LABORATORIUM PZWJS

Data wykonania ćwiczenia:	19/10/2023
Rok studiów:	3
Semestr:	*5*
Grupa studencka:	2a
Grupa laboratoryjna:	3

Laboratorium nr 2

Osoby wykonujące ćwiczenia:

1. Jakub Światłoch
Katedra Informatyki i Automatyki

Zadania Liczniki:
1.Za pomocą liczników wykonaj zdarzenie wyświetlania czasu, który upłynął od startu aplikacji, co 1 sekundę,

```
function displayElapsedTime() {
  const elapsedSeconds = Math.floor((Date.now() - startTime) / 1000);
  console.log(`Czas od startu aplikacji: ${elapsedSeconds} sekundy`);
}
```

```
const startTime = Date.now();
setInterval(displayElapsedTime, 1000);
```

Efekt:

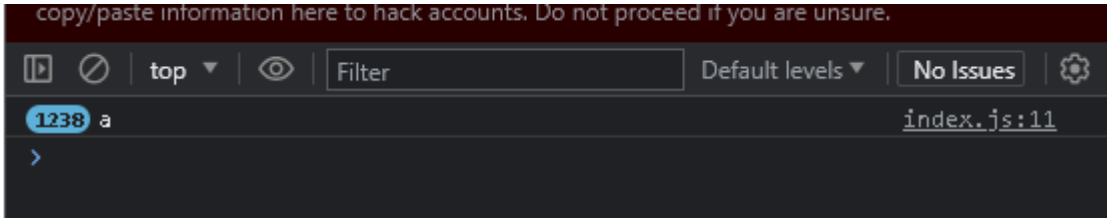
```
Czas od startu aplikacji: 314 sekundy                    index.js:4
Czas od startu aplikacji: 315 sekundy                    index.js:4
Czas od startu aplikacji: 316 sekundy                    index.js:4
Czas od startu aplikacji: 317 sekundy                    index.js:4
Czas od startu aplikacji: 318 sekundy                    index.js:4
Czas od startu aplikacji: 319 sekundy                    index.js:4
Czas od startu aplikacji: 320 sekundy                    index.js:4
Czas od startu aplikacji: 321 sekundy                    index.js:4
Czas od startu aplikacji: 322 sekundy                    index.js:4
Czas od startu aplikacji: 323 sekundy                    index.js:4
Czas od startu aplikacji: 324 sekundy                    index.js:4
Czas od startu aplikacji: 325 sekundy                    index.js:4
Czas od startu aplikacji: 326 sekundy                    index.js:4
Czas od startu aplikacji: 327 sekundy                    index.js:4
Czas od startu aplikacji: 328 sekundy                    index.js:4
Czas od startu aplikacji: 329 sekundy                    index.js:4
Czas od startu aplikacji: 330 sekundy                    index.js:4
Czas od startu aplikacji: 331 sekundy                    index.js:4
```

2. Co jedną sekundę wyświetl pięć razy literę ‘a’ w odstępach 10 ms,

```
function displayLetterA() {
  for (let i = 0; i < 5; i++) {
    setTimeout(() => {
      console.log("a");
    }, i * 10);
  }
}
```

```
setInterval(displayLetterA, 10);
```

Efekt:



