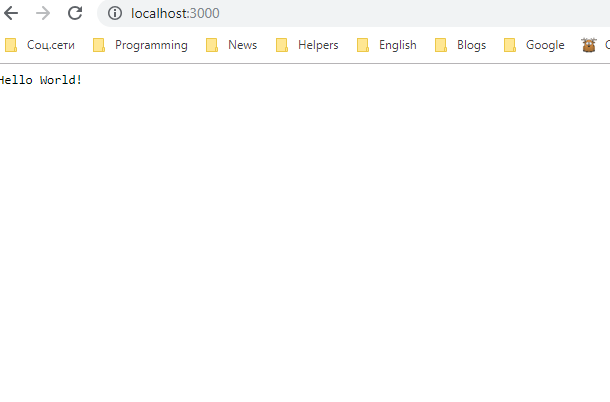
**Лабораторна робота 2\_8**

**Тема «Node.js»**

Завдання:

Створити і запусти HTTP-сервер на Node.js, що видає «Hello, world»

Розв’язок:



Контрольні питання

1. Розкажіть що собою представляє Node.js.
2. Які переваги використання Node.js?
3. Яка послідовність встановлення Node.js?
4. Поясніть принцип роботи Node.js?

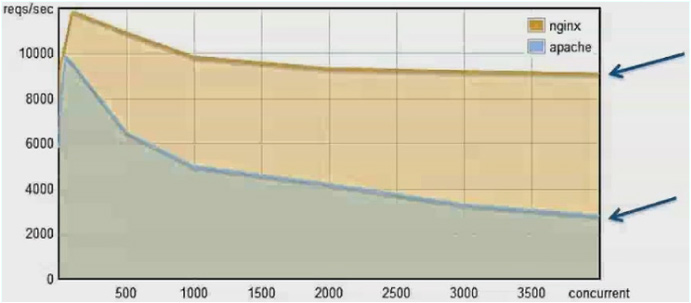
**Відповіді:**

1. **Node.js** — це JavaScript–оточення побудоване на JavaScript–рушієві Chrome V8. Node.js використовує подієву, неблокуючу I/O модель, що робить його легким та ефективним. Пакетна екосистема Node.js, npm, є найбільшою у світі екосистемою бібліотек з відкритим кодом.

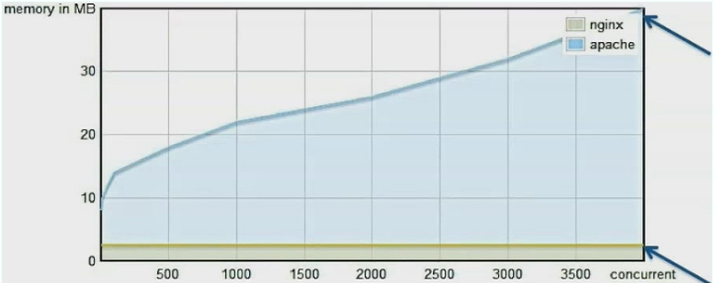
2. **Переваги Node.js**

Як ви могли помітити, в Node.js є тільки один потік, який обслуговує клієнтські запити і отримує POSIX результати. Цикл між користувачем і інтерфейсом POSIX дає можливість передавати дані туди і назад, а самі асинхронні операції здійснюються в POSIX. Дана модель отримала назву event-driven non-blocking IO model.

Продуктивність в такій системі набагато вище, ніж, якщо використовувалася многопоточная модель (multi-threaded blocking model). Прикладом многопоточної моделі є веб-сервер Apache і Nginx. Нижче представлені їх показники продуктивності.



І нижче наводиться порівняння пам'яті між ними.



3. Встановлення

Для налаштуванн оточення буде досить зробити два кроки:

1. Спочатку встановіть сам сервер Node.JS.

* Якщо у вас Unix-система - рекомендується зібрати останню версію з початкових кодів, а також NPM. Ви впораєтеся.
* Якщо Windows - відвідайте сайт http://nodejs.org або скачайте установник (32 або 64-бітний) з розширенням .msi з http://nodejs.org/dist/latest/.

1. Виберіть директорію, в якій будете вирішувати завдання. Запустіть в ній:

npm install node-static. Це встановить в поточну директорію модуль node-static, який стане автоматично доступним для скриптів з піддиректорій.

Якщо у вас Windows і команда не спрацювала, то швидше за все справа в тому, що не «підхопилися» нові шляхи. Перезапустіть ваш файловий менеджер або консоль.

Перевірка. Перевірте інсталяцію.

Для цього:

Створіть якусь піддиректорію і в ній файл server.js з таким вмістом:

var http = require('http');

var static = require('node-static');

var file = new static.Server('.');

http.createServer(function(req, res) {

file.serve(req, res);

}).listen(8080);

console.log('Server running on port 8080');

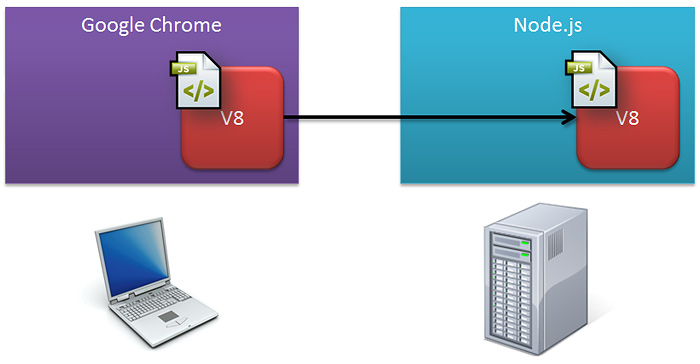
Запустіть його: node server.js.

