

Wybrane elementy praktyki projektowania oprogramowania

Zestaw 5

Javascript - programowanie asynchroniczne

2025-11-04

Liczba punktów do zdobycia: **10/45**

Zestaw ważny do: 2025-11-25

1. (1p) Zademonstrować eksportowanie i importowanie modułów w standardzie CommonJS (za pomocą `require`). Zademonstrować eksportowanie i importowanie modułów w standardzie ECMAScript modules (za pomocą `import` i `export`). Czy możliwa jest sytuacja w której dwa moduły tworzą cykl (odwołują się do siebie nawzajem)? W którym podsystemie modułów jest to łatwiejsze?
2. (1p) Napisać program, który wypisze na ekranie zapytanie o imię użytkownika, odczyta z konsoli wprowadzony tekst, a następnie wypisze `Witaj ***` gdzie puste miejsce zostanie wypełnione wprowadzonym przez użytkownika napisem. Użyć dowolnej techniki do spełnienia tego wymagania, ale nie korzystać z zewnętrznych modułów z `npm` a wyłącznie z obiektów z biblioteki standardowej (wszystkie te techniki sprowadzają się do jakiejś formy dojścia do strumienia `process.stdin`).

Wskazówka: <https://stackoverflow.com/q/8128578/941240>, jak zwykle w takim przypadku kiedy korzysta się ze SO, proszę przejrzeć odpowiedzi, nie tylko tę najwyżej punktowaną.

3. (2p) Skoro już wiadomo jak uzyskać efekt promtu na konsoli, napisać program, który wylosuje liczbę od 0 do 100 i zaproponuje użytkownikowi "zgadywanie". Program prosi użytkownika o podanie wartości. Na podaną przez użytkownika wartość program odpowiada na jeden z trzech sposobów:

- jeśli podana przez użytkownika liczba jest równa wylosowanej - "to jest właśnie ta liczba" i zabawa się kończy
- jeśli podana przez użytkownika liczba jest mniejsza od wylosowanej - "moja liczba jest większa"
- jeśli podana przez użytkownika liczba jest większa od wylosowanej - "moja liczba jest mniejsza"

Gra toczy się w pętli aż do zwycięstwa użytkownika, czyli do podania prawidłowej wartości.

4. (1p) Napisać program używający modułu (`fs`), który przeczyta w całości plik tekstowy a następnie wypisze jego zawartość na konsoli (użyć funkcji zwrotnej lub promise, wedle uznania).

5. **(1p)** Napisać program używający modułu (`http/https`), który odczyta zawartość jakiegoś zasobu sieciowego (np. strony) i wypisze ją na konsoli. Pokazać jak dostępny w bibliotece standardowej podstawowy interfejs programowania (metoda `get` i zdarzenia `data/end`) opakować w wygodniejszy interfejs programowania zwracający obiekt `Promise`.
6. **(2p)** Pokazać w jaki sposób odczytywało duże pliki linia po linii za pomocą modułu `readline`. Działanie zademonstrować na przykładowym kodzie analizującym duży plik logów hipotetycznego serwera WWW, w którym każda linia ma postać

```
08:55:36 192.168.0.1 GET /TheApplication/WebResource.axd 200
```

gdzie poszczególne wartości oznaczają czas, adres klienta, rodzaj żądania HTTP, nazwę zasobu oraz status odpowiedzi.

W przykładowej aplikacji wydobyć listę adresów IP trzech klientów, którzy skierowali do serwera aplikacji największą liczbę żądań. Przykładowe dane dla aplikacji proszę sobie wygenerować we własnym zakresie (niekoniecznie ściągać logi z rzeczywistego serwera!)

Wynikiem działania programu powinien być przykładowy raport postaci:

```
12.34.56.78 143  
23.45.67.89 113  
123.245.167.289 89
```

7. **(2p)** Wybrać jeden z modułów i funkcję asynchroniczną do odczytu danych (`fs::readFile`) i pokazać klasyczny interfejs programowania asynchronicznego, w którym asynchroniczny wynik wywołania funkcji jest dostarczany jako argument do funkcji zwrotnej (callback).

Następnie pokazać jak taki klasyczny interfejs można zmienić na `Promise` na trzy sposoby:

- za pomocą "ręcznie" napisanej funkcji przyjmującej te same argumenty co `fs::readFile` ale zwracającej `Promise`
- za pomocą `util.promisify` z biblioteki standardowej
- za pomocą `fs.promises` z biblioteki standardowej

Na zademonstrowanym przykładzie pokazać dwa sposoby obsługi funkcji zwracającej `Promise`

- "po staremu" - wywołanie z kontynuacją (`Promise::then`)
- "po nowemu" - wywołanie przez `async/await`

Wiktor Zychla