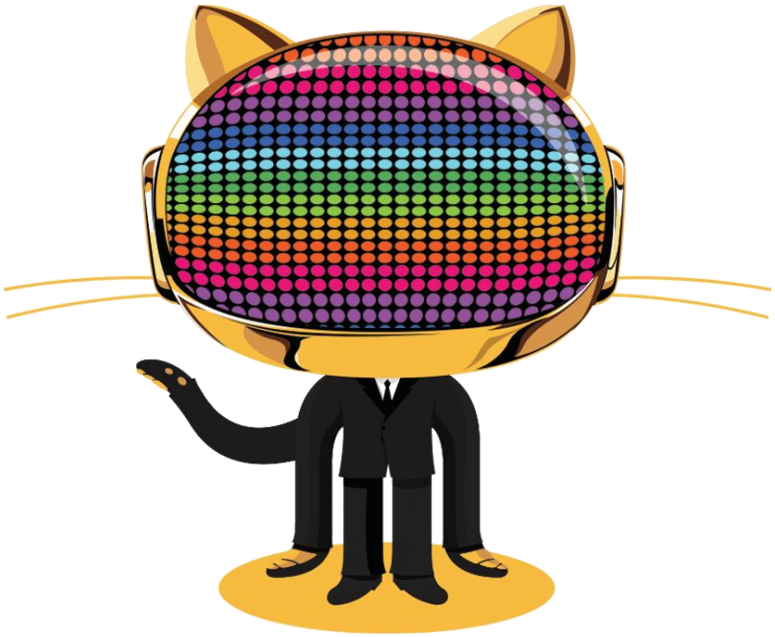


JavaScript:
git, функции, объекты, массивы

Tinkoff.ru

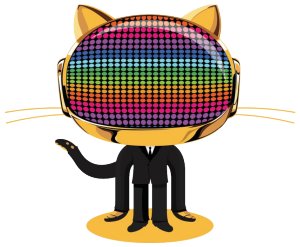
Советы по ДЗ

Не пишите обфусцированный код. Имена должны быть понятны, читабельны.

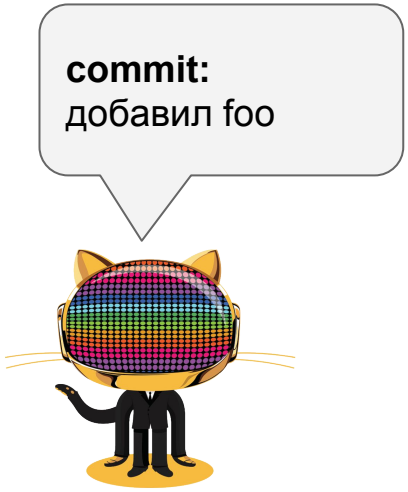
Видите, что тесты не покрывают все случаи — напишите дополнительный тест.

