Упражнения: Основи на програмирането с JavaScript част 3 - Mасиви

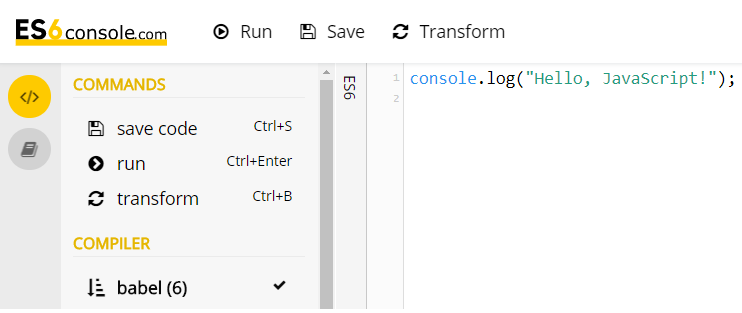
<https://www.w3schools.com/jsref/jsref_obj_array.asp>

## 1. Игра с масиви

## (Тук ще разгледаме **деклариране на масив**, метода **toString()** върху масиви и метода **join()**)

Напишете **програма**, която заделя в масив числата от 1 до 10 и отпечатва елементите в масива разделени със запетая.

1. Може да използвате онлайн редактор като: <https://es6console.com/>, **VS Code** или **IntelliJ IDEA**



Напишете програмния код в белия прозорец отдясно и стартирайте c бутона **run** или ползвайте клавишната комбинация Ctrl+Enter.

|  |
| --- |
| let arr = new Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);    console.log(arr); |

Помислете за оптимизация на кода. Как можем да изпълним програмата ползвайки **фор- цикъл**? Разгледайте как може да реализирате решението, ползвайки метода[**join()**](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_join). Сега

отпечатайте числата разделени с празно място без запетая.

Решение:

С **фор-цикъл** генерираме числа от 1 до 10 (i = 1, i = 2... i=10) и на всяка итерация (завъртане на цикъла) с метода **push(елемента)** вкарваме съответната стойност в края на масива. Накрая отпечатваме резултата с метода **масива.join(разделител)** като в този случай сме задали за разделител празно място.

|  |
| --- |
| let arr = new Array();  for(let i = 1; i <= 10; i++)  {  arr.push(i);  }    console.log(arr.join(" ")); |

## 2. Сбор на елементи в масив

## (Тук ще разгледаме обхождане на масив ползвайки **фор-цикъл**, достъпване на елемент от масива по индекс **[index]** )

Допишете дадената **javaScript програма**, която сумира стойностите на елементите в дадения масив - **let array = [32, 4, 3, 8, 6, 2, 66, 5];**

Отпечатайте сумата като краен резултат.

|  |
| --- |
| let arr = [32, 4, 3, 8, 6, 2, 66, 5];  let sum = 0;  for(let i = 0; i < arr.length; i++)  {  let number = //TODO    }  console.log(sum); |

Сумата на елементите в масив може да изчислите, използвайки функционалния метод **array.reduce((a, b) => a + b)**

Решение 1:

С **фор-цикъл** обхождаме масива и на всяка итерация декларираме променлива **number**, която приема за стойност текущия елемент на масива (**arr[i]**). Ползвайки индексаторите **[индекс]** със зададен индекс ние можем да достъпим стойност от масива, която стои на индекс - 1 позиция. Всеки път прибавяме към променливата **sum** стойността на **number**. Накрая отпечатваме **sum**.

|  |
| --- |
| let arr = [32, 4, 3, 8, 6, 2, 66, 5];  let sum = 0;  for(let i = 0; i < arr.length; i++)  {  let number = arr[i];  sum += number;  }  console.log(sum); |

Решение 2:

JavaScript е функционален език и можем да решим задачата използвайки метода .**reduce()**, в който има зададена **ламбда** функция, с този метод можем да редуцираме всички елементи до тяхната сума.

|  |
| --- |
| let arr = [32, 4, 3, 8, 6, 2, 66, 5];  let sum = arr.reduce((a, b) => a + b);  console.log(sum); |

## 3. Сбор на първия и последния елемент в масив

Напишете  **javaScript програма**, която сумира стойностите на първия и последния елемент в дадения масив - **let array = [32, 4, 3, 8, 6, 2, 66, 5];**

Отпечатайте сумата като краен резултат.

