

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

Факультет программной инженерии и компьютерной техники

Направление (специальность) – 09.04.02 Информационные системы и технологии

Специализация – Веб-технологии

**Дисциплина – Проектирование и анализ языков веб-сайтов/Исследование экосистем
веб-языков и веб-технологий**

Курсовой проект (работа)

ТЕМА: Исследование отладчика Node.JS

ВЫПОЛНИЛ

Студент группы

Р41621

№ группы

подпись, дата

Раджабов А.И.

ФИО

ПРОВЕРИЛ

к.пед.н., доцент

ученая степень, должность

подпись, дата

Государев И.Б.

ФИО

САНКТ-ПЕТЕРБУРГ

2019 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 ВВЕДЕНИЕ В NODE.JS	4
1.1 Установка и запуск	4
1.2 Модули для Node.JS	7
2 ОТЛАДКА СКРИПТОВ ПОД NODE.JS	10
2.1 Отладчик node debugger	10
1.2 Отладка под браузер Chrome	12
1.3 Отладка под IDE	14
ЗАКЛЮЧЕНИЕ	17
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	18

ВВЕДЕНИЕ

Node.js представляет собой среду выполнения кода на JavaScript, которая построена на основе движка JavaScript Chrome V8, позволяющего транслировать вызовы на языке JavaScript в машинный код [4]. Node.js прежде всего предназначен для отладки серверных приложений на языке JavaScript. Хотя также существуют проекты по написанию десктопных приложений (Electron) и даже по созданию кода для микроконтроллеров. Но прежде всего мы говорим о Node.js, как о платформе для создания веб-приложений.

Node.js — это open-source кроссплатформенная среда выполнения кода на JavaScript, которая работает на серверах. С момента выпуска этой платформы в 2009 году она стала чрезвычайно популярной и в наши дни играет весьма важную роль в области веб-разработки. Если считать показателем популярности число звёзд, которые собрал некий проект на GitHub, то Node.js, у которого более 50000 звёзд – это очень и очень популярный проект [6].

Платформа Node.js построена на базе JavaScript движка V8 от Google, который используется в браузере Google Chrome. Данная платформа, в основном, используется для создания веб-серверов, однако сфера её применения этим не ограничивается.

Node.js включает в себя полную внепроцессную утилиту по отладке, доступ к которой осуществляется через простой протокол на основе TCP и встроенный клиент. Для использования нужно запустить Node.js с аргументом `debug`, где нужно указать путь к скрипту отладки. Запрос будет осуществляться с отображением успешного запуска отладчика [5].

В курсовой работе будет исследована функциональность отладчика Node.js, реализованная с помощью вышеуказанной технологии, с подведением итогов проведенного анализа возможностей Node.js.

1 ВВЕДЕНИЕ В NODE.JS

1.1 Установка и запуск

Будут рассмотрены установка Node.JS и выполнение скриптов. Для этого необходимо перейти на сайт <http://nodejs.org> [3]. На сайте расположена ссылка, которая позволяет скачать наиболее подходящий пакет для определённой ОС. При работе с Node.JS лучше использовать последнюю версию пакета. Для дальнейшей работы будет использована операционная система Windows. После нажатия на ссылку скачивается файл с названием и расширением «node-v10.16.0-x64.msi». По окончании загрузки необходимо запустить данный файл установки и согласиться со всеми условиями, которые поставлены разработчиком. По умолчанию директорией установки Node.JS является C:\Program Files\nodejs. После в созданной по данному пути папке можно найти такие файлы, как node.exe и npm.cmd. NPM – это пакетный менеджер, он будет рассмотрен позже.

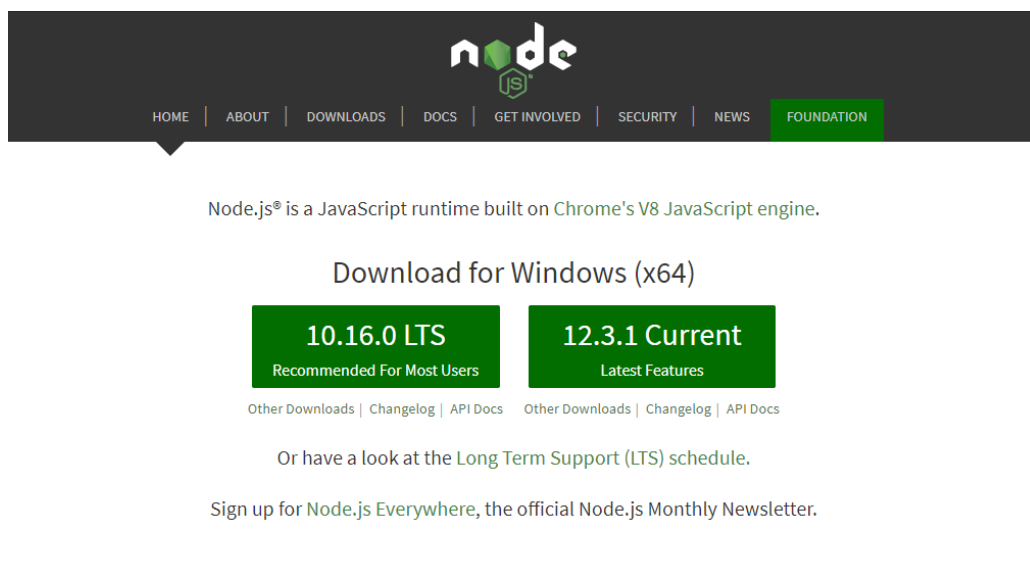


Рисунок 1 – Окно для скачивания Node.JS

Установленный Node.JS позволяет вызывать собственные инструменты через контекстное меню. Чтобы в этом удостовериться, в Windows 10 можно проделать следующее: нажать правой кнопкой мыши на клавишу «Пуск». В выпавшем контекстном меню следует выбрать пункт «Система». Далее в открывшемся окне выбирается пункт «Дополнительные параметры системы», и в новом окне во вкладке «Дополнительно» – клавиша «Переменные среды...».

