PIISW, W08, IO, 2017/2018, semestr letni
Lista zadań nr 2: HTML, CSS i Javascript

mgr inż. Maciej Małecki
maciej.malecki@pwr.edu.pl

19 lutego 2018

Wprowadzenie

- Rozwiązania zadań z tej listy muszą znaleźć się w prywatnym repozytorium na `github.com` (może to być to samo repozytorium, które założono w ramach prac nad listą nr 1).
- Prowadzący zajęcia musi mieć uprawnienia do odczytu i zapisu do tego repozytorium.
- Rozwiązanie każdego zadania musi się znaleźć w podkatalogu `zadanie_x`, gdzie `x` jest numerem zadania.
- Do wykonania zadań z Javascript można użyć interpretera Nashorn (uruchamianego komendą `jjs`) dostarczanego wraz z środowiskiem JDK (tworzymy pliki źródłowe `*.js`). Należy zwrócić uwagę na poprawne deklarowanie zmiennych i stałych z użyciem słów kluczowych `let` i `const`. W interpreterze Nashorn należy w tym celu włączyć tryb `ECMAScript 6` (`--language=es6`).
- Wszystkie argumenty wywołania komendy `jjs`, które wystąpią po ciągu znaków `--` są traktowane jako argumenty uruchamianego skryptu i są dostępne w tablicy `arguments`.
- Starsze wersje Nashorn pracują nieprawidłowo z notacją *fat arrow*. W takim przypadku deklaruj funkcje z użyciem słowa kluczowego `function`.
- Do wykonania zadań z zakresu HTML/CSS należy użyć dowolnej przeglądarki internetowej (tworzymy pliki źródłowe `*.html` oraz `*.css`).

Oceny

Punkty:	< 7	$7 - 8$	$9 - 10$	$11 - 12$	$13 - 14$	15
Ocena:	$2,0$	$3,0$	$3,5$	$4,0$	$4,5$	$5,0$

Zadania

1. (3 pkt) Dane są dokument HTML:

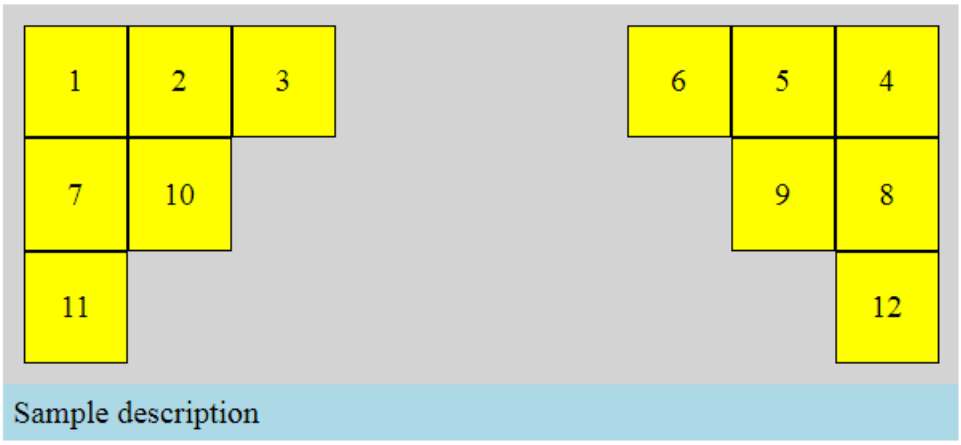
```
<!doctype html>
<html>
<head>
```

```
  <link rel="stylesheet" href="zad1.css" type="text/css"/>
</head>
<body>
<ul>
  <li>1</li><li>2</li><li>3</li><li>4</li><li>5</li><li>6</li>
  <li>7</li><li>8</li><li>9</li><li>10</li><li>11</li><li>12</li>
</ul>
<p>Sample description</p>
</body>
</html>
```

oraz arkusz styli CSS:

```
ul:after {
  display:block;
  height:0;
  content:'';
  clear:both;
}
```

Modyfikując jedynie arkusz styli CSS spraw, aby dokument HTML został wyrenderowany przez przeglądarkę w sposób pokazany na rysunku 1.



Rysunek 1: Prawidłowo wystylowany przykład dla zadania 1

.

Wskazówki: ustaw właściwą wartość dla `list-style` oraz użyj atrybutów `float` i `clear` dla elementu `<li>`. Zapoznaj się z selektorem `nth-child(x)`.

2. (3 pkt) Napisz program w języku Javascript, który akceptuje dowolną ilość liczb całkowitych jako argumentów wejściowych oraz wyświetla numeryczny wynik na wyjściu będący sumą: argumentów powiększonych o 1, jeśli argumenty są liczbami nieparzystymi oraz argumentów pomniejszonych o 1, jeśli argumenty są liczbami parzystymi.

   Program należy napisać „klasycznie”, z wykorzystaniem pętli `for` oraz instrukcji warunkowej.

   Przykładowe wyniki działania programu:

- $\emptyset \rightarrow$ `0`
- `0` $\rightarrow$ `-1`
- `1` $\rightarrow$ `2`
- `1 2 3` $\rightarrow$ `7`

3. (5 pkt) Przepisz program z zadania 2 z wykorzystaniem technik programowania funkcyjnego tak, aby wyeliminować konieczność użycia instrukcji: pętli, warunkowych oraz deklaracji zmiennych.

   Wskazówka: użyj funkcji `filter`, `map`, `reduce`.

4. (4 pkt) Rozwiń następujący przykład z wykładu:

```
var account = {
  balance: 0,
  debit: function(amount) {
    if (amount < 0) { throw new Error('Illegal amount'); }
    if (this.balance < amount) {
        throw new Error('Insufficient account balance');
    }
    this.balance -= amount;
    return this.balance;
  },
  credit: function(amount) {
    if (amount < 0) {
        throw new Error('Illegal amount');
    }
    this.balance += amount;
    return this.balance;
  }
};
```

   tak, aby

   (a) umożliwić tworzenie wielu obiektów o identycznej strukturze jak obiekt `account`,

   (b) rozszerzyć listę pól obiektu o następujące pola: numer konta, data utworzenia, waluta,

   (c) uniemożliwić bezpośrednią manipulację wartością atrybutów `balance`, nr konta, data utworzenia oraz waluta,

   (d) umożliwić inicjalizację wszystkich pól poza `balance` dowolnie wybranymi wartościami w momencie tworzenia obiektu (później wartości tych atrybutów powinny być niezmienne),

   (e) umożliwić odczyt wartości wszystkich atrybutów przy pomocy pojedynczego wywołania metody oraz obiektu opisującego.

   Wskazówka: użyj metody fabrykującej.