Git

Git



Git: коммит



file.js

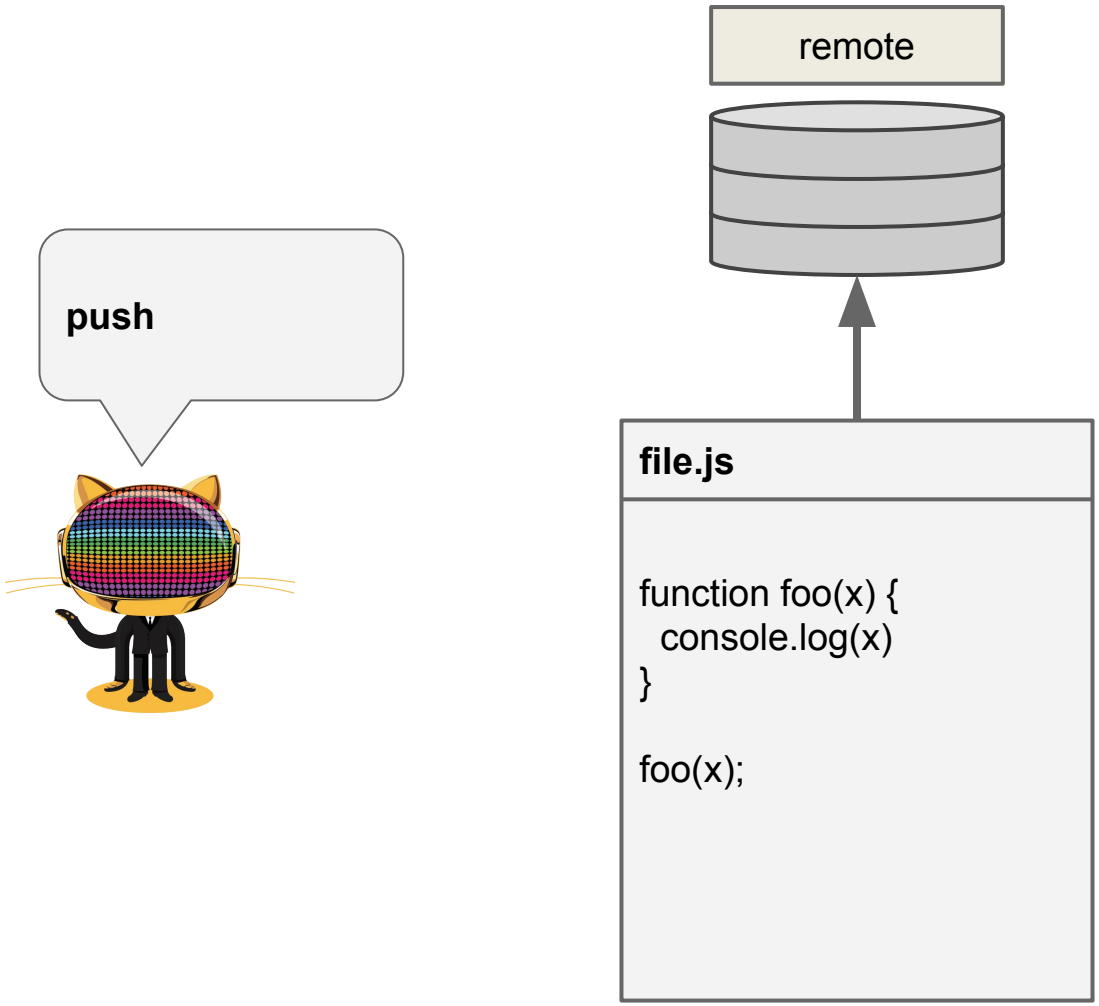
```
function foo(x) {
  console.log(x)
}
```