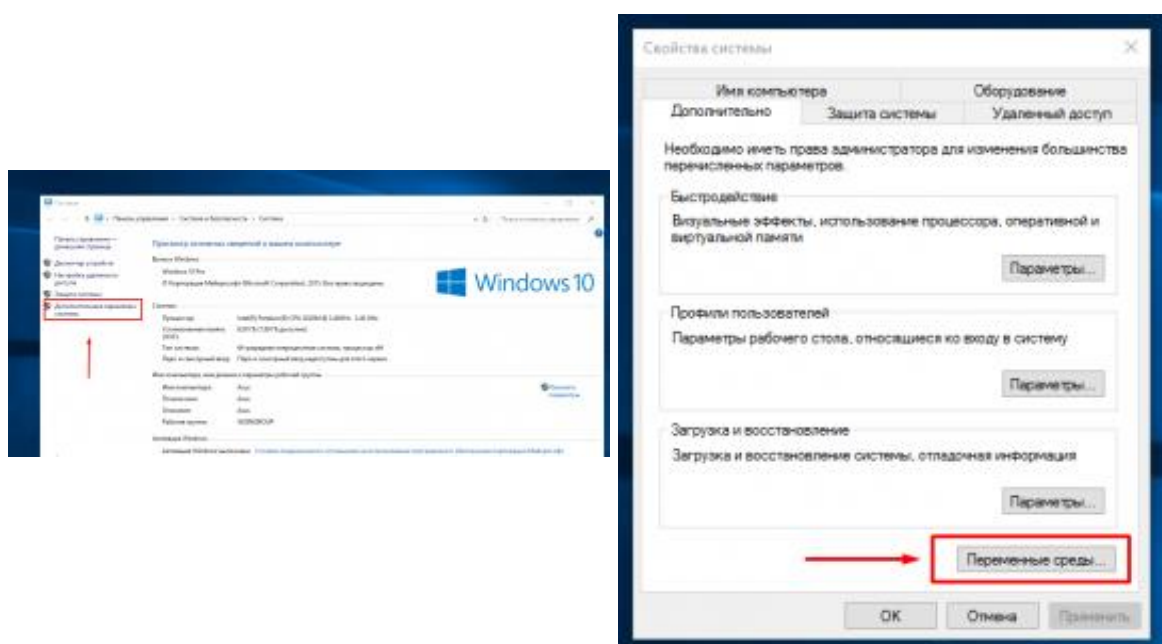


Рисунок 2.1-2.2 – Настройка параметров для Node.JS

После этого появится окно «Переменные среды», в котором нас интересуют записи в среду PATH [8], причем не важно, для всех пользователей или для конкретного лица (в нижнем или верхнем окне). Главное, что присутствуют записи «C:\Users\Username\AppData\Roaming\npm» и «C:\Program Files\nodejs\».

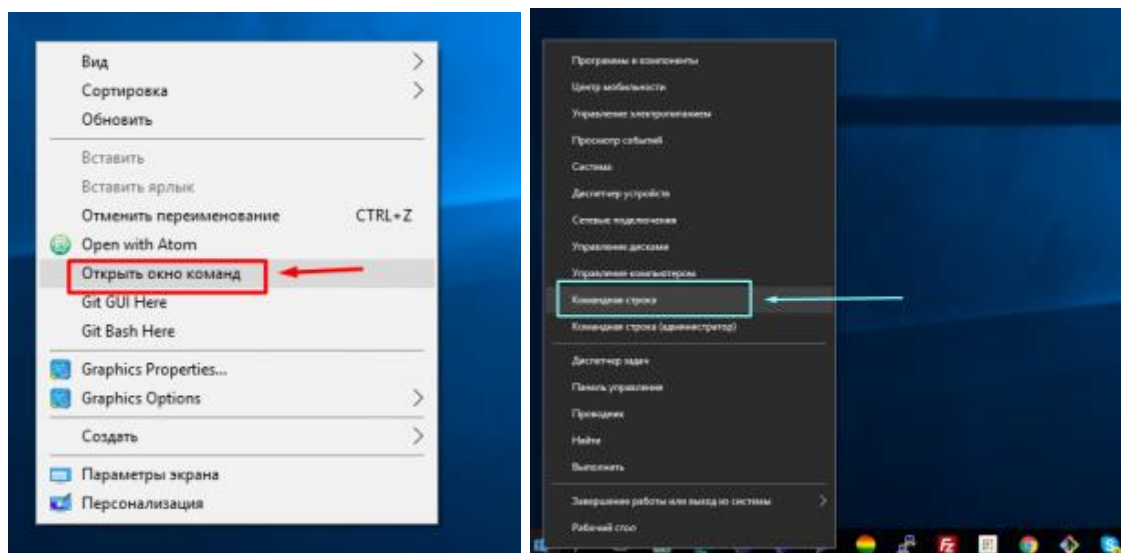


Рисунок 3.1-3.2 – Способы запуска командной строки

Далее для проверки работоспособности запускается консоль и вводится команда «node». Появляется приглашение ввести javascript выражение. Это так называемый режим « REPL », в котором можно вводить javascript выражения для дальнейшего их выполнения [1].

Двойное нажатие сочетания клавиш «Ctrl + C» позволяет выйти из этого режима. Этот режим используется достаточно редко, но в данном случае он потребовался для проверки того, успешно ли прошла установка [4]. Конечно же, такой запуск должен работать не только в Windows, но и в любой операционной системе. Как правило, в Node.JS запускают программы, написанные на JavaScript. Например, создадим файл «1.js» и введём в нём команду: “ console.log("Hello, world!");”. Она сделает то же самое, что и в браузере. Далее введём в консоли команду «node 1.js» и нажмём клавишу «Enter». Результат показан на Рисунке 4.

