Ćwiczenie: Tworzymy własny serwer HTTP z wykorzystaniem Node.js

Cel ćwiczenia:

Samodzielne stworzenie prostego serwera HTTP, który wyświetla stronę HTML, wczytuje plik CSS oraz uruchamia skrypt JavaScript.

Krok 1: Przygotowanie projektu

- Utwórz folder projektu np. `serwer-http`.
- Otwórz terminal w tym folderze.
- Zainicjuj projekt Node.js komendą:

  ```
  npm init -y
  ```

Krok 2: Tworzenie pliku serwera (`server.js`)

- W folderze projektu utwórz plik `server.js`.
- Załaduj wymagane moduły Node.js:
  - `http` – obsługa serwera HTTP,
  - `fs` – odczytywanie plików,
  - `path` – zarządzanie ścieżkami do plików.
- Serwer powinien działać pod adresem:
  `127.0.0.1`, na porcie `5555`.

Krok 3: Obsługa ścieżek URL na serwerze

Serwer powinien obsługiwać następujące adresy (ścieżki URL):

- `/` – wyświetla plik HTML
- `/style.css` – wyświetla arkusz CSS
- `/script.js` – wyświetla skrypt JavaScript
- `/favicon.ico` – opcjonalnie obsługuje ikonę strony

Dla każdej ścieżki powinieneś stworzyć odpowiednią instrukcję warunkową.

Ogólny schemat instrukcji warunkowej dla obsługi pliku:

```
if (req.url === 'adres URL żądanego pliku') {
    // krok 1: Utwórz ścieżkę do pliku z wykorzystaniem modułu 'path'
    // krok 2: Odczytaj zawartość pliku za pomocą funkcji 'fs.readFile()'
```

```
    // Jeśli odczyt jest poprawny (brak błędu):
    // - Ustaw status odpowiedzi HTTP na 200
    // - Ustaw odpowiedni nagłówek 'Content-Type':
    //   * dla HTML: 'text/html; charset=utf-8'
    //   * dla CSS: 'text/css; charset=utf-8'
    //   * dla JavaScript: 'text/javascript; charset=utf-8'
    //   * dla favicon: 'image/x-icon'
    // - Wyślij zawartość pliku do klienta (`res.end()`)
    // Jeśli wystąpił błąd odczytu (plik nie istnieje):
    // - Ustaw status HTTP na 404
    // - Wyślij komunikat błędu do klienta (np. „Plik nie istnieje")
}
```

Krok 4: Tworzenie pliku HTML (`index.html`)

- Utwórz plik `index.html`.
- Dodaj podstawową strukturę HTML.
- Dodaj odwołanie do arkusza CSS (`style.css`) w `<head>`.
- W treści umieść kontener (`div`) o id `container`, a wewnątrz niego kontener (`div`) o id `page`. Umieść tam przykładowy tekst.
- Na końcu strony (przed zamknięciem `</body>`) dodaj odwołanie do skryptu JavaScript (`script.js`).

Krok 5: Tworzenie arkusza CSS (`style.css`)

- Utwórz plik `style.css`.
- Utwórz style dla dwóch kontenerów:
  - `#container`: szerokość: `800px`, wyśrodkowanie na stronie.
  - `#page`: czcionka `150%`, wyśrodkowanie tekstu, odstęp `20px`, ramka `1px` czarna, tło żółte.

Krok 6: Tworzenie skryptu JS (`script.js`)

- Utwórz plik `script.js`.
- Po załadowaniu strony wyświetl alert informujący, że skrypt załadowano poprawnie.

Krok 7: Uruchomienie i testowanie serwera

- W terminalu uruchom serwer poleceniem:

  ```
  node server.js
  ```

- Otwórz przeglądarkę i wpisz adres:

`http://127.0.0.1:5555`

- Sprawdź czy wyświetliła się strona HTML, poprawnie wczytał się CSS oraz czy działa skrypt JavaScript.

Dodatkowe zadanie – Obsługa błędu 404

Twoje zadanie polega na obsłudze nieprawidłowych (nieistniejących) ścieżek.

Kroki realizacji zadania:

1. Stwórz dodatkowy plik HTML o nazwie `404.html`:
   - Przygotuj prostą strukturę HTML.
   - W treści dodaj komunikat informujący użytkownika o błędzie, np. „Strona, której szukasz, nie istnieje (błąd 404).”
2. W pliku `server.js` dodaj **ogólną obsługę wszystkich innych, nieznanych ścieżek**:
   - Jeśli użytkownik wpisze adres URL, który nie pasuje do żadnej ze wcześniej obsługiwanych ścieżek, serwer powinien automatycznie:
     – wczytać zawartość pliku `404.html`,
     – wysłać ją do użytkownika z kodem błędu HTTP: `404` oraz odpowiednim typem MIME:
     `'Content-Type': 'text/html; charset=utf-8'`.

Ogólny schemat obsługi błędu 404:

```
// Jeśli żaden wcześniejszy warunek (if) nie został spełniony,
// obsłuż ogólnie wszystkie pozostałe żądania (else):

// 1. Utwórz ścieżkę do pliku 404.html (moduł 'path')

// 2. Odczytaj zawartość pliku 404.html ('fs.readFile()')

//   - Jeśli odczyt jest poprawny (plik istnieje):
//     * Ustaw status HTTP na 404
//     * Ustaw Content-Type na 'text/html; charset=utf-8'
//     * Wyślij zawartość pliku 404.html

//   - Jeśli wystąpił błąd odczytu (plik 404.html nie istnieje):
//     * Wyślij podstawowy komunikat błędu bezpośrednio z kodu serwera
```

Wskazówki pomocnicze:

- Upewnij się, że poprawnie wywołujesz metodę `res.end()`.
- Jeśli masz wątpliwości, obserwuj konsolę (`console.log()`) podczas działania serwera.

Oczekiwany efekt końcowy:

Po wykonaniu wszystkich kroków powinieneś mieć działający serwer HTTP, który:

- Poprawnie wyświetla plik HTML na ścieżce `/`.
- Ładuje pliki CSS oraz JavaScript.
- W przypadku nieznanych adresów wyświetla użytkownikowi stronę `404.html` z odpowiednim komunikatem i kodem błędu HTTP: 404.
