

Объекты.(продолжение).

Массивы

Деструктуризация.

for .. of

Оператор for...of выполняет цикл обхода итерируемых объектов

```
let iterable = [10, 20, 30];  
  
for (let value of iterable) {  
  
    value += 1;  
  
    console.log(value);  
  
}
```

Array.from

Есть универсальный метод `Array.from`, который принимает итерируемый объект или псевдомассив и делает из него «настоящий» `Array`. После этого мы уже можем использовать методы массивов.

```
let obj = {  
  0: 34,  
  1: 12,  
  length: 2  
};  
let arr = Array.from(obj);  
  
for (let item of arr) {  
  alert(item); // 34, 12  
}
```

Задача 1

Создать произвольный массив и пройти по нему с помощью цикла `for .. of`

Задача 2

Существует ul список на странице. Получить все текстовые значения элементов списка. Создать из них массив и к каждому элементу массива добавить его порядковый номер. Вывести полученный массив

Object.keys, values, entries

Для простых объектов доступны следующие методы:

- `Object.keys(obj)` – возвращает массив ключей.
- `Object.values(obj)` – возвращает массив значений.
- `Object.entries(obj)` – возвращает массив пар [ключ, значение].

Используем `Object.fromEntries(array)` на результате, чтобы преобразовать его обратно в объект.

Задача 3

Есть объект `prices` с произвольным количеством свойств, содержащих цены продуктов.

Напишите функцию `sumPrices(prices)`, которая возвращает сумму всех цен с помощью метода `Object.values`

Деструктурирующее присваивание

Довольно часто требуется извлечь одно или несколько значений из группы и присвоить их переменным. Возьмем для примера массив, содержащий имя и значение свойства. Чтобы сохранить каждый элемент в отдельной переменной, требуется две инструкции, например:

```
let nextValue = ["color", "red"];
```

```
let name = nextValue[0];
```

```
let value = nextValue[1];
```


Деструктурирующее присваивание

Присваивание с деструктуризацией (destructuring assignment) позволяет назначить

два элемента массива переменным с помощью одной инструкции:

```
let [name, value] = ["color", "red"];  
alert(name); // "color"  
alert(value); // "red"
```

Деструктурирующее присваивание

Присваивание с де структуризацией указывает, что переменным в массиве слева от знака равенства необходимо присвоить значения из массива, указанного справа от знака равенства. В результате переменная name получает значение "color", а переменная value - значение "red".

Деструктурирующее присваивание

Объекты также поддерживают присваивание с деструктуризацией, например:

```
let person = {  
  name : "Nicholas" , age: 29 }  
  
let { name : personName, age : personAge } = person;  
  
alert(personName) ; // "Nicholas"  
  
alert(personAge); // 29
```

Остаточные параметры «...»

Если мы хотим не просто получить первые значения, но и собрать все остальные, то мы можем добавить ещё один параметр, который получает остальные значения, используя оператор «остаточные параметры» – троеточие ("..."):

```
let [name1, name2, ...rest] = ["Julius", "Caesar", "Consul", "of the Roman Republic"];
```

```
alert(name1); // Julius
```

```
alert(name2); // Caesar
```

```
alert(rest.length); // 2
```

Задача 4

Есть объект

```
let product = {
```

```
  name: "John",
```

```
  price: 30,
```

```
  sold: false
```

```
};
```

Присвоить каждое свойство объекта к отдельной переменной

Задача 5

Есть массив ['Tony', 'Stark', 1 , 45, 2, 5, 34, 9, 11]

Присвоить первое и второе значения массива к соответствующим переменным, а остальные значения сложить

Конструкторы объектов

Обычный синтаксис `{...}` позволяет создать только один объект. Чтобы создать множество однотипных объектов можно использовать функции-конструкторы и оператор `"new"`

Конструкторы объектов

```
function User(name) {  
    this.name = name;  
    this.isAdmin = false;  
}  
  
let user = new User("Вася");  
  
alert(user.name); // Вася  
  
alert(user.isAdmin); // false
```