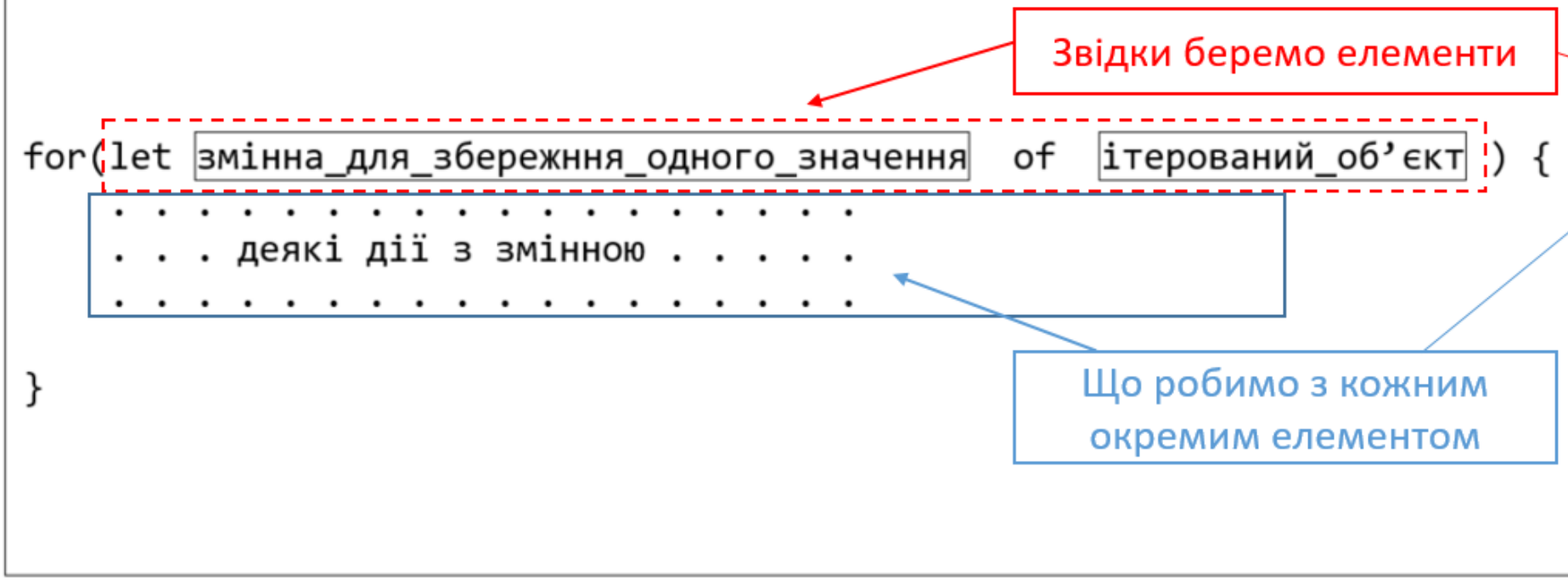
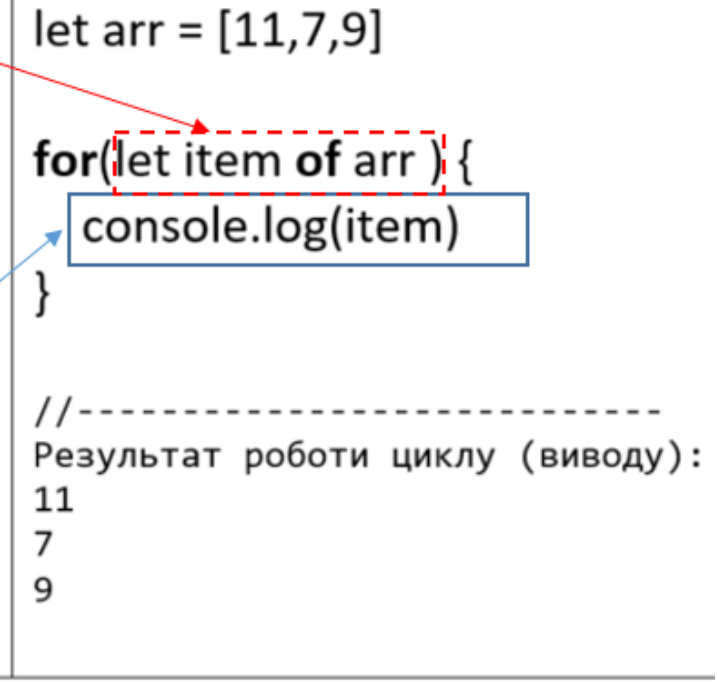
Рекомендації з обробки масивів

