Front-end

JavaScript

JS. Arrays

- Массивы
- Создание массивов
- Ссылочный тип данных
- Проверка на массив
- Удаление элементов
- Методы массивов
- Использование циклов. Копирование массива

JS. Arrays

Массив - это коллекция значений. Значения в массиве называются элементами, и каждый элемент характеризуется числовой позицией в массиве, которая называется индексом.

Массивы в языке JavaScript являются не типизированными: элементы массива могут иметь любой тип, причем разные элементы одного и того же массива могут иметь разные типы. Элементы массива могут даже быть объектами или другими массивами, что позволяет создавать сложные структуры данных, такие как массивы объектов и массивы массивов.

Отсчет индексов массивов в языке JavaScript начинается с нуля - первый элемент массива имеет индекс 0.

Массивы в JavaScript являются динамическими: они могут увеличиваться и уменьшаться в размерах по мере необходимости;

JS. Arrays

Создание массива

Можно создать пустой массив, используя квадратные скобки (использование литерала массива) или конструктор Array:

```
const users = new Array();
const people = [];
```

Еще один способ инициализации массивов представляет метод Array.of() - он принимает элементы и инициизирует ими массив:

```
const people = Array.of("Tom", "Bob", "Sam");
```

И еще один способ представляет функция Array.from().

```
const array = Array.from("Hello");
console.log(array); // ["H", "e", "l", "l", "o"]
```

JS. Arrays

Ссылочный тип данных

Переменные (и константы) в JavaScript могут хранить два вида данных: примитивные и ссылочные. К примитивным относятся все примитивные типы: числа, строки, булеан и т.д. К ссылочным – объекты.

Массив внутри – это объект:

```
typeof []; // =>  'object'
```

JS. Arrays

Ссылочный тип данных

Массив можно менять, даже если он записан в константу. Здесь как раз и проявляется ссылочная природа. Константа хранит ссылку на массив, а не сам массив, и эта ссылка не меняется. А вот массив поменяться может. С примитивными типами такой трюк не пройдет.

Сравнение массивов тоже происходит по ссылке. Это может быть очень неожиданно с непривычки. Одинаковые по структуре массивы имеют разные ссылки и не равны друг другу:

```
[1,2,3] === [1,2,3]; // => false
```

JS. Arrays

Проверка на массив

Время от времени необходимо проверить, хранится в переменной массив или что-то другое. Так как массивы не являются в JavaScript отдельным типом, то проверка с помощью typeof не подойдёт:

```
const a = []
console.log(typeof a)
// 'object'
```

JS. Arrays

Проверка на массив

Статический метод Array.isArray() проверяет, является ли переданный аргумент массивом.

Array.isArray() принимает один аргумент — переменную или значение, которое вы хотите проверить.

Возвращает true, если в переменной хранится массив и false во всех остальных случаях.

JS. Arrays

Удаление элемента из массива

Оператор delete удаляет свойство из объекта

Когда с помощью оператора delete удаляется элемент массива, длина массива не меняется. Например, если вы удалите a[3], a[4] по прежнему a[4], а a[3] не определено. Так будет даже если вы удалите последний элемент массива (delete a[a.length-1]).

Когда оператор delete удаляет элемент массива, этот элемент больше не существует в массиве.

```
delete arr[1];

console.log(arr[1])
```

JS. Arrays

- split - метод строки для преобразования в массив
- push - добавляет новый элемент в конце
- pop - удаляет последний элемент
- shift - удаляет первый элемент
- unshift - добавляет новый элемент в начале
- concat - создает новый массив путем слияния
- join - объединяет все элементы в строку
- slice – вырезает часть массива
- indexOf - возвращает индекс искомого элемента
- includes - определяет, содержит ли массив определённый элемент
- reverse - обращает порядок следования элементов (переворачивает)
- at - возвращает элемент массива по индексу
- fill - заполняет все элементы массива одинаковым значением
- splice - универсальный, умеет всё, удалять, изменять, добавлять

JS. Arrays

split

Метод split() используется для разбиения строки на массив подстрок и возвращает новый массив.

Если в качестве разделителя используется пустая строка (""), то строка разделяется между каждым символом.

Примечание: split() - метод не изменяет исходную строку.

JS. Arrays

split

Синтаксис

str.split(separator, limit)

separator - условие по которому будет разделена строка. В качестве разделителя может выступать строковый литерал или регулярное выражение. Также можно использовать специальные символы, например перевод строки, кавычки или юникод. Параметр является необязательным.

limit - количество элементов, которые должен вернуть метод. Параметр необязательный, если пропустить в массив попадут все подстроки. Если задать, то split() все-равно разделит всю строку по разделителям, но возвратит только указанное количество.

При нахождении разделителя метод удаляет separator и возвращает подстроку.

Если задать в качестве разделителя пустое значение '', тогда каждый символ строки запишется в отдельный элемент массива.

Если при записи split() пропустить separator, то метод вернет массив с одним элементом, который будет содержать всю строку.

JS. Arrays

push

Метод push() добавляет один или более элементов в конец массива и возвращает новую длину массива.

