Wybrane elementy praktyki projektowania oprogramowania
Zestaw 4

Javascript - dziedziczenie prototypowe, podstawy biblioteki standardowej

08-11-2016

Liczba punktów do zdobycia: **10/40**
Zestaw ważny do: 22-11-2016

1. (**1p**) Pokazać jak zdefiniować funkcję konstruktorową tworzącą węzeł drzewa binarnego. Pokazać przykład użycia tej funkcji do utworzenia instancji drzewa o głębokości 3 (nie musi być to drzewo pełne). Każdy węzeł powinien przechowywać referencje do swojego lewego i prawego syna oraz wartość.

2. (**1p**) Do prototypu funkcji drzewa binarnego z poprzedniego przykładu dodać iterator (zaimplementowany jako generator przy użyciu `yield`), który pozwoli napisać

```
var root = new Tree( ..... );

// enumeracja wartości z węzłów
for ( var e of root ){
    console.log( e );
}
```

3. (**1p**) Na dowolnym przykładzie zademonstrować jak za pomocą techniki IIFE (*immediately invoked function expression*) można uzyskać efekt "składowych prywatnych" znany z wielu języków obiektowych.

4. (**1p**) Zademonstrować w praktyce tworzenie własnych modułów oraz ich włączanie do kodu za pomocą `require`. Czy możliwa jest sytuacja w której dwa moduły tworzą cykl (odwołują się do siebie nawzajem)? Jeśli nie - wytłumaczyć dlaczego, jeśli tak - pokazać przykład implementacji.

5. (**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. Użyj dowolnej techniki do spełnienia tego wymagania.

6. (**1p**) Napisać program używający modułu (`fs`), który przeczyta w całości plik tekstowy a następnie wypisze jego zawartość na konsoli.

7. (**2p**) Pokazać w jaki sposób odczytywać duże pliki linia po linii. 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.

Wynikiem działania programu powinien być przykładowy raport postaci:

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

8. (**2p**) Wybrać jeden z modułów i funkcję asynchroniczną do odczytu danych: `fs::readFile` - odczyt zawartości pliku lokalnego lub `http::get` - odczyt zawartości zasobu sieciowego i pokazać jak klasyczny interfejs programowania asynchronicznego, w którym asynchroniczny wynik wywołania funkcji jest dostarczany jako argument do funkcji zwrotnej.

   Następnie pokazać jak taki klasyczny interfejs można "unowocześnić" za pomocą obiektów `Promise`.

   Na zademonstrowanym przykładzie pokazać czym różni się wywołanie klasyczne wywołanie z funkcją zwrotną (callback) od wywołania z kontynuacją (`Promise::then`) a następnie zademonstrować na przykładzie i wyjaśnić działanie funkcji `Promise::all` i `Promise::race`.

Wiktor Zychla