Масиви

План

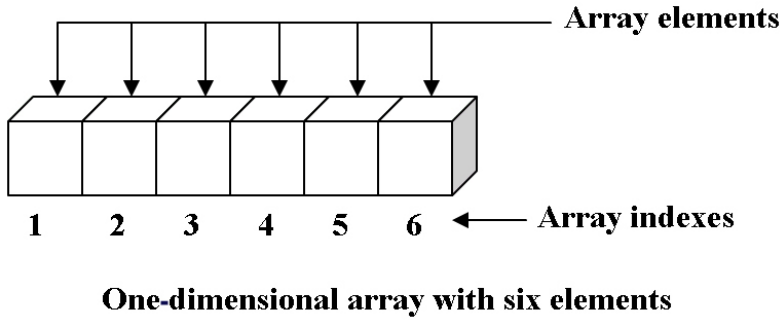
- Оголошення масивів
- Методи додавання і видалення елементів
- Багатовимірні масиви
- Методи роботи з масивами
- Сортування масивів
- Перебираючі методи

Оголошення масивів

- упорядкований набір значень, до якого ви посилаєтеся по імені і індексу

```
let array = [‘html’, ‘css’, ‘javascript’];
array[0] // ‘html’;
array[1] // ‘css’;
array[2] // ‘javascript’;

array[3] = ‘jquery’;
```



One-dimensional array with six elements

https://learn.javascript.ru/array
https://www.w3schools.com/js/js_arrays.asp

Оголошення за допомогою new Array()

```
let array = new Array('html', 'css', 'javascript');
array[0] // 'html';

let arrayNew = new Array(5);
```

Створює масив без елементів, але довжиною 5

```
arrayNew[0]; // undefined
```

Довжина масивів length

- не кількість елементів масиву, а останній індекс + 1

```
let arr = [];
arr[1000] = true;
arr.length; // 1001
```

- При зменшенні length масив коротшає

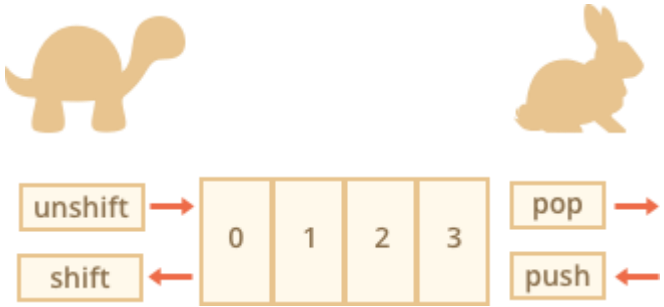
```
let arr = [1, 2, 3, 4, 5];
arr.length = 2; // вкоротити до 2 елементов
alert( arr ); // [1, 2]
```

Методи додавання і видалення елементів

- `pop()` - видаляє останній елемент з масиву і повертає його
- `push(el)` - додає елемент в кінець масиву
- `shift()` - видаляє з масиву перший елемент і повертає його
- `unshift(el)` - додає елемент на початок масиву

Швидкодія методів

- Методи `push / pop` виконуються швидко, а `shift / unshift` - повільно



Приклади

```
let arr = [‘html’, ‘css’, ‘javascript’];
arr.pop(); // ‘javascript’
alert(arr); // [‘html’, ‘css’]


arr.push(‘jquery’);
alert(arr); // [‘html’, ‘css’, ‘javascript’, ‘jquery’]


arr.shift();
alert(arr); // [‘css’, ‘javascript’, ‘jquery’]


arr.unshift(‘bootstrap’);
alert(arr); // [‘bootstrap’, ‘css’, ‘javascript’, ‘jquery’]
```

Багатовимірні масиви

- масив може містити в собі інші масиви

```
let matrix = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
];

alert( matrix[1][1] ); 5
```

Практика

- додавати елементи в масив через метод prompt() поки користував ненатисне cancel
- знайти суму елементів двовимірного масиву

Метод split()

- `split(s)` - дозволяє перетворити рядок в масив, розбивши її по розділювачу s

```
let names = 'html, css, javascript';

let arr = names.split(', ');
alert(arr); //  ['html’, ‘css’, ‘javascript']
```