Метод push() изменяет исходный массив, а не создаёт его модифицированную копию.

Синтаксис

arr.push(element1, ..., elementN);

Аргументы

elementN: элементы, добавляемые в конец массива. Элементы добавляются в том порядке, в котором они были переданы методу.

Возвращаемое значение

Новое значение свойства length того массива, для которого был вызван метод.

JS. Arrays

pop

Метод pop() удаляет последний элемент массива, уменьшает длину массива и возвращает удалённое им значение.

Метод pop() изменяет исходный массив, а не создаёт его модифицированную копию.

Синтаксис

```
arr.pop();
```

Аргументы - нет.

Возвращаемое значение

Удалённое значение или undefined, если массив пуст.

JS. Arrays

shift

Метод shift() удаляет первый элемент массива, уменьшает индекс всех последующих элементов на единицу, уменьшает длину массива и возвращает удалённое им значение.

Метод shift() изменяет исходный массив, а не создаёт его модифицированную копию.

Синтаксис

arr.shift();

Аргументы - нет.

Возвращаемое значение

Удалённое значение или undefined, если массив пуст.

JS. Arrays

unshift

Метод unshift() добавляет один или более элементов в начало массива и возвращает новую длину массива.

Индексы всех элементов, изначально присутствующих в массиве, увеличиваются на единицу (если методу был передан всего один аргумент) или на число, равное количеству переданных аргументов.

Метод unshift() изменяет исходный массив, а не создаёт его модифицированную копию.

Синтаксис

arr.unshift([element1[, ...[, elementN]]]);

Аргументы

elementN: Элементы, добавляемые в начало массива. Элементы добавляются в том порядке, в котором они были переданы методу.

Возвращаемое значение

Новое значение свойства length того массива, для которого был вызван метод.

JS. Arrays

concat

Метод concat() создаёт и возвращает новый массив, содержащий элементы массива, на котором он был вызван, и значения, переданные в качестве аргументов.

Если из переданных аргументов какой-либо аргумент является массивом, то в возвращаемый массив добавляется не сам массив, а его элементы. В новом массиве сначала идут элементы исходного массива, затем значения переданные в качестве аргументов. Значения добавляются в том порядке, в котором они были переданы методу.

Метод concat() не изменяет исходный массив.

JS. Arrays

concat

Синтаксис

arr.concat(value1[, value2[, ...[, valueN]]]);

Аргументы

valueN: Значения, которые будут добавлены в возвращаемый массив.

Возвращаемое значение

Массив, содержащий элементы исходного массива, и значения, переданные в качестве аргументов.

JS. Arrays

join

Метод join() преобразует все элементы массива в строки, объединяет их и возвращает получившуюся строку.

Если элемент массива в качестве значения содержит undefined или null, то он преобразуется в пустую строку.

Синтаксис

arr.join([separator]);

Аргументы

separator (необязательный): Строка, которая будет использоваться в качестве разделителя элементов в возвращаемой строке. Если строка-разделитель не указана, то по умолчанию в качестве разделителя используется "," (запятая).

Возвращаемое значение

Строка: строка, содержащая все элементы массива. Если длина массива равна 0, то будет возвращена пустая строка.

JS. Arrays

slice

Метод Array.slice() используется для копирования указанного участка из массива и возвращает новый массив содержащий скопированные элементы.

Исходный массив при этом не меняется.

Синтаксис метода:

arr.slice(begin, end);

Метод принимает два аргумента, которые определяют начало и конец возвращаемого участка массива. Метод копирует участок массива, начиная от begin до end, не включая end.

Если указан только один аргумент, возвращаемый массив будет содержать все элементы от указанной позиции до конца массива. Можно использовать отрицательные индексы - они отсчитываются с конца массива.

JS. Arrays

indexOf

Метод indexOf возвращает индекс элемента, значение которого равно значению, переданному методу в качестве аргумента.

Синтаксис методов indexOf() и lastIndexOf():

arr.indexOf(искомый_элемент, индекс)

arr.lastIndexOf(искомый_элемент, индекс)

Первый аргумент метода указывает значение элемента, индекс которого нужно найти, второй аргумент (необязательный), указывает индекс с которого будет начинаться поиск. Если одинаковых вхождений несколько, выбирается наименьший (первый) индекс. Если элемент с искомым значением не найден, метод вернет -1. Внутри метода для поиска используется строгое сравнение (===).

JS. Arrays

includes

Метод includes() определяет, содержит ли массив указанный элемент.

Этот метод возвращает true, если массив содержит элемент, и false, если нет.

Синтаксис

arr.includes(element, start)

Значения параметров

element Требуемый. Элемент для поиска

start Необязательный. По умолчанию 0. В какой позиции массива начать поиск

JS. Arrays

reverse

Метод reverse() меняет порядок следования элементов в массиве на обратный и возвращает переупорядоченный массив.

Перестановка элементов выполняется непосредственно в исходном массиве. Это означает, что метод reverse() не создаёт новый массив с переупорядоченными элементами, а переупорядочивает их в уже существующем массиве.

