**[Работа с прототипами](https://learn.javascript.ru/prototype-inheritance" \l "rabota-s-prototipami)**

важность: 5

В приведённом ниже коде создаются и изменяются два объекта.

Какие значения показываются в процессе выполнения кода?

let animal = {

jumps: null

};

let rabbit = {

\_\_proto\_\_: animal,

jumps: true

};

alert( rabbit.jumps ); // ? (1)

delete rabbit.jumps;

alert( rabbit.jumps ); // ? (2)

delete animal.jumps;

alert( rabbit.jumps ); // ? (3)

Должно быть 3 ответа.

Ответ:

1. True
2. Null
3. undefined

### [Алгоритм поиска](https://learn.javascript.ru/prototype-inheritance" \l "algoritm-poiska)

важность: 5

Задача состоит из двух частей.

У нас есть объекты:

let head = {

glasses: 1

};

let table = {

pen: 3

};

let bed = {

sheet: 1,

pillow: 2

};

let pockets = {

money: 2000

};

1. С помощью свойства \_\_proto\_\_ задайте прототипы так, чтобы поиск любого свойства выполнялся по следующему пути: pockets → bed → table → head. Например, pockets.pen должно возвращать значение 3 (найденное в table), а bed.glasses – значение 1 (найденное в head).
2. Ответьте на вопрос: как быстрее получить значение glasses – через pockets.glasses или через head.glasses? При необходимости составьте цепочки поиска и сравните их.

Ответ:

let head = {

glasses: 1

};

let table = {

pen: 3,

\_\_proto\_\_: head

};

let bed = {

sheet: 1,

pillow: 2,

\_\_proto\_\_: table

};

let pockets = {

money: 2000,

\_\_proto\_\_: bed

};

С точки зрения производительности, для современных движков неважно, откуда берётся свойство – из объекта или из прототипа. Они запоминают, где было найдено свойство, и повторно используют его в следующем запросе.

### [Куда будет произведена запись?](https://learn.javascript.ru/prototype-inheritance" \l "kuda-budet-proizvedena-zapis)

важность: 5

Объект rabbit наследует от объекта animal.

Какой объект получит свойство full при вызове rabbit.eat(): animal или rabbit?

let animal = {

eat() {

this.full = true;

}

};

let rabbit = {

\_\_proto\_\_: animal

};

rabbit.eat();

Ответ: rabbit

### [Почему наедаются оба хомяка?](https://learn.javascript.ru/prototype-inheritance" \l "pochemu-naedayutsya-oba-homyaka)

важность: 5

У нас есть два хомяка: шустрый (speedy) и ленивый (lazy); оба наследуют от общего объекта hamster.

Когда мы кормим одного хомяка, второй тоже наедается. Почему? Как это исправить?

let hamster = {

stomach: [],

eat(food) {

this.stomach.push(food);

}

};

let speedy = {

\_\_proto\_\_: hamster

};

let lazy = {

\_\_proto\_\_: hamster

};

// Этот хомяк нашёл еду

speedy.eat("apple");

alert( speedy.stomach ); // apple

// У этого хомяка тоже есть еда. Почему? Исправьте

alert( lazy.stomach ); // apple

Ответ:

И при lazy.stomach.push(...) и при speedy.stomach.push(), свойство stomach берётся из прототипа (так как его нет в самом объекте), затем в него добавляются данные.

Можно к каждому хомяку добавить stomach: []