

The slide features decorative geometric patterns in dark blue and gold. These patterns consist of overlapping squares, rectangles, and lines, some of which are filled with a dotted texture. The patterns are located in the top-left, bottom-left, and bottom-right corners of the slide.

Методы массивов. Объекты и их свойства

Подготовил:
Глеб Завертяев

The image features decorative geometric patterns in the corners. The top-left and bottom-right corners contain overlapping blue and gold geometric shapes, including rectangles and triangles, some with a dotted texture. The text is centered on a white background.

Повторение: метод split

Метод - split

Описание:

Метод `split()` разбивает строку на массив строк по заданному разделителю

Синтаксис:

```
str.split(separator, limit)
```

Метод - `split`

Параметры:

`separator`

Необязательный параметр. Указывает символы, используемые в качестве разделителя внутри строки. Параметр `separator` может быть как строкой, так и регулярным выражением. Если параметр опущен, возвращённый массив будет содержать один элемент со всей строкой. Если параметр равен пустой строке, строка `str` будет преобразована в массив символов.

`limit`

Необязательный параметр. Целое число, определяющее ограничение на количество найденных подстрок. Метод `split()` всё равно разделяет строку на каждом сопоставлении с разделителем `separator`, но обрезает возвращаемый массив так, чтобы он содержал не более `limit` элементов.

Метод - split

Возвращаемое значение:

Метод split() возвращает новый массив.

Примеры:

```
1 let names = 'Вася, Петя, Маша';
2
3 let arr = names.split(', ');
4
5 for (let name of arr) {
6   alert( `Сообщение получат: ${name}.` ); // Сообщение получат: Вася (и другие имена)
7 }
```

```
1 let arr = 'Вася, Петя, Маша, Саша'.split(', ', 2);
2
3 alert(arr); // Вася, Петя
```

The image features a white background with decorative geometric patterns in the corners. The top-left and bottom-right corners contain overlapping blue and dark blue shapes with gold outlines and a pattern of small blue dots. The title is centered in the middle of the page.

Методы массивов

Метод массива - map

Описание:

Метод map вызывает переданную функцию callback один раз для каждого элемента, в порядке их появления и конструирует новый массив из результатов её вызова.

Синтаксис:

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

Метод массива - map

Параметры:

callback

Функция, вызываемая для каждого элемента массива arr. Каждый раз, когда callback выполняется, возвращаемое значение добавляется в new_array.

Функция callback, создающая элемент в новом массиве, принимает три аргумента:

- **currentValue**
Текущий обрабатываемый элемент в массиве.
- **index** (необязательный)
Индекс текущего обрабатываемого элемента в массиве.
- **array** (необязательный)
Массив, по которому осуществляется проход.

Метод массива - map

Пример:

```
1 const array1 = [1, 4, 9, 16];  
2  
3 // Pass a function to map  
4 const map1 = array1.map((x) => x * 2);  
5  
6 console.log(map1);  
7 // Expected output: Array [2, 8, 18, 32]  
8
```

Метод массива - `map`

Возвращаемое значение:

Новый массив, где каждый элемент является результатом callback функции.

!Метод `map` не изменяет массив, для которого он был вызван

Метод массива - `forEach`

Описание:

Метод `forEach()` выполняет функцию `callback` один раз для каждого элемента, находящегося в массиве в порядке возрастания.

Синтаксис:

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

Метод массива - `forEach`

Параметры:

`callback`

Функция, которая будет вызвана для каждого элемента массива, принимает три аргумента:

- `currentValue`
Текущий обрабатываемый элемент в массиве.
- `index` (необязательный)
Индекс текущего обрабатываемого элемента в массиве.
- `array` (необязательный)
Массив, по которому осуществляется проход.

Метод массива - forEach

Пример:

```
1 const array1 = ['a', 'b', 'c'];
2
3 array1.forEach((element) => console.log(element));
4
5 // Expected output: "a"
6 // Expected output: "b"
7 // Expected output: "c"
8
```

Метод массива - **forEach**

Возвращаемое значение:

undefined

!Метод `forEach()` не изменяет массив, для которого он был вызван.

