Homework 2 - JavaScript Arrays, Functions and Objects

Deadline 5th June, 2022, 23:59.

Tapşırıq

Verilmiş məsələləri həll edin.

Qaydalar

Məsələləri həll edərkən hər bir məsələ üçün ayrıca JavaScript faylı yaradın (e.g. `sum.js`) və həmin məsələyə aid bütün kodlar o faylın içərisində olsun. Məsələləri həll edərkən heç bir üçüncü partiya kodlardan, hətta JavaScriptin öz built-in funksiyalarından da işlətmək olmaz (Bəzi hallar istisna olmaqla).

Misalçün, `map` funksiyasını tətbiq edərkən, JavaScript-in `map` metodunu işlətməyin. Digər hallarda, built-in funksiyalardan istifadə etmək olar (misalçün `Math.random`, `String.prototype.charCodeAt` və s.)

Heç bir funksiyada `console.log` işlətməyin, hər bir funksiya deyilən parametrləri götürməli və geriyə yeni dəyər qaytarmalıdır (`return`).

Öz yazdığınız funksiyaları işlədə bilərsiniz. Misalçün, 2-ci məsələdə yazdığınız funksiya 6-cı və 19-cu məsələlərdə işinizə yaraya bilər.

Göndərmək

Tapşırığı göndərmək üçün, əvvəlcə yazdığınız kodu hər hansı bir repossitoryə push edin, daha sonra isə orkhan.r.huseyn@gmail.com emailinə ***Homework 2 - JavaScript Arrays, Functions and Objects*** başlığı altında göndərin.

Məsələlər

1. `sum` adlı funksiya yazın. Funksiya ədədlərdən ibarət bir `array`i parametr kimi qəbul etməli və `array`ın bütün elementlərinin cəmini qaytarmalıdır:

```
sum([1, 2, 3, 4, 5]); // cavab: 15
sum([-1, -2, -3, -4, -5]); // cavab: -15
```

```
sum([-1, 0, 1, 2, -2]); // cavab: 0
sum([]); // cavab: 0
sum(); // cavab: 0
```

2. `random` adlı funksiya yazın. Funksiya `lower` və `upper` adlı iki parametr qəbul etməli və həmin iki ədəd arasında (hər ikisi daxil olmlaqla) ixtiyari bir natural ədəd qaytarmalıdır:

```
random(1, 6); // cavab: 1 və 6 arasında ixtiyari natural ədəd
random(0, 10); // cavab: 0 ilə 10 arasında ixtiyari natural ədəd
```

3. `arithmeticMean` adlı funksiya yazın. Funksiyanız ədədlərdən ibarət bir `array` i parametr kimi qəbul etməli və `array` in elementlərinin ədədi ortasını qaytarmalıdır:

```
arithmeticMean([3, 3, 3, 3, 3]); // cavab: 3
arithmeticMean([1, 5, 6, 10, 4]); // cavab: 5.2
arithmeticMean([]); // cavab: 0
arithmeticMean(); // cavab: 0
```

4. `geometricMean` adlı funksiya yazın. Funksiyanız ədədlərdən ibarət bir `array` i parametr kimi qəbul etməli və `array` in elementlərinin həndəsi ortasını qaytarmalıdır:

```
geometricMean([1, 2, 3, 4, 5, 6, 7, 8]); // təxmini cavab: 3.76435
geometricMean([15, 12, 13, 19, 10]); // təxmini cavab: 13.477
geometricMean(); // cavab: 0
geometricMean([]); // cavab: 0
```

5. `euclideanDistance` adlı funksiya yazın. Funksiya iki vektoru parametr kimi qəbul etməli və onlar arasındakı Evklid məsafəsini qaytarmalıdır. Vektor dedikdə içərisində iki element olan `array` dən söhbət gedir. Arrayin birinci elementi `x` ikinci elementi isə `y` i bildirir: `[x, y]`.

```
// (3, 4) və (7, 7) nöqtələri arasındakı məsafə
euclideanDistance([3, 4], [7, 7]); // cavab: 5

// (3, 4) və (4, 3) nöqtələri arasındakı məsafə
euclideanDistance([3, 4], [4, 3]); // cavab: 1.41421

euclideanDistance([], []); // cavab: 0
euclideanDistance(); // cavab: 0
```

6. `pickOne` adlı funksiya yazın. Funksiya bir ədəd `array` qəbul etməli və içərisindən ixtiyari bir elementi qaytarmalıdır:

```
pickOne(['Cəfər', 'Aynur', 'Leyla', 'Zöhrə', 'Günay']);
// cavab: adı çəkilmiş şəxslərdən hər hansı biri
```