Метод join()

- `arr.join (str)` робить в точності протилежне split. Він бере масив і склеює його в рядок, використовуючи str як роздільник

```
let arr = ['html’, ‘css’, ‘javascript'];

let str = names.join(', ');
alert(str); //  'html, css, javascript’
```

Практика

- розбити строку lorem по розділювачу ‘ ’ і вивести в консоль слова довжиною більше 5

Метод splice()

- універсальний метод для роботи з масивами. Вміє все: видаляти елементи, вставляти елементи, замінювати елементи

```
let arr = [‘html’, ‘css’, ‘javascript’, ‘jquery’];
arr.splice(1, 1); // починаючи з індексу 1 видалити 1 елемент
alert(arr); // [‘html’, ‘javascript’, ‘jquery’];


arr.splice(1, 2, ‘ajax’, ‘json’); // починаючи з 1 видалити 2 елемент і вставити 2 ел
alert(arr); // [‘html’, ‘ajax’, ‘json’ ‘javascript’, ‘jquery’];


arr.splice(1, 0, ‘scss’); // просто вставляє елемент елемент після індекса 2
alert(arr); // [‘html’, ‘ajax’, ‘scss’, ‘json’ ‘javascript’, ‘jquery’];
```

Метод slice()

- `slice (begin, end)` копіює ділянку масиву від `begin` до `end`, не включаючи `end`. Вихідний масив при цьому не змінюється

```
let arr = [‘html’, ‘css’, ‘javascript’, ‘jquery’];
let arr2 = arr.slice(1, 3); // копіює з індексу 1 до 3 елементи
alert(arr2); // [css, ‘javascript’];
```

- якщо не вказати end - копіювання буде до кінця масиву:

```
let arr2 = arr.slice(1); // копіює з індексу 1 до кінця масиву
alert(arr2); // [‘css’, ‘javascript’, ‘jquery’];
```

Практика

- дано масив [1, 9, 22, 7, 6] додати число 8 після 22

Сортування масивів

- метод sort() - сортує масив

```
let arr = [ 1, 2, 15 ];
arr.sort();
alert( arr );  // 1, 15, 2
```

- за замовчуванням sort сортує, перетворюючи елементи в строки

Своє сортування

```
function compareNumeric(a, b) {  // порівнюється два значення
  if (a > b) return 1;              // якщо перше значення більше другого то ф-я повертає додатнє
число
  if (a == b) return 0;             // якщо рівні то ф-я повертає нуль
  if (a < b) return -1;             // якщо перше значення менше другого то ф-я повертає відємне
число
}
```

від зміни оператора порівняння буде залежати напрям сортування

```
let arr = [ 1, 2, 15 ];

arr.sort(compareNumeric);

alert(arr);  // 1, 2, 15
```

Функція реверсу reverse()

- змінює порядок елементів в масиві на зворотний

```
let arr = [1, 2, 3];
arr.reverse();

alert( arr ); // 3,2,1
```

З’єднання кількох масивів в один

- Метод `arr.concat (value1, value2, ... valueN)` створює новий масив, в який копіюються елементи з arr, а також value1, value2, ... valueN

```
let arr = [1, 2];
let newArr = arr.concat(3, 4);

alert( newArr ); // 1,2,3,4
```

Пошук в масивові

- Метод «arr.indexOf (searchElement [, fromIndex])» повертає номер елемента searchElement в масиві arr або -1, якщо його немає.
- Пошук починається з номера fromIndex, якщо він вказаний. Якщо немає - з початку масиву.

```
let arr = [1, 0, false];

alert( arr.indexOf(0) ); // 1
alert( arr.indexOf(false) ); // 2
alert( arr.indexOf(null) ); // -1
```

Практика

- написати функцію randomNumber(number) яка наповнює масив випадковими числами від 0 до 100, де number - довжина масиву
- перевірити чи в масивові є число, яке вводиться через prompt()

Цикл for..of

- цикл по масивах
  https://goo.gl/d7c84b

```
for (змінна of масив) {
 // тіло циклу
}
```