Git: коммит

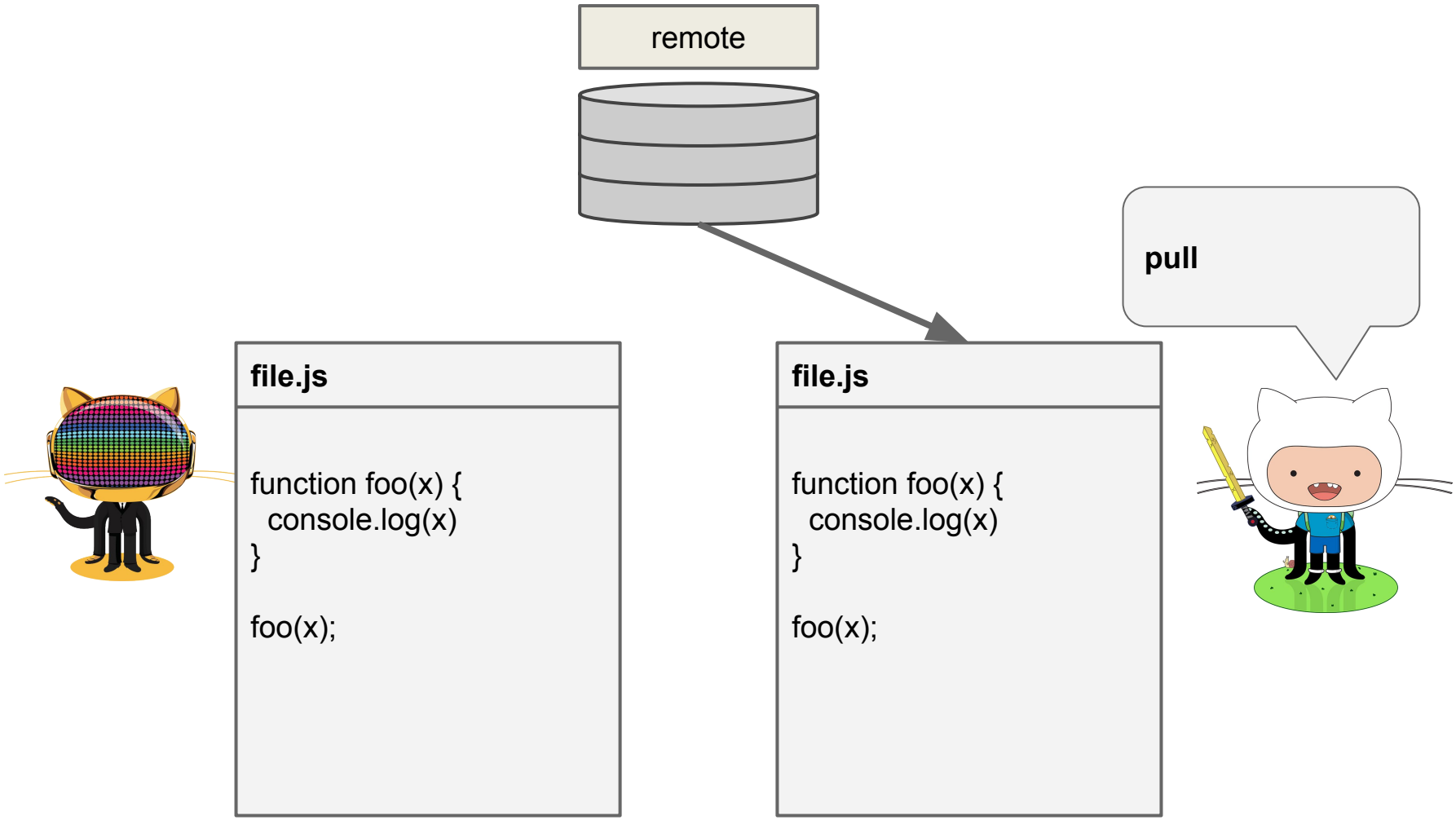
file.js

```
function foo(x) {
  console.log(x)
}

foo(x);
```

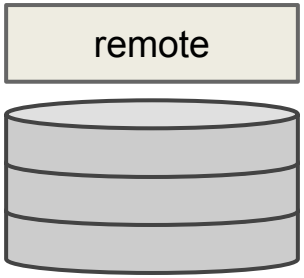
Git: remote

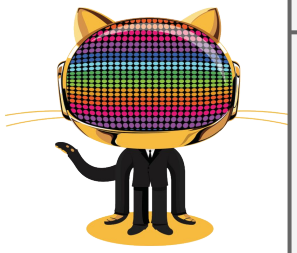


Git: remote


remote
pull
file.js
function foo(x) {
console.log(x)
}
foo(x);
file.js
function foo(x) {
console.log(x)
}
foo(x);

