ОБЪЕКТНЫЙ ТИП

ОБЪЕКТЫ

СОЗДАНИЕ ОБЪЕКТОВ

```javascript
let obj = {
  str: "value",
  num: 1,
  fn: function() { /* ... */ },
  bool: true,
  unknown: null,
  arr: [1, 2],
  obj: {
    key: "value"
  },
  "some other key": 1
};

let emptyObj = {};
let constructedObj = new Object();
let absolutelyEmpty = Object.create(null);
let str = "white space",
    fn = function() { /* ... */ },
    star = "★";

let obj = {
  str,
  fn,
  [star]: star,
  [`a star ${star}`]: star,
  someFunction() {
    // return expression;
  }
};
```

ДОСТУП К СВОЙСТВАМ ОБЪЕКТОВ

```
let obj = {
  age: 50,
  name: "Geogre",
  "last name": "Smith",
  isMarried: true
};

obj.name;                                    // "George"
obj["last name"];                            // "Smith"
obj["last " + "name"];

let age = obj.age;
console.log(age);                            // 50

obj.isMarrie                                 // undefined
```

ИЗМЕНЕНИЕ И ДОБАВЛЕНИЕ СВОЙСТВ

```
let obj = {
  key: 1
},
    str = "some string";

obj.newStr = str,
obj["prop"] = 2,
obj["prop #" + obj.key] = "prop #1";

obj.key = 5;

obj.someProp.val = 1;         // Uncaught TypeError:
                              // Cannot set property 'val' of undefi
```

ПОНЯТИЕ МЕТОД. СОЗДАНИЕ И ВЫЗОВ МЕТОДОВ

```
let o = {
  someMethod: function(param) {
    return param;
  },
  anotherMethod(param) {
    return param;
  }
};

let str = "string";
o.getString = function() {
  return str;
};
```

УДАЛЕНИЕ СВОЙСТВ

```
let obj = {
  a: 1,
  b: 2
};

delete obj.a
console.log(obj.a);
console.log(obj);

delete obj.b
console.log(obj.b);

delete obj.c
```

ПРОВЕРКА НА НАЛИЧИЕ СВОЙСТВА

```
let obj = {
  prop: 1,
  str: "hello",
  undef: undefined,
  empty: null
};

obj.prop;                                   // 1
obj.props;                                  // undefined

if(!obj.props) obj.props();
if(typeof obj.props !== "undefined") obj.props();

obj.undef;                                  // undefined
```

```
let obj = {
  prop: 1,
  undef: undefined,
  empty: null
};

"prop" in obj                     // true
"empty" in obj                    // true
"undef" in obj                    // true
"props" in obj                    // false

props in obj                      // Uncaught ReferenceError:
                                  // props is not defined
// проверка на наличие свойства
// исключающая наследованные свойства

let obj = Object.create({prop: "value"});
obj.ownProp = "own";

obj.prop;                                 // "value"
"prop" in obj                             // true

obj.hasOwnProperty("prop");               // false
obj.hasOwnProperty("ownProp");            // true
```

КЛЮЧЕВОЕ СЛОВО THIS

```javascript
let obj = {
  key: "some value",
  getKey() {
    return obj.key;
  },
  setKey(newValue) {
    obj.key = newValue;
  }
};

console.log(obj.key);
console.log(obj.getKey());

obj.setKey("new value");

console.log(obj.getKey());
```

```javascript
let obj = {
  key: "some value",
  getKey() {
    return this.key;
  },
  setKey(newValue) {
    this.key = newValue;
  }
};

console.log(obj.key);
console.log(obj.getKey());

obj.setKey("new value");

console.log(obj.getKey());
```

```
let obj = {
  key: "some value",
  innerObj: {
    key: "inner key",
    getKey() {
      return this.key;
    }
  }
};

console.log(obj.key);
console.log(obj.innerObj.getKey());
```

```
let firstObj = {
  key: "first object",
  getKey
};

let secondObj = {
  key: "second object",
  getKey
};

function getKey() {
  return this.key;
}

console.log(firstObj.getKey());
console.log(secondObj.getKey());
```

НЕПРЯМОЙ ВЫЗОВ МЕТОДОВ

CALL, APPLY, BIND

call, apply

```
let firstObj = {
  key: "first object",
  getKey
};

let secondObj = {
  key: "second object",
  getKey
};

function getKey(...someArg) {
  return `${this.key} and ${someArg.join("; ")}`;
}

firstObj.getKey.call(secondObj, 1, 2, 3);
secondObj.getKey.apply(firstObj, [1, 2, 3]);
```

bind

```
let firstObj = {
  key: "first object",
  getKey
};

function getKey(...someArg) {
  return `${this.key} and ${someArg.join("; ")}`;
}

let boundToFirst = getKey.bind(firstObj, "boundToFirst");

boundToFirst(1, 2, 3);
```

```
let obj = {
  a: 3
};

let ar = [1, 2, 3];

ar.forEach(function(el) {
  let result = el * this.a;
  console.log(result);
}, obj);
```