Потрібно переглянути усі елементи ***<u>без зміни елементів масиву.</u>*** Можна використати цикл **for..of**

Загальна форма	Приклад
	


```
//Дано масив чисел. Вивести елементи, які більші за 7
let arr = [2,3,9,8,1,51]

for(let  element of arr){
     if(element > 7)
       document.write( element )
}
```

Потрібно переглянути ***усі елементи*** або ***елементи з деякими*** номерами ***можливо із зміною елементів масиву.***

Можна використати цикл **for** ***(використовувати коли треба індекси або треба змінювати)***

```
for( індекс  = поч.індекс;  індекс <= кінц.індекс; індекс += крок ){
    ……
    …… операції з елементом : масив [ індекс ] ……
    ……
}
```

```
// Дано величини прибутків за рік. Знайти загальний прибуток

let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]

let sum = 0
for (let i = 0; i < arr.length; i++) {
  sum += arr[i]
}

alert(`Загальний прибуток = ${sum}`)
```

Приклад ***перегляду усіх елементів*** масиву (індекси від 0 до arr.length)

(у цьому випадку можна було використати for..of)

Потрібно переглянути ***усі елементи*** або ***елементи з деякими*** номерами ***<u>можливо із зміною елементів масиву.</u>***

Можна використати цикл **for**

for(**індекс** = **поч.індекс**; **індекс** <= **кінц.індекс**; **індекс += крок**){

…….

……. операції з елементом : масив [індекс] ……

…….

}

```
// Дано величини прибутків за рік.
// Знайти прибуток за 2 і 3 квартал (місяці від 4 до 9 )

let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]

let sum = 0
for (let i = 4; i <= 9; i++) {
  sum += arr[i]
}

alert(`Загальний прибуток = ${sum}`)
```

Приклад ***<u>перегляду елементів</u>*** масиву у ***<u>заданому діапазоні індексів</u>***
(індекси від 4 до 9)
(у цьому випадку використати for..of <u>неможна</u> було, бо не всі елементи)

Потрібно переглянути ***усі елементи*** або ***елементи з деякими*** номерами ***<u>можливо із зміною елементів масиву.</u>***

Можна використати цикл **for**

```
for( індекс  = поч.індекс;  індекс <= кінц.індекс; індекс += крок ){
    ……
    …… операції з елементом : масив [ індекс ] ……
    ……
}
```

Приклад ***<u>зміни елементів</u>*** масиву у ***<u>заданому діапазоні індексів</u>***
(індекси від 4 до 9)
(у цьому випадку використати for..of <u>неможна</u> було, бо не всі елементи і елементи змінюються)

```
// Дано величини прибутків за рік. Для 2 і 3 кварталів (місяці від 4 до 9 )
// елементи, що менші за 1000 замінити на 1000

let arr = [11, 4, 12, 5, 23, 70, 1, 34, 22, 15, 43, 7]

for (let i = 4; i <= 9; i++) {
  if (arr[i] < 1000) arr[i] = 1000
}

alert(arr)
```

Потрібно переглянути усі елементи ***можливо із зміною елементів масиву.*** Можна використати цикл **forEach**

За допомогою

forEach

(***перегляд*** і можливо ***зміна*** елементів)

item – змінна, у яку передаються (копіюються) поступово елементи з базового

index - індекс поточного елемента (елемента, над яким зараза проводять обчислення)

baseArrRef – змінна, що містить посилання на масив (з яким проводимо операції)

якщо *index* і *baseArrRef* не використовуються, то їх писати не потрібно

назва_масиву . **forEach** (

(item, index, baseArrRef) => дії_над_елементами_масиву

)

```
//Дано масив чисел. Вивести елементи, які більші за 7
let arr = [2,3,9,8,1,51]

arr.forEach(
   element =>{
     if(element > 7)
       document.write(element)
   }
)
```

