|  |  |  |  |
| --- | --- | --- | --- |
| Метод № | Повикване | Описание | Примери |
| 1 | **find()** | **Връща стойността на първия елемент** в масив, който удовлетворява предоставената функция за тестване. Ако нито една стойност не отговаря на функцията за тестване, се връща **undefined**. | arr.find(callback(element[, index[, array]])[, thisArg])  const array1 = [5, 12, 8, 130, 44];  const found = array1.find(element => element > 10);  console.log(found);  // expected output: 12 |
| 2 | **join()** | **Създава и връща нов стринг чрез обединяване** на всички елементи в масив (или подобен на масив обект), разделен със запетаи или определен сепаратор. Ако масивът има само един елемент, тогава този елемент ще бъде върнат без използване на разделителя. | const elements = ['Fire', 'Air', 'Water'];  console.log(elements.join());  // expected output: "Fire,Air,Water"  console.log(elements.join(''));  // expected output: "FireAirWater"  console.log(elements.join('-'));  // expected output: "Fire-Air-Water" |
| 3 | **split()** | **Разделя стринга в подреден списък с поделементи**, слага ги в масив и връща масива. Разделянето става по схема (),(‘’),(“ ”),(“,”), (“\_”) и т.н. | const **str** = 'The quick brown fox jumps over the lazy dog.';  const **words** = str.split(' ');  console.log(words[3]);  // expected output: "fox"  const **chars** = str.split('');  console.log(chars[8]);  // expected output: "k"  const **strCopy** = str.split();  console.log(strCopy);  // expected output: Array ["The quick brown fox jumps over the lazy dog."] |
| 4 | **findIndex()** | **Връща индекса на първия елемент в масива**, който удовлетворява предоставената функция за тестване. В противен случай връща -1, което показва, че нито един елемент не е преминал теста. | const array1 = [5, 12, 8, 130, 44];  const isLargeNumber = (element) => element > 13;  console.log(array1.findIndex(isLargeNumber));  // expected output: 3 |
| 5 | **indexOf()** | **Връща първия индекс, при който даден елемент може да бъде намерен в масива**, или -1, ако той не присъства. | const beasts = ['ant', 'bison', 'camel', 'duck', 'bison'];  console.log(beasts.indexOf('bison'));  // expected output: 1  // start from index 2  console.log(beasts.indexOf('bison', 2));  // expected output: 4  console.log(beasts.indexOf('giraffe'));  // expected output: -1 |
| 6 | **lastIndexOf()** | **Връща индекса в извикващия String обект на последното появяване на посочената стойност**, търсейки назад от fromIndex. Връща -1, ако стойността не е намерена. | const paragraph = 'The quick brown fox jumps over the lazy dog. If the dog barked, was it really lazy?';  const searchTerm = 'dog';  console.log(`The index of the first "${searchTerm}" from the end is ${paragraph.lastIndexOf(searchTerm)}`);  // expected output: "The index of the first "dog" from the end is 52" |
| 7 | **map()** | **Обхожда всички елементи от един масив и прилага дадена функция спрямо всеки един от тях.** Запазва резултата в нов масив, който е идентичен по дължина с първия. Не променя оригиналния масив. | const array1 = [1, 4, 9, 16];  // pass a function to map  const map1 = array1.map(x => x \* 2);  console.log(map1);  // expected output: Array [2, 8, 18, 32] |
| 8 | **push()** | **Добавя един или повече елементи в края на масив** и връща новата дължина на масива. | const animals = ['pigs', 'goats', 'sheep'];  const count = animals.push('cows');  console.log(count);  // expected output: 4  console.log(animals);  // expected output: Array ["pigs", "goats", "sheep", "cows"]  animals.push('chickens', 'cats', 'dogs');  console.log(animals);  // expected output: Array ["pigs", "goats", "sheep", "cows", "chickens", "cats", "dogs"] |
| 9 | **pop()** | **Премахва последния елемент от масив и връща този елемент.** Този метод променя дължината на масива. Връща undefined, ако масивът е празен. Ако искаме да вземем елемент на последния индекс от даден масив, изпозлваме или този метод или arr.length – 1. Аналогично е. | const plants = ['broccoli', 'cauliflower', 'cabbage', 'kale', 'tomato'];  console.log(plants.pop());  // expected output: "tomato"  console.log(plants);  // expected output: Array ["broccoli", "cauliflower", "cabbage", "kale"]  plants.pop();  console.log(plants);  // expected output: Array ["broccoli", "cauliflower", "cabbage"] |
| 10 | **forEach()** | Методът forEach () **изпълнява предоставена функция веднъж за всеки елемент на масив.** | const array1 = ['a', 'b', 'c'];  array1.forEach(element => console.log(element));  // expected output: "a"  // expected output: "b" // expected output: "c"  arr.forEach(element => element += “a”) |
