Kodavstämning – JSU 2021

Instruktioner

Nedan har du 3st problem, läs igenom beskrivningen av varje problem noga och skapa en bild av hur du ska angripa dem. Du behöver inte ha klarat alla problem men du bör ha hunnit börja på alla. Du väljer själv i vilken ordning du löser dem.

Inlämning

Inlämning sker via formuläret som skickats till din @iths-mail. Deadline för inlämning är kl12.30.
Du lämnar in endast en javascript-fil med dina funktioner. Städa bort statements i ditt global scope innan du lämnar in.

Problem

Varje problem består av en funktion du skall implementera. Det är helt OK att skapa flera funktioner om du behöver. Varje problem har också lite testdata du kan testköra din funktion med.

Problem

Password validation – Without RegExp

Skapa en funktion `validatePassword` som validerar ett lösenord. Funktionen ska ta emot ett lösenord som parameter och returnera en Boolean.

Valideringen ska se till att lösenordet följer dessa regler

- Minst vara 8 tecken långt och högst vara 50 tecken långt
- Innehålla minst en liten bokstav
- Innehålla minst en stor bokstav

- Innehålla minst en siffra
- Innehålla minst ett specialtecken Special Character List

Det är inte tillåtet att använda reguljära uttryck i denna uppgift.

Testdata

```
validatePassword("Abcd123!") // => true
validatePassword("omg") // => false
validatePassword("123") // => false
validatePassword("........") // => false
validatePassword("B3ng70l550n?") // => true
validatePassword("_F00b4R_") // => true
```

Order sum

Skapa en funktion `orderTotal` som ger priset för en order. En order är modellerad på följande sätt.
Varje föremål i ordern är ett objekt med följande struktur.

```
{
  name: PRODUCT_NAME,
  amount: AMOUNT_IN_ORDER,
  price: COST_PER_PIECE
}
```

En order skulle därför kunna se ut såhär

```
[
  {name:'Gurka', amount: 2, price: 10},
  {name:'Vattenmelon', amount: 1, price: 30},
  {name:'Kokosnöt', amount: 2, price: 25},
]
```

Två gurkor, en vattenmelon och två kokosnötter blir därför 100.

Testdata

```
orderTotal([
  {name: 'Kaffe', amount: 1, price: 25},
  {name: 'Muffin', amount: 1, price: 30}
]) // => 55

orderTotal([
  {name: 'Porche', amount: 3, price: 130000}
]) // => 390000

orderTotal([
  {name:'Gurka', amount: 2, price: 10},
  {name:'Vattenmelon', amount: 1, price: 30},
  {name:'Kokosnöt', amount: 2, price: 25},
]) // => 100
```

E-mail Data Extractor

Skapa en funktion `extractEmailData` som läser av delarna av en e-mail-adress och lägger i ett objekt.
En e-mail består av två delar, en local och en domain.
`local@domain`.

Vissa e-mail tjänster tillåter att man kan lägga till "tags" i slutet på sin `local`. Dessa tags läggs till i slutet på `local` med ett `+`-tecken framför. Ex `kalle.svensson+mytag@example.com`, den kan också innehålla flera tags `jonas.sjoberg+spam+trash@example.com`.

Testdata

```
extractEmailData("kalle.svensson+spam+trash@example.com")
  // => {
  //   local: "kalle.svensson",
  //   domain: "example.com",
  //   tags: ["spam", "trash"]
```

```
  // }

extractEmailData("barack.obama@murica.gov")
  // => {
  //   local: "barack.obama",
  //   domain: "murica.gov",
  //   tags: []
  // }

extractEmailData("test+test@test.dev")
  // => {
  //   local: "test",
  //   domain: "test.dev",
  //   tags: ["test"]
  // }
```

```
* Write a function that extracts the local part and the domain and a
list of tags from an e-email string
```