Git: взаимодействие





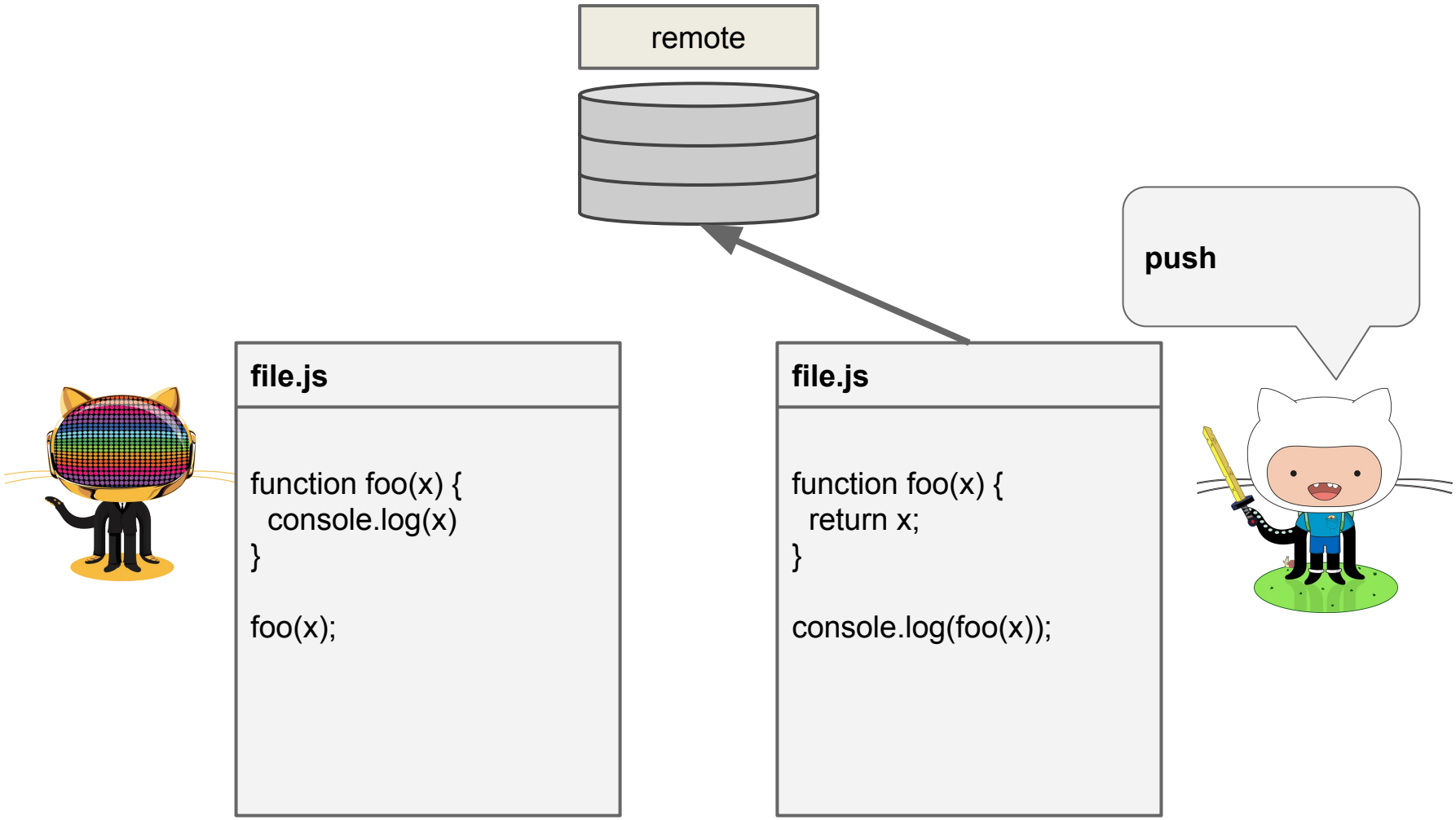
file.js

```
function foo(x) {
  console.log(x)
}

foo(x);
```