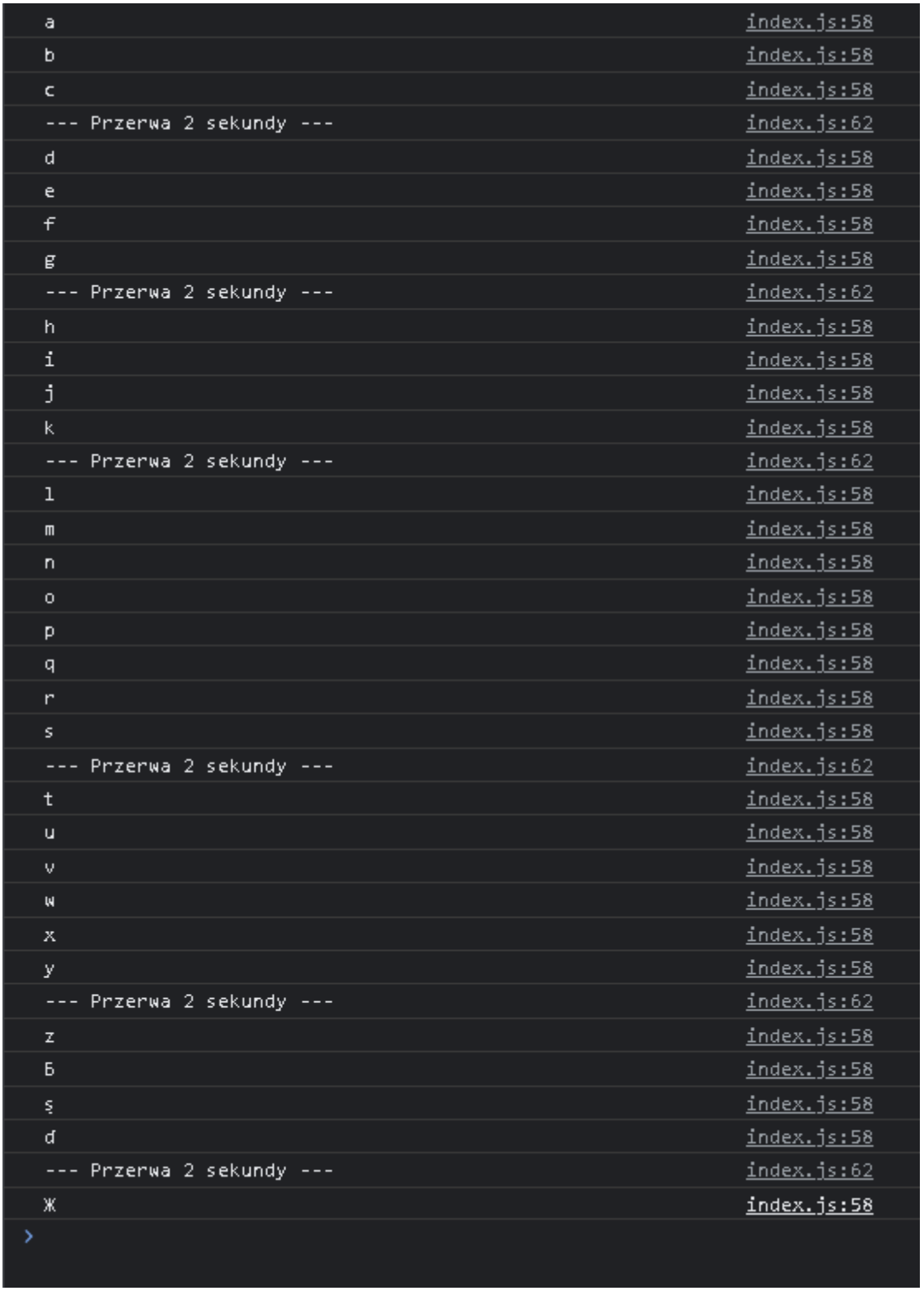
3.Dla 30 elementowej tablicy (z alfabetem) wyświetlić po kolei co 0,5s litery przy czym jeżeli jest to samogłoska dodać przerwę 2 sekundową.

```
function displayAlphabetWithDelay()
  const alphabet = [
    "a",
    "b",
    "c",
    "d",
    "e",
    "f",
    "g",
    "h",
    "i",
    "j",
    "k",
    "l",
    "m",
    "n",
    "o",
    "p",
    "q",
    "r",
    "s",
    "t",
    "u",
    "v",
    "w",
    "x",
    "y",
    "z",
    "Б",
    "ś",
    "ď",
    "Ж",
  ];
```

```
  let index = 0;

  const intervalId = setInterval(() => {
    if (index < alphabet.length) {
      const letter = alphabet[index];
      console.log(letter);

      if (isVowel(letter)) {
        setTimeout(() => {
          console.log("--- Przerwa 2 sekundy ---");
        }, 2000);
      }

      index++;
    } else {
      clearInterval(intervalId);
    }
  }, 500);
}
```