ПЕРЕБОР СВОЙСТВ ОБЪЕКТА

Метод Object.keys()

```
let obj = {
  someNumber: 1,
  someMethod: function() {},
  someArray: [1, 2, 3]
};

let objectKeys = Object.keys(obj);

objectKeys.forEach(function(el) {
  console.log(this[el]);
  this[el] = "new value";
}, obj);
```

Цикл for in

```
let obj = {
  someNumber: 1,
  someMethod: function() {},
  someArray: [1, 2, 3]
};

for(let prop in obj) {
  console.log(obj[prop]);
}
```

СВОЙСТВА АКСЕССОРЫ (ГЕТТЕРЫ И СЕТТЕРЫ)

```
let keyName = "☘";
let obj = {
  get someKey() {
    return "someKey";
  },
  get [keyName]() {
    return keyName;
  }
};

obj.someKey;
obj[keyName];
```

```javascript
let keyName = "☘";
let obj = {
  [`_${keyName}`]: 0,
  set someKey(value) {
    console.log(`attempt to set value ${value}`);
  },
  set [keyName](value) {
    this[`_${keyName}`] = value;
  }
};

obj.someKey = 1;
obj[keyName] = 3;
```

```
let obj = {
  _prop: "less than 20 chars",
  get prop() {
    return this._prop;
  },
  set prop(val) {
    this._prop = val.length < 20 && typeof val === "string" ?
                 val :
                 this._prop;
  }
};

obj.prop = 10;
console.log(obj.prop);
obj.prop = "Hello";
console.log(obj.prop);
```

СВОЙСТВА ДЕСКРИПТОРЫ

```
let obj = {
  prop: "prop value",
  get anotherProp() {},
  set anotherProp() {}
};

Object.getOwnPropertyDescriptor(obj, "prop");
Object.getOwnPropertyDescriptor(obj, "anotherProp");

// value
// writable
// enumerable
// configurable
```

```
Object.defineProperty(obj, "name", {
  writable: false,
  enumerable: true,
  configurable: true
});
```

```javascript
let obj = {a: 1};
Object.defineProperties(obj, {
  b: {
    value: "it's b",
    writable: true,
    enumerable: true,
    configurable: false
  }
});

console.log(obj.b);
delete obj.b;
console.log(obj.b);
```

```javascript
let obj = {};
Object.defineProperties(obj, {
  b: {
    get() {
      console.log("it's getter");
    },
    set(value) {
      console.log("it's setter");
    },
    enumerable: false
  }
});

Object.keys(obj);                     // []
obj.hasOwnProperty("b");              // true
obj.propertyIsEnumerable("b");        // false
```

МЕТОДЫ КОНСТРУКТОРА OBJECT

```
let obj = {
  a: 1
};

Object.preventExtensions(obj);
Object.isExtensible(obj);
obj.b = 2;

console.log(obj.b);
```

```
let obj = {
  a: 1
};

Object.seal(obj);
Object.isSealed(obj);
delete obj.a;

console.log(obj.a);
```

```
let obj = {
  a: 1
};

Object.freeze(obj);
Object.isFrozen(obj);

obj.a = 2;
console.log(obj.a);

delete obj.a
console.log(obj.a);
```

```
let parentObj = {a: 1};
let childObj = Object.create(parentObj);

console.log(childObj.a);                // 1

Object.preventExtensions(childObj);
childObj.b = 3;
console.log(childObj.b);                // undefined

parentObj.b = 2;

console.log(childObj.b);                // 2
```

```
let obj = {a: 1};
Object.defineProperties(obj, {
  b: {
    value: 2,
    enumerable: false
  }
});

Object.keys(obj);                          // ["a"]
Object.getOwnPropertyNames(obj);           // ["a", "b"]
```

```
// объекты - ссылочный тип
let obj = {a: 1};
let copy = obj;

copy.a = 5;
console.log(obj.a);                      // 5
```

```javascript
// Object.assign(target, ...sources)

let obj = {a: 1};
let copy = {};

Object.assign(copy, obj);

copy.a = 5;
console.log(obj.a);
```

ПОЛЕЗНЫЕ ССЫЛКИ

- Learn Javascript by Example
- Codecademy