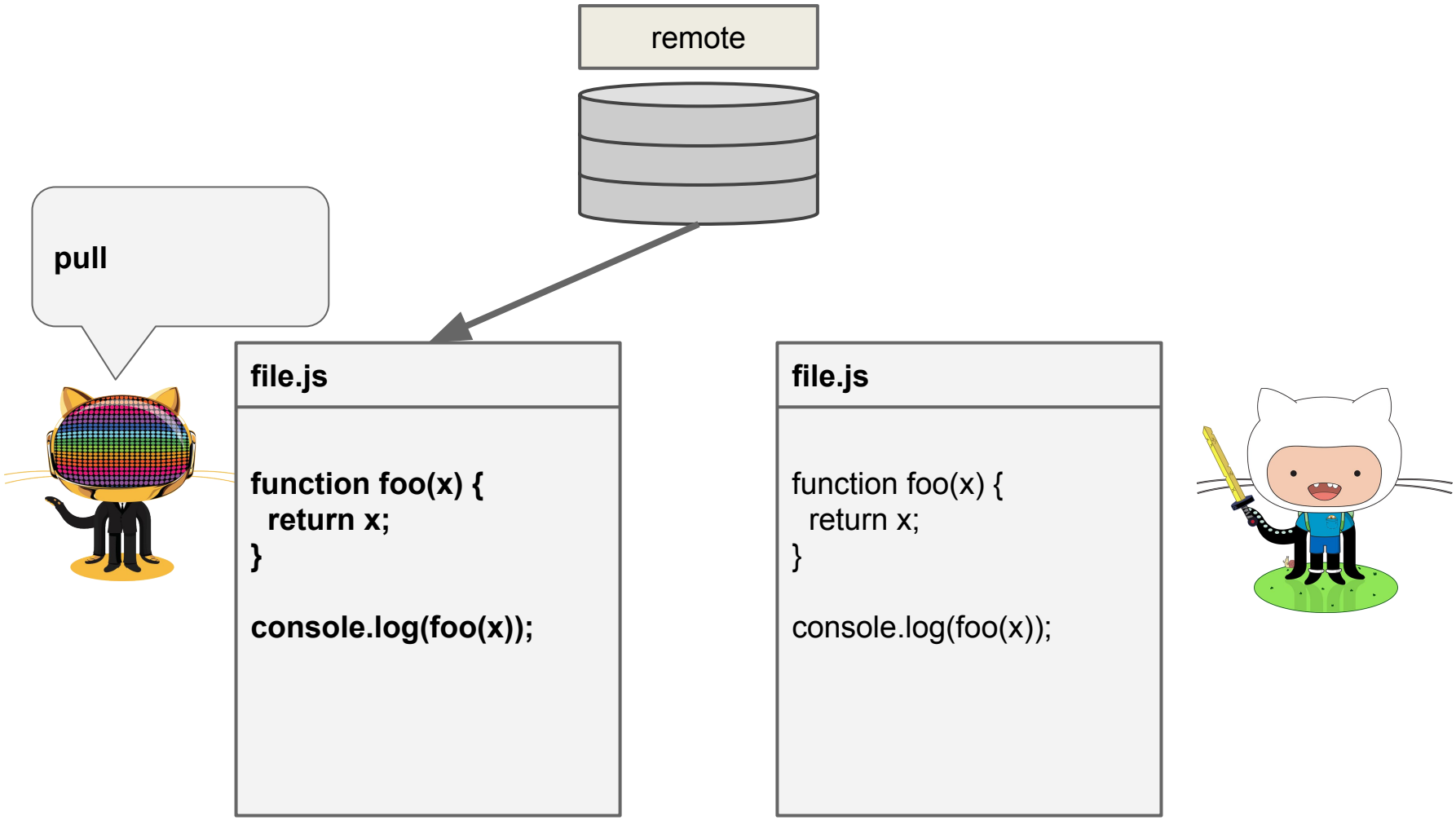
file.js

```
function foo(x) {
  console.log(x)
  return x;
}

foo(x);
console.log(foo(x));
```