Метод массива - **filter**

Описание:

Метод `filter()` создаёт новый массив со всеми элементами, прошедшими проверку, задаваемую в передаваемой функции.

Он вызывает переданную функцию `callback` один раз для каждого элемента, присутствующего в массиве, и создаёт новый массив со всеми значениями, для которых функция `callback` вернула значение, которое может быть приведено к **true**.

Метод массива - filter

Синтаксис:

```
// Стрелочная функция
filter((element) => { ... } )
filter((element, index) => { ... } )
filter((element, index, array) => { ... } )

// Колбэк-функция
filter(callbackFn)
filter(callbackFn, thisArg)

// Встроенная колбэк-функция
filter(function callbackFn(element) { ... })
filter(function callbackFn(element, index) { ... })
filter(function callbackFn(element, index, array){ ... })
```


Метод массива - filter

Параметры:

`callbackFn`

Функция, которая будет вызвана для проверки каждого элемента массива. Если функция возвращает `true`, то элемент остаётся в массиве, если `false`, то удаляется.

Принимает три аргумента

- `element`

Текущий обрабатываемый элемент в массиве.

- `index`(необязательный)

Индекс текущего обрабатываемого элемента в массиве.

- `array`(необязательный)

Обрабатываемый массив, на котором был вызван метод `filter()`.

Метод массива - filter

Пример:

```
const words = ['spray', 'elite', 'exuberant', 'destruction', 'present'];  
  
const result = words.filter((word) => word.length > 6);  
  
console.log(result);  
// Expected output: Array ["exuberant", "destruction", "present"]
```

Метод массива - **filter**

Возвращаемое значение:

Вернётся новый массив с элементами, которые прошли проверку. Если ни один элемент не прошёл проверку, то будет возвращён пустой массив.

!Метод `filter()` не изменяет массив, для которого он был вызван.

Метод массива - find

Описание:

Метод `find()` возвращает значение первого найденного в массиве элемента, которое удовлетворяет условию переданному в `callback` функции. В противном случае возвращается `undefined`.

Синтаксис:

```
arr.find(callback)
```

Метод массива - find

Параметры:

callback

Функция, вызываемая для каждого значения в массиве, принимает три аргумента:

- **element**

Текущий обрабатываемый элемент в массиве.

- **index** (необязательный)

Индекс текущего обрабатываемого элемента в массиве.

- **array** (необязательный)

Массив, по которому осуществляется проход.

Метод массива - find

Пример:

```
1  let users = [  
2    {id: 1, name: "Вася"},  
3    {id: 2, name: "Петя"},  
4    {id: 3, name: "Маша"}  
5  ];  
6  
7  let user = users.find(item => item.id == 1);  
8  
9  alert(user.name); // Вася
```

Метод массива - **find**

Возвращаемое значение:

Значение элемента из массива, если элемент прошёл проверку, иначе undefined.

!Метод `find()` не изменяет массив, для которого он был вызван.

Метод массива - reduce

Описание:

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

Синтаксис:

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


Метод массива - reduce

callback

Функция, выполняющаяся для каждого элемента массива, принимает четыре аргумента:

- **previousValue**

Значение, которое возвращает функция callback после посещения очередного элемента, либо значение initialValue, если оно предоставлено (смотрите пояснения ниже).

- **currentValue**

Текущий обрабатываемый элемент в массиве.

- **index** (необязательный). Индекс текущего обрабатываемого элемента в массиве.

- **array** (необязательный). Массив, для которого была вызвана функция reduce.

initialValue (необязательный)

Необязательный параметр. Объект, используемый в качестве первого аргумента при первом вызове функции callback.

