Методы в массивах и объектах

`every` , `some`

Методы массива длят условий: `Array#every`

- `Array#every`
  - Сигнатура: `[].every(callback);`
  - Колбэк: `callback(item [, index [,arr]])`
  - Возвращает: `Boolean`
  - Поведение: возвращает `true` если **все** если все элементы массива удовлетворяют критериям `callback()`
    - Возвращает `false` если **какой-либо** из методов не удовлетворяет условию переданному в `callback()`
  - Поддерживается: везде

Условия: `Array#every`

- Проверить все элементы на четность?

```
function isOdd(number) {
  return !!(number % 2);
}
console.log([1, 2, 3, 4].every(isOdd)); //false
console.log([1, 3, 5, 7].every(isOdd)); //true
```

- Проверка всех чисел на условие того, что они больше 18

```
function isGreaterThan18(number) {
  return number > 18;
}

console.log([22, 23].every(isGreaterThan18)); //true
console.log([19, 18].every(isGreaterThan18)); //false
```

Методы массивов для условий: `Array#some`

- `Array#some`
  - Сигнатура: `[].some(callback);`
  - Колбэк: `callback(item [, index [,arr]])`
  - Возвращает: `Boolean`
  - Поведение: возвращает `true` если **какой-либо** удовлетворяет критериям указанным в `callback()`
    - -Возвращает `false` если не один из элементов не удовлетворяет усвлоиям `callback()`
  - Поддержка везде

Условия: `Array#some`

- Проверка массива на наличие хотя бы одного четного числа

```
function isOdd(number) {
  return !!(number % 2);
}
console.log([1, 2, 3, 4].some(isOdd)); //true
console.log([1, 3, 5, 7].some(isOdd)); //true
console.log([2, 4, 6, 8].some(isOdd)); //false
```

- Хотя бы одно число больше 18

```
function isGreaterThan18(number) {
  return number > 18;
}
console.log([22, 23].some(isGreaterThan18)); //true
console.log([19, 18].some(isGreaterThan18)); //true
console.log([17, 18].some(isGreaterThan18)); //false
```

Методы трасформации массивов `map`, `reduce`, `filter`

Метод `Array#filter`

- `Array#filter`
  - Сигнатура: `[].filter(callback);`
  - Колбэк: `callback(item [, index [, arr]])`
  - Возвращает: `Array`
  - Поведение: **извлекает** из массива элементы которые **удовлетворяют условию** в `callback()`
    - Возвращает **пустой массив**, если нет элементов удовлетворяющих критериям.
  - Поддержка: везде

Трансформация: Array#filter

- Извлечь четные элементы из массива

```
function isOdd(number) {
  return !!(number % 2);
}
console.log([1, 2, 3, 4].filter(isOdd)); // [1, 3]
console.log([1, 3, 5, 7].filter(isOdd)); // [1, 3, 5, 7]
console.log([2, 4, 6, 8].filter(isOdd)); // []
```

- Возвращает элементы водящие в диапазон

```
function InRange(min, max) {
  return function(item) { return min <= item && item <= max; };
}
var numbers = [2, 3, 4, 5, 6, 7, 8];
console.log(numbers.filter(inRange(4, 7))); // [4, 5, 6, 7]
console.log(numbers.filter(inRange(2, 4))); // [2, 3, 4]
```

Методы трансформации массивов: `Array#reduce`

- `Array#reduce`
  - Сигнатура: `[].reduce(callback, initial);`
  - Колбэк: `callback(accumulator, item [, index [, arr]])`
  - Возвращает: `Object`
  - Поведение: возвращает **объект**, с результатами `callback()`
  - Поддержка: везде

Трансформация: Array#reduce

- Вычислить сумму и произведение элементов

```
var sum = [1, 2, 3, 4].reduce(function(sum, number) {
  return sum + number;
}, 0);
var product = [1, 2, 3, 4].reduce(function(sum, number) {
  return sum * number;
}, 1);
console.log(sum);               //10
console.log(product);           //24
```

Методы трансформации массивов: `Array#map`

- `Array#map`
  - Сигнатура: `[].map(callback);`
  - Колбэк: `callback(item [, index [, arr]])`
  - Возвращает: `Object`
  - Порведение: возвращается новый массив с таким же размером. каждый элемент преобразуется в соответствии с `callback()`
  - Поддержка: везде

Трансформация: `Array#map`

- Вычисляет квадрат каждого из чисел

```
var squares = [1, 2, 3, 4, 5].map(function(number) {
  return number * number;
});
console.log(squares); //prints [ 1, 4, 9, 16, 25 ]
```

- Трансформирует матрицу заданную в массиве строк, в массив массивов

```
var lines = ['1 2 3',
             '4 5 6'];
var matrix = lines.map(function(line) {
  return line.split(' ')
             .map(Number);
});                                          // [[1, 2, 3],
console.dir(matrix);                         //  [4, 5, 6]]
```

Итераторы массивов

Итератор: `Array#forEach`

- `Array#forEach`
  - Сигнатура: `[].forEach(callback);`
  - Колбэк: `callback(item [, index [, arr]])`
  - Возвращает: `undefined`
  - Поведение: **проходит по элементами** и применяет к каждому элементу функцию callback
    - Очень похоже на цикл for-of где функцией колбеком является блок цикла.
  - Поддержка: везде