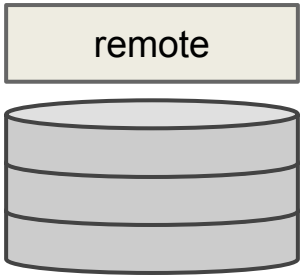
Git: взаимодействие



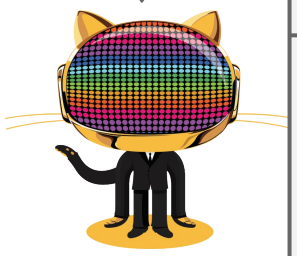
Git: взаимодействие



Git: конфликты



commit:
переимовал

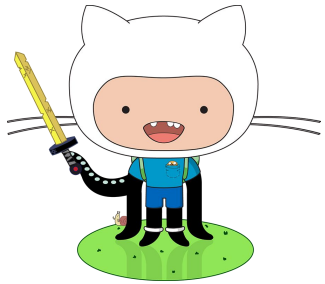


file.js

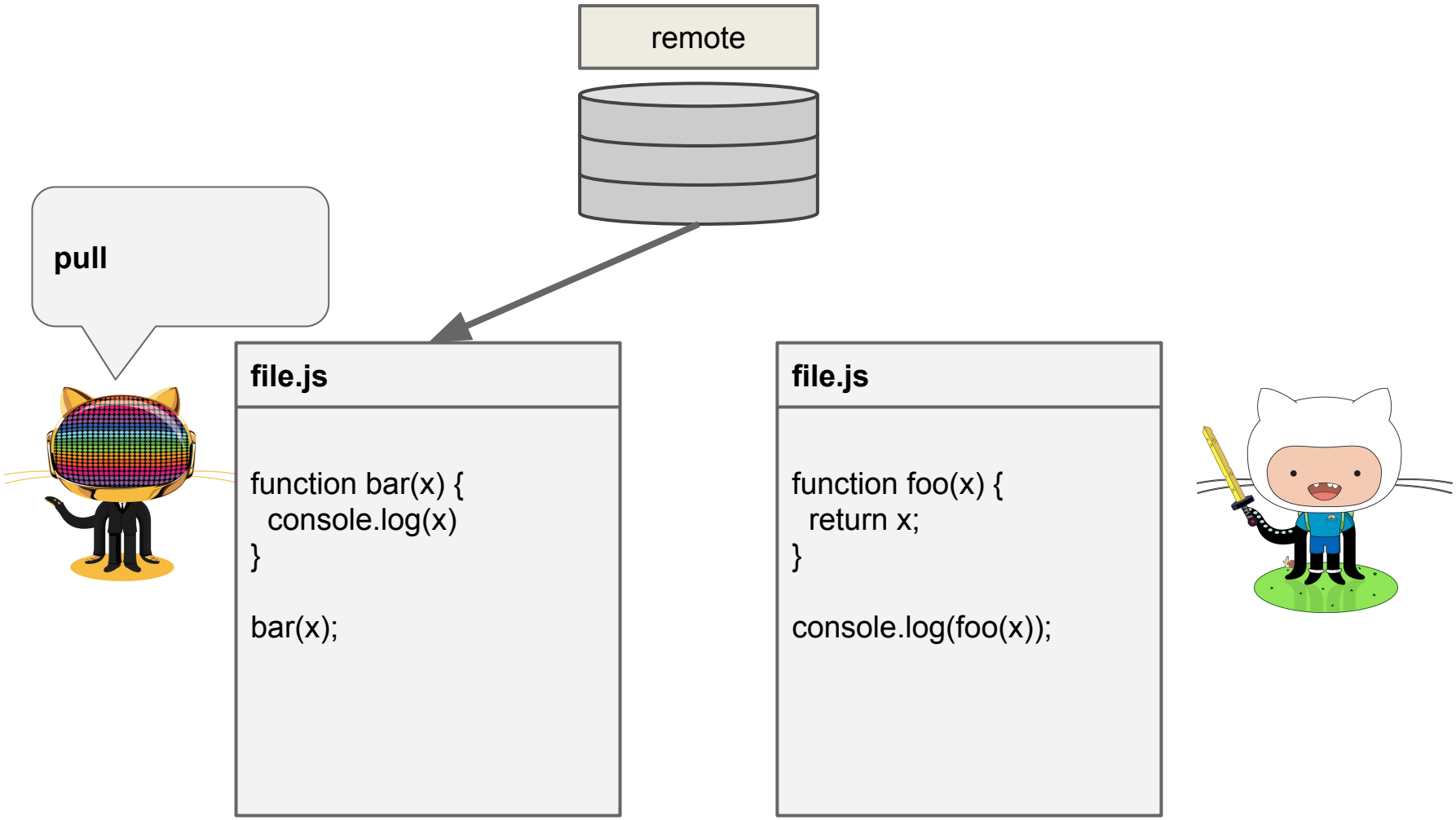
```
function foo(x) {
function bar(x) {
  console.log(x)
}

foo(x);
bar(x);
```

