Лабораторная работа №1 «Введение в Node.js»

Цель: изучить основы Node.js

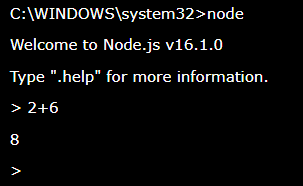
Node.js представляет среду выполнения кода на JavaScript, которая построена на основе движка JavaScript Chrome V8, который позволяет транслировать вызовы на языке JavaScript в машинный код. Node.js прежде всего предназначен для создания серверных приложений на языке JavaScript. Хотя также существуют 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.

**Ход работы:**

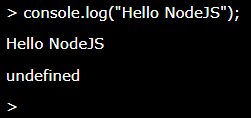
1. Установить node.js с официального сайта (<https://nodejs.org/en/download/>)
2. Установить любой редактор кода.
3. Произвести контроль версии при помощи команды *node -v* в командной строке или терминале.

После установки NodeJS нам становится доступным такой инструмент как REPL. REPL (Read Eval Print Loop) представляет возможность запуска выражений на языке JavaScript в командной строке или терминале.

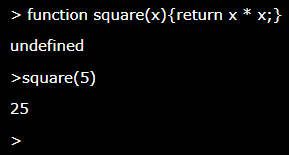
Так, запустим командную строку (на Windows) или терминал (на OS X или Linux) и введем команду *node*. После ввода этой команды мы можем выполнять различные выражения на JavaScript:



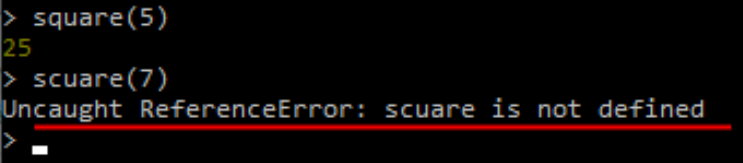
Можно использовать стандартные функции JS:



Или определять свои функции, а затем их вызывать:



При вводе неверного значения REPL указывает на ошибку:



Вместо ввода кода в консоль удобнее использовать внешние файлы. Создадим на жестком диске новый каталог в который поместим js файл со следующим кодом:

console.log(“Hello world!”);

Для выполнения внешнего файла необходимо в командной строке при помощи команды cd перейти к созданному каталогу и выполнить команду:

node helloworld.js

Данная команда выполняет код из файла helloworld.js.

Создадим файл lib.js и определим в нем следующий код:

const sum = (a, b) => a + b;

const PI = 3.1415;

const person = {

name: "nick",

age: 20

}

// module.exports.sum = sum;

// module.exports.PI = PI;

// module.exports.person = person;

module.exports = {

sum: sum,

PI: PI,

person: person

};

Создадим файл app.js, который использует функционал модуля lib:

const mylib = require("./lib");

console.log(mylib.sum);

Для запуска приложения перейдем с помощью команды *cd* к его каталогу, затем вызовем команду *node app.js*.

При разработке консольного приложения предполагается интерактивное взаимодействие пользователя с программой. Модуль обработки **readline** предоставляет интерфейс для чтения данных из потока ввода (например, process.stdin) по одной строке за раз. Следующий простой пример иллюстрирует базовое использование модуля:

const readline = require("readline");

const rl = readline.createInterface({

    input: process.stdin,

    output: process.stdout

});

rl.question('Enter your name:\n', (name) => {

   console.log(`Hello, ${name}`);

    rl.question('Enter your age:\n', (age) => {

        console.log(`You are ${age} years old!!!`);

        if (Number(age) > 20) console.log(`You may already vote!!!`);

        else console.log(`You need to grow up`);

        rl.close();

     });

});

**Задание**

Задание 1. В соответствии со своим вариантом написать скрипт для вычисления значения функции b=f(x,y,z). Значения x, y и z должны вводиться пользователем. Скрипт оформить в модуль, экспортирующий функцию b. В файле app.js импортировать функция и вычислить значение функции для константных значений x, y и z. Запустить скрипт при помощи node.js.

| Вариант | Вид функции | Вариант | Вид функции |
| --- | --- | --- | --- |
| 1 |  | 16 |  |
| 2 |  | 17 |  |
| 3 |  | 18 |  |
| 4 |  | 19 |  |
| 5 |  | 20 |  |
| 6 |  | 21 |  |
| 7 |  | 22 |  |
| 8 |  | 23 |  |
| 9 |  | 24 |  |
| 10 |  | 25 |  |
| 11 |  | 26 |  |
| 12 |  | 27 |  |
| 13 |  | 28 |  |
| 14 |  | 29 |  |
| 15 |  | 30 |  |

Задание2. Выполнить задание1, передав значения x, y и z как параметры командной строки.

Задание 3. Выполнить задание 1, используя модуль readline и запрашивая ввод значений x, y, z. При выводе исходных данных и результата предусмотреть форматирование.

**Порядок выполнения**

Создание проекта

Для создания проекта необходимо настроить package.json. Он представляет собой файл манифеста для проекта Node.js и содержит сведения о метаданных проекта. Кроме того, он регулирует, например, порядок управления зависимостями, то, какие файлы попадут в пакет, предназначенный для npm, и многое другое.

Файл package.json не пишется вручную. Он создается в результате выполнения команды init. Эту команду можно выполнять двумя основными способами.

* npm init. Эта команда запускает мастер, предлагающий заполнить такие данные, как имя проекта, его версия, описание, точка входа, команда тестирования, репозиторий GIT, ключевые слова, автор и лицензия.
* npm init -y. Эта команда с флагом -y представляет собой ускоренную версию команды npm init. Ускорение связано с тем, что эта команда не интерактивная. Вместо этого все поля, которые требуется заполнить при выполнении команды npm init, получают значения по умолчанию.

Ниже представлена структура файла package.json

"scripts" : {

"start" : "node ./dist/index.js",

"test": "jest",

"build": "tsc",

"lint": "eslint"

}

Независимо от того, используется ли Node.js, любой проект, скорее всего, потребует запуска, тестирования и сборки. Разработчики Node.js это поняли и предоставили рекомендации по именованию скриптов. Идея в том, чтобы во всех проектах Node.js имена скриптов были согласованы. Возможность быстро перемещаться между проектами Node.js и ориентироваться в них по единообразному набору действий намного упрощает разработку.

Действие start запускает приложение. Действие test выполняет тесты, используя платформу тестирования jest. Далее действие build использует компилятор TypeScript tsc для компиляции кода из TypeScript, чтобы код был понятен для браузера. Наконец, программа анализа кода eslint ищет несоответствия и возможные ошибки в коде.

Для вызова действий нужно ввести npm run <action>. При этом существуют особые действия: start и test. Они являются специальными, так как можно опустить run. Поэтому можно вместо npm run start ввести чуть более короткую команду npm start.

Для установки пакетов используется встроенная программа командной строки npm. Команда установки – npm install имя\_пакета.