JS Fundamentals
Array methods

•••

Автор презентации: Макухина Марина

Функция обратного вызова

Функция обратного вызова (callback) — это функция, переданная в другую функцию в качестве аргумента, которая вызывается по завершению какого-либо действия.

```
function getUpdatedArray(values, fn) {
  let result = [];
  for (let value of values) {
    if (fn(value)) result.push(value);
  }
  return result;
}
function isPositiveNumber(value) {
  return typeof value === "number" && value > 0;
}
getUpdatedArray([0, 11, 5, -2, -4], isPositiveNumber); // [11, 5]
```

Методы массива: forEach()

Метод *forEach()* принимает *callback* функцию, которая будет вызвана для каждого элемента массива. Она принимает от одного до трех аргументов:

- *currentValue* — текущий обрабатываемый элемент в массиве;
- *index* (необязательный) — индекс текущего обрабатываемого элемента массива;
- *array* (необязательный) — массив, по которому осуществляется проход.

```
let numbers = [1, 3, 5];
numbers.forEach(function (currentValue, index, array) {
  console.log(currentValue);
}); // 1 3 5
```

Методы массива: find()

Метод *find()* возвращает значение первого найденного в массиве элемента, которое удовлетворяет условию переданному в *callback* функции или *undefined*, если такого элемента нет. Метод *find()* не изменяет массив, для которого он был вызван.

```
let numbers = [1, 3, 5];
let result = numbers.find( function (currentValue, index, array) {
  return currentValue > 1;
});
console.log(result); // 3
```

Подобно методу *forEach()*, *find()* принимает *callback* функцию с тремя аргументами: *currentValue*, *index* и *array*.

Методы массива: filter()

Метод *find()* вернет первый попавшийся элемент, которые удовлетворит условие. Чтобы найти все элементы, необходимо использовать метод *filter()*. Этот метод вернет новый массив с элементами, которые удовлетворяют условию переданному в *callback* функции. Если ни один элемент не пройдет проверку — метод вернет пустой массив.

```
let numbers = [1, 3, 5];
let newNumbers = numbers.filter( function (currentValue, index, array) {
  return currentValue > 1;
});
console.log(newNumbers);  // [3 ,5]
```

Методы массива: sort()

Метод *sort()* на месте сортирует элементы массива и возвращает отсортированный массив. Порядок сортировки по умолчанию соответствует порядку кодовых точек Unicode. Если предоставить методу *sort()* функцию сравнения *compareFunction*, элементы массива отсортируются в соответствии с ее возвращаемым значением.

```
[5, 21, 2, 25, 200, 10].sort(function(a, b) {
  return a - b;
}); // [ 2, 5, 10, 21, 25, 200 ]
```

Если *compareFunction* вернет значение меньше 0, сортировка поставит переданный в качестве первого аргумента элемент по меньшему индексу, чем второй. Если compareFunction вернет значение больше 0, сортировка поставит второй аргумент по меньшему индексу, чем первый.

Методы массива: map()

Метод *map()* создает новый массив с результатом вызова указанной функции для каждого элемента массива. Обратите внимания, что этот метод не изменяет массив, для которого он был вызван.

```
let numbers = [1, 3, 5];
let cubed = numbers.map( function (currentValue) {
  return Math.pow(currentValue, 3);
});
console.log(cubed); // [1, 27, 125]
```

Метод *map()* также принимает *callback* функцию с тремя аргументами: *currentValue*, *index* и *array*. Обратите внимания, что можно в аргументах функции написать только *currentValue*, если другие аргументы не будут использоваться.

Методы массива: reduce()

В то время как *map()* преобразует каждый элемент в массиве, метод *reduce()* преобразует весь массив. Этот метод используется для сведения (reduce) массива к единому значению. Принимает 2 параметра:

1) функцию, выполняющеюся для каждого элемента массива с аргументами:
   - *accumulator* — результат предыдущего вызова этой функции;
   - *currentValue* — текущий обрабатываемый элемент массива;
   - *index* (необязательный) — индекс текущего обрабатываемого элемента массива;
   - *array* (необязательный) — для которого была вызвана функция *reduce()*.
2) initial (необязательный) — объект. Если этот параметр передан, то при первом вызове функции *accumulator* будет равен этому значению.

Методы массива: reduce()

Рассмотрим пример. Для первого элемента массива вызывается функция. Первоначально, accumulator имеет значение 0, а *currentValue* — значение 1. Функция возвращает сумму этих двух значений (0+1), что становится значением *accumulator* на следующем этапе.

На следующем шаге *currentValue* равняется значению 3, а *accumulator* будет 1. Произойдет сложение этих чисел и переход к следующему шагу. И так далее.

```
let numbers = [1, 3, 5];
for (let currentValue of numbers) {
  result += currentValue;
}
console.log(result); // 9
```

```
let result = numbers.reduce(
function (accumulator, currentValue) {
  return accumulator + currentValue;
}, 0);
console.log(result); // 9
```

Стрелочные функции

Стрелочные функции – это более простой и краткий синтаксис для создания функций. Всегда являются анонимными. Кроме того, для стрелочных функций недоступна специальная переменная *arguments*.

```
let getSum = (a, b)  => a + b;
console.log(getSum( 2, 4)); // 6
```

Стрелочные функции позволяют упростить синтаксис тремя способами:

- Опустить слово *function*;
- Если функции передается один аргумент, опустить круглые скобки;
- Если тело функции — одно выражение, опустить фигурные скобки и *return*.

Пример использования стрелочных функций

При использовании методов массива, которые принимаю функции, предпочтительно использовать в качестве аргумента стрелочные функции.

```
let numbers = [1, 3, 5];
numbers.forEach(currentValue => console.log(currentValue));   // 1 3 5
```

```
console.log(numbers.find(el  => el > 1)); // 3
```

```
console.log(numbers.map(value  => Math.pow(value,  3))); // [1, 27, 125]
```

```
console.log(numbers.reduce((a, v)  => a + v)); // 9
```

Полезные ссылки

https://learn.javascript.ru/arrow-functions-basics

https://learn.javascript.ru/array-methods

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/find

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/map

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce

На этом всё!

...

Спасибо за внимание