Решение:

С индексаторите взимаме първия елемент от масива, този на нулева позиция и последния, като за него използваме свойството на масива length, което връща дължината му. От дължината изваждаме единица (масивите започват от 0, а не 1)

|  |
| --- |
| let arr = [32, 4, 3, 8, 6, 2, 66, 5];  let lastIndex = arr.length - 1;  let sum = arr[0] + arr[lastIndex];  console.log(sum); |

## 4. Триене на числа с нечетна стойност

## (Тук ще разгледаме обхождане на масив ползвайки **фор-цикъл**, достъпване на елемент от масива по индекс **[index]** и триене на елемент от масива с метода delete arr[index])

Напишете **javaScript програма**, която изтрива всички нечетни стойности на елементите в дадения масив - **let array = [2, 4, 3, 99, 6, 123, 66, 5];**

При всяко триене изведете на конзолата съобщението - "**Element with index { 1 } was successfully deleted**" .

Отпечатайте елементите в масива.

|  |
| --- |
| let arr = [2, 4, 3, 99, 6, 123, 66, 5];    for(let i = 0; i < arr.length; i++)  {  //TODO  }  //TODO |

Решение:

На всяка итерация проверяваме текущия елемент от масива дали има четна или нечетна стойност ползвайки оператора **%** (мода), който ни връща остатъка от целочислено деление, в случай че имаме нечетна стойност трием елемента от масива.

|  |
| --- |
| let arr = [2, 4, 3, 99, 6, 123, 66, 5];  for(let i = 0; i < arr.length; i++)  {  let number = arr[i];  if(number % 2 == 1){  delete arr[i];  }  }  console.log(arr.join("")); |

## 5. Триене на елемент от масив

# (Тук ще разгледаме метода **indexOf()** с който можем да вземем индекса на даден елемент)

Допишете **javaScript функция**, която получава като параметър даден елемент, открива го в масив и го изтрива.

Отпечатайте изтрития елемент на първи ред и елементите в масива на втори.

|  |
| --- |
| function deleteItem(item){  let array = ["Lemon", "Mango", "Cucumber", "Tomato"];  //TODO  console.log(array.join(' '));  }    deleteItem("Lemon"); |

Решение:

Имаме функция, която приема като параметър външния вход (елемента който трябва да изтрием. В нея имаме деклариран масив, който държи в себе си няколко елемента ("Lemon", "Mango", "Cucumber", "Tomato"), с метода indexOf(elementa) можем да вземем индекса на елемент, който търсим, а с delete arra[index] да го изтрием.

|  |
| --- |
| function deleteItem(item){  let array = ["Lemon", "Mango", "Cucumber", "Tomato"];  let index = array.indexOf(item);  delete array[index];  console.log(array.join(' '));  }    deleteItem("Lemon"); |

## 6. Вторият съвпадащ елемент от масив

# (Тук ще разгледаме метода **indexOf(element, startIndex)** с който можем да вземем индекса на даден елемент като започнем търсенето от зададен стартов индекс)

Допишете **javaScript функция**, която получава като параметър даден елемент, открива втория съществуваш такъв в масив и го изтрива.

|  |
| --- |
| function deleteItem(item){  let array = ["Lemon", "Mango", "Cucumber", "Tomato", "Lemon","Mango", "Cucumber", "Tomato"];  let index = array.indexOf(item);  let currentIndex = //TODO  delete array[currentIndex];  console.log(array.join(' '));  }    deleteItem("Lemon"); |

Отпечатайте изтрития елемент на първи ред и елементите в масива на втори.

Решение:

Имаме функция, която приема като параметър външния вход (елемента който трябва да изтрием). В нея имаме деклариран масив, който държи в себе си няколко елемента ("Lemon", "Mango", "Cucumber", "Tomato" "Lemon", "Mango", "Cucumber", "Tomato"), с метода **indexOf(elementa)** можем да вземем индекса на първия елемент, който търсим, а с **.indexOf(item, index)** търсим елемент след определен индекс.

|  |
| --- |
| function deleteItem(item){  let array = ["Lemon", "Mango", "Cucumber", "Tomato", "Lemon", "Mango", "Cucumber", "Tomato"];  let index = array.indexOf(item);  let currentIndex = array.indexOf(item, index)  delete array[currentIndex];  console.log(array.join(' '));  }  deleteItem("Lemon"); |

## 7. Най-голямо число в масив

Допишете **javaScript функция**, която получава като параметър поредица от числа в масив и отпечатва като резултат елемента с най-голяма числова стойност.

|  |
| --- |
| function max(array)  {  let maxNumber = Number.MIN\_VALUE;  for(let i = 0; i < array.length; i++){  //TODO  }  console.log(maxNumber);  }    max([1, 3, 5, 4, 3]); |

Решение:

|  |
| --- |
| function max(array)  {  let maxNumber = Number.MIN\_VALUE;  for(let i = 0; i < array.length; i++){  let number = array[i];  if(number > maxNumber){  maxNumber = number;  }  }  console.log(maxNumber);  }    max([1, 3, 5, 4, 3]); |

## 8. Най-малко число в масив

Допишете **javaScript функция**, която получава като параметър поредица от числа в масив и отпечатва като резултат елемента с най-малка числова стойност.