7. `includes` adlı funksiya yazın. Funksiya bir ədəd `array` və bir ədəd axtarış üçün dəyər qəbul etməlidir. Əgər həmin dəyər `array`in içərisində mövcuddursa, `true` əks halda `false` qaytarmalıdır:

```
includes([5, 4, 3, 2, 1], 4); // cavab: true
includes([-1, 4, 3, 7, 9], 10) // cavab: false
includes(['a', 'b', 'c', 'd'], 'b'); // cavab: true
includes([], 13); // cavab: false
includes(); // cavab: false
```

8. `unique` adlı funksiya yazın. Funksiya bir ədəd `array`i parametr kim qəbul etməli, içərisində təkrar elementlər olmayan **yeni** bir `array` qaytarmalıdır:

```
unique(['a', 'a', 'b', 'c', 'd', 'b']); // cavab: ['a', 'b', 'c', 'd']
unique([1, 1, 1, 1, 5]); // cavab: [1, 5]
unique([]); // cavab: []
```

9. `intersection` adlı funksiya yazın. Funksiya iki `array`i parametr kimi qəbul etməli və onların kəsişməsini **yeni** array olaraq qaytarmalıdır:

```
intersection([2, 1], [3, 2]); // cavab: [2]
intersection([0, 1, 2, 5], [4, 7, 1, 5]); // cavab: [1, 5]
intersection([1, 2, 3, 4], [5, 6, 7, 8]); // cavab: []
intersection([], []); // cavab: []
intersection(); // cavab: []
```

10. `flat2D` adlı funksiya yazın. Funksiya bir ədəd iki ölçülü `array`i parametr kimi qəbul etməli və bir ölçülü **yeni** bir array qaytarmalıdır:

```
flat2D([[1, 2, 3], [4, 5, 6], [7, 8, 9]]); // cavab: [1, 2, 3, 4, 5, 6, 7, 8, 9]
flat2D([[12, 3, 1], [54]]); // cavab: [12, 3, 1, 54]
flat2D([[]]);// cavab: []
flat2D([]); // cavab: []
flat2D(); // cavab: []
```

11. `union` adlı funksiya yazın. Funksiya iki `array`i parametr kimi qəbul etməli və onların birləşməsini **yeni** array olaraq qaytarmalıdır:

```
union([20, 12], [8, 15, 6]); // cavab: [20, 12, 8, 15, 6]
union([2], [1, 2]); // cavab: [1, 2]
union([1, 2, 3]); // cavab: [1, 2, 3]
union([], []); // cavab: []
union(); // cavab: []
```

12. `reverse` adlı funksiya yazın. Funksiya bir ədəd `array`i parametr kimi qəbul etməli və onun tərsini **yeni** bir array kimi qaytarmalıdır:

```
reverse([1, 2, 3]); // cavab: [3, 2, 1]
reverse([10, -1, 3, 1, 4, 2]); // cavab: [2, 4, 1, 3, -1, 10]
reverse(); // cavab: []
```

13. `map` adlı funksiya yazın. Funksiya bir ədəd `array`i və bir ədəd funksiyanı parametr kimi qəbul etməli, funksiyanın nəticəsini arrayın hər bir elementinə tətbiq etməli və **yeni bir array** qaytarmalıdır:

```
map([1, 2, 3], function(x) { return x * 2; }); // cavab: [2, 4, 6]
map([3, 6, 9], function(x) { return x * x; }); // cavab: [9, 36, 81]
map([1, 2, 3], function(x) { return [x]; }); // cavab: [[1], [2], [3]]
map(['a', 'b', 'c', 'd'], x => null); // cavab: [null, null, null, null]


const people = ['Cəfər', 'Ümid', 'Etibar'];
function fn(name) {
   return {
      tag: 'div',
      textContent: name,
   };
}

map(people, fn);
/*
[
  {tag: 'div', textContent: 'Cəfər'},
  {tag: 'div', textContent: 'Ümid'},
  {tag: 'div', textContent: 'Etibar'}
]
*/
```