Метод массива - reduce

Пример

```
[0, 1, 2, 3, 4].reduce(function (previousValue, currentValue, index, array) {  
    return previousValue + currentValue;  
});
```

	previousValue	currentValue	index	array	возвращаемое значение
первый вызов	0	1	1	[0, 1, 2, 3, 4]	1
второй вызов	1	2	2	[0, 1, 2, 3, 4]	3
третий вызов	3	3	3	[0, 1, 2, 3, 4]	6
четвёртый вызов	6	4	4	[0, 1, 2, 3, 4]	10

Значение, возвращённое методом reduce() будет равным последнему результату выполнения колбэк-функции — **10**.

Метод массива - reduce

Возвращаемое значение:

Результирующее значение

!Метод `reduce()` не изменяет массив, для которого он был вызван.

Метод массива - reverse

Описание:

Метод reverse() на месте обращает порядок следования элементов массива. Первый элемент массива становится последним, а последний — первым.

Синтаксис:

```
array.reverse()
```

Метод массива - reverse

Пример:

```
var myArray = ["один", "два", "три"];  
myArray.reverse();  
  
console.log(myArray); // ['три', 'два', 'один']
```

Метод массива - reverse

Параметры: нет

Возвращаемое значение:

Перевернутый массив

!Метод reverse изменяет исходный массив

Метод массива - some

Описание:

Метод `some()` вызывает переданную функцию `callback` один раз для каждого элемента, присутствующего в массиве до тех пор, пока не найдёт такой, для которого `callback` вернёт истинное значение. Если такой элемент найден, метод `some()` немедленно вернёт `true`. В противном случае, вернёт `false`.

Синтаксис:

```
arr.some(callback(element, index, array))
```

Метод массива - `some`

Параметры

`callback`

Функция проверки каждого элемента, принимает три аргумента:

- `element`

Текущий обрабатываемый элемент в массиве.

- `index` (необязательный).

Индекс текущего обрабатываемого элемента в массиве.

- `array` (необязательный).

Массив, для которого была вызвана функция `reduce`.

Метод массива - some

Пример:

```
function isBiggerThan10(element, index, array) {  
  return element > 10;  
}  
[2, 5, 8, 1, 4].some(isBiggerThan10); // false  
[12, 5, 8, 1, 4].some(isBiggerThan10); // true
```

Метод массива - **some**

Возвращаемое значение:

true, если функция проверки возвращает true значение хотя бы для одного элемента массива. Иначе, false.

!Метод `some()` не изменяет массив, для которого он был вызван.

Метод массива - every

Описание:

Метод every() проверяет, удовлетворяют ли все элементы массива условию, заданному в передаваемой функции.

Он вызывает переданную функцию callback один раз для каждого элемента, присутствующего в массиве до тех пор, пока не найдёт такой, для которого callback вернёт ложное значение (значение, становящееся равным false при приведении его к типу Boolean).

Синтаксис:

```
arr.every(callback(currentValue, index, array))
```

Метод массива - **every**

Параметры:

callback

Функция проверки каждого элемента, принимает три аргумента:

- **currentValue**
Текущий обрабатываемый элемент массива.
- **index** (необязательный)
Индекс текущего обрабатываемого элемента массива.
- **array** (необязательный)
Массив, по которому осуществляется проход.

Метод массива - every

Примеры:

```
function isBigEnough(element, index, array) {  
  return element >= 10;  
}  
  
[12, 5, 8, 130, 44].every(isBigEnough); // false  
[12, 54, 18, 130, 44].every(isBigEnough); // true
```

```
[12, 5, 8, 130, 44].every((elem) => elem >= 10); // false  
[12, 54, 18, 130, 44].every((elem) => elem >= 10); // true
```

Метод массива - **every**

Возвращаемое значение:

true если функция проверки возвращает значение **true** для каждого элемента массива. Иначе, **false**.

!Метод `every()` не изменяет массив, для которого он был вызван.

Метод массива - includes

Описание:

Метод `includes()` определяет, содержит ли массив определённый элемент, возвращая в зависимости от этого `true` или `false`.