file.js

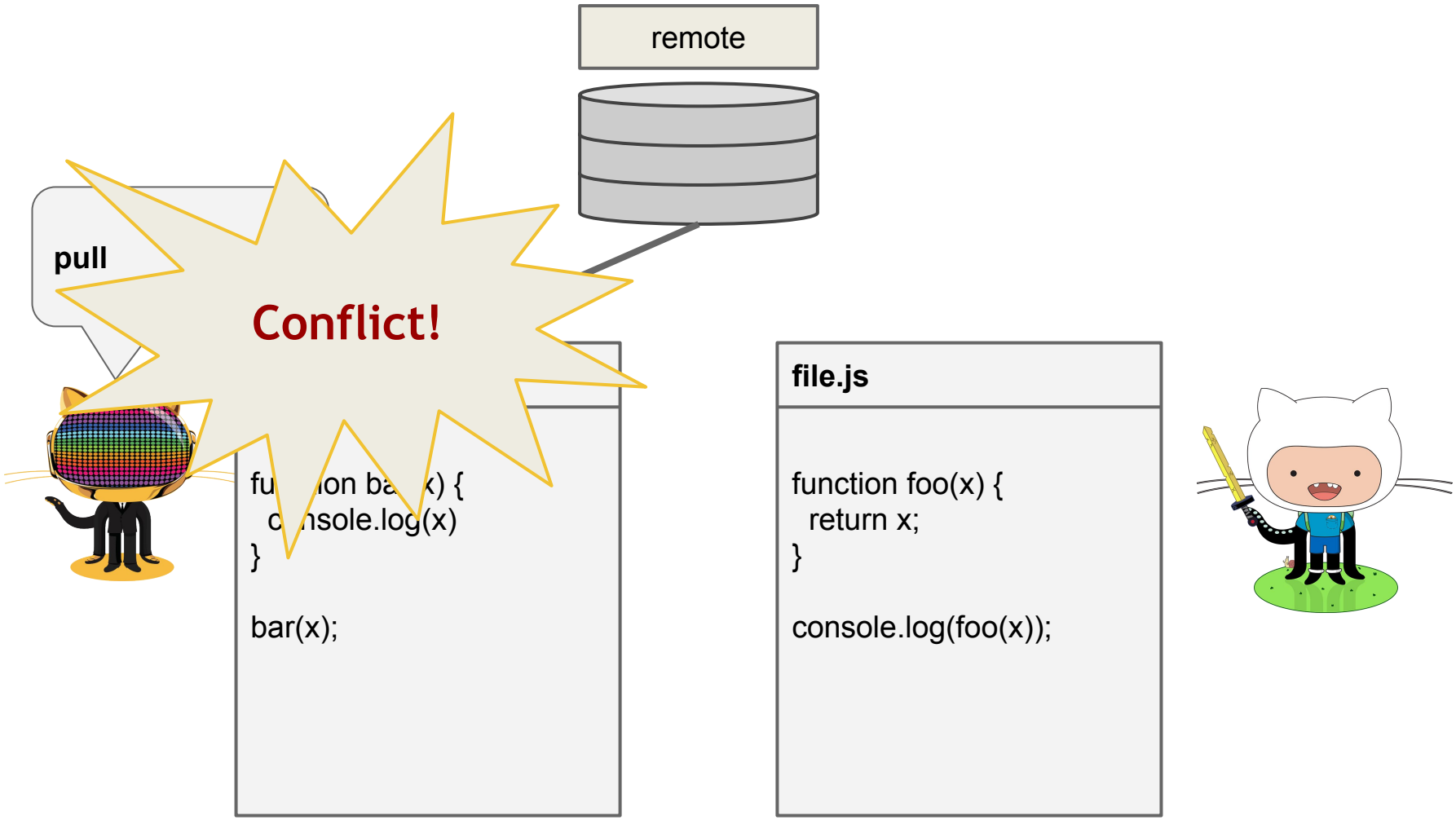
```
function foo(x) {
  return x;
}

console.log(foo(x));
```