Приклад ***перегляду усіх елементів*** масиву *(без зміни елементів)*

```
//Дано масив чисел. Елементи більші за 7 замінити на 0
let arr = [2,3,9,8,1,51]

arr.forEach(
   ( element, index, baseArrRef)=>{
     if(element > 7)
       baseArrRef [ index ] =0  // рівносильно arr[index]=0
   }
)
```

Приклад ***перегляду усіх елементів*** масиву *(зі зміною елементів)*

Задача. Дано величини прибутків за рік. Кожен елемент помножити на 200

Розв'язок зі зміною існуючого масив

З використанням циклу	`let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]`
	
	`for (let i = 0; i <= arr.length; i++) {`
	`arr[i] *= 200`
	`}`
	
	`alert(arr)`

З використанням forEach	`let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]`
	
	`arr.forEach(`
	`(element, index, arrBaseRef) => arrBaseRef[i] *= 200`
	`)`
	
	`alert(arr)`

Створення на основі елементів іншого ітерованого об’єкта з додатковими перетвореннями елементів

(обчисленнями нових елементів нового масиву за деяким правилом на основі елементів базової колекції)

Зміст	Загальний вигляд	Приклад

Створення на основі деякої колекції елементів ***ітерованого об’єкта*** (в якості ітерованого об’єкта може бути і масив)
item – змінна, у яку передаються (копіюються) поступово елементи з базової колекції
index - індекс поточного елемента (елемента, над яким зараза проводять обчислення)
	Створення нового масив на основі елементів базового ітерованого об’єкта, над якими додатково проводяться деякі обчислення
	
	назва = Array.**from**(
	базова_колекція_елементів ,
	(item, index) => вираз_перетворення ,
	thisArg
	)
		//Створити масив на базі іншого, де кожен елемент множиться на 3
		
		const arr1 = [1, 33, 2, 1, 9]
		
		const arr2= **Array.from(**
		arr1,
		(element, index) => element * 3
		)
		// arr2 = [3,99,6,3,27]

Створення на основі ***іншого масиву***
 	
item – змінна, у яку передаються (копіюються) поступово елементи з базового
index - індекс поточного елемента (елемента, над яким зараза проводять обчислення)
baseArrRef – змінна, що містить посилання на базовий масив (з якого беремо елементи для обчислень)
	Створення нового масив на основі елементів базового ітерованого об’єкта, над якими додатково проводяться деякі обчислення
	
	назва_нового_масиву = базовий_масив.**map** (
	(item, index, baseArrRef) => вираз_перетворення
	)
		//Створити масив на базі іншого, де кожен елемент множиться на 3
		
		const **oldArr** = [1, 33, 2, 1, 9]
		
		const arr2= **oldArr.** map **(**
		(element, index, baseArrRef) => element * 3
		)
		// arr2 = [3,99,6,3,27]
		
		//Якщо індексі посилання на базовий масив не використовуються, то їх можна не писати
		
		const arr2= arr1. **map (**element => element * 3)

Задача. Дано величини прибутків за рік. Кожен елемент помножити на 200

Розв'язок зі створенням нового масиву

З використанням циклу ***for***	`let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]`
	
	`let newArr = []`
	`for (let i = 0; i <= arr.length; i++) {`
	`  newArr.push( arr[i] * 200 )`
	`}`
	
	`alert(newArr)`

З використанням циклу ***for..of***	`let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]`
	
	`let newArr = []`
	`for (const element of arr) {`
	`  newArr.push( element * 200 )`
	`}`
	
	`alert(newArr)`

З використанням **map**	`let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]`
	
	`let newArr = arr.map(element => element * 200)`
	
	`alert(newArr)`

Фільтрація елементів масиву

(створення нового масиву, у якому елементи задовольняють деякій умові)

Зміст	Загальний вигляд	Приклад

Повертає масив елементів, які задовольняють заданій умові **condition**
(якщо таких немає, то масив буде порожнім)
	назва_масиву . **filter** (
	
	(element, index, baseArrRef) => condition
	
	)
		//Елементи, що є більшим за 20
		const arr1 = [1, 33, 2, 1, 9, 2, 90]
		const elements = arr1. **filter** (
		(el, index, baseArrRef) => el > 20
		)
		// elements = [33, 90]
		
		// index, baseArrRef не використовуються, тому можна не писати
		//const elements = arr1. **filter** (el => el > 20)

Задача. Дано величини прибутків за рік. Сформувати новий масив, де є тільки значення, що менші за 10.