Синтаксис:

```
arr.includes(searchElement, fromIndex)
```

Метод массива - includes

Параметры:

searchElement

Искомый элемент.

fromIndex (необязательный)

Позиция в массиве, с которой начинать поиск элемента searchElement. При отрицательных значениях поиск производится начиная с индекса `array.length + fromIndex` по возрастанию. Значение по умолчанию равно 0.

Возвращаемое значение:

Boolean.

!Метод `includes()` не изменяет массив, для которого он был вызван.

Метод массива - includes

Пример:

```
1 const array1 = [1, 2, 3];  
2  
3 console.log(array1.includes(2));  
4 // Expected output: true  
5  
6 const pets = ['cat', 'dog', 'bat'];  
7  
8 console.log(pets.includes('cat'));  
9 // Expected output: true  
10  
11 console.log(pets.includes('at'));  
12 // Expected output: false  
13
```

Метод массива - join

Описание:

Метод join() преобразует все элементы массива в строки и объединяет их в одну большую строку.

Элемент массива с типом undefined или null преобразуется в пустую строку.

Синтаксис:

```
arr.join(separator)
```

Метод массива - join

Параметры:

separator (необязательный)

Определяет строку, разделяющую элементы массива. В случае необходимости тип разделителя приводится к типу Строка. Если он не задан, элементы массива разделяются запятой ','. Если разделитель - пустая строка, элементы массива ничем не разделяются в возвращаемой строке.

Возвращаемое значение:

Строка, содержащая все элементы массива. Если `arr.length == 0`, то будет возвращена пустая строка.

!Метод `join()` не изменяет массив, для которого он был вызван.

Метод массива - join

Пример:

```
1 let arr = ['Вася', 'Петя', 'Маша'];  
2  
3 let str = arr.join(';'); // объединить массив в строку через ;  
4  
5 alert( str ); // Вася;Петя;Маша
```

Метод массива - `findIndex`

Описание:

Метод `findIndex()` возвращает индекс в массиве, если элемент удовлетворяет условию проверяющей функции. В противном случае возвращается -1.

Синтаксис:

```
arr.findIndex(callback)
```

Метод массива - `findIndex`

Параметры:

`callback`

Функция, вызываемая для каждого значения в массиве, принимает три аргумента:

- `element`

Текущий обрабатываемый элемент в массиве.

- `index` (необязательный)

Индекс текущего обрабатываемого элемента в массиве.

- `array` (необязательный)

Массив, по которому осуществляется проход.

Метод массива - `findIndex`

Пример:

```
1 let users = [  
2   {id: 1, name: "Вася"},  
3   {id: 2, name: "Петя"},  
4   {id: 3, name: "Маша"},  
5   {id: 4, name: "Вася"}  
6 ];  
7  
8 // Найти индекс первого Васи  
9 alert(users.findIndex(user => user.name == 'Вася')); // 0
```

Метод массива - `findIndex`

Возвращаемое значение:

Индекс элемента из массива, если элемент прошёл проверку, иначе -1.

!Метод `findIndex()` не изменяет массив, для которого он был вызван.

Метод массива - isArray

Описание:

Метод `Array.isArray()` возвращает `true`, если объект является массивом и `false`, если он массивом не является.

Синтаксис:

```
Array.isArray(obj)
```

Метод массива - isArray

Параметры:

obj

Объект для проверки.

Возвращаемое значение:

true если объект является Array. Иначе, **false**.

!Метод isArray() не изменяет элемент, для которого он был вызван.

Метод массива - isArray

Пример:

```
// Все следующие вызовы вернут true
Array.isArray([]);
Array.isArray([1]);
Array.isArray(new Array());
// Малоизвестный факт: Array.prototype сам является массивом:
Array.isArray(Array.prototype);

// Все следующие вызовы вернут false
Array.isArray();
Array.isArray({});
Array.isArray(null);
Array.isArray(undefined);
Array.isArray(17);
Array.isArray("Array");
Array.isArray(true);
Array.isArray(false);
Array.isArray({ __proto__: Array.prototype });
```

Методы, изменяющие исходный массив

- `pop()`
- `push()`
- `shift()`
- `unshift()`
- `reverse()`
- `sort()`
- `splice()`



Свойства объектов

Свойство объекта- keys

Описание:

Метод `Object.keys` возвращает массив строковых элементов, соответствующих именам перечисляемых свойств, найденных непосредственно в самом объекте. Порядок свойств такой же, как и при ручном перечислении свойств в объекте через цикл.