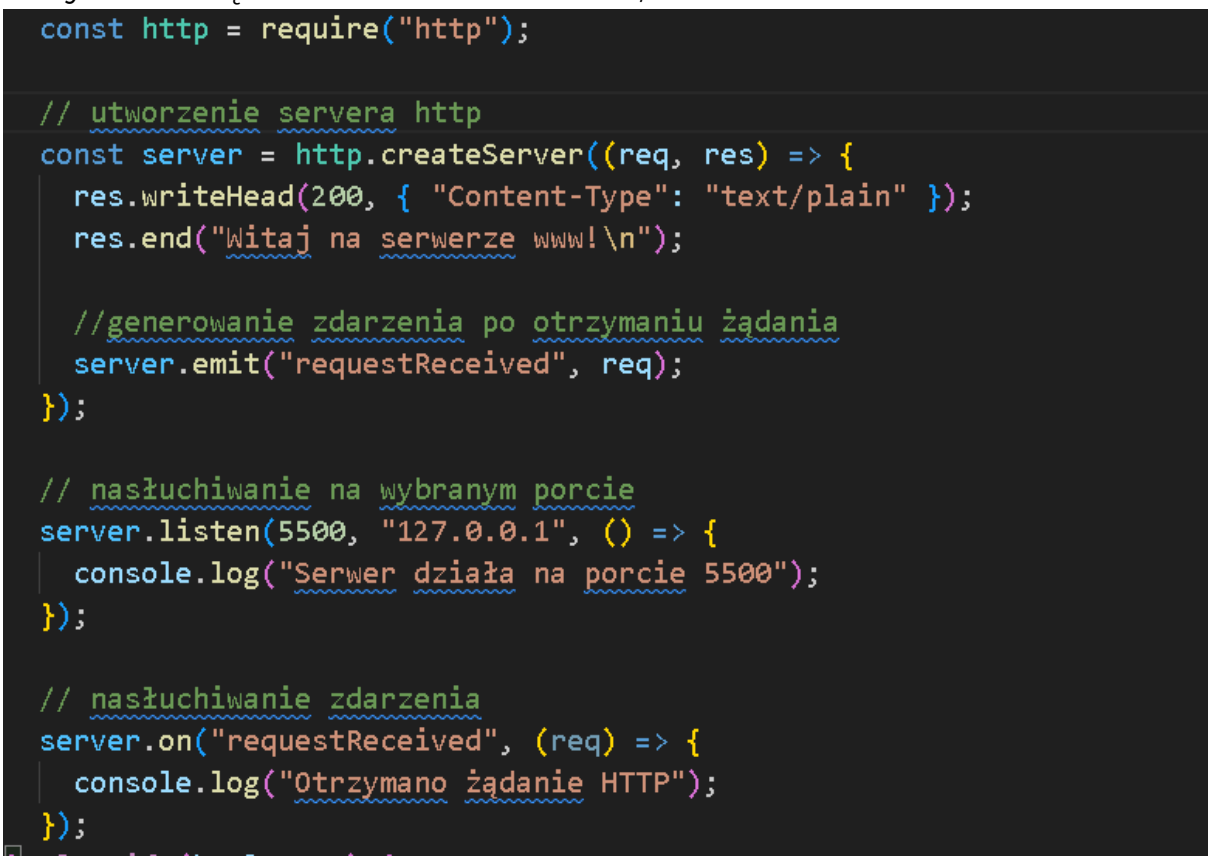
```
  //sprwadz czy to samogloska
  function isVowel(letter) {
    const vowels = ["a", "e", "i", "o", "u", "y"];
    return vowels.includes(letter);
  }
```

Efekt:


a index.js:58
b index.js:58
c index.js:58
--- Przerwa 2 sekundy --- index.js:62
d index.js:58
e index.js:58
f index.js:58
g index.js:58
--- Przerwa 2 sekundy --- index.js:62
h index.js:58
i index.js:58
j index.js:58
k index.js:58
--- Przerwa 2 sekundy --- index.js:62
l index.js:58
m index.js:58
n index.js:58
o index.js:58
p index.js:58
q index.js:58
r index.js:58
s index.js:58
--- Przerwa 2 sekundy --- index.js:62
t index.js:58
u index.js:58
v index.js:58
w index.js:58
x index.js:58
y index.js:58
--- Przerwa 2 sekundy --- index.js:62
z index.js:58
Б index.js:58
ş index.js:58
d index.js:58
--- Przerwa 2 sekundy --- index.js:62
Ж index.js:58

