. Выполните приложение запишите общую продолжительность всего цикла запросов.
15. Запустите приложение поочередно еще в двух вкладках браузера и запишите продолжительность всего цикла запросов для каждой вкладки.
16. Запустите приложение одновременно в трех вкладках браузера и запишите продолжительность всего цикла запросов для каждой вкладки.

Задание 04

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

Задание 05

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

Задание 06. Ответьте на следующие вопросы

21. Перечислите основные свойства глобальные объекты Node.js и поясните их предназначение.
22. Поясните понятие «асинхронная функция».
23. Поясните понятие стандартные «системные потоки».
24. Поясните назначение функций **process.nextTick**, **setImmediate**, поясните в чем разница.