Приклад

```
var arr = [3,5,7,9]
for (let i of arr) {
 console.log(i);
}
```

Перебираючі методи

- `forEach` - для перебору масиву
- `filter` - для фільтрації масиву
- `every / some` - для перевірки масиву
- `map` - для трансформації масиву в масив
- `reduce / reduceRight` - для проходу по масиву з обчисленням значення
- `find` - для пошуку в масиві з обєктів
- `findIndex` - повертає індекс для якого був знайдений елемент

forEach

```
arr.forEach(callback(item, i, arr) {});
```

item - черговий елемент масиву
i - його номер
arr - масив, який перебирається

```
let arr = [‘html’, ‘css’, ‘javascript’];
arr.forEach(function(item, i, arr) {
  alert( i + ": " + item + " (массив:" + arr + ")" );
});
```

filter

- створює новий масив, в який увійдуть тільки ті елементи arr, для яких виклик callback (item, i, arr) поверне true

```
let arr = [1, -1, 2, -2, 3];

let positiveArr = arr.filter(function(number) {
  return number > 0;
});

alert( positiveArr ); // 1,2,3

let positiveArr = arr.filter((number) => number > 0); // новіший синтаксис
```

map

- створює новий масив, який буде складатися з результатів виклику callback (item, i, arr) для кожного елемента arr

```
let names = ['HTML', 'CSS', 'JavaScript'];

let nameLengths = names.map(function(name) {
  return name.length;
});

alert( nameLengths ); // 4,3,10
```

find

- повертає перший елемент для якого виконується умова

```
let users = [
  {id: 1, name: "Вася"},
  {id: 2, name: "Петя"},
  {id: 3, name: "Маша"}
];
let user = users.find(function(item){ return item.id == 1});
alert(user.name); // Вася
```

findIndex

- майже такий же метод як find але повертає індекс елемента який знайдено, якщо не знайдено то повертає -1

```
let users = [
  {id: 1, name: "Вася"},
  {id: 2, name: "Петя"},
  {id: 3, name: "Маша"}
];
let id = users.findIndex(function(item){ return item.id == 1});
alert(id); // 1
```

every/some

- Метод «arr.every (callback [, thisArg])» повертає `true`, якщо виклик callback поверне true для `кожного елемента` arr.
- Метод «arr.some (callback [, thisArg])» повертає `true`, якщо виклик callback поверне true для `якого-небудь елементу` arr.

```
function isNumber(value){
  return typeof value == 'number';
}
let a1 = [1, 2, 3];
console.log(a1.every(isNumber)); // поверне true
let a2 = [1, '2', 3];
console.log(a2.every(isNumber)); // поверне false
```

```
let a1 = [1, 2, 3];
console.log(a1.some(isNumber)); // поверне true
let a2 = [1, '2', 3];
console.log(a2.some(isNumber)); // поверне true
let a3 = ['1', '2', '3'];
console.log(a3.some(isNumber)); // поверне false
```

reduce

- arr.reduce(previousValue, currentItem, index, arr) використовується для послідовної обробки кожного елемента масиву із збереженням проміжного результату
- previousValue - останній результат виклику функції, він же «проміжний результат»
- currentItem - поточний елемент масиву, елементи перебираються по черзі зліва-направо
- index - номер поточного елемента
- arr - оброблюваний масив

```
let arr = [1, 2, 3, 4, 5]

// для кожного елемента масиву запустити функцію,
// проміжний результат передавати першим аргументом далі
let result = arr.reduce(function(sum, current) {
  return sum + current;
}, 0); // 0 initialValue - початкове значення
alert( result ); // 15
```

- Метод `arr.reduceRight` працює аналогічно, але йде по масиву справа-наліво

Практика

- за допомогою методу forEach знайти найдовше слово в строці lorem
- за допомогою методу map створити масив квадратів масиву
  arr = [8, 6, 12, 10];

lodash

- бібліотека з методами для роботи з масивами і об’єктами

https://lodash.com/docs/

http://underscorejs.ru/

Посилання

https://learn.javascript.ru/array
https://learn.javascript.ru/array-methods методи масивів
https://learn.javascript.ru/array-iteration перебираючі методи

Відео

https://www.youtube.com/playlist?list=PLM7wFzahDYnEcQh1G_fxFXBZ4O1tlVsDG