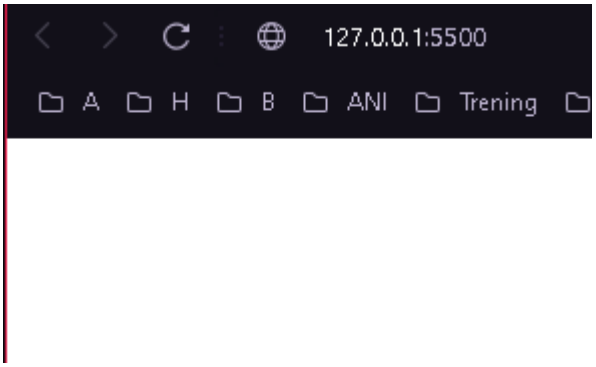
3.Zadania Zdarzenia

1.Napisać program, który generuje zdarzenie w przypadku otrzymania żądania do serwera www,

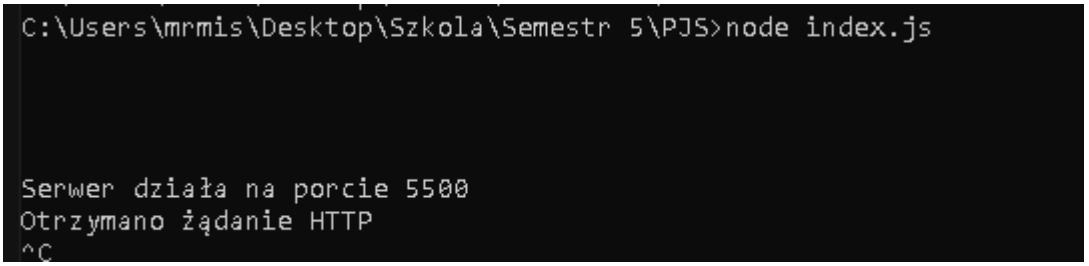


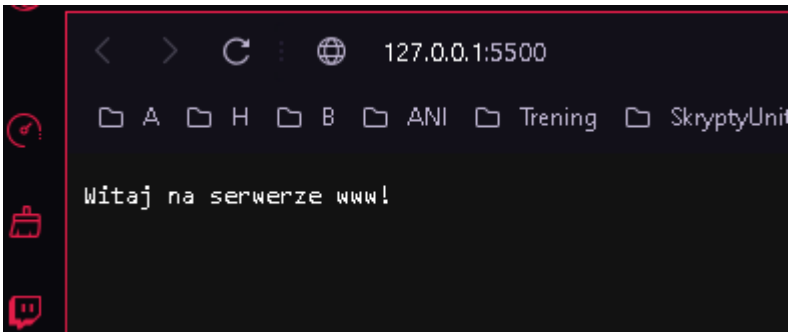
```
const http = require("http");

// utworzenie servera http
const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "text/plain" });
  res.end("Witaj na serwerze www!\n");

  //generowanie zdarzenia po otrzymaniu żądania
  server.emit("requestReceived", req);
});

// nasłuchiwanie na wybranym porcie
server.listen(5500, "127.0.0.1", () => {
  console.log("Serwer działa na porcie 5500");
});

// nasłuchiwanie zdarzenia
server.on("requestReceived", (req) => {
  console.log("Otrzymano żądanie HTTP");
});
```

Efekt:
przed odświeżeniem strony



po odświeżeniu strony





2.Stworzyć własny obiekt użytkownik i zaimplementować w nim zdarzenie przy zmianie imienia, które wyświetla informacje o osobie

```js
const { EventEmitter } = require("events");
//klasa uzytkownika ktora dziedziczy z EventEmitter
//ma metodę changeName, która zmienia imię użytkownika i emituje zdarzenie "nameChanged" z nowym imieniem
class User extends EventEmitter {
  constructor(name) {
    super();
    this.name = name;
  }

  changeName(newName) {
    this.name = newName;
    this.emit("nameChanged", this.name);
  }
}

// utworzenie uzytkownika
const user = new User("John Doe");

// nasłuchiwanie na zdarzenie nameChanged
user.on("nameChanged", (newName) => {
  console.log(`Imię użytkownika zostało zmienione na: ${newName}`);
});

// Zmiana imienia użytkownika
user.changeName("Jane Smith");
```