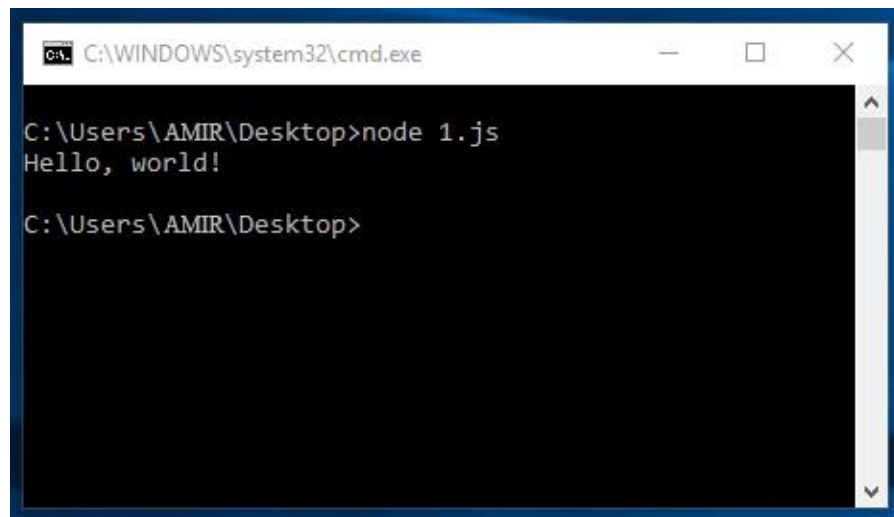


Рисунок 4 – Программа, запущенная в консоли с помощью Node.JS

1.2 Модули для Node.JS

Открывается WebStorm или любой другой редактор кода в новой пустой директории «start». В этой директории будет находиться проект Node.JS. Так как используется WebStorm, понадобится инициализация. Запускаем WebStorm, и в меню «File» выбираем пункт «Settings...» В левой колонке появившегося окна выбираем пункт и подпункт «Languages & Frameworks» и «Node.js and NPM», соответственно. Далее в правой части окна в секции «Coding Assistance» рядом с надписью «Node.js Core library is not enabled» нажимаем кнопку «Enable».

По умолчанию WebStorm считает, что редактируется html код с обычным браузерным JS, и предлагает варианты автодополнения, соответствующие браузерному скрипту. Под наши задачи же следует выбрать вариант для работы с Node.JS. Поэтому необходимо нажать на кнопку «Usage scope», которая появилась рядом с бывшей кнопкой «Enable». В появившемся окне выделяем строку с названием проекта и разворачиваем список подключенных библиотек, который находится на пересечении строки названия проекта со столбцом «Library» [6].

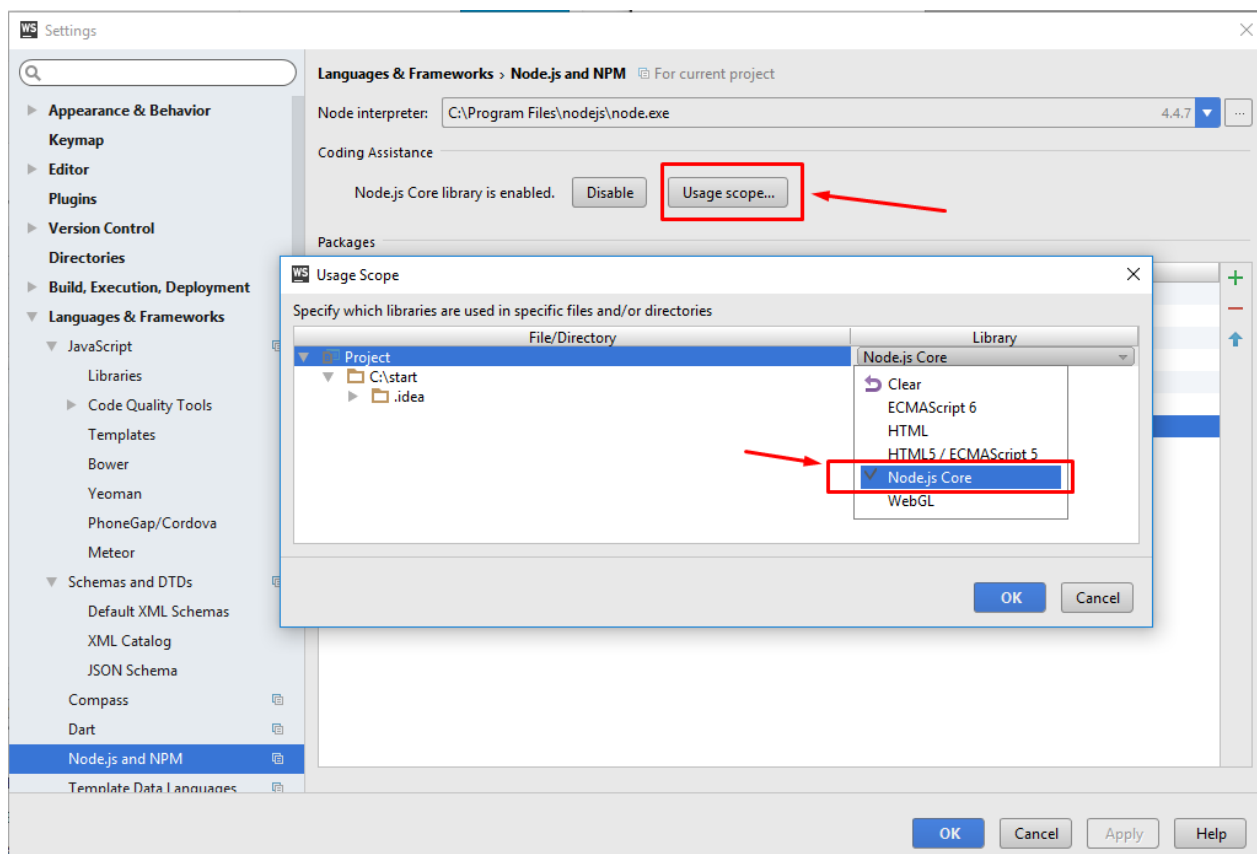


Рисунок 5 – Выбор варианта автодополнения для Node.JS

Итак, добавлена библиотека «Node.JS Core» для всего проекта. Инициализация завершена. Создаётся первый файл с названием «server.js». Для этого правой кнопкой мыши кликаем по названию проекта, в выпавшем меню наводим на New и кликаем по появившемуся пункту File. В новом окне вводим название будущего файла с разрешением «server.js».