| 11 | **filter()** | **Създава нов масив с всички елементи, които са преминали проверка**, имплементирана в зададена, като параметър функция – т.нар. предикат – функция, която връща true или false. Връща само тези елементи, които отговарят на условията. Word.startsWith(‘F’); **Не променя стария масив!** | const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];  const result = words.filter(word => word.length > 6);  console.log(result);  // expected output: Array ["exuberant", "destruction", "present"] |
| 12 | **sort()**  **localCompare()** | **Сортира елементите в един масив, като го модифицира и връща същия, но вече сортиран масив**. По подразбиране подреждането е по възходящ ред, но това се случва едва след превръщането на елементите в стринг, т.е. **по азбучен ред**. Ако искаме да сортираме числа, трябва да подадем подходяща сравняваща функция, иначе и те ще бъдат подредени по азбучен ред => 1, 1000, 21, 3, 489 и т.н.  - Ако резултатът < 0, „а“ се сортира преди „б“; [a, b];  - Ако резултатът > 0, „б“ се сортира преди „а“; [b, a];  - Ако резултатът = 0, „а“ и „б“ са равни, няма промяна;  **localeCompare()** е стринг метод, който сравнява две букви лексикографски, независимо дали са главни или малки. Това е пример за сравняваща функция. | const months = ['March', 'Jan', 'Feb', 'Dec'];  months.sort();  console.log(months);  // expected output: Array ["Dec", "Feb", "Jan", "March"]  const array1 = [1, 30, 4, 21, 100000];  array1.sort();  console.log(array1);  // expected output: Array [1, 100000, 21, 30, 4]  months.sort((a, b) => a.localeCompare(b)); |
| 13 | **reverse()** | **Обръща масива на място. Първият елемент на масива става последен**, а последният елемент на масив става първият. | const array1 = ['one', 'two', 'three'];  console.log('array1:', array1);  // expected output: "array1:" Array ["one", "two", "three"]  const reversed = array1.reverse();  console.log('reversed:', reversed);  // expected output: "reversed:" Array ["three", "two", "one"]  // Careful: reverse is destructive -- it changes the original array.  console.log('array1:', array1);  // expected output: "array1:" Array ["three", "two", "one"] |
| 14 | **shift()** | **Премахва първия елемент от масив и връща този премахнат елемент**. Този метод променя дължината на масива. Аналогично е с arr[0], когато искаме да вземем първия елемент на 0 индекс от даден масив. | const array1 = [1, 2, 3];  const firstElement = array1.shift();  console.log(array1); expected output: Array [2, 3]  console.log(firstElement); expected output: 1 |
| 15 | **unshift()** | Добавя един или повече елементи в началото на масив и **връща новата дължина на масива.** | const array1 = [1, 2, 3];  console.log(array1.unshift(4, 5));  expected output: 5 / console.log(array1);expected output: Array [4, 5, 1, 2, 3] |
| 16 | **slice()** | **Връща нов масив като част от първоначален масив, изрязан от начало до край** (краят може да не е посочен и тогава взима от посочения индекс-старт до края, края на се включва!), където start и end представляват индекса на елементите в този масив. **Оригиналният масив няма да бъде модифициран.** Така се прави копие на масива.  Може да се използва отрицателен индекс, показващ отместване от края на масива. slice(-2) извлича последните два елемента от масива. **За порция от масив.** | const animals = ['ant', 'bison', 'camel', 'duck', 'elephant']; console.log(animals.slice(2));  // expected output: Array ["camel", "duck", "elephant"]  console.log(animals.slice(2, 4));  // expected output: Array ["camel", "duck"]  console.log(animals.slice(1, 5));  // expected output: Array ["bison", "camel", "duck", "elephant"] |
| 17 | **splice()** | **Променя съдържанието на масива, като добавя, изтрива или заменя съществуващи елементи или добавя нови**.  let arrDeletedItems = array.splice(start[, deleteCount[, item1[, item2[, ...]]]])  Ако delete count = 0 или е отрицателно число, няма да се премахне елемент, а ще се добави. Иначе, той посочва колко елемента искаме да премахнем. **Променя оригиналния масив.** | const months = ['Jan', 'March', 'April', 'June'];  months.splice(1, 0, 'Feb');  // inserts at index 1  console.log(months);  // expected output: Array ["Jan", "Feb", "March", "April", "June"]  months.splice(4, 1, 'May');  // replaces 1 element at index 4  console.log(months);  // expected output: Array ["Jan", "Feb", "March", "April", "May"] |
| 18 | **toString()** | **Връща обекта, като стринг** и съответно всички методи за стринг могат да се прилагат върху него, както и да се итерира с цикъл. | function Dog(name) {  this.name = name;  }  const dog1 = new Dog('Gabby');  Dog.prototype.toString = function dogToString() {  return `${this.name}`;  };  console.log(dog1.toString());  // expected output: "Gabby" |
| 19 | **concat()** | **Свързва/споява два масива в едно.** Не подменя съществуващите два масива, а връща нов масив. | const array1 = ['a', 'b', 'c'];  const array2 = ['d', 'e', 'f'];  const array3 = array1.concat(array2);  console.log(array3);  // expected output: Array ["a", "b", "c", "d", "e", "f"] |
