

Лабораторная работа № 14. Новый ECMAScript 2017

Цель: изучить новшества ECMAScript 2017: метод `Object.values()`, дополнение строк до заданной длины, метод `Object.getOwnPropertyDescriptors()`.

Теория

Object.values

Метод **`Object.values()`** — это новая функция, которая похожа на `Object.keys()`, но возвращает все значения собственных свойств объекта, исключая любые значения в цепочке прототипов. Возвращает в том же порядке, что и цикл *for...in*, но цикл перечисляет свойства и из цепочки прототипов.

```
const cars = { BMW: 3, Tesla: 2, Toyota: 1 };
// С помощью Object.keys
const vals = Object.keys(cars).map(key => cars [key] );
console.log (vals);      // [ 3, 2, 1 ]
// С помощью Object.values
const values = Object.values(cars);
console.log(values);      // [ 3, 2, 1 ]
```

Дополнение строк до заданной длины

У объектов типа `String` есть два новых метода после появления ECMAScript 2017: **`String.prototype.padStart()`** и **`String.prototype.padEnd()`**. Они позволяют присоединять к строкам, в их начало или конец, некоторое количество символов для дополнения строк до заданной длины. Это удобно, если нужно выровнять текст.

Пример 1. Есть список чисел разной длины. Нужно добавить в начало этих чисел «0» таким образом, чтобы сделать все их состоящими из 10 цифр. Результат – аккуратный вывод чисел в консоль. Можно воспользоваться командой `padStart(10, '0')`.

```
const formatted = [0, 1, 12, 123, 1234, 12345].map(num=>
  num.toString().padStart(10, '0') );
console.log(formatted);
// Результат
// [
//   '0000000000',
//   '0000000001',
//   '0000000012',
//   '0000000123',
```

```
// '0000001234',
// '0000012345',
// ];
```

Пример 2. Совместное использование методов *padEnd()*, *padStart()*, и *Object.entries()* для формирования строк, которые аккуратно смотрятся при выводе на экран.

```
const cars = {
  'BMW': '10',
  'Tesla': '5',
  'Lamborghini': '0'
}
Object.entries(cars).map(([name, count])=>{
  console.log(`${name.padEnd(20, '-')}Count: ${count.padStart(3, '0')}')
});
// Результат
// BMW - - - - - Count: 010
// Tesla - - - - - Count: 005
// Lamborghini - - - Count: 000
```

Object.getOwnPropertyDescriptors()

Метод возвращает все сведения (включая данные о геттерах и сеттерах) для всех свойств заданного объекта. Главная причина добавления этого метода заключается в том, чтобы позволить создавать мелкие копии объектов и клонировать объекты, создавая новые объекты, при этом копируя, помимо прочего, геттеры и сеттеры. Метод *Object.assign()* этого не умеет. Он позволяет выполнять мелкие копии объектов, но не работает с их геттерами и сеттерами.

В следующих примерах показана разница между *Object.assign()* и *Object.getOwnPropertyDescriptors()*, а также продемонстрировано использование метода *Object.defineProperties()* для копирования исходного объекта *car* в новый объект, *ElectricCar*. Тут можно заметить, что, благодаря использованию *Object.getOwnPropertyDescriptors()*, функция *discount()*, играющая роль и геттера, и сеттера, также копируется в целевой объект.

```
// Код с использованием Object.assign()
var Car = {
  name: 'BMW',
  price: 1000000,
  set discount(x) {
    this.d = x;
  },
  get discount() {
    return this.d;
  }
}
```

```

    },
};
console.log(Object.getOwnPropertyDescriptor(Car, 'discount'));
// Результат
// {
//   get: [Function: get],
//   set: [Function: set],
//   enumerable: true,
//   configurable : true
// }
//Код с использованием Object.getOwnPropertyDescriptors()
var Car = {
  name : 'BMW',
  price: 1000000,
  set discount(x) {
    this.d = x;
  },
  get discount() {
    return this.d;
  },
};
const      ElectricCar2      =      Object.defineProperties({},
Object.getOwnPropertyDescriptors(Car));
console.log(Object.getOwnPropertyDescriptor(ElectricCar2,
'discount'));
// Результат
// {
//   get: [Function: get],
//   set: [Function: set],
//   enumerable: true,
//   configurable : true
// }

```

Задания к лабораторной работе № 14

Задание 1. Создать объект с несколькими свойствами. Вывести на экран значения этих свойств с помощью метода `Object.values()` и `Object.keys`.

Задание 2. Создать список чисел разной длины. Добавить в начало и в конец любое число с помощью `padStart()` и `padEnd()` так, чтобы выравнивать границы списка. Суммарное количество цифр в каждой строчке – 15. Вывести на экран список до и после выравнивания.

Задание 3. Написать свои имя и фамилию так, чтобы в начале имени были произвольные символы, а в конце фамилии свой возраст.

Задание 4. Создать массив символов. Вывести на экран значения массива так, чтобы в начале были единицы, а в конце ноль с отступом 4.

Задание 5. Создать объект со свойствами. Присвоить отдельной переменной описание свойства. Вывести его.

Задание 6. Создать объект. Добавить свойства объекту через *defineProperty()* с дескриптором доступа.