**Лабораторная работа № 1**

**Цель работы:**

* установить NVM;
* установить последнюю версию Node.js;
* ознакомиться с составляющими Node;
* вывести строчку “Hello, World!” через консоль.

**Ход работы:**

Для начала работы необходимо установить Node, используя Node Version Manager (nvm). Для этого воспользуемся установщиком с сайта

<https://github.com/coreybutler/nvm-windows#installation--upgrades>

Затем когда установка завершится, откроем PowerShell (рекомендуется открыть с повышенными правами администратора) и попробуйем использовать windows-nvm, чтобы узнать, какие версии узла сейчас установлены (на этом этапе они не указаны): **nvm ls** (рисунок 1).



Рисунок 1 – Текущая версия Node, установленная на устройстве

В случае если node не установлен(а он и не будет установлен) нужно будет прописать в консоль следующую команду **nvm install <version>**(вместо <version> написать интересующую вас версию). Затем чтобы использовать напишите **nvm use <version>**

А чтобы посмотреть используемую версию node воспользуемся командой **node -v**

**Node: что это такое, для чего нужен и из чего состоит**

Что такое Node.js

Node или Node.js — это программная платформа, которая позволяет использовать JavaScript не только для браузера. В этой статье мы подробно разберем, что такое Node.js и какие особенности она имеет.

Что такое Node.js 1

Node.js

Node.js ― это не отдельный язык программирования, а среда для выполнения JavaScript. Или, другими словами, платформа для работы с JavaScript на стороне сервера. Раньше JavaScript мог выполнять только браузер. С помощью Node.js это можно делать без него.

Чаще всего Node используют для написания серверной части сайта (для неё он и был создан изначально), однако сейчас на нём пишут и десктопные программы. На данный момент эту технологию для своего бэкенда используют PayPal, eBay, Amazon и Netflix.

Платформу разработал Райан Дал, а спонсором разработки выступила компания Joyent. Среда была представлена в 2009 году. Node.js основана на движке V8. V8 ― это программа, которая преобразует JavaScript в машинный код, то есть код понятный процессору. Этот же движок используется в основе браузера Google Chrome.

Одного V8 для удобной работы мало, так как он может работать только с JavaScript. Он не умеет читать файлы, не может работать с сетью вне браузера, поэтому Райан Дал вместе с разработчиками с помощью дополнительных библиотек и кода добавили дополнительные возможности к V8. Это позволило сделать из JavaScript веб-сервер.

Плюсы Node.js:

Использование одного языка JavaScript как на клиентской стороне в браузере, так и на сервере.

NPM (Node Package Manager) ― это менеджер пакетов, или набор готовых модулей. Это своеобразный склад, где программисты делятся своим кодом. Комьюнити очень большое, поэтому в NPM на данный момент собрано огромное количество решений для разных задач.

Увеличение скорости работы. Node.js ― это однопоточная и асинхронная система. Вы можете читать файлы, отправлять электронные письма, запрашивать данные и совершать другие действия одновременно. Подробнее об однопоточной работе мы расскажем ниже.

Node.js основы: что такое многопоточная и однопоточная работа

При изучении серверных процессов нужно знать, что есть 2 схемы работы с сервером ― многопоточная и однопоточная.

Работа с веб-приложением ― это передача запроса и получение ответа от сервера. Многопоточная система линейна. Вначале выполняется один запрос, затем другой и так далее по очереди. Например, нам нужно обратиться к базе данных. Веб-сервер делает запрос и ждёт, пока не придёт ответ. Как только ответ вернулся, он обрабатывает его и может приступить к отправке других запросов. Как видите, в процессе ожидания веб-сервер тратит ресурсы, но ничего не делает.

При выполнении запроса потребляются ресурсы сервера: память, процессорное время и количество ядер. Получается, при многопоточном подходе сервер может выполнить только один запрос за раз? Нет. Каждый запрос помещается в отдельный поток. Поток ― это время и ресурсы, которые выделяет сервер для выполнения небольшого блока инструкций. Таким образом, количество потоков зависит от мощности машины. Чем больше ресурсов, тем больше потоков и тем больше запросов может выполнить сервер одновременно. Такая модель работы называется thread-per-request model (модель потока на запрос).