З використанням циклу ***for***	```let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]
	
	let newArr = []
	for (let i = 0; i <= arr.length; i++) {
	  if(arr[i]<10)
	  newArr.push(arr[i])
	}
	
	alert(newArr)```

З використанням циклу ***for..of***	```let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]
	
	let newArr = []
	for (const element of arr) {
	  if(element<10)
	  newArr.push(element)
	}
	
	alert(newArr)```

З використанням ***filter***	```let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7] let newArr = arr.filter(element => element < 10) alert(newArr)```

Обчислення агрегованого (акумулючого) значення на основі елементів усього масиву

Зміст	Загальний вигляд	Приклад

reduce (
  callback-func ,
  initialAgregateValue
)
послідовно викликає функцію callback-func один раз для кожного елемента масиву зліва-направо (за виключенням порожніх елементів undefined), обчислюючи при цьому деяку агреговану величину. Значення величини при цьому поступово (зліва-направо) обраховується (накопичується) на основі кожного елемента масиву. Наприклад, при знаходженні суми до величини суми поступово додається кожен елемент масиву.
	назва_масиву . **reduce** (
	(***prevAgregateValue***, element, index, baseArrRef)
	=> ***nextAgregateValue*** ,
	initialAgregateValue
	)
	
	За замовчуванням (якщо не вказати) ***initialAgregateValue*** *дорівнює першому елементу масиву*
	
	InitialAgregateValue=назва_масиву *[0]*
		`//Знайти кількість додатних`
		`function checkFunc(prevResult, x, i, arr) {`
		`  if (x > 0) { prevResult ++ };`
		`  return prevResult`
		`}`
		`let a = [1, 2, -3, 4, -5];`
		`let res = a.reduce(checkFunc, 0); //спочатку result=0`
		`  // res = 3`
		
		`//------------------------------`
		`//Знайти суму елементів масиву`
		`function sum(prevResult, x, i, arr) {`
		`  prevResult += x;`
		`  return prevResult;`
		`}`
		`let a = [1, 2, -3, 4, -5];`
		`let res = a.reduce(sum);  //спочатку prevResult=a[0]`
		`  // res = -1`
		
		`// скорочений запис`
		`//let res = a.reduce((prevResult, x)=> prevResult +x);`

reduceRight (
  callback-func ,
  initialAgregateValue
)
послідовно викликає функцію callback-func один раз для кожного елемента масиву справа-наліво (за виключенням порожніх елементів undefined)
	назва_масиву . **reduceRight** (
	(***prevAgregateValue***, element, index, baseArrRef)
	=> ***nextAgregateValue*** ,
	initialAgregateValue
	)
	
	За замовчуванням (якщо не вказати) ***initialAgregateValue*** *дорівнює останньому елементу масиву*
	initialAgregateValue=назва_масиву.at(-1)
		Початкове значення суми дорівнює останньому елементу. Поступово розглядаючи елементи справа-наліво додаємо наступний елемент тільки у тому випадку, якщо його значення менше за поточне значення суми.
		
		`let a = [1, 2, 3];`
		`let res = a.reduceRight(`
		`  (prevResult, x) =>`
		`  x < prevResult?`
		`  prevResult + x :`
		`  prevResult`
		`);`
		`  // res = 6`

Задача. Дано величини прибутків за рік. Знайти суму тих, які менші за 10.

З використанням циклу ***for***	```let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]
	
	let sum = 0
	for (let i = 0; i <= arr.length; i++) {
	  if(arr[i]<10)
	  sum = sum + arr[i]
	}
	
	alert(sum)```

З використанням циклу ***for..of***	```let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7]
	
	let sum = 0
	for (const element of object) {
	  if(element<10)
	  sum = sum + element
	}
	
	alert(sum)```

З використанням ***reduce*** У функцію передаємо: ***prevSum*** – сума, яка була до цього елемента, ***element*** – копія поточного елемента Функція повинна повернути новий результат: – сума яка була до + елемент, якщо він < 10, – інаше – повернути суму, яка була до цього	```let arr = [11, 4, 12, 5, 23, 76, 1, 34, 22, 15, 43, 7] let newArr = arr.reduce((prevSum, element) => element < 10 ? prevSum + element : prevSum) alert(sum)```