14. `filter` adlı funksiya yazın. Funksiya bir `array`i və bir funksiyanı parametr kimi qəbul etməli və geriyə **yeni bir array** qaytarmalıdır. Verilmiş callback funksiyası

arrayin hər bir elementini yoxlamalı və onun `true` qaytardığı bütün elementlər yeni arrayə daxil edilməlidir:

```
filter([1, 2, 3, 4, 5, 6], function (x) { return x % 2 === 0; });
// cavab: [2, 4, 6]

filter([1, 2, 3, 4, 5, 6], function (x) { return x > 3; });
// cavab: [4, 5, 6]

filter([-2, -1, 0, 1, 2], x => x < 0); // cavab: [-2, -1]
filter([-2, -1, 0, 1, 2], num => num > 0); // cavab: [1, 2]


const sites = ['my.gov.az', 'report.az', 'google.com', 'asan.gov.az', 'now.sh'];

const ourDomains = site => {
   return site.endsWith('.az');
}

filter(sites, ourDomains); //cavab: ['my.gov.az', 'asan.gov.az', 'report.az']
```

15. `find` adlı funksiya yazın. Funksiya bir ədəd `array`i və bir funksiyanı parametr kimi qəbul etməlidir. Verilmiş callback funksiyası arrayin hər bir elementini yoxlamalı və onun `true` qaytardığı **ilk** element `find` funksiyasını cavabı olmalıdır:

```
find([1, 2, 3, 4, 5], function(num) { return num === 1; }); // cavab: 1
find(['Ümid', 'Etibar', 'Uday', 'Könül'], name => name.startsWith('Ud')); // cavab: Uday

find([-3, -2, -1, 0, 1, 2, 3], x => x > 0); // cavab: 1
find([0, 1, 2, 3, 4], x => x < 0); // cavab: undefined
```

16. `some` adlı funksiya yazın. Funksiya bir ədəd `array`i və bir funksiyanı parametr kimi qəbul etməli və geriyə `Boolean` tipli dəyər qaytarmalıdır. Verilmiş callback funksiyası arrayin hər bir elementini yoxlamalı və əgər **ən azı bir element belə** `true` nəticəsi versə, `some` funksiyasının cavabı `true` olmalıdır. Bütün digər hallarda `some` funksiyası `false` qaytarmalıdır:

```
some([-1, -2, -5, -123, 0, 1], x => x > 0); // cavab: true
some([-1, -2, -5, -123, 0], x => x > 0); // cavab: false

function isBiggerThan10(element, index, array) {
  return element > 10;
}

some([2, 5, 8, 1, 4], isBiggerThan10); // cavab: false
some([12, 5, 8, 1, 4], isBiggerThan10); // cavab: true
```

17. `every` adlı funksiya yazın. Funksiya bir ədəd `array`i və bir funksiyanı parametr kimi qəbul etməli və geriyə `Boolean` tipli dəyər qaytarmalıdır. Verilmiş callback funksiyası arrayin hər bir elementini yoxlamalı və əgər bütün elementlər `true` nəticəsi versə, `every` funksiyasının nəticəsi `true` olmlalıdır. Qalan bütün hallarda `every` funksiyası `false` qaytarmalıdır:

```
const sites1 = ['my.gov.az', 'report.az', 'google.com', 'asan.gov.az', 'now.sh'];
const sites2 = ['my.gov.az', 'report.az', 'asan.gov.az'];

every(sites1, domain => domain.endsWith('.az')); // cavab: false
every(sites2, domain => domain.endsWith('.az')); // cavab: true
```

18. `reduce` adlı funksiya yazın. Funksiya bir ədəd `array` və biri ədəd callback funksiyasını parametr kimi götürməli və geriyə yeni dəyər qaytarmalıdır. `reduce` funksiyasının necə işləməli olduğunu öyrənmək üçün reduce funksiyasının dokumentasiyasını oxuyun.

```
const array = [1, 2, 3, 4];
const initialValue = 0;
function reducer(acc, curr) {
   return acc + curr;
}

const sum = reduce(array, reducer, 0);
console.log(sum); // cavab: 10
```

19. `generatePassword` adlı funksiya yazın. Funksiya `passwordLength` adlı bir parametr götürməli və həmin uzunluqda şifrə generasiya etməlidir. Şifrə rəqəmlərdən, böyük və kiçik ingilis hərflərindən ibarət olmalıdır:

```
generatePassword(16); // nümunə cavab: zNOkFPta0juCbIm5
generatePassword(32); // nümunə cavab: kHe0k2pwERrhPi4RB4mYtW5MRB876PS0
```

20. `ceasarCipher` adlı funksiya yazın. Funksiya bir ədəd `string`i və açar rəqəmi parametr kimi qəbul etməlidir. Verilən stringin ingilis hərflərindən ibarət bir cümlə olduğunu nəzərə alaraq, onu Sezar şifrəsi ilə şifrələyin və nəticəni başqa bir string kimi qaytarın.

```
ceasarCipher('ATTACKATONCE', 4); // cavab: EXXEGOEXSRGI
ceasarCipher('Salam', 3); // cavab: Vdodp
```