Рисунок 2 – Многопоточная система

**Движок V8**

V8 — движок JavaScript с открытым исходным кодом, разработанный датским отделением компании Google.

Краеугольными камнями V8 по мнению ведущего разработчика Ларса Бака являются:

* Компиляция исходного кода JavaScript непосредственно в собственный машинный код, минуя стадию промежуточного байт-кода.
* Эффективная система управления памятью, приводящая к быстрому объектному выделению и маленьким паузам сборки «мусора».
* V8 приостанавливает исполнение кода во время выполнения сборки «мусора».
* Уменьшает влияние и воздействие приостановки приложения при сборке «мусора».
* V8 может точно определять, где находятся в памяти объекты и указатели, что позволяет избежать утечки памяти при ложной идентификации объектов в качестве указателей.
* Введение скрытых классов и встроенных кэшей, ускоряющих доступ к свойствам и вызовам функций.

V8 исполняет JavaScript-сценарии в особых «контекстах», которые по сути являются отдельными виртуальными машинами. Правда в одном процессе может работать только одна виртуальная машина, несмотря на возможность использования нескольких потоков. Таким образом, несмотря на динамическую природу JavaScript, разработчикам удалось применить методы, характерные для реализации классических объектно-ориентированных языков, такие как компиляция кода «на лету», внутреннее кэширование, точный процесс сборки мусора, снэпшоттинг при создании контекстов.

Движок V8 отличается от других движков (JScript, SpiderMonkey, JavaScriptCore, Nitro) высокой производительностью.

**Библиотека Libuv**

Libuv - это кроссплатформенная библиотека поддержки, написанная на C. Изначально он был написан для поддержки Node.js , но теперь он также используется в Julia, Luvit, Neovim и т. Д… Он разработан на основе асинхронной модели, управляемой событиями.

Некоторые из наиболее важных функций Libuv включают:

* Полнофункциональный цикл событий, поддерживаемый epoll, kqueue, IOCP, портами событий;
* TCP-сокеты, которые в узле представляют модуль net;
* Разрешение DNS;
* Сокеты UDP, которые представляют модуль dgram в Node;
* Файловый ввод-вывод, который включает просмотр файлов, операции с файловой системой;
* Дочерние процессы;
* Пул рабочих потоков для обработки блокирующих задач ввода-вывода (также, начиная с узла 10.5, доступен модуль worker-threads для параллельного выполнения Javascript);
* Примитивы синхронизации для потоков (например, блокировки, мьютексы, семафоры, барьеры);

Высокоуровневую архитектуру Libuv можно рассматривать как два уровня. Верхний уровень, который включает сетевой ввод-вывод, файловый ввод-вывод, операции DNS и пользовательский код, представляет собой API, который может использоваться клиентами Libuv. Нижний уровень представляет собой код, который обрабатывает опрос ввода-вывода и рабочие потоки. IOCP обрабатывает опрос ввода-вывода в Windows. Epoll, Kqueue и порты событий обрабатывают опрос ввода-вывода в системах Unix (epoll используется в Linux, kqueue - в BSD, порты событий - в Solaris).

Цикл событий является центральной частью Libuv и выполняется в основном потоке. Это полубесконечный цикл. Он продолжает выполняться до тех пор, пока не будет выполнена какая-либо работа (например, неисполненный код, активные обработчики или запросы), и когда работы больше нет цикл продолжается.

**Кодинг**

Работать будем в текстовом редакторе VS Code

Откроем проект в виде папки и в ней создадим файл app.js, в котором запишем следующую команду(рисунок 3):

console.log(“Hello, World!”)



Рисунок 3 – Создание файла и первой строчки кода

Данная команда выводит в консоли сообщение “Hello, World!”. Запустим данный файл через консоль и получим результат, представленный на рисунке 4.

**Ссылка на проект в GitHub**

<https://github.com/r0uven/WebProgramming_1Lab>



Рисунок 4 – Результат работы файла Index.js

**Вывод:** в ходе лабораторной работы был установлен Node.js и изучены его основополагающие компоненты, а также запущен всеми известный код “Hello, World!”.