Синтаксис:

```
Object.keys(obj)
```

Свойство объекта- keys

Параметры:

`obj`

Объект, чьи собственные перечисляемые свойства будут возвращены.

Возвращаемое значение:

Массив строковых элементов, соответствующих именам перечисляемых свойств, найденных непосредственно в самом объекте

Свойство объекта- keys

Пример:

```
var arr = ["a", "b", "c"];  
console.log(Object.keys(arr)); // консоль: ['0', '1', '2']  
  
// Массивоподобный объект  
var obj = { 0: "a", 1: "b", 2: "c" };  
console.log(Object.keys(obj)); // консоль: ['0', '1', '2']
```


Свойство объекта- values

Описание:

Object.values() возвращает массив, чьи элементы это значения перечисляемых свойств найденных в объекте. Порядок такой же как если пройти по объекту циклом вручную.

Синтаксис:

```
Object.values(obj)
```

Свойство объекта- values

Параметры:

obj

Объект, чьи значения перечисляемых свойств будут возвращены.

Возвращаемое значение:

Массив содержащий значения перечисляемых свойств объекта

Свойство объекта- values

Пример:

```
var obj = { foo: "bar", baz: 42 };  
console.log(Object.values(obj)); // ['bar', 42]  
  
// Массив как объект  
var obj = { 0: "a", 1: "b", 2: "c" };  
console.log(Object.values(obj)); // ['a', 'b', 'c']
```

Свойство объекта- `entries`

Описание:

`Object.entries()` возвращает массив, элементами которого являются массивы, соответствующие перечисляемому свойству пары `[key, value]`, найденной прямо в `object`. Порядок свойств тот же, что и при прохождении циклом по свойствам объекта вручную.

Синтаксис:

```
Object.entries(obj)
```

СВОЙСТВО объекта- `entries`

Параметры:

`obj`

Объект, чьи перечислимые свойства будут возвращены в виде массива `[key, value]`.

Возвращаемое значение:

Массив перечислений собственных свойств объекта с парами `[key, value]`.

Свойство объекта- entries

Пример:

```
var obj = { foo: "bar", baz: 42 };  
console.log(Object.entries(obj)); // [ ['foo', 'bar'], ['baz', 42] ]  
  
// массив как объект  
var obj = { 0: "a", 1: "b", 2: "c" };  
console.log(Object.entries(obj)); // [ ['0', 'a'], ['1', 'b'], ['2', 'c'] ]
```

Свойство объекта- delete method

Описание:

Оператор delete удаляет свойство из объекта.

Синтаксис:

```
delete выражение
```

где результат вычисления выражения должен быть ссылкой на свойство (объекта), например:

```
delete object.property  
delete object['property']  
delete object[index]  
delete property // удаляет свойства глобального объекта, или,  
                  // используя инструкцию with, свойства объекта, на который ссылается инструкция
```

Свойство объекта- delete method

Параметры:

object

Имя объекта или выражение, результатом вычисления которого является объект.

property

Удаляемое свойство.

index

Целое число, представляющее собой индекс массива, который должен быть удалён.

Возвращаемое значение:

Возвращает false, только если свойство существует в самом объекте, а не в его прототипах, и не может быть удалено. Во всех остальных случаях возвращает true.

Свойство объекта- `delete` method

Проблемы при использовании метода `delete`:

1. Проблема с кроссбраузерностью. Использование метода `delete` с Internet Explorer приводит к некоторым неожиданным результатам
2. Когда с помощью оператора `delete` удаляется элемент массива, длина массива не меняется, то есть образуется своего рода “дыра” с неопределённым значением