Efekt:

```
C:\Users\mrmis\Desktop\Szkola\Semestr 5\PJS>node index.js

Imię użytkownika zostało zmienione na: Jane Smith
```

4.Zadania domknęcie
1.Zaprojektować listę oraz wyświetlić jej zawartość asynchronicznie (poprzez funkcję nextTick)

```
class Lista {
  constructor() {
    this.elements = [];
  }

  dodajElement(element) {
    this.elements.push(element);
  }

  wyswietlZawartosc() {
    process.nextTick(() => {
      this.elements.forEach((element, index) => {
        console.log(`Element ${index + 1}: ${element}`);
      });
    });
  }
}

const lista = new Lista();
lista.dodajElement("Pierwszy element");
lista.dodajElement("Drugi element");
lista.dodajElement("Trzeci element");

// wywołanie asynchroniczne za pomocą nextTick
lista.wyswietlZawartosc();

console.log("Proszę czekać na wyświetlenie zawartości listy...");
```

process.nextTick pozwala na odblokowanie głównego wątku i wykonanie kodu asynchronicznie

Efekt:

```
C:\Users\mmmis\Desktop\Szkoła\Semestr 5\PJS>node index.js


Proszę czekać na wyświetlenie zawartości listy...


Element 1: Pierwszy element
Element 2: Drugi element
Element 3: Trzeci element
```

5. Zadania JSON
• Stworzyć bazę danych (3 osoby) w zmiennej opartej o strukturę JSON. (zmienna tekstowy)
• Skonwertować tekstowy na obiekt (JSON.parse),
• Wyświetlić osoby w pętli,
• Dodać osobę,
• Zamienić na ciąg znaków i wyświetlić.

```
// baza danych w formie JSON
const jsonDatabaseText = `
    {
    "people": [
        { "id": 1, "name": "Adam niezgoda", "age": 21 },
        { "id": 2, "name": "Mieszko Walus", "age": 20 },
        { "id": 3, "name": "Andrzej Lajsota", "age": 19 },
        { "id": 3, "name": "krystyna Ruchlionski", "age": 13 }
    ]
    }
    `;

// konwersja tekstu na obiekt
const database = JSON.parse(jsonDatabaseText);

// wyswietlenie osob w petli
console.log("Persons in database:");
database.people.forEach((person) => {
  console.log(`id: ${person.id}, name: ${person.name}, age: ${person.age}`);
});
```

```
  // dodawanie osoby
  const newPerson = { id: 4, name: "Alicja kombajn", age: 78 };
  database.people.push(newPerson);

  // zamiana na ciąg znakow
  const updatedJsonDatabaseText = JSON.stringify(database, null, 2);

  // wyswietlenie w formie tekstu
  console.log("Zaktualizowana baza danych w formie tekstu:");
  console.log(updatedJsonDatabaseText);

  // zapisanie zaktualizowej bazy do pliku
  fs.writeFileSync("database.json", updatedJsonDatabaseText, "utf8");
```

Efekt:

```
Persons in database:
id: 1, name: Adam niezgoda, age: 21
id: 2, name: Mieszko Walus, age: 20
id: 3, name: Andrzej Lajsota, age: 19
id: 3, name: krystyna Ruchlionski, age: 13
Zaktualizowana baza danych w formie tekstu:
{
  "people": [
    {
      "id": 1,
      "name": "Adam niezgoda",
      "age": 21
    },
    {
      "id": 2,
      "name": "Mieszko Walus",
      "age": 20
    },
    {
      "id": 3,
      "name": "Andrzej Lajsota",
      "age": 19
    },
    {
      "id": 3,
      "name": "krystyna Ruchlionski",
      "age": 13
    },
    {
      "id": 4,
      "name": "Alicja kombajn",
      "age": 78
    }
  ]
}
```

```
{
  "people": [
    {
      "id": 1,
      "name": "Adam niezgoda",
      "age": 21
    },
    {
      "id": 2,
      "name": "Mieszko Walus",
      "age": 20
    },
    {
      "id": 3,
      "name": "Andrzej Lajsota",
      "age": 19
    },
    {
      "id": 3,
      "name": "krystyna Ruchlionski",
      "age": 13
    },
    {
      "id": 4,
      "name": "Alicja kombajn",
      "age": 78
    }
  ]
}
```