Объекты

Мікалай Янкойць

Адукар

Повестка дня

1. Что такое объект и как его создать
2. Свойства объектов
3. Объект – ссылочный тип данных
4. Методы. Доступ через this

Что такое объект

Объект – это структура, которая позволяет хранить **любые** данные в формате ключ-значение.
Значения, которые содержатся в объекте, называются **свойствами** этого объекта. Доступ к ним осуществляется по **именам** (они же ключи).
Имена свойств могут иметь **только тип String***. Значения могут иметь любой тип (в том числе могут сами быть объектами).

*с версии ES6 – String или Symbol

Зачем нужны объекты

Объекты позволяют описывать любые сложные данные, которые невозможно «вместить» в переменные типов string, number, boolean. Часто они описывают объекты из реальной жизни.


```
let passport = {
    number: "07BC00000",
    name: "Anais",
    surname: "Specimen",
    birthday: "1973-07-12",
    birthplace: "Perpignan"
}
```

Как создать (пустой) объект

Как и в случае с массивами, есть несколько вариантов синтаксиса для создания пустого объекта.

```
1  let obj1 = {};
2  let obj2 = new Object();
```

Способ 1, с фигурными скобками, используется чаще.

Как наполнить объект данными

```
1  let dog = {};
2
3  dog.name = "Buddy";
4  dog.breed = "rottweiler";
5  dog.age = 5;
6  dog.color = "black with mahogany marks";
```

Объекту можно добавлять любое количество свойств. Для этого используется синтаксис вида **объект.имя_свойства**.
Свойства похожи на обычные переменные, им можно задавать значения любого типа.

Как объявить объект со свойствами

Свойства можно добавлять сразу при объявлении объектов. Для этого используется синтаксис вида **имя: значение**. Отдельные свойства внутри фигурных скобок разделяются запятыми.

```
let person = {
    name: "Ніл Гілевіч",
    birth: 1931,
    death: 2016,
    job: "паэт"
}
```

```
let person = {};
person.name = "Ніл Гілевіч";
person.birth = 1931;
person.death = 2016;
person.job = "паэт";
```

Код слева делает то же самое, что код справа.

Доступ к свойствам

Читать значения свойств можно по их именам, используя всё тот же синтаксис «через точку»: **объект.имя_свойства**

```
let address = {
    city: "Minsk",
    street: "Staravilenski trakt",
    building: 26,
    apartment: 124
}

console.log("Посылка доставлена по адресу " +
                address.street + ", " + address.building);
                // "... по адресу Staravilenski trakt, 26"
```

Проверка наличия свойства

Проверить, если у объекта свойство с конкретным именем, можно с помощью оператора in. Он возвращает true, если свойство есть, и false, если его нет.

```
let ball = {
    color: "cyan",
    diameter: 14,
    material: "glass"
}

console.log("color" in ball); // true
console.log("width" in ball); // false
// обратите внимание, что имя свойства передаётся как строка!
```

Доступ к несуществующим свойствам

В JS не запрещено обращаться к несуществующим свойствам объекта. При попытке их «чтения» возвращается undefined.

```
1  let ball = {
2      color: "cyan",
3      diameter: 14,
4      material: "glass"
5  }
6
7  console.log(ball.color, ball.width, ball.material);
8  // "cyan undefined glass"
```

Доступ к свойствам через квадратные скобки

Ещё один способ доступа к свойствам (и на чтение, и на запись) – через указание их имён в квадратных скобках.

```
let station = {
    name: "Міхалова",
    line: "Аўтазаводская"
}

console.log(station["name"]); // "Міхалова"
station["line"] = "Маскоўская";
console.log(station["line"]); // "Маскоўская". исправили!
station["code"] = 112;
console.log("code" in station); // true, свойство появилось
```

Доступ к свойствам через квадратные скобки

Если мы обращаемся к свойству «через точку», его имя должно соответствовать обычным правилам именования переменных. При обращении через квадратные скобки таких ограничений нет.

```
let city = {
    name: "Hrodna",
    population: 365600
}

city["postal code"] = "230000";
console.log(city["postal code"]); // работает!
console.log(city.postal code); // ошибка!
city["название на русском"] = "Гродно"; // работает!
```