| 20 | **includes()** | **Определя дали масивът включва определена стойност сред елементите си, връщайки true или false** според случая.  Когато сравнявате стрингове и символи, includes() е чувствителен към малки и големи букви, да се внимава.  Може да търси от даден символ нататък, като втори параметър. | const array1 = [1, 2, 3];  console.log(array1.includes(2));  // expected output: true  const pets = ['cat', 'dog', 'bat'];  console.log(pets.includes('cat'));  // expected output: true  console.log(pets.includes('at'));  // expected output: false |
| 21 | **some()** | **Проверява дали поне един елемент в масива преминава проверката**, която се изпълнява от предоставената от нас функция. Връща резулта - булева стойност. | const array = [1, 2, 3, 4, 5];  // checks whether an element is even  const even = (element) => element % 2 === 0;  console.log(array.some(even));  // expected output: true |
| 22 | **reduce()** | **Изпълнява редуктор функция (която предоставяте вие) за всеки елемент от масива**, и връща единичен output (един краен резултат).  Използваме го при сумиране на елементи в даден масив. | const array1 = [1, 2, 3, 4];  const reducer = (accumulator, currentValue) => accumulator + currentValue;  // 1 + 2 + 3 + 4  console.log(array1.reduce(reducer));  // expected output: 10  // 5 + 1 + 2 + 3 + 4  console.log(array1.reduce(reducer, 5));  // expected output: 15 |
| 23 | **reduceRight()** | **Прилага функция срещу акумулатор и всяка стойност на масива (от дясно наляво)**, за да намали масива до една единствена стойност. | const array1 = [[0, 1], [2, 3], [4, 5]].reduceRight(  (accumulator, currentValue) => accumulator.concat(currentValue)  );  console.log(array1);  // expected output: Array [4, 5, 2, 3, 0, 1] |
| 24 | **every()** | **Проверява дали всички елементи в масива преминават проверката**, която се изпълнява от предоставената от нас функция. Връща резулта - булева стойност. | const isBelowThreshold = (currentValue) => currentValue < 40;  const array1 = [1, 30, 39, 29, 10, 13];  console.log(array1.every(isBelowThreshold));  // expected output: true |
| 25 | **delete()** | **Премахва определена стойност от обект Set**, ако тя се съдържа в него. | const set1 = new Set();  set1.add({ x: 10, y: 20 }).add({ x: 20, y: 30 });  // Delete any point with `x > 10`.  set1.forEach((point) => {  if (point.x > 10) {  set1.delete(point);  }  });  console.log(set1.size);  // expected output: 1 |
| 26 | **fill()** | **Променя всички елементи в масива на статична стойност**, от начален индекс (по подразбиране 0) до краен индекс (по подразбиране array.length). Връща същия, но модифициран масив. | const array1 = [1, 2, 3, 4];  // fill with 0 from position 2 until position 4  console.log(array1.fill(0, 2, 4));  // expected output: [1, 2, 0, 0]  // fill with 5 from position 1  console.log(array1.fill(5, 1));  // expected output: [1, 5, 5, 5]  console.log(array1.fill(6));  // expected output: [6, 6, 6, 6] |
| 27 | **Spread syntax** | **Разпределящият синтаксис** позволява на итериращ се израз като масив или string да бъде разширен на места, където се използват нула или повече аргументи (или извиквания на функции), елементи (дефиниция на масиви), както и обект да бъде разширен на места, където се очакват нула или повече двойки от тип ключ-стойност (дефиниция на обекти). Изпозлва се оператор – „...“. Използва се за намаляване на изписвания код. Добра статия: https://www.freecodecamp.org/news/an-introduction-to-spread-syntax-in-javascript-fba39595922c/ | const numbers = [1, 2, 3];  console.log(sum(...numbers));  function sum(x, y, z) {  return x + y + z;  }  const numbers = [1, 2, 3];  console.log(sum(...numbers));  // expected output: 6 |
| OBJECTS AND CLASSES | | | |
| 30 | **Object.entries()** | **Връща масив от двойки [ключ, стойност]** на даден обект  Редът на масива, върнат от Object.entries (), не зависи от начина на дефиниране на обект. Ако има нужда от определено подреждане, тогава масивът трябва първо да бъде сортиран, като Object.entries (obj) .sort ((a, b) => b [0] .localeCompare (a [0])) ; | const object1 = {  a: 'somestring',  b: 42  };  for (const [key, value] of Object.entries(object1)) {  console.log(`${key}: ${value}`);  }  // expected output:  // "a: somestring"  // "b: 42"  // order is not guaranteed |
| 31 | **Object.keys()** | **Връща масив с всичките свойства (ключове)** на даден обект, итерирани в същия ред, в който нормалният цикъл би ги итерирал. | const object1 = {  a: 'somestring',  b: 42,  c: false  };  console.log(Object.keys(object1));  // expected output: Array ["a", "b", "c"] |
| 32 | **Object.values()** | **Връща масив от стойностите на свойствата** на даден обект, в същия ред като този, предоставен от цикъл for ... in. | const object1 = {  a: 'somestring',  b: 42,  c: false  };  console.log(Object.values(object1));  // expected output: Array ["somestring", 42, false] |