Języki programowania i GUI

Lista 1 - 2022

1. Napisz funkcję, `bigger(a)`, która zwraca funkcję strzałkową sprawdzającą czy jej argument `x` jest większy od `a`. Przetestuj ją za pomocą metody `.filter()`:
   `console.log([2,31,5,3,6].filter(bigger(3))); // 31, 5, 6`
   oraz metod `.find()`, `.findIndex()`, `.every()`, `.some()`, dla różnych wartości argumentu `a`. Następnie napisz i przetestuj funkcje `smaller(a)` oraz `between(a,b)` zwracające funkcje sprawdzające, czy $x < a$, oraz czy $a \leq x \leq b$.

2. Napisz funkcję `f(a,b)` taką by: `[1,2,3,271,12313,123,21313,541,42].sort(f);` dawało:
   (a) tablicę posortowaną malejąco.
   Odp. `function f(a,b){return b-a}` lub funkcjca strzałkowa `(a,b)=>b-a;`
   (b) posortowaną względem cyfry jedności.
   (c) posortowaną względem cyfry dziesiątek.
   (d) posortowaną względem sumy cyfr.

   Przetestuj każdą ze swych funkcji dla kilku różnych tablic.

3. (a) Napisz funkcję, która zwraca sumę wszystkich swoich argumentów, które są liczbami, natomiast pozostałe argumenty ignoruje.
   Wskazówka: użyj destrukturyzacji `function suma(...args)` lub magicznej zmiennej `arguments`

   ```
   suma(1,2,3,10,[],20,30,"marek",{a:4}); //   66
   ```

   (b) Zmodyfikuj swoją funkcję tak, aby potrafiła też uwzględniać sumę liczb argumentach w typu `Array`.

   ```
   suma(1,2,3,[4,5,"aa"],10,"asda"); //   25
   ```

   (c) Zmodyfikuj swoją funkcję, tak aby potrafiła też uwzględniać sumę liczb za wartych w tablicach będących elementami innych tablic..... Ma to działać dla dowolnie głęboko zagnieżdżonej tablicy. Wskazówka: można użyć rekurencji.

   ```
   suma(1,2,3,[4,5,[5,5]],10); //   35
   ```

4. Właściwości (properties). Wzorując się na klasie:

   ```
   class Kwadrat {
       constructor(a) { this.a = a; }
       get bok() { return this.a; }
       set bok(a) { this.a = a; }
       get obwód() { return 4 * this.a; }
       set obwód(len) { this.a = len / 4; }
       get pole() { return this.a * this.a; }
       set pole(P) { this.a = Math.sqrt(P); }
       toString() {return `a=${this.bok}\nL=${this.obwód}\nP=${this.pole}\n`;}
   }

   let k = new Kwadrat(1);
   k.bok = 12;console.log(k+"");
   k.obwód = 12;console.log(k+"");
   k.pole=12;console.log(k+"");
   ```

Napisz klasę `Koło` z właściwościami `promień`, `średnica`, `obwód`, `pole`. Wewnętrznie jedyną prawdziwą daną ma być promień `r` a pozostałe powinny być obliczane na jego podstawie. Powinny być jednak możliwe przypisania:

```
var k=new Koło(3);   // Koło o promieniu 3
k.promień=8;         // zmienia promień na 8
k.średnica=10;       // zmienia promień ma 5
k.pole=4;            // zmienia promień na sqrt(4/Pi)
k.obwód=7;           // zmienia promień na 3.5/Pi
```

Napisz też funkcję demonstrującą, że wszystkie sety i gety działają poprawnie. Funkcja, powinna pokazywać stan wszystkich setów po każdym gecie z treści zadania.

Następnie utwórz tablicę zawierającą koła i kwadraty i oblicz w pętli ich łączne pole i obwód.

5. Korzystając z informacji zawartych na stronie:
`https://developer.mozilla.org/pl/docs/Web/JavaScript/Guide/Iterators_and_Generators`
Napisz funkcje gwiazdkowe (`function*`) zwracające generatory:

   (a) `function* dzielniki(n)` - zwraca generator dzielników liczby n,
   (b) `function* pierwsze(n)` - zwraca generator liczb pierwszych nie większych od n.
   (c) `function* rozkład(n)` - zwraca rozkład liczby n na czynniki pierwsze.

   Wypróbuj działanie każdego z nich za pomocą funkcji `Array.from` oraz funkcji:

```
function wypisz(gen) {
    for (let x of gen)
        console.log(x);
}

function wypisz2(gen) {
    while (!(x = gen.next()).done)
        console.log(x.value);
}
```

   napisanych w analogiczny sposób funkcji:
   `sklej(gen,glue=','), suma(gen), iloczyn(gen)`, których argumentem jest dowolny skończony generator liczb, a wynikiem odpowiednio: sklejenie za pomocą przecinków, suma oraz iloczyn generowanych liczb.