Объявление свойств с «плохими» именами

Свойства, имена которых не соответствуют правилам именования переменных (например, содержат пробелы или начинаются с цифры), можно объявить и вместе с объектом:

```
let oneMoreCity = {
    name: "Barysaŭ",
    population: 143050,
    "1970population": 84000, // просто добавь кавычки!
    "football club": "BATE"
}

console.log(oneMoreCity["1970population"]); // работает!
```

Доступ к свойствам через квадратные скобки

Ещё одно преимущество доступа через квадратные скобки – возможность передачи имени свойства через переменную.

```
let book = {
    name: "Slaughterhouse-Five",
    author: "Kurt Vonnegut",
    year: 1969
}

let whatWeWant = "author";
console.log(book[whatWeWant]); // Kurt Vonnegut

let newProperty = "genre";
book[newProperty] = "sci-fi, postmodern";
console.log(book.genre); // sci-fi, postmodern
```

Удаление свойств

Свойства объектов можно удалять с помощью оператора delete.

```
let film = {
    name: "Forrest Gump",
    director: "Robert Zemeckis",
    year: 1994
}

delete film.year;
console.log(film.year); // undefined
console.log("year" in film); // false
```

Перебор свойств объекта

Перебор всех свойств объекта осуществляется в цикле for..in.

```
let company = {
    name: "Samsung",
    country: "South Korea",
    employees: 489000
}

for (let propName in company)
    console.log(propName + ": " + company[propName]);
// name: Samsung
// country: South Korea
// employees: 489000
```

Имена всех свойств объекта поочерёдно попадают в переменную (в примере она названа propName; её имя может быть любым).

Объект – ссылочный тип данных

Переменные типа Object хранятся не так, как переменные других типов.

В переменных, значения которых имеют **примитивные** типы (string, number, boolean, null, undefined), хранятся сами значения.

В переменной, которой в качестве значения присвоен объект, **хранится не он сам**, а только **ссылка на этот объект**, то есть адрес его места в памяти.

Объект – ссылочный тип данных

Если скопировать объект из одной переменной в другую, обе будут ссылаться на один и тот же объект. Изменение свойств в одной переменной будет менять их и в другой.

```
1  let human = {
2      name: "Joe"
3  }
4
5  let sameHuman = human;
6  sameHuman.prop = "Fred";
7  console.log(human.prop); // Fred
8
9  human.surname = "Astor";
10 console.log(sameHuman.surname); // Astor
```

Объект – ссылочный тип данных

Даже если свойства двух объектов полностью идентичны (имеют одинаковые имена и одинаковые значения), эти объекты – разные.

```
let obj1 = {
    prop1: "value1",
    prop2: "value2",
}

let obj2 = {
    prop1: "value1",
    prop2: "value2",
}

console.log(obj1 === obj2); // false
```

```
let emptyOne = {};
let emptyTwo = {};

console.log(emptyOne === emptyTwo);
    // false
```

Клонирование объектов

Если вам нужно, чтобы переменная B содержала точно такой же объект, как переменная A, но не ссылалась на то же место в памяти, вы можете клонировать объект, скопировав каждое его свойство.

```
let circleA = {
    x: 24,
    y: 36,
    D: 12
};

let circleB = {};

for (let key in circleA)
    circleB[key] = circleA[key];
```

```
console.log(circleB.D); // 12

circleB.y = 10;
console.log(circleA.y); // всё ещё 36!

circleB.color = "yellow";
console.log("color" in circleA); // false
```

Практика

1. Создайте объект obj = {a: 1, b: 2, c: 3}. Выведите в консоль элемент с ключом 'c' двумя способами: через квадратные скобки и через точку. Затем выведите все свойства объекта через цикл for..in.
2. Создайте объект city, добавьте ему свойства name (название города, строка) и population (население, число).
3. Создайте массив из шести объектов такого же вида, как city из задачи 2 – по одному для каждого областного города Беларуси.
4. Напишите функцию, которая принимает массив из задачи 3 в качестве параметра и выводит в консоль информацию о каждом городе.