Решение:

|  |
| --- |
| function min(array)  {  let minNumber = Number.MAX\_VALUE;  for(let i = 0; i < array.length; i++){  let number = array[i];  if(number < minNumber){  minNumber = number;  }  }  console.log(minNumber);  }    min([1, 3, 5, 4, 3]); |

//Тези задачи ще бъдат разгледани на следващото упражнение

## 9. Двете най-големи числа в масив

# (Тук ще разгледаме сортиране на елементи с методите **sort()** и **slice(start, end + 1)**)

Допишете **javaScript функция**, която получава като параметър поредица от числа в масив и отпечатва като резултат двата елемента с най-голяма числова стойност.

|  |
| --- |
| function biggestTwoNumbers(array)  {  //TODO  console.log(biggestNumber)  }    biggestTwoNumbers([1, 3, 5, 4, 3, 5, 23]); |

## 10. Масив от 10 елемента

(Тук ще разгледаме метода **push()**)

Напишете **програма**, която взима първите 10 символа от текста ("Push all elements to array.") и ги добавя в празен масив. Отпечатвайте дължината на масива при всяко едно добавяне със съобщението "Count of array is {дължината}". Накрая изведете елементите на масива, които сте добавили разделени със запетая.

|  |
| --- |
| let array = [];  let text = "Push all elements to array.";  let index = 0;  let length = array.push(text[index++]);  while(length < 10){  //TODO  }  console.log(array.join(", ")); |

## 11\*. Плодове

(Тук ще разгледаме метода **shift()**)

Даден ти следния **html** в него има скрипт таг в който са декларирани две функции (**start()** и **eat()**). Първата изрисува на документа плодовете от масива, а втората (която трябва да допишеш) изтрива плодове от колекцията и изписва с **alert()** съобщение за успешно действие. Функцията **eat()** се изпълнява при натискане на бутона "take fruit". При всяко натискане на бутона трябва да се вземе първия елемент от колекцията.

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Fruits</title>  </head>  <style>  img{  width: 60px;  height: 60px;  }  </style>  <body onload="start()">  <div id="fruits"></div>  </body>  <script>  let arr = ["apple", "mango", "lemon"];  function start(){  for(let fruit of arr){  switch(fruit){  case "apple":  document.getElementById("fruits").innerHTML += `  <h1>${fruit}</h1>  <img src="http://clipground.com/images/red-apple-clipart-1.jpg"/>`;  break;  case "mango":  document.getElementById("fruits").innerHTML += `  <h1>${fruit}</h1>  <img src="https://png.pngtree.com/element\_origin\_min\_pic/17/08/02/ccfe07798e6584690a56beb358c07a74.jpg"/>`;  break;  case "lemon":  document.getElementById("fruits").innerHTML += `  <h1>${fruit}</h1>  <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS18qkt61wHMNG8IVfr\_p\_7N2SIGyOB\_kpQBvDywY2X-D3lrN2pxQ"/>`;  break;  }  }    if(arr.length > 0)  document.getElementById("fruits").innerHTML += `<div><button onclick="eat()">take fruit!</button></div>`;  }    function eat(){  //TODO  document.getElementById("fruits").innerText = "";  start();  alert(`The ${fruit} is taken!`)  }  </script>  </html> |

Бележки:

**Метод** **Описание**

**concat()** Комбинира елементите на два или повече масива в нов масив

**join()**  Комбинира елементите на масив в един низ с разделителен знак

**pop()**  Премахва последния елемент от масива, след което връща премахнатия елемент

**push()** Добавя елемент в края на масив, след което връща числената стойност на новата дължина на масива

**reverse()**  Обръща посоката на елементите в масива - първия елемент се премества на последна позиция, а последния на първа и т.н.

**shift()** Премахва първия елемент на масива, след което връща този елемент

**unshift()**  Добавя елемент в началото на масива, след което връща числената стойност на новата дължина на масива

**slice()** Извлича указана секция от масив, след което връща тази секция като нов масив

**splice()** Премахва или заменя елементи от масив

**sort()**  Сортира елементите на масив по азбучен ред, базирайки се на техните низови стойности