Лабораторна робота № 5

Тема: Node.js як засіб написання сценаріїв

Мета: Засвоїти елементи програмування серверної частини застосувань мовою Node.js.

Основні елементи Node.js

`Node.js` - це опенсорсне кросплатформове середовище виконання для JavaScript, яке працює на серверах.

Платформа `Node.js` побудована на базі `JavaScript` двигун V8 від Google, який використовується в браузері Google Chrome. Ця платформа в основному використовується для створення веб-серверів, проте сфера її застосування цим не обмежується.

Однією з основних привабливих особливостей `Node.js` є швидкість. `JavaScript`-код, що виконується в середовищі `Node.js`, може бути вдвічі швидше, ніж код, написаний компілюваними мовами, на кшталт `C` або `Java`, і на порядки швидше за інтерпретовані мови на кшталт `Python` або `Ruby`.

У традиційних мовах програмування (`C, Java, Python, PHP`) всі інструкції за замовчуванням є блокуючими, якщо тільки розробник явно не подбає про асинхронне виконання коду. В результаті якщо, наприклад, у такому середовищі, зробити мережевий запит для завантаження якогось JSON-коду, виконання потоку, з якого зроблений запит, буде припинено доти, доки не завершиться отримання та обробка відповіді.

`JavaScript` значно спрощує написання асинхронного та неблокуючого коду з використанням єдиного потоку, функцій зворотного виклику (коллбеків) та підходу до розробки, що базується на подіях. Щоразу, коли нам потрібно виконати важку операцію, ми передаємо відповідному механізму коллбек, який буде викликаний одразу після завершення цієї операції. В результаті, щоб програма продовжила роботу, чекати результатів виконання подібних операцій не потрібно.

Подібний механізм виник у браузерах. Ми не можемо дозволити собі чекати, скажімо, закінчення виконання AJAX-запиту, не маючи при цьому можливості реагувати на дії користувача, наприклад на клацання по кнопках. Для того, щоб користувачам було зручно працювати з веб-сторінками, все, і завантаження даних з мережі, і обробка натискання на кнопки має відбуватися одночасно, в режимі реального часу.

Якщо ви створювали колись обробник події натискання на кнопку, то ви вже користувалися методиками асинхронного програмування.

Завдяки простоті та зручності роботи з менеджером пакетів для `Node.js`, який називається `npm`, екосистема `Node.js` розвивається. Зараз у реєстрі `npm` є понад півмільйона опенсорсних пакетів, які може вільно використовувати будь-який `Node.js`-розробник.

Інсталяція Node.js

Менеджер пакетів macOS, який є фактичним стандартом у цій галузі, називається `Homebrew`. Якщо у вашій системі є, ви можете встановити `Node.js`, виконавши цю команду в командному рядку:

```
brew install node
```

Список менеджерів пакетів для інших операційних систем, у тому числі для Linux і Windows, можна знайти тут

https://nodejs.org/en/download/package-manager/

Найпопулярнішим менеджером версій `Node.js` є `nvm`.

Прикладом першої програми `Node.js` можна назвати простий веб-сервер. Ось його код:

```
const http = require('http')
const hostname = '127.0.0.1'
const port = 3000
const server = http.createServer((req, res) => {
  res.statusCode = 200
  res.setHeader('Content-Type', 'text/plain')
  res.end('Hello World\n')
})
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`)
})
```

Щоб запустити цей код, збережіть його у файлі `server.js` і виконайте в терміналі таку команду:

```
node server.js
```

Для перевірки сервера відкрийте який-небудь браузер і введіть в адресному рядку http://127.0.0.1:3000, тобто ту адресу сервера, яка буде виведена в консолі після його успішного запуску. Якщо все працює як треба - на сторінці буде виведено Hello World.

`Node.js` – це низькорівнева платформа. Для того, щоб спростити розробку для неї та полегшити життя програмістам, було створено величезну кількість бібліотек. Деякі з них стали популярними.

Наприклад:

`Express` – Ця бібліотека надає розробнику дуже простий, але потужний інструмент для створення веб-серверів. Ключем до успіху `Express` став мінімалістичний підхід та орієнтація на базові серверні механізми без спроби нав'язати якесь бачення «єдино правильної» серверної архітектури.

`Socket.io` Це бібліотека для розробки мережевих програм реального часу.
https://socket.io

Завдання до лабораторної роботи № 5:

Спроектувати веб сторінку згідно макета **Messages**
(**https://cacoo.com/diagrams/ZvVhYS3UpG5PdbBy/EDE3A**
Обираємо для розробки шаблон Messages
Використовуємо Gitlab (https://gitlab.com)

Розробити 2 складові частини проекту :

- Використати бібліотеку socket.io
- Сайт з формою, що надсилає на сервер дані чату
- Серверна частина перевіряє чи повідомлення конкретному студенту чи декільком/всім студентам і повертає відповідь (response) всім кому це повідомлення адресовано
- Сайт реагує на відповіді. І в потрібних чатах показує повідомлення