**Mutating Arrays**

Напоминаю, что мутация чего-либо - это всегда плохо и её стоит избегать.

Однако, некоторые рабочие задачи мы будем вынуждены решать через мутации.

Сегодня рассмотрим методы мутации массивов.

**fill()**

Метод заполняет массив одинаковыми знаниями.

У метода есть 3 аргумента:

* value - значение, заполняющее массив
* start - начальный индекс
* end - конечный индекс

start и end являются необязательными аргументами и их можно опускать.

Используя аргументы start и end, мы можем заполнять значениями какой-то определенный промежуток элементов.

Пример:

const array = new Array(10).fill(0, 5, 7)

console.log(array) //отобразит массив [,,,,,0,0,,,]

Как мы можем видеть, наш массив заполнен элементами типа undefined и только элементы с индексами 5 и 6 получили значения 0.

**copyWithin()**

Метод заполняем массив его же элементами.

У метода есть 3 аргумента:

* target - начальный индекс цели перезаписи элементов
* start - начальный индекс источника копируемых элементов
* end - конечный индекс источника копируемых элементов

start и end являются необязательными аргументами и их можно опускать. При этом в массиве будут перезаписываться элементы от индекса из аргумента target и до конца массива.

const array = new Array(10).fill(0).map((it,index) => index)

array.copyWithin(4, 6, 7)

console.log(array) //отобразит массив [0,1,2,3,6,5,6,7,8,9]

Как мы видим, нам не нужно присваивать массив другой переменной, так как в любом случае мы мутируем изначальный массив.

**unshift() и push()**

Данные методы **добавляют** 1 элемент в массив с начала и с конца, соответственно:

const array = new Array(10).fill(0).map((it,index) => index)

array.unshift(-1)

console.log(array) //отобразит массив [-1,0,1,2,3,4,5,6,7,8,9]

array.push(10)

console.log(array) //отобразит массив [-1,0,1,2,3,4,5,6,7,8,9,10]

shift() и pop()

Данные методы удаляют 1 элемент из массива с начала и с конца, соответственно:

const array = new Array(10).fill(0).map((it,index) => index)

array.shift()

console.log(array) //отобразит массив [1,2,3,4,5,6,7,8,9]

array.pop()

console.log(array) //отобразит массив [1,2,3,4,5,6,7,8]

Удаленные элементы мы можем положить в переменные:

const arr = [1,2,3]

const firstElem = arr.shift()

const lastElem = arr.pop()

console.log(firstElem, lastElem) //отобразит 1 3

**reverse()**

Обращает порядок следования элементов в массиве. Первый элемент массива становится последним, а последний — первым.

const arr = [1,2,3]

arr.reverse()

console.log(arr) //отобразит массив [3,2,1]

**sort()**

Сортирует элементы массива по правилу кодовых точек Unicode. То есть, сортировка числового массива будет выстраивать элементы следующим образом:

const array = new Array(12).fill(0).map((it,index) => index)

array.sort()

console.log(array) //отобразит [0,1,10,11,2,3,4,5,6,7,8,9]

У метода sort() в аргументе может быть функция сравнения, которая будет задавать правила сортировки

const array = new Array(12).fill(0).map((it,index) => index)

array.sort((a, b) => a - b) //сортировка числовых элементов по возрастанию

console.log(array)

array.sort((a, b) => Math.random() - 0.5) //сортировка элементов в случайном порядке

console.log(array)

*Math.random() - возвращает случайное значение от 0 до 1*

**Сортировка массива строк**

const arrayStrings = ['abra', 'cadabra', 'acadraba', 'bcabra']

arrayStrings.sort((a, b) => a.localeCompare(b))

console.log(arrayStrings) //отобразит массив отсортированный в алфавитном порядке

// ["abra","acadraba","bcabra","cadabra"]

arrayStrings.sort((a, b) => -1 \* a.localeCompare(b))

console.log(arrayStrings) //отобразит массив отсортированный в обратном алфавиту порядке

// ["cadabra","bcabra","acadraba","abra"]

**splice()**

Метод удаляет элементы массива и может добавить на их место новые элементы.

splice() имеет 3 аргумента:

* start начальный индекс удаления элементов
* deleteCount количество удаляемых элементов
* itemN добавляемые элементы вместо удалённых

const array = [1,2,3,4,5]

array.splice(2, 2, 30, 40, 50)

console.log(array) //отобразит [1,2,30,40,50,5]