Итератор: `Array#forEach`

- Печатает элементы вместе с их индексом

```
var numbers = ['One', 'Two', 'Three', 'Four', 'Five'];
numbers.forEach(function(item, index) {
  console.log('Item at' + index + 'has value' + item);
});
```

- Вызывает метод для каждом элементе в массиве

```
function createPerson(name, age) { //… }
var people = [createPerson('Peter', 13),
              createPerson('John', 18),
              createPerson('Susan', 21)];
people.forEach(function(person) {
   person.introduce();
});
}
```

Метды для поиска по массиву `find`, `findIndex`

Поиск: `Array#find`

- `Array#find`
  - Сигнатура: `[].find(callback);`
  - Колбэк: `callback(item [, index [, arr]])`
  - Возвращает: `Object` or `undefined`
  - Поведение: возвращает **крайний левый элемент** в массиве, который **удовлетворяет критериям** в `callback()`
    - И **нет такого элемента** возвращается `undefined`
  - Поддержка: Почти нигде, необходим полифил - класс, имплементирующий поддержку функционала

Поиск: `Array#find`

- Ищет крайнее левое нечетное число, большее 5

```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
console.log(numbers.find(function(item) {
  return !!(item % 2) && item > 5;
}));            //prints 7
```

- Ищет крайнее левое нечетное число с индексом больше 3

```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
console.log(numbers.find(function(item, index) {
  return index > 3 && !!(item % 2);
}));            //prints 5
```

Поиск по массиву: `Array#findIndex`

- `Array#findIndex`
  - Сигнатура: `[].findIndex(callback);`
  - Коллбэк: `callback(item [, index [, arr]])`
  - Возвращает: `Number` or `-1`
  - Поведение: возвращает **индекс крайнего левого** элемента в массиве, который **удловлеторяет критерию** в `callback`
    - Если **элемент не найде** возвращает `-1`
  - Поддержка: Почти нигде, необходим полифил - класс, имплементирующий поддержку функционала

Поиск: Array#findIndex

- Найти индекс крайнего левого элемента большего 5

```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
console.log(numbers.findIndex(function(item) {
  return !!(item % 2) && item > 5;
}));            //prints 6(element 7)
```

- Поиск индекса крайнего левого элмента с индексом большим 3

```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
console.log(numbers.findIndex(function(item, index) {
  return index > 3 && !!(item % 2);
}));            //prints 4(element 5)
```

Другие методы массива `sort`, `fill`

Методы: `Array#sort`

- `Array#sort`
  - Сигнатура: `[].sort(callback);`
  - Коллбэк: `callback(obj1, obj2)`
  - Возвращает: `Array`
  - Поведение: **сортирует элементы** в массиве в соответствии с `callback()`
  - Поддержка: везде

`Array#sort`

- Сортирует массив в обратном порядке

```
var numbers = [5, 1, 2, 4, 6];
numbers.sort(function(x, y) {
  return y - x;
});
console.log(numbers); //[ 6, 5, 4, 2, 1 ]
```

- Сотирует массив людей по имени

```
var people = [createPerson('Peter', 13), createPerson('John', 18), ..];
people.sort(function(p1, p2) {
  return p1.name > p2.name;
});
console.log(people);  // John, Peter, Susan
```

Методы: `Array#fill`

- `Array#fill`
  - Сигнатура: `[].fill(value [, from [, to]])`
  - Возвращает: `Array`
  - Поведение: **заполняет массив** переданными значениями
  - Поддержка: Почти нигде, необходим полифил - класс, имплементирующий поддержку функционала

`Array#fill`

- Заполняет массив единицами

```
var count = 15,
  arr = Array.from({length: count})
    .fill(1);
console.log(arr);
```

- Создает массив массивов

```
var count = 5,
  arr = Array.from({length: count}) // [ [ 1, 2, 3, 4, 5 ],
    .fill([1, 2, 3, 4, 5]);         // [ 1, 2, 3, 4, 5 ],
                                    // [ 1, 2, 3, 4, 5 ],
arr.fill([1, 2, 3, 4, 5]);          // [ 1, 2, 3, 4, 5 ],
console.log(arr);                   // [ 1, 2, 3, 4, 5 ] ]
```

Цепочки методов

Что такое цепочки?

- Цепочки это шаблон для вызова методов в функциональном программировании
  - Каждый метод возвращает объект
    - Другой метод может вызан по этому объекту
    - и.т.д…
- Для того, чтобы вызывать цепочки, следуйте следующим правилам:
  - Если метод должен возвращать результат -> верните его
    - Иначе, верите объект для того, чтобы продолжить цепочку

Цепочки методов массива

- Большинство методов массива возвращают результат
  i. Заполнить массив случайных чисел
  ii. Удалить все четные цифры
  iii. Вернуть массив с именами остальных цифр

```
var n = 10;
function getRandomDigit() { return (Math.random() * 10) | 0; }
function isOdd(item) { return !!(item % 2); }
function digitToDigitName(digit) {
  return ['Zero', 'One', /* … */, 'Nine'][digit];
}
  var digitNames = Array.from({length: n})
    .fill(0)
    .map(getRandomDigit)
    .filter(isOdd)
    .map(digitToDigitName);
  console.log(digitNames);
```

- Используя ES 2015

```
var digitNames = digits.fill(0)
  .map(x => Math.random()*10 | 0)
  .filter(x=> !!(x%2))
  .map(x=> […][x]);
console.log(digitNames);
```