Методы массивов

- Презентация: ссылка

Метод map()

узнать больше

создаёт новый массив с результатом вызова указанной функции для каждого элемента массива.

Синтаксис (в квадратных скобках необязательные параметры)

```
const new_array = arr.map(function callback( currentValue[, index[, array]]) {
    // Возвращает элемент для new_array
}[, thisArg])
```

Параметры

- **callback** Функция, вызываемая для каждого элемента массива arr. Каждый раз, когда callback выполняется, возвращаемое значение добавляется в new_array.
- **thisArg** - Необязательный параметр. Значение, используемое в качестве this при вызове функции callback Функция callback, создающая элемент в новом массиве, принимает три аргумента:
- **currentValue** - Текущий обрабатываемый элемент массива.
- **index** Необязательный - Индекс текущего обрабатываемого элемента в массиве.
- **array** Необязательный - Массив, по которому осуществляется проход.

Возвращаемое значение Новый массив, где каждый элемент является результатом callback функции.

Array.prototype.reduce()

узнать больше

Метод reduce() применяет функцию reducer к каждому элементу массива (слева-направо), возвращая одно результирующее значение.

Синтаксис

```
array.reduce(callback[, initialValue])
```

- **callback** - Функция, выполняющаяся для каждого элемента массива, принимает четыре аргумента:
    - **accumulator** - Аккумулятор, аккумулирующий значение, которое возвращает функция callback после посещения очередного элемента, либо значение initialValue, если оно предоставлено (смотрите пояснения ниже).
    - **currentValue** - Текущий обрабатываемый элемент массива.
    - **index Необязательный** - Индекс текущего обрабатываемого элемента массива.
    - **array****Необязательный** - Массив, для которого была вызвана функция reduce.
- **initialValue** Необязательный параметр. Объект, используемый в качестве первого аргумента при первом вызове функции callback.

Array.prototype.forEach()

узнать больше

Метод forEach() выполняет указанную функцию один раз для каждого элемента в массиве.

Синтаксис

```
arr.forEach(function callback(currentValue, index, array) {
//your iterator
}[, thisArg]);
```

Параметры

- callback Функция, которая будет вызвана для каждого элемента массива. Она принимает от одного до трёх аргументов:
    - currentValue

Текущий обрабатываемый элемент в массиве.

- index Необязательный Индекс текущего обрабатываемого элемента в массиве.
- array Необязательный Массив, по которому осуществляется проход.
- thisArg Необязательный параметр. Значение, используемое в качестве this при вызове функции callback.

Дополнительная информация:

- filter
- every
- some

Array methods Cheat Sheet

```
// 1. concat()
const fruits = ['🍉', '🍎', '🍒', '🍌'];
const vegetables = ['🍆', '🌽', '🥕', '🥑'];

const food = fruits.concat(vegetables); // ["🍉", "🍎", "🍒", "🍌", "🍆", "🌽", "🥕", "🥑"]

// 2. copyWithin()
const fruits = ['🍉', '🍎', '🍒', '🍌'];

const fruitsCopy = fruits.copyWithin(0, 2); // ["🍒", "🍌", "🍒", "🍌"]

// 3. every()
const fruits = ['🍉', '🍎', '🍒', '🍌'];

const allBananas = fruits.every(fruit => fruit === '🍌'); // false

// 4. fill()
const fruits = ['🍉', '🍎', '🍒', '🍌'];

const lemons =  fruits.fill('🍋'); // ["🍋", "🍋", "🍋", "🍋"]

// 5. filter()
const fruits = ['🍉', '🍎', '🍒', '🍌'];

const onlyBananas = ['🍉', '🍎', '🍌', '🍌'].filter(fruit => fruit === '🍌'); // ["🍌", "🍌"]

// 6. find()
const fruits = ['🍉', '🍎', '🍒', '🍌'];

const cherry = fruits.find(fruit => fruit === '🍒'); // "🍒"

// 7. findIndex()
const fruits = ['🍉', '🍎', '🍒', '🍌'];

const cherryIndex = fruits.findIndex(fruit => fruit === '🍒'); // 2

// 8. forEach()
const vegetables = ['🍆', '🌽', '🥕', '🥑'];

vegetables.forEach(vegetable => console.log(vegetable));
// "🍆"
// "🌽"
// "🥕"
// "🥑"

// 9. includes()
const vegetables = ['🍆', '🌽', '🥕', '🥑'];

const includesCorn = vegetables.includes('🌽'); // true
const includesTomato = vegetables.includes('🍅'); // false

// 10. join()
const vegetables = ['🍆', '🌽', '🥕', '🥑'];

const vegetablesGroup = vegetables.join(''); // "🍆🌽🥕🥑"

// 11. map()
const vegetables = ['🍆', '🌽', '🥕', '🥑'];

const doubledVegetables = vegetables.map(vegetable => vegetable + vegetable);
// ["🍆🍆", "🌽🌽", "🥕🥕", "🥑🥑"]

// 12. push()
const fruits = ['🍉', '🍎', '🍒', '🍌'];

fruits.push('🥝'); // ["🍉", "🍎", "🍒", "🍌", "🥝"]

// 13. reverse()
const fruits = ['🍉', '🍎', '🍒', '🍌'];

const reversedFruits = fruits.reverse(); // ["🍌", "🍒", "🍎", "🍉"]

// 14. slice()
fruits.slice(2); // ["🍒", "🍌"]

// 15. some()
```