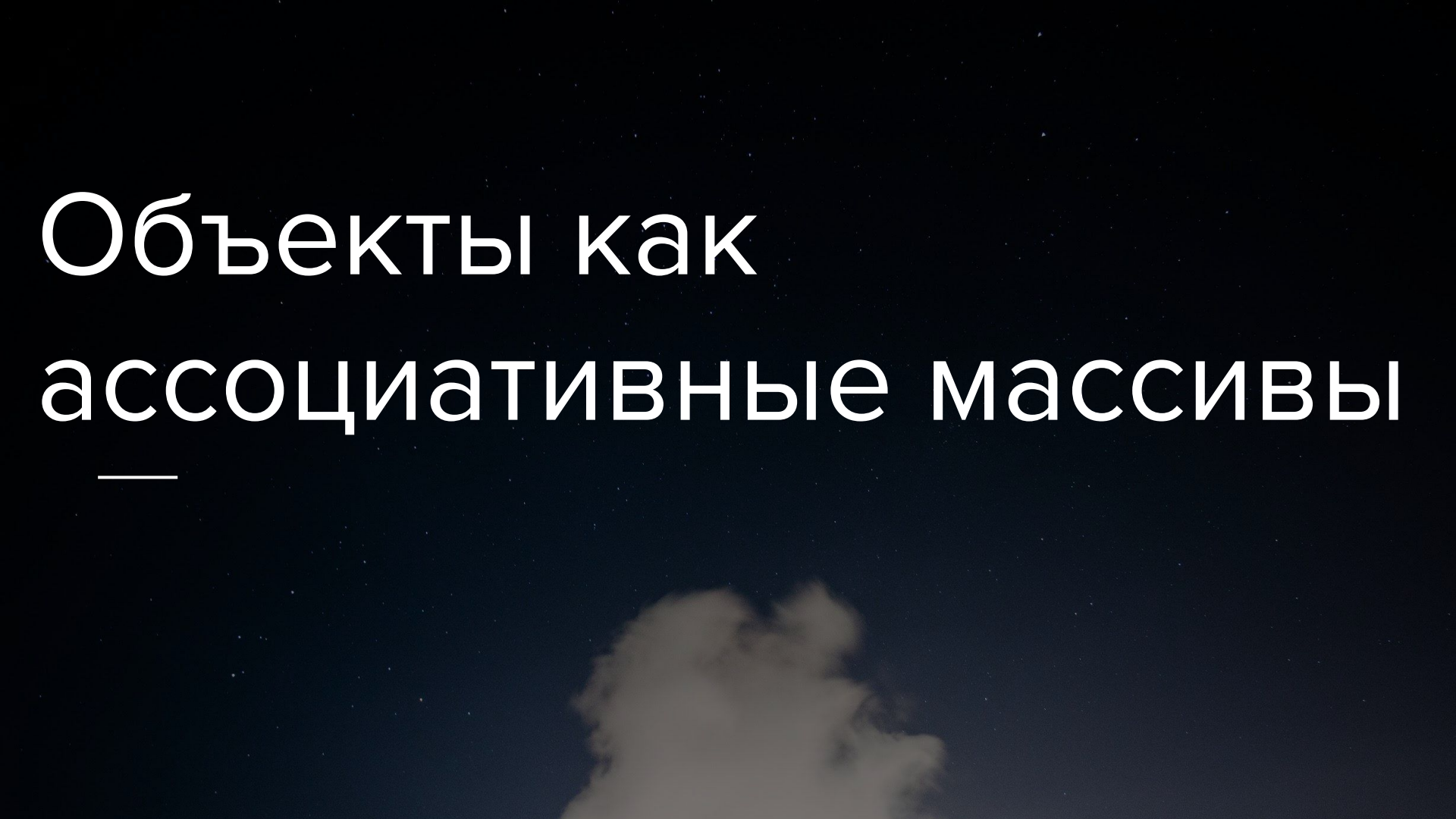
Объекты как ассоциативные массивы



Объект

это набор свойств, каждое свойство состоит из имени и значения, ассоциированного с этим именем. Значением свойства может быть функция, которую можно назвать методом объекта.

Объекты передаются по ссылке.

Применение объектов в JavaScript:

- как ассоциативный массив, для хранения данных в формате ключ-значение;
- как возможности языка для ООП

Объекты как ассоциативные массивы

Ассоциативный массив – структура данных, в которой можно хранить данные в формате ключ-значение.