Объект как свойство объекта

Свойство объекта может иметь тип Object. Это позволяет выстраивать сложные вложенные структуры данных.

```
let person = {
    name: {
        first: "Alessandro",
        second: "Nesta"
    },
    birth: {
        year: 1976,
        place: "Italy"
    }
};

console.log(person.name.second); // Nesta
person.birth.place = "Rome, Italy";
```

Функция как свойство объекта

Функция тоже может быть свойством объекта. Свойства, значения которых – функции, называются методами объекта.

```
let machine = {
    type: "Coffee grinder",
    makeSound: function() {
        console.log("Drrrrrrrr!!!");
    }
};
machine.makeSound(); // Drrrrrrrr!!!
```

Функция как свойство объекта

Метод можно создать и после объявления объекта – точно так же, как любое другое свойство.

```
let cat = {
    name: "Мурзилка",
    color: "рыжий"
};

cat.sayHello = function() {
    console.log("Мур!");
}

cat.sayHello(); // Мур!
```

ES6: краткое объявление методов

Начиная с ES6 (2015 год), методы объекта можно объявлять с помощью сокращённого синтаксиса. Его поддерживают все браузеры, кроме IE.

```
let obj = {
    usefulMethod() {
        /* ... */
    },
    anotherUsefulMethod(arg1, arg2) {
        /* ... */
    }
}
// раньше было так:
let oldStyleObj = {
    usefulMethod: function() {
        /* ... */
    }
}
```

Массив – это тоже объект

Массив – это всего лишь вид объекта с числовыми именами свойств.

Он хранится в памяти особым образом, его методы созданы и оптимизированы для операций с последовательными индексами, но в целом он ведёт себя как обычный объект (например, хранится по ссылке).

Массив – это тоже объект

```
1  let arr = [1, 2, 3, 4];
2  console.log(typeof arr); // object
3
4  let arr2 = arr;
5  arr2[2] = 16;
6  console.log(arr[2]); // 16, ссылочный тип
7
8  arr["stringKey"] = "some value";
9      // можно добавлять свойства с любыми именами
10     // (но лучше не нужно)
11
12 console.log(arr[2] === arr["2"]); // true
13     // на самом деле, индексы хранятся как имена типа String
14     // и при обращении через [скобки]
15     // значения в скобках неявно приводятся к String
```

Функция – это тоже объект

Функция – ещё один специальный тип объекта. Все функции можно передавать в качестве значений, и у них есть встроенные методы и свойства.
Это одна из важных особенностей языка JavaScript: за счёт такого поведения язык может реализовывать функциональную парадигму программирования[1] [2].

[1] И это очень круто!
[2] Но мы поговорим об этом позже

Использование this в методах объекта

Чтобы получить внутри метода доступ к текущему объекту, используется ключевое слово this.

```
1  let user = {
2      name: "Олег",
3      introduce: function() {
4          console.log("Меня зовут " + this.name);
5      }
6  };
7
8  user.introduce(); // Меня зовут Олег
```

this в методах объекта

this в методах объекта – всегда ссылка на сам объект.

```
1  let testObj = {
2      testThis: function() {
3          console.log(this === testObj);
4      }
5  }
6
7  testObj.testThis(); // true
```

Почему this, а не имя переменной?

this позволяет писать более надёжный код. Даже если имя переменной, которая ссылается на объект, изменится, ссылка в this будет верной.

```
let oldVariable = {
    test: "всё работает!",
    first: function() {
        console.log(oldVariable.test);
    },
    second: function() {
        console.log(this.test);
    }
};
```

```
let newVariable = oldVariable;
    // теперь newVariable ссылается на тот же объект
oldVariable = {};
    // а oldVariable больше на него не ссылается!

newVariable.first(); // undefined
newVariable.second(); // всё работает!
```

this можно использовать в любой функции

Даже если функция не является методом объекта, в ней можно использовать this. Его значение определится при вызове функции*.

```
1  let circleA = {x: 10, y: 12};
2  let circleB = {x: -4, y: 2};
3
4  function coords() {
5      console.log("X: " + this.x);
6      console.log("Y: " + this.y);
7  }
8  circleA.printXY = coords;
9  circleB.tellCoords = coords;
10
11 circleA.printXY();       // X: 10   Y: 12
12 circleB.tellCoords();    // X: -4    Y: 2
```

*Но подробнее об этом мы тоже поговорим позже

Внеклассное чтение

https://learn.javascript.ru/object
https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Working_with_Objects
https://learn.javascript.ru/object-methods

Практика

5. Создайте в объектах с городами из задачи 3 метод getInfo, который возвращает строку с информацией о городе (например, в таком формате: "Город Добруш, население – 18760 человек").

6. Создайте объект с информацией о себе (имя, фамилия, любимое занятие). Добавьте в него метод, который выводит эту информацию в консоль в удобочитаемом формате.