PHASE 1

WEEK 1

DAY 1

ELBRUS/BOOTCAMP

План

1. Node.js и REPL
2. NPM и NPX
3. CommonJS: экспорт и подключение модулей
4. FS: модуль работы с файловой системой
5. process.argv

Node.js

Node.js

Среда выполнения JS-кода, которая даёт доступ к операционной системе.

Даёт возможность:

- запускать JS на сервере
- разбивать JS-программу на модули
- принимать / обрабатывать запросы от других компьютеров
- работать с файловой системой

Установка Node.js

Можно устанавливать одним из двух способов:

- Напрямую
  https://nodejs.org/en/download/
- Через NVM (Node Version Manager) - рекомендуется
  Сторонняя программа, позволяет переключаться между версиями node в системе.
  - Для Linux, MacOS
    https://github.com/nvm-sh/nvm
  - Для Windows
    https://github.com/coreybutler/nvm-windows

Проверка версии Node

Проверить, установлен ли Node, можно командой в терминале.

```
node -v
```

Команда выводит версию Node, которая установлена и используется сейчас.

Запуск .js файла в Node

Запустить файл script.js из активной папки и выполнить все операции внутри.

Команда в терминале:

node script.js

Использование расширения Code Runner в VSCode (появляется кнопка Play):

https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner

REPL в Node

После установки Node.js нам становится доступным такой инструмент как REPL.

REPL (Read Eval Print Loop) представляет возможность запуска выражений на языке JavaScript в командной строке или терминале.

Команда в терминале для запуска REPL:

```
node
```

Команда в терминале для выхода из REPL:

```
.exit
```

NPM

Node Package Manager

NPM

Программа для работы с пакетами (модулями) Node. Устанавливается автоматически вместе с Node.

Пакет (модуль) - завершённый изолированный кусок кода.

https://npmjs.com - общедоступный регистр опубликованных пакетов от сторонних разработчиков.

Yarn - альтернатива NPM. Программа от Facebook для работы с пакетами. Устанавливается отдельно.

NPM-пакет

Любая папка, внутри которой есть файл package.json - это NPM-пакет (NPM-проект).

Инициализировать NPM-проект в активной папке можно командой:

npm init

Инициализировать NPM-проект, автоматически отвечая “да” на каждый вопрос:

npm init --yes

NPM-пакет

Важно!

Сразу после того, как сделали init, надо обязательно создать файл .gitignore и добавить в него папку node_modules

или воспользоваться командой (рекомендуется):

npx create-gitignore Node

Dependencies / Зависимости

Проект может требовать для работы какие-то пакеты из NPM.
Эти пакеты называются *зависимостями*.

У этих пакетов могут быть свои зависимости, и так далее.

Информация о зависимостях проекта хранится в package.json

Информация точных версиях всех используемых зависимостей и подзависимостей находится в package-lock.json

package.json, example code

```
{
  "name": "d1",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "node app.js",
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.17.1"
  }
}
```

Основные типы зависимостей

- **dependencies** – список пакетов, необходимых для работы приложения
- **devDependencies** – пакеты, которые нужны только для разработки
  - Jest,
  - ESLint,
  - nodemon
  - и др. инструменты

Установка / удаление зависимостей

- Установка обычной зависимости: npm install <pack>
- Установка dev зависимостей: npm install -D <pack> <pack> <pack>
- Удаление зависимости: npm uninstall <pack>

Пример установки пакета для проверки числа на нечётность:

npm install is-odd

Варианты установки пакета

- Локально – работают только в текущей папке. Зашли в папку другого проекта – надо ставить по новой
- Глобально – установлены в папку текущего пользователя компьютера. Работают из любой папки.

Важно!

Надо ставить зависимость глобально (флаг -g) только если точно понимаешь, зачем это нужно в этом конкретном случае. Рекомендуется всё ставить локально.

Работа с готовым проектом

1. Скачать репозиторий, перейти в папку проекта
2. Проверить наличие package.json
3. Установить зависимости командой npm i / npm ci (для строгой установки)
4. Приступить к работе :)

NPX

- npx — программа, автоматически устанавливается вместе с Node и NPM.
- npx позволяет скачивать и сразу запускать Node-пакеты.

  Например:
  npx eslint --init — пакет eslint скачается на компьютер, выполнится команда init, затем скачанные файлы удалятся.

CommonJS

require / exports

Синтаксис require / export используется для импорта и экспорта переменных, функций, классов между модулями (файлами) в проекте Node.js

Важно!
require / export не работают вне Node.js!

```
// экспортирующий файл lib.js:

const x = 5;
const addX = (value) => value + x;


module.exports = { x, addX };
```

```
// импорт из файла lib.js:

const { x, addX } = require('./lib');


console.log(x); // 5
console.log(addX); // [Function: addX]
```

FS

File System

Подключение и использование модуля fs

```
// подключение модуля файловой системы
const fs = require('fs');

// название и расширение создаваемого файла
const fileName = './text.txt';

// запись файла на диск + его данные
fs.writeFileSync(fileName, 'FS module work!');

// чтение файла с диска в указанной кодировке
fs.readFileSync(fileName, 'utf-8');
```

FS: синхронная работа с файлами

```
// создать файл
fs.writeFileSync('./newFile.txt', 'data in file');

// дописать в созданный файл
fs.appendFileSync('./currentFile.txt', 'newData in currentFile.txt');

// удалить файл
fs.unlinkSync('./removeFile.txt');

// копировать файл
fs.copyFileSync('./origFile.txt', './copyFile.txt');
```

FS: синхронная работа с папками

```
// создать папку
fs.mkdirSync('./newFolder');

// удалить папку
fs.rmdirSync('./removeFolder');

// переименовать файл или папку
fs.renameSync('./newFolder', './node_example');

// проверка существования файла или папки
fs.existsSync('./someFile.txt');
```

process.argv

Argument values

Аргументы командной строки

process.argv – массив, содержащий аргументы командной строки. Первым элементом будет node, вторым элементом будет название файла JavaScript запущенного в текущем процессе. Следующие элементы будут любыми дополнительными аргументами командной строки.

Командная строка:

```
node index.js --par1 --par2
```

index.js:

```
console.log(process.argv); // ['node', 'index.js', '--par1', '--par2']
```

process.argv, example code

```
const fs = require('fs');

const userCard = `*****************************
* ${('User: ' + process.argv[2]).padEnd(26)} *
* ${('Group: ' + process.argv[3]).padEnd(26)} *
* ${('Phase: ' + process.argv[4]).padEnd(26)} *
*****************************`;

fs.appendFileSync('./users.txt', `${userCard}\n\n`);
```