Целью этого процесса является знакомство с модулями, поэтому будет написана простая программа на языке JavaScript.

Эта программа создаёт объект «User», который хранит имена пользователей. У пользователей могут быть методы: например, «hello», где «(who)» – это другой User. Далее будут созданы два имени пользователей и будет запущен код, вызывающий метод «hello» для одного из пользователей с именем другого пользователя в качестве аргумента.



```
JS server.js x
1 function User(name){
2   |   this.name = name;
3 };
4
5 User.prototype.hello = function(who){
6   |   console.log("Hello, " + who.name);
7 };
8
9 var amir = new User("Амир");
10 var ali = new User("Али");
11
12 amir.hello(ali);
```

Рисунок 6 – Пример кода в Node.JS

Такой метод уже был рассмотрен выше, где показано, как открыть окно команд в нужной папке. Открывается папка с проектом и вводится команда «C:\start>node server.js.»



```
C:\WINDOWS\system32\cmd.exe
C:\start>node server.js
Hello, Амир
C:\start>
```

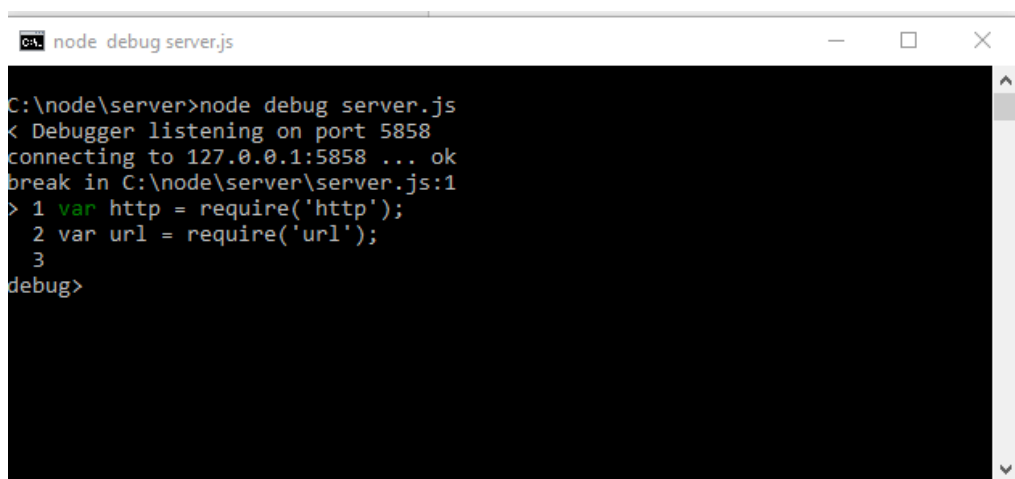
Рисунок 7 – Консоль для работы с Node.JS

Теперь надо усовершенствовать наш код. «User» – это достаточно мощный объект, который может иметь множество свойств. Со временем появится необходимость вынести этот «User» в отдельный файл. И для реализации этого используются модули.

2 ОТЛАДКА СКРИПТОВ ПОД NODE.JS

2.1 Отладчик node debugger

В этой главе будет исследована отладка с помощью Node.JS. Для этого рассмотрим самый простой встроенный отладчик, который называется «node debug». Например, если server.js выдаёт ошибку, то следует остановить его выполнение на некотором моменте для наблюдения за переменными. Наблюдение же осуществляется следующим образом: в месте прерывания хода программы вставляется «debugger» и запускается скрипт режима отладки – «C:\node\server> node debug server.js».

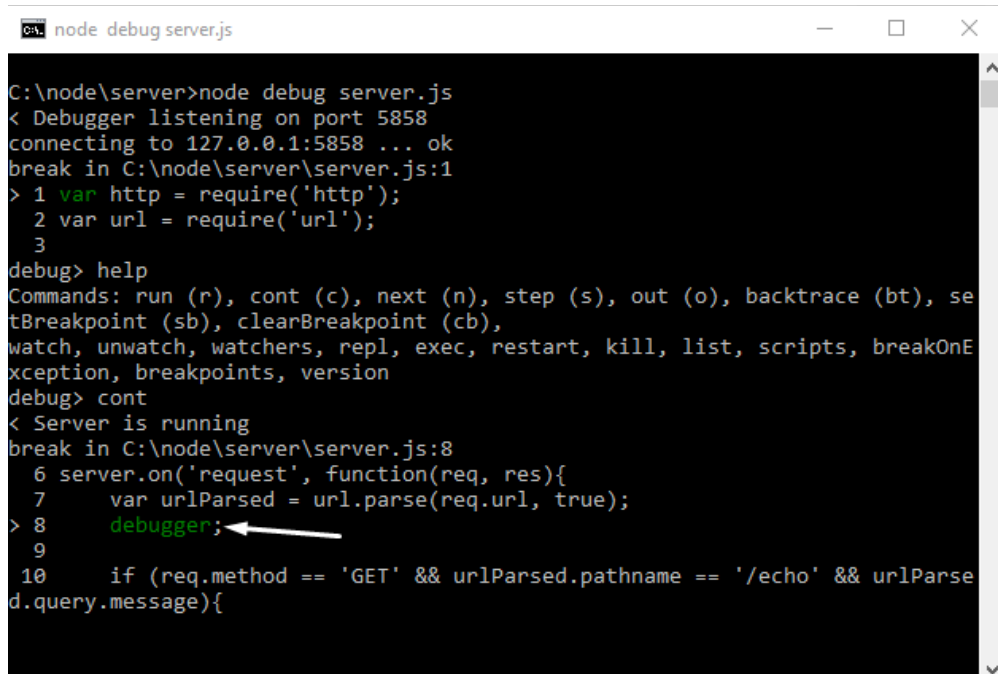


```
C:\node\server>node debug server.js
< Debugger listening on port 5858
connecting to 127.0.0.1:5858 ... ok
break in C:\node\server\server.js:1
> 1 var http = require('http');
  2 var url = require('url');
  3
debug>
```