Объявление объекта (2 варианта):

чаще используется первый

1) `let obj = {};`
2) `let obj = new Object();`

Свойства объекта

Объект может содержать в себе любые значения, которые называются свойствами объекта. Свойства можно понимать как переменную, закрепленную за объектом.

Доступ к свойствам осуществляется по имени:

- имяОбъекта.имяСвойства - запись «через точку» или
- объект[свойство] - запись через квадратные скобки

Имена свойств объекта могут быть строками JavaScript, или тем, что может быть сконвертировано в строку.

Неопределенные свойства объекта являются **undefined** (не null).

Свойства объекта (пример)

```
let user = {}; // создали объект user

user.id = 1;  // создали свойство с именем id, присвоили ему значение 1
user.login = 'nikname';
user.email = 'nikname@email.ru';

console.log('Логин: ', user.login);  // прочитали свойство name

delete user.email; // удалили свойство email
```

Проверка на наличие свойства объекта

1. Сравнить свойство с undefined (такой способ не подойдет, если свойство есть, а его значением является undefined);
2. Использовать оператор in: `if ("prop" in obj) {//Свойство name существует}`
   При написании программ для значения, которое обозначает неизвестность, используется null.
3. obj.hasOwnProperty(prop)

Доступ к свойству через переменную

Квадратные скобки позволяют обратиться к свойству, имя которого хранится в переменной

```
var idKey = 'id';    // обратились к свойству объекта через переменную
console.log(user[idKey]);  // аналогично записи console.log(user.id);
```

Такой вариант подходит для случаев, когда свойство определяется по ходу выполнения программы.

Литеральный способ объявления объекта

объявление объекта со свойствами, указанными в фигурных скобках.

```
let имяОбъекта= {
    свойство1: значение,
    свойство2: значение,
    свойство3: значение
};
```

```
let user = {
    id: 3,
    login: 'qwerty',
    email: 'qw@email.com'
};
```

Перечисление всех свойств объекта

1. **цикл for...in** - перебирает все перечисляемые свойства объекта и его цепочку прототипов;
2. **Object.keys(obj)** - возвращает массив со всеми собственными (те, что в цепочке прототипов, не войдут в массив) именами перечисляемых свойств объекта **obj**;
3. **Object.getOwnPropertyNames(obj)** - возвращает массив содержащий все имена своих свойств (перечисляемых и неперечисляемых) объекта o.

Методы объекта

Если в качестве свойства объекта указывать функцию - мы получим метод объекта.

```
let имяОбъекта= {
    свойство1: значение,
    свойство2: значение,
    свойство3: function () {
     // метод объекта
     }
};
```

Обращение в методу такое же, как и к свойствам рассмотренным ранее:

***имяОбъекта*.свойство3();**

Для доступа к другим свойствам объекта из метода используется **ключевое слово this**. **this.свойство2** из объекта аналогично ***имяОбъекта*.свойство2** снаружи

Деструктуризация

“возможность ES6/ES2015”

особый синтаксис присваивания, при котором можно присвоить массив или объект сразу нескольким переменным, разбив его на части.

Деструктуризация массива

```
let [name, age] = ["Иван", 67];

console.log(name); // Иван

console.log(age);  // 67
```

При таком присвоении **первое значение** массива пойдёт **в первую переменную**, **второе – во вторую**, а **последующие** (если есть) – **будут отброшены**.

Если нужны и последующие элементы, можно использовать **оператор ...**

```
let [name, age, ...otherElems] = ["Иван", 67, "ivan@gmail.com", "+79991112233"];
console.log(name); // Иван
console.log(age);  // 67
console.log(otherElems);  // ["ivan@gmail.com", "+79991112233"]
```

Деструктуризация

“возможность ES6/ES2015”

Деструктуризация объекта

```
let {перем1, перем2} = {перем1: "значение", перем2: "значение"};
```

Объект справа - объект, который мы хотим разбить, это уже существующий объект.

Слева - список переменных, в которые попадут соответствующие свойства.

Если необходимо присвоить свойство объекта в переменную с другим именем, нужно использовать двоеточие.

```
let {свойство1:перем1, свойство2:перем2} =

        {свойство1: "значение", свойство2: "значение"};
```

Деструктуризация в параметрах

“возможность ES6/ES2015”

Если функция получает объект, то она может его тут же разбить в переменные:

```
let userData = {
   name: "Иван",
   login: "qwe",
   age: 44
};
function showUserData({name, login, age, surname="значение по умолчанию"}) {
   console.log(name + ' ' + login + ' ' + age + ' ' + surname); объект будет разбит на
                                                                переменные
}

showUserData(userData); // Иван qwe 44 значение по умолчанию
```