Синтаксис

arr.reverse();

Аргументы - нет.

Возвращаемое значение

Ссылка на переупорядоченный массив.

JS. Arrays

at

Метод at() принимает целочисленное значение и возвращает элемент по этому индексу с учетом положительных и отрицательных целых чисел. Отрицательные целые числа отсчитываются от последнего элемента в массиве.

Синтаксис

arr.at(index)

Возвращает элемент в массиве, соответствующий данному индексу. Возвращает undefined если указанный индекс не может быть найден.

JS. Arrays

fill

Метод fill() заполняет указанные элементы массива статическим значением.

Вы можете указать положение, с которого начинается и заканчивается заполнение. Если не указано, то все элементы будет заполнены.

Примечание: этот метод перезаписывает исходный массив.

Синтаксис

arr.fill(value, start, end)

value	Требуемый. Значение для заполнения массива
start	Необязательный. Индекс для начала заполнения массива (по умолчанию 0)
end	Необязательный. Индекс для остановки заполнения массива (по умолчанию array.length)

JS. Arrays

splice

Метод splice() изменяет массив, добавляя или удаляя элементы в нем.

Метод splice() изменяет существующий массив, а не возвращает новый.

JavaScript синтаксис:

// только с указанием индекса – является обязательным параметром

arr.splice(start)

// с указанием индекса и количества удаляемых элементов - отрицательные значения не допускаются

arr.splice(start, deleteCount)

// с указанием индекса, количества удаляемых элементов и с добавлением элементов

arr.splice(start, deleteCount, element1, element2, ..., elementX)

JS. Arrays

flat

Метод flat() возвращает новый массив и уменьшает вложенность массива на заданное количество уровней

Метод принимает необязательный аргумент depth — количество уровней, на которые нужно уменьшить вложенность. Значение по умолчанию — 1.

arr.flat()

Результатом вызова метода flat() будет новый массив меньшей вложенности.

Если вложенность неизвестна, но нужно получить из массива с вложенными элементами плоский массив, то передайте аргумент Infinity. Тогда метод рекурсивно обойдёт массив и сделает на его основе новый плоский.

JS. Arrays

toString()

Метод toString() возвращает строковое представление указанного массива и его элементов.

Синтаксис

arr.toString()

Параметры

Нет.

Описание

Для объектов класса Array, метод toString соединяет массив и возвращает одну строку, содержащую каждый элемент массива, разделённый запятыми.

JS. Arrays

Циклы и массивы. Копирование

Цикл for...of в JavaScript позволяет перебирать итерируемые (перебираемые) объекты: массивы, множества, Map, строки и т.д. for...of появился в JavaScript ES6.

Синтаксис

```
for (элемент of итерируемый_объект) {
    // тело цикла for of
}
```

итерируемый_объект — массив, множество, строка и т.д.

элемент — элементы итерируемого объекта.

На русском языке этот код можно прочитать так: для каждого элемента в итерируемом объекте выполнить тело цикла.

JS. Arrays

Циклы и массивы. Копирование

```
const arr = [1, 2, 3]
for (let element of arr) {
 console.log(element)
}
```

```
// массив
const students = ['Андрей', 'Маша', 'Даня'];

// используем for...of
for ( let element of students ) {

    // выводим значения
    console.log(element);
}
```

Отличия for...of от for...in

for...of	for...in
Для перебора значений итерируемого объекта.	Для перебора ключей объекта.
Нельзя использовать для перебора объекта.	Можно использовать для перебора объекта, но не стоит использовать для перебора итерируемых объектов.

JS. Arrays

Циклы и массивы. Копирование

Оператор for позволяет организовывать циклы, которые, в частности, можно использовать и для перебора (или инициализации) массивов, обращаясь к их элементам по индексам. Обычно индекс очередного элемента получают, пользуясь счётчиком цикла.

```
const arr = [1, 2, 3]

for (let i = 0; i < arr.length; i += 1) {

  console.log( arr[i])

}
```

JS. Arrays

Копирование

В JavaScript все присваивания объектов реализуются через передачу ссылок на них.

```
const arr = [1, 2, 3];
const arrCopy = arr;

console.log(arr === arrCopy); // => true
```

```
const arr = [1, 2, 3];
const arrCopy = [1, 2, 3];

console.log(arr === arrCopy); // => false
```

let arr = ["a", "b", "c"];

let arrCopy = arr;

Хоть мы и получили две разные переменные, но тем не менее они обе ссылаются на один и тот же объект массива. Если сейчас в одном массиве произвести какие-либо манипуляции с элементами, то аналогичные изменения можно будет увидеть и в другом.

Если вы хотите сделать независимую копию массива, то нужно использовать метод slice без аргументов.

let arr = ["a", "b", "c"];

let arrCopy = arr.slice();

Массивы arr и arrCopy будут состоять из одних и тех же элементов, но фактически это будут разные объекты.

JS. Arrays

Копирование с помощью цикла

```
const arr = [1,2,3]
const arrCopy = []

for (let i = 0; i < arr.length; i++) {
    arrCopy[i] = arr[i]
}

console.log(arrCopy)
```