Рисунок 8 – Запуск отладчика Node.JS в командной строке

Изначально скрипт приостановлен. В режиме отладки есть несколько команд: например, «help», которая выводит список всех остальных, или необходимая нам сейчас команда «cont», которая продолжит выполнение скрипта. Можно увидеть, что последний вывод в console текст «Server is running» [5].

Откроем браузер и перейдём по ссылке «<http://127.0.0.1:1337/>». В консоли видно, что произошло событие ‘request’, запустился обработчик и выполнение остановилось на команде «debugger».



```
C:\node\server>node debug server.js
< Debugger listening on port 5858
connecting to 127.0.0.1:5858 ... ok
break in C:\node\server\server.js:1
> 1 var http = require('http');
  2 var url = require('url');
  3
debug> help
Commands: run (r), cont (c), next (n), step (s), out (o), backtrace (bt), se
tBreakpoint (sb), clearBreakpoint (cb),
watch, unwatch, watchers, repl, exec, restart, kill, list, scripts, breakOnE
xception, breakpoints, version
debug> cont
< Server is running
break in C:\node\server\server.js:8
  6 server.on('request', function(req, res){
  7     var urlParsed = url.parse(req.url, true);
>  8     debugger;
  9
 10     if (req.method == 'GET' && urlParsed.pathname == '/echo' && urlParse
d.query.message){
```

Рисунок 9 – Ход отладки Node.JS, возвращённый в консоль

Теперь через консоль можно перейти в «repl» — режим выполнения команд и выяснить, что такое «urlParsed». Например, запустить выполнение строки «res.end(«Hallo NodeJS!»)».

Надо сказать, что встроенный отладчик, который было рассмотрено-самое простое. Его используют в тех редких случаях, когда более продвинутые способы отладки, не работают. Гораздо удобнее осуществлять отладку используя инструменты разработчика в браузере Chrome.

1.2 Отладка под браузер Chrome

Для этого понадобится утилита, которая называется «node inspector». Поставим её глобально, и подготовка завершена. У Node.JS при запуске есть специальный параметр «—debug» для которого можно задать путь к файлу скрипта “C:\node\server> node --debug server.js”. Когда Node.JS запускается с этим параметром, то он не только запускает server.js, но и начинает отслеживать происходящее на порту «5858» [9]. К этому порту может подключиться другая программа и задавать команды Node.JS, относящиеся к отладке.

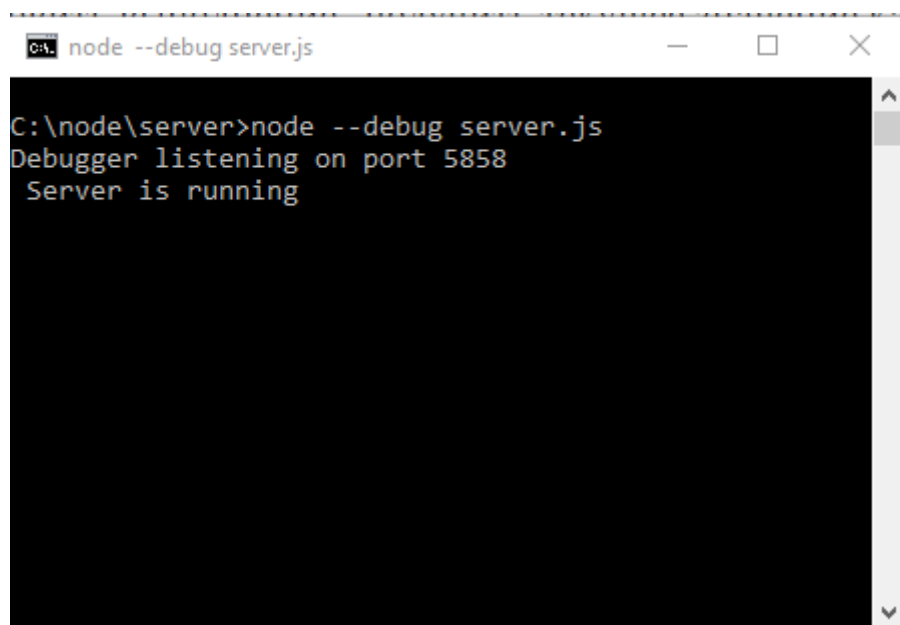


Рисунок 10 – Сводка о состоянии сервера

Node-inspector — это веб-сервер, к которому можно подсоединиться посредством URL «<http://127.0.0.1:8080/?port=5858>» для работы с отладчиком. Node-inspector отправляет команды веб-серверу, который транслирует их далее в Node для отслеживания с использованием языка отладки «V8».

При переходе по заданному адресу сработала функция-обработчик запросов «request» — та функция, которая передает эти данные вторым параметром метода «server.on()».