Повинно вивести:

Server running on port 8080

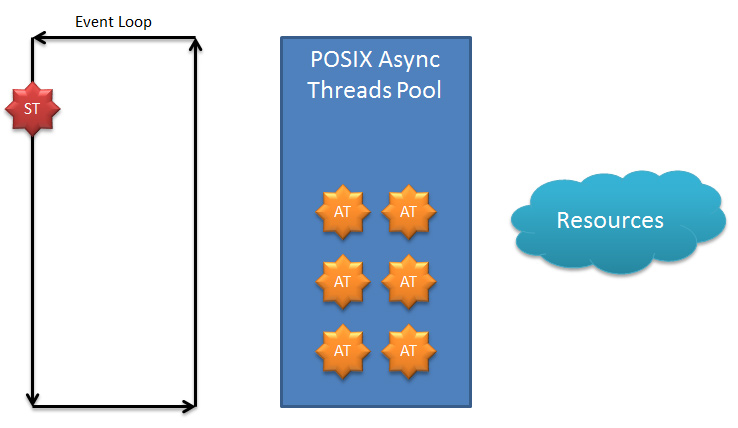
Відкрийте в браузері <http://127.0.0.1:8080/server.js>. Повинно вивести код файлу server.js. Якщо все працює - відмінно, тепер ви готові працювати з Node.js.

**4. Перш за все, Node.js** відрізняється від класичного JavaScript тим, що виконуваний код виконується на стороні сервера (backend), а не на стороні браузера. Для інтерпретації коду Node.js використовує движок V8, який в даний час застосовується в Google Chrome.

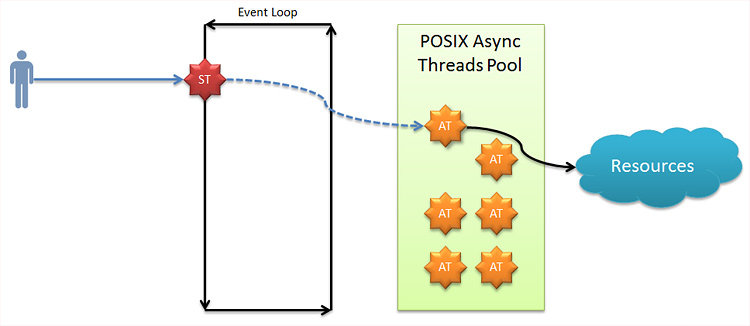


Крім того, всі механізми обробки запитів та інших операцій введення / виводу (I / O) побудовані на події. Це означає, що в Node.js немає ніякого способу, щоб заблокувати працює в даний момент потік. Кожна операція в Node.js виконується асинхронно. Це є величезною перевагою, особливо якщо ваш код повинен бути побудований на операціях введення-виведення: читання дисків, підключення до бази даних, веб-сервіси і т.д.

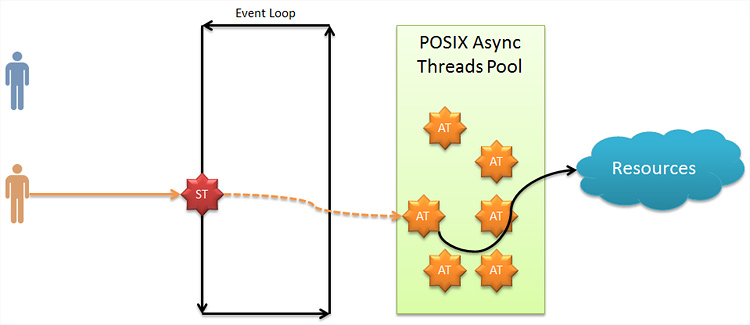
На відміну від IIS або Apache, Node.js не використовують многопоточную модель. У Node.js є тільки один робочий потік, який обслуговує всі запити користувачів і відповідають ресурси (на малюнку позначено у вигляді ST зірки). І існує POSIX басейн асинхронних потоків Node.js, який містить безліч асинхронних потоків (AT зірки) для операцій введення-виведення.



Коли користувач відправляє запит вводу-виводу, окремий ST потік обслуговує його, однак він не буде виконувати операції введення-виведення. Замість цього, ST потік буде йти в басейн POSIX асинхронних потоків, щоб забрати AT потік, зробити операцію до кінця, а потім назад повернутися в початковий стан для проведення будь-яких інших запитів. Таким чином, AT потік буде виконувати операції введення-виведення асинхронно.



Припустимо, перед завершенням попереднього AT потоку операції введення-виведення заходить інший користувач. При цьому ST потік буде обслуговувати новий запит користувача і для нього буде підібраний вже інший AT потік з POSIX.



Повернення потоку у двох користувачів буде працювати за тією ж схемою, що і відправлення запиту.

