Лабораторна робота №1

Тема: Програмна платформа Node.js

Теоретичні відомості

Node.js - програмна платформа, яка дозволяє виконувати **JavaScript** за межами браузера, а це дає змогу писати серверну частину проекту на **JavaScript**. Таким чином, **JavaScript** перетворюється з вузькоспеціалізованої мови клієнтських сценаріїв в мову загального призначення.

Node.js не є мовою програмування чи веб-сервером (не містить жодних **config**-файлів), це середовище для виконання **JavaScript**-файлів.

Node.js надає **JavaScript** можливість взаємодіяти з пристроями вводу-виводу через свій **API** (написаний на **C++**), підключати зовнішні бібліотеки, які написані на інших мовах, забезпечує взаємодію з ними з **JavaScript**-коду. **Node.js** базується на асинхронному (реактивному) програмуванні (орієнтованому на події) з неблокованим вводом-виводом.

Пакетом (модулем) в **Node.js** називають один або кілька **JavaScript**-файлів, які є певною бібліотекою або інструментом.

NPM (**Node Package Manager**) - стандартний менеджер пакетів, який автоматично встановлюється з **Node.js**. Він використовується для стягування пакетів з хмарного сервера **npm**, або для завантаження пакетів на цей сервер.

Файл **package.json** містить інформацію про проект:назва, версії, залежності та інше. Будь-який каталог, який містить цей файл, інтерпретується як Node.js - пакет, навіть якщо ви не збираєтесь його публікувати.

Для встановлення пакету використовується команда

npm install ***<назва_пакету>***

Для запуску **JavaScript** файлу на виконання використовується команда

node ***<назва_файлу>***

Хід роботи:

1) Заходим на сайт **https://nodejs.org**, завантажуєм та інсталюємо останню версію **Node.js** для вашої операційної системи;

2) Запускаєм **CommandPrompt** і дивимось на встановлені версії платформи **node.js** та менеджера пакетів **npm**

node -v

npm -v

3) Створюєм на робочому столі папку **myapp** з файлом **script.js**

```
function User(first,last){
    this.first=first;
    this.last=last;
}
User.prototype.fullName=function(){
    console.log('fullname: '+this.first+" "+this.last)
}
var user=new User('Petro','Petrenko');
console.log('firstname:'+user.first);
console.log('lastname:'+user.last);
user.fullName();
```

4) Запускаєм файл **script.js** на виконання за межами браузера за допомогою **CommandPrompt**

cd desktop/myapp

node script

5) Створюєм в папці **myapp** файл-модуль **user.js**, в який переносимо вмістиме файлу **script.js** за виключенням створення екземпляру та виводів, і додаєм в **user.js** конструкцію **module.exports**, яка повертає назовні об'єкт

```
function User(first,last){
    this.first=first;
    this.last=last;
}
User.prototype.fullName=function(){
    console.log('fullname: '+this.first+" "+this.last)
}
module.exports=User;
```

6) В файлі **script.js** підключаємо створений користувацький модуль **user.js** . Команда **require** задає відносний шлях до файлу

```
var User=require('./user');
var user=new User('Petro','Petrenko');
console.log('firstname:'+user.first);
console.log('lastname:'+user.last);
user.fullName();
```

7) Запускаєм файл **script.js** на виконання, отримуємо той самий результат

node script

8) В папці myapp створюємо файл **package.json**

```
{
    "name": "myapp",
    "version": "1.0.0",
    "main": "server.js",
    "author": "username",
    "license": "ISC"
  }
```

9) Будемо розгортати веб-сервер на платформі **Node.js** за допомогою фреймворку веб-проектів **express.** Інсталювати фреймворк **express** в каталог **myapp** за допомогою пакетного менеджера **npm.** Для цього необхідно виконати команду

npm install express --save-dev

В каталозі **myapp** з'явиться підкаталог **node_modules** з встановленим фреймворком **express** та всіма його залежностями. Ключ **save** означає локальне встановлення в текучий каталог, а ключ **dev** пропише залежність встановленого модуля в файл **package.json** (в ньому з'явиться властивість **devDependencies**)

```
"devDependencies": {
        "express": "^4.16.3"
}
```

Ключова властивість **main** в файлі **package.json** задає точку входу для нашого проекту - файл **server.js**, який нам необхідно створити.

10) Створюємо в каталозі **myapp** файл **server.js**

```
//підключаємо модуль express
var express=require('express');
//створюємо проект
 var app=express();
//папка для віддачі статичного контенту (каталог проекту)
 app.use(express.static(__dirname));
//опрацювання кореневого шляху -віддати клієнту index.html
 app.get('/',function(req,res){
     res.sendFile(__dirname+'/index.html');
 })
//порт прослуховування для сервера
//автоматичний підбір для віддаленого сервера,
//або 8080 для localhost
app.listen(process.env.PORT||8080);
//повідомлення про запуск сервера
console.log('Run server!');
```

11) Створюємо в каталозі **myapp** файл **index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>MyApp</title>
</head>
<body>
    <h1>MyApp</h1>
</body>
</html>
```

12) Запускаємо сервер. Виконаємо команду

node server.js

В браузері в адресній стрічці вводимо **localhost:8080** і здійснюємо перехід - отримаємо від сервера файл **index.html.**

13) Платформа **Node.js** дозволяє працювати з файлами, для цього існує модуль **fs**, який встановлюється разом з платформою **Node.js**. Створюємо в папці **myapp** файл з даними **data.json**

```
[
    {"username":"Ivan","password":"123","age":34},
    {"username":"Andriy","password":"124","age":38},
    {"username":"Petro","password":"125","age":41}
]
```

14) Створюємо в папці **myapp** файл **client.js** і підключаємо його в файлі **index.html** після бібліотеки **jquery**

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>MyApp</title>
    <script
        src="https://code.jquery.com/jquery-3.3.1.min.js"
        integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
        crossorigin="anonymous">
    </script>
    <script src='client.js'></script>
</head>
```

15) При завантаженні файлу **index.html** ввідправляєм з клієнта **get**-запит на сервер для отримання даних з файлу **data.json**. Логіку реалізуєм в файлі **client.js**

```
$(document).ready(function(){
    function getUsers(){
        $.get('/getusers',function(data){
            console.log(data);
        })
    }
    getUsers();
})
```

16) Підключаємо модуль **fs** і додаєм обробник шляху **'/getusers'** в файлі **server.js**

```
var fs=require('fs');
app.get('/getusers',function(req,res){
    fs.readFile('data.json','utf-8',function(err,data){
        console.log(data);
        res.send(data);
    })
 })
```

Для читання даних з файлу використовується асинхронний метод **readFile**, який приймає параметрами назву файду, кодування та функцію зворотнього виклику, другий параметр якої (**data**) містить дані файлу. Метод **send** відправляє дані клієнту (відповідь сервера).

17) Додайте в тіло файлу **index.html** тег **div** з **id='table'**

```
<body>
    <h1>MyApp</h1>
    <div id='table'></div>
</body>
```

18) Напишіть функцію створення таблиці **createTable(container,mas)**, де **container** - елемент, в якому необхідно створити таблицю, а **mas** - дані для таблиці (масив об'єктів).

19) Відповідь сервера (масив об'єктів) повинна вдображатись на **html** -сторінці у вигляді таблиці. Модифікуйте функцію **getUsers()** в файлі **client.js** наступним чином

```
function getUsers(){
        $.get('/getusers',function(data){
            createTable($('#table',data))
        })
    }
```

20) Створіть в каталозі **myapp** файл **style.css**, який буде задавати стилі для таблиці (всі необхідні класи додавати програмно в функції **createTable**).