Если бы Node.JS был запущен без флагов отладки, то команда debugger была бы проигнорирована. В нашем же случае она сработала, V8 приостановил выполнение программы на JavaScript и послал node-inspector, который подключен к Node по порту «5858». Node-inspector получил данные от функции «request» и при помощи протокола «WebSocket», с которым ознакомимся далее, переслал их далее для выполнения на «сервер». Здесь на это отреагировал уже веб-интерфейс.

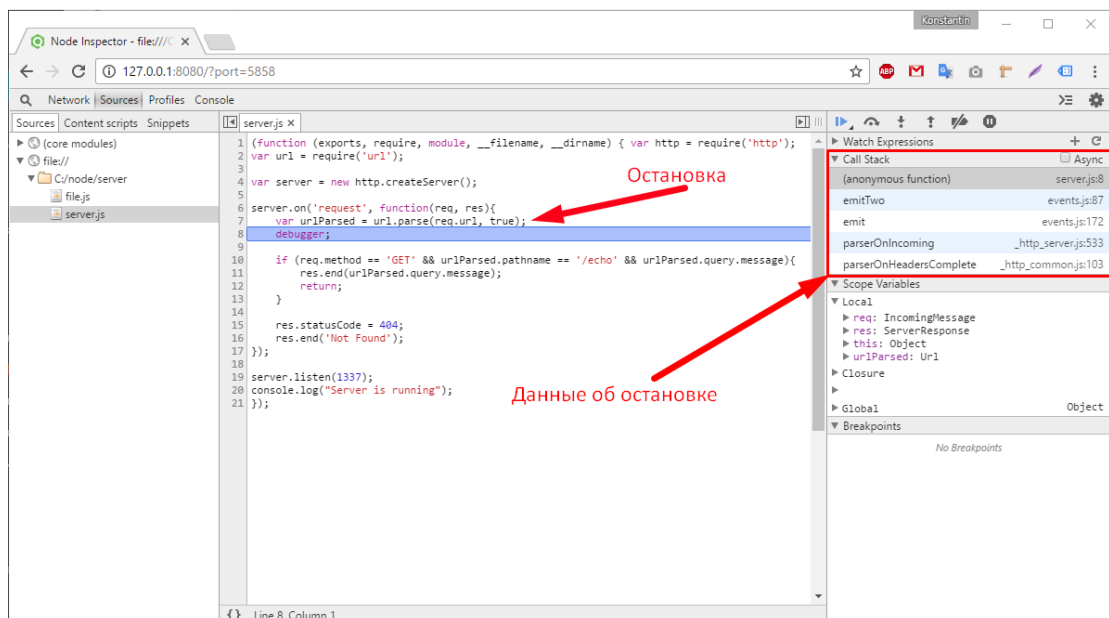


Рисунок 11 – Веб-интерфейс отладчика Node.JS

Теперь, если перейти по адресу сервера (http://127.0.0.1:1337/), то никакой отладки не произойдет, потому что «server.js» перестанет работать. Для того чтобы отладить программу в этом случае, можно зайти в отладчик и повторить действия, как в отладчике браузера.

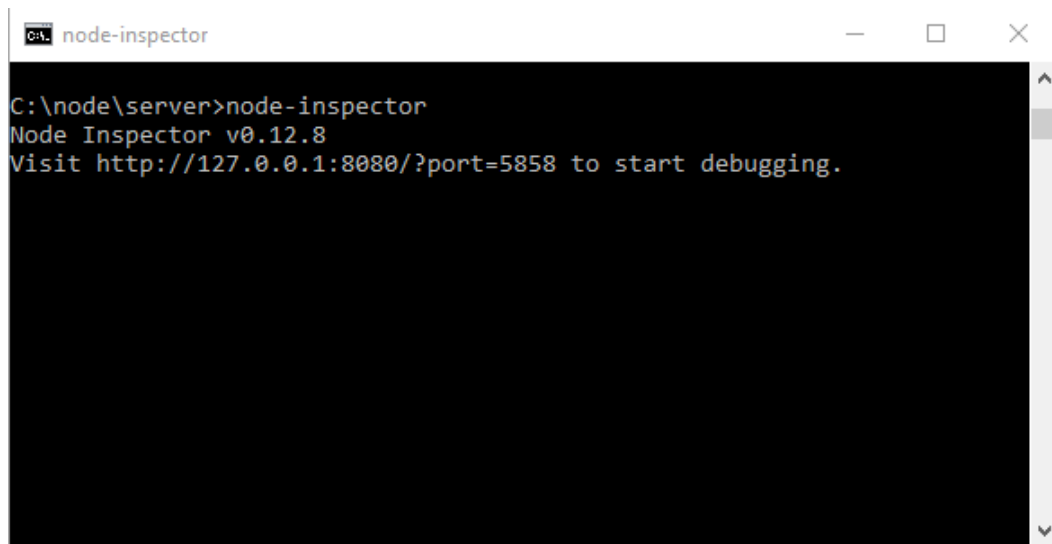


Рисунок 12 – Консоль для работы с Node.JS

1.3 Отладка под IDE

В качестве IDE будет использоваться WebStorm, и для того, чтобы запустить отладку, нам подойдет уже готовая конфигурация.

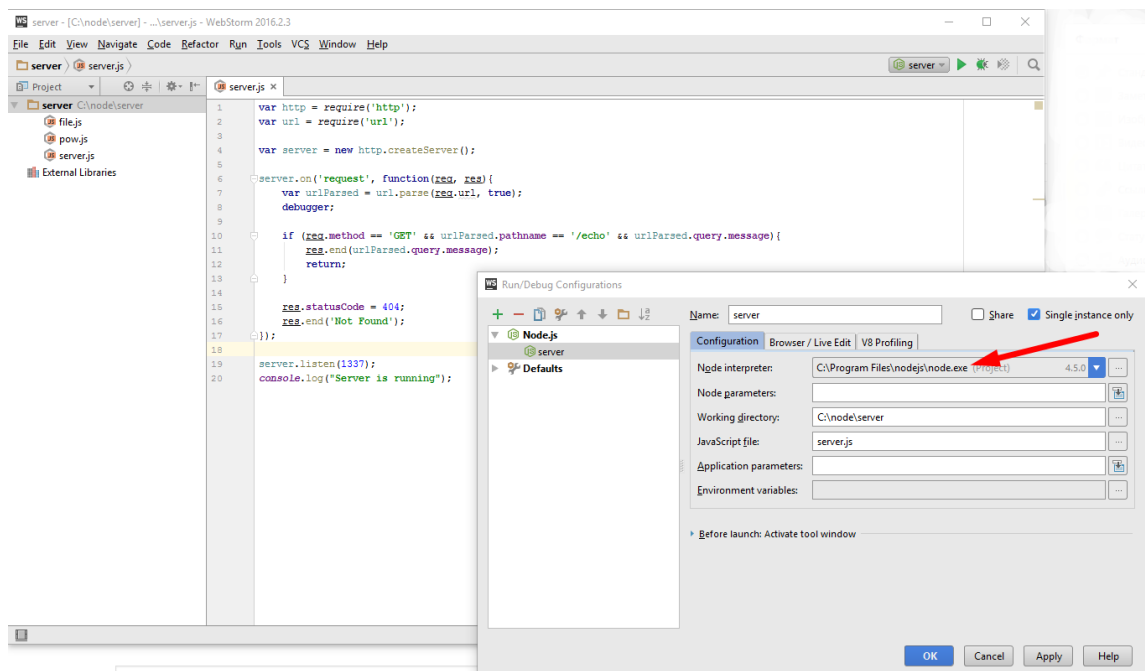


Рисунок 13 – Конфигурация Node.JS

Нам понадобится лишь, чтобы запускалась именно Node, а не supervisor.

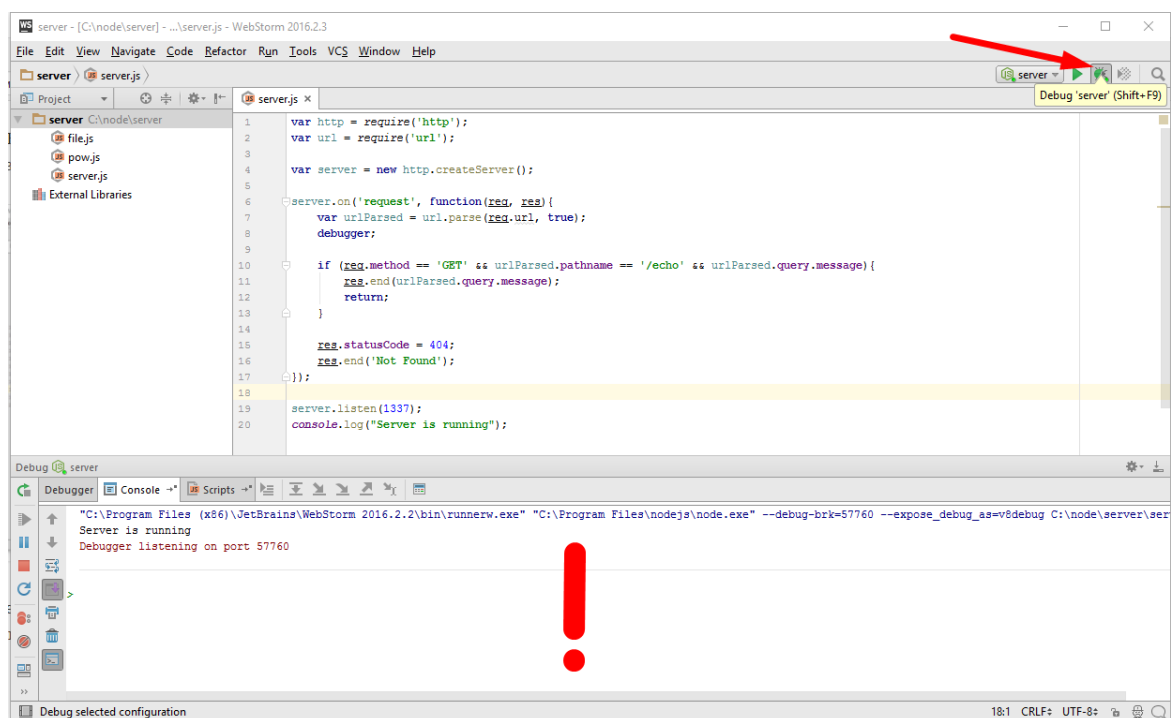


Рисунок 14 – Запуск Node.JS

Смотрим в консоли, что получилось. При запуске в режиме отладки WebStorm добавляет к Node соответствующий параметр «—debug-brk» и указывает ему значение. Данное значение – порт, на котором Node должна ожидать подключение отладчика (по умолчанию — 5858). Ранее к этому порту подключался node-inspector, но сейчас к нему подключен сам WebStorm [6]. Он реализует необходимый интерфейс, и если зайти в браузере по URL «<http://127.0.0.1:1337/echo?message=TEST>» и перейти в окно WebStorm, то получается результат, продемонстрированный на Рисунке 15.

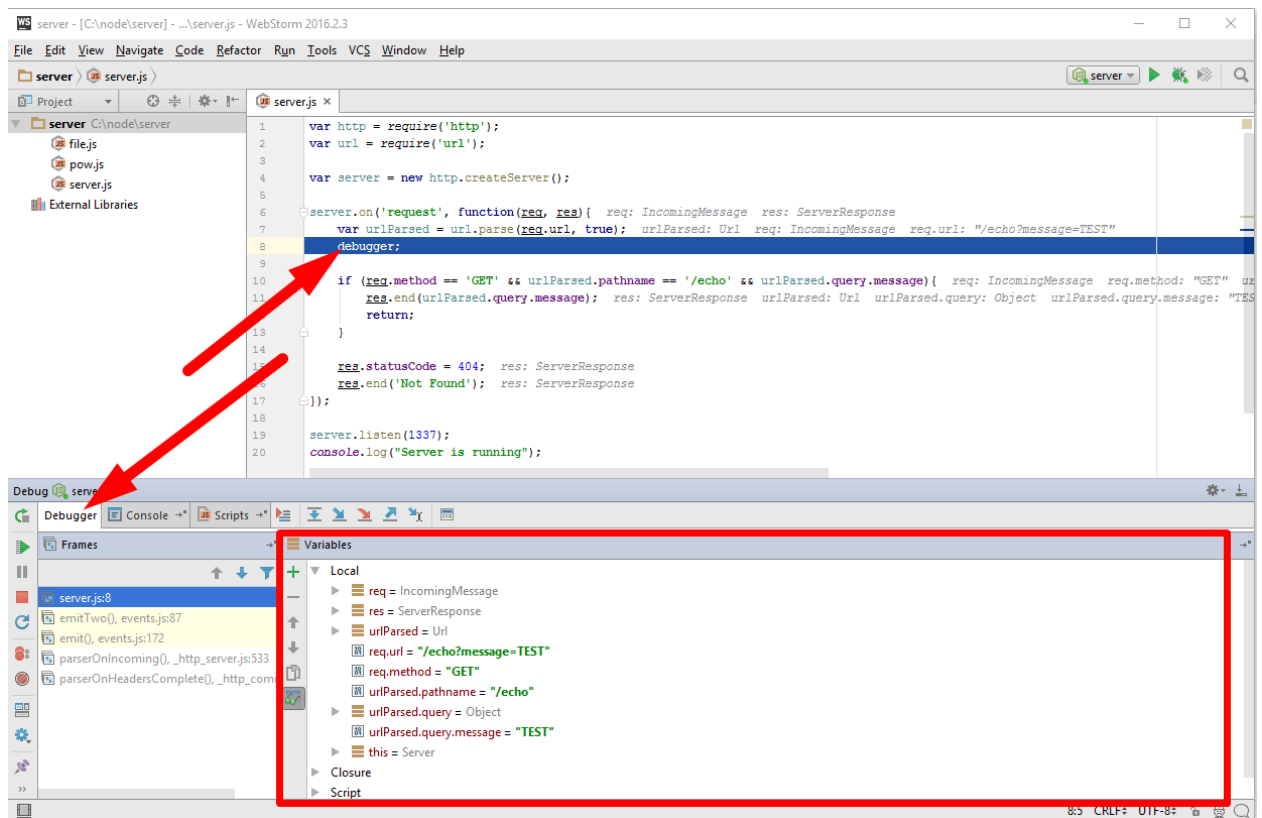


Рисунок 15 – Окна с результатами анализа debugger'а

Процесс останавливается, и в окне Variables можно наблюдать за переменными, анализировать полученные данные, смотреть urlParsed и так далее.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была изучена программная платформа Node.JS и проанализирована функциональность отладчика, реализованная на ней. Были продемонстрированы установка и настройка Node.JS для Windows 10, задокументировано и продемонстрировано в скриншотах каждое действие, выполненное по ходу работы.

Далее был создан тестовый файл “server.js”, который был открыт в редакторе WebStorm, загружен в качестве тестовой программы и запущен на локальном хосте по адресу “http://127.0.0.1:1337/” для тестирования отладчика Node.JS. Были настроены и добавлены расширения для работы с Node.JS в редакторе WebStorm для более удобной работы и тестирования отладчика. Рассмотрен отладчик Node-debug, отладчик под браузер Chrome и отладчик под IDE, каждый из которых был продемонстрирован в консоли и в веб-интерфейсе.

Таким образом сделан вывод, что при работе с программной платформой NodeJS и реализации проектов, отладчик может стать отличным помощником при выявлении ошибок и их исправлении для корректной работы. Также было решено использовать в модулях библиотеку «Node.JS Core» для распознавания Node.JS кода и работы с ним.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Введение в JavaScript [Электронный ресурс] URL: <https://learn.javascript.ru/intro>
2. JavaScript: Возможности и ограничения [Электронный ресурс] URL: https://puzzleweb.ru/javascript/00_teacher.php
3. Портал для скачивания Node.JS [Электронный ресурс] URL: <https://nodejs.org/en/>
4. Отладка Node.js с помощью Google Chrome [Электронный ресурс] URL: <https://medium.com/nuances-of-programming/>
5. Как отлаживать приложения Node.js [Электронный ресурс] URL: <http://qaru.site/questions/917/how-do-i-debug-nodejs-applications>
6. Debugger [Электронный ресурс] URL: <https://js-node.ru/site/article?id=18>
7. Отладка Node.js в Chrome DevTools [Электронный ресурс] URL: <https://canonium.com/articles/debugging-nodejs-in-chrome-devtools/>
8. Отладчик node js с древовидным [Электронный ресурс] URL: <https://ru.stackoverflow.com/questions/768414/>
9. Среда разработки на node js с отладчиком [Электронный ресурс] URL: <https://ru.stackoverflow.com/questions/568388/>
10. Node.js debugger based on Blink Developer Tools [Электронный ресурс] URL: <https://github.com/node-inspector/node-inspector>
11. Изучаем Node.js. — СПб.: Питер, 2014. — 400 с.: ил. — (Серия «Бестселлеры O'Reilly»).
12. JavaScript. Подробное руководство, 6 е издание. — Пер. с англ. — СПб: Символ-Плюс, 2012. — 1080 с.

Ссылка лабораторных работ на GitHub: <https://github.com/amir9595/goss>