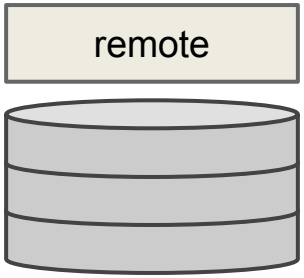
Git: конфликты



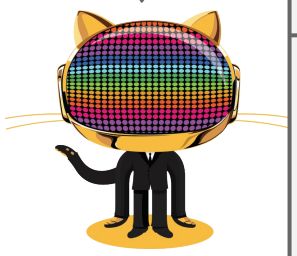
Git: конфликты



Git: конфликты



commit:
fix conflicts

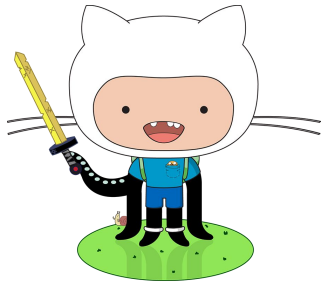


file.js

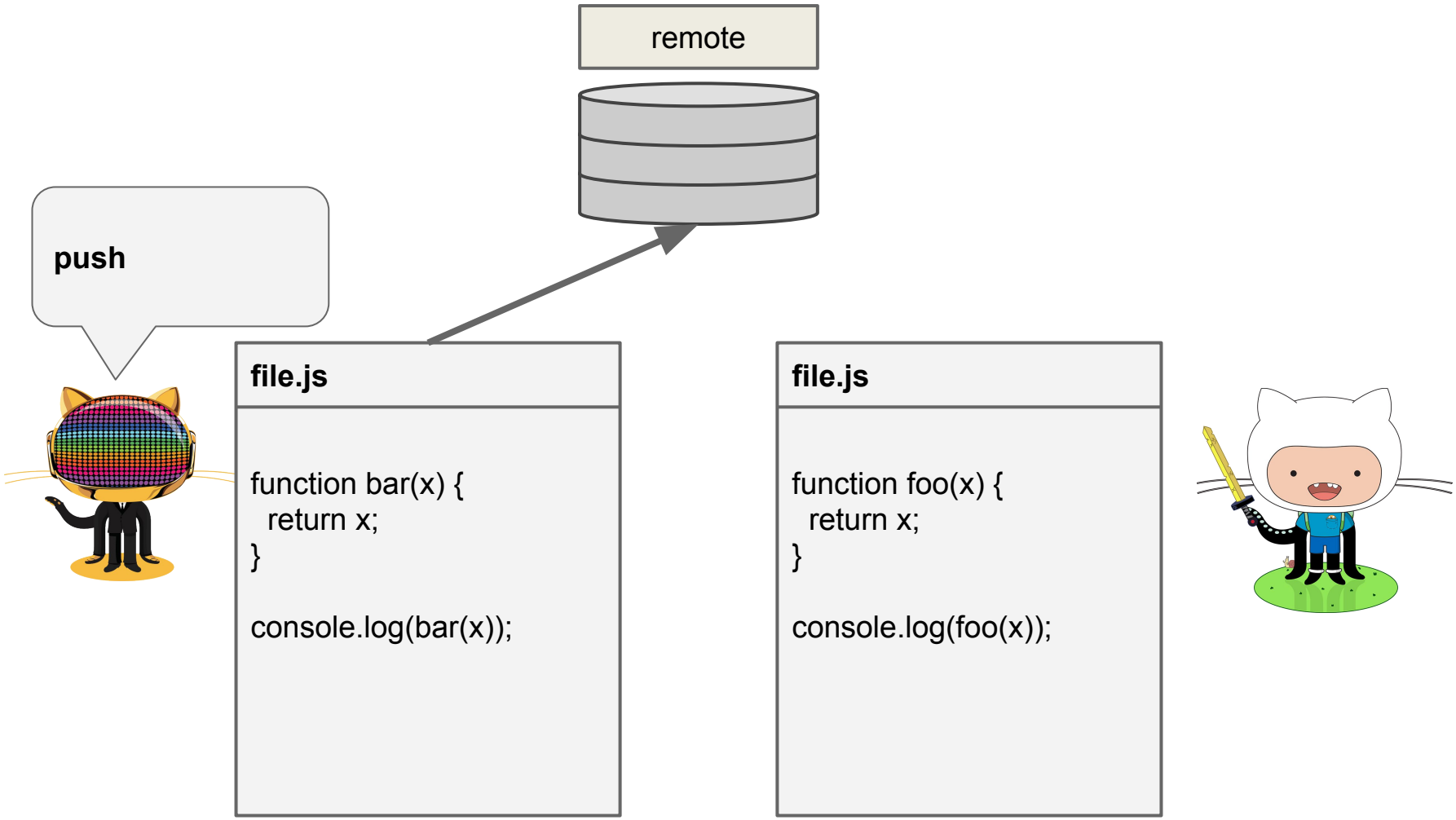
```
function foo(x) {
function bar(x) {
  console.log(x)
  return x;
}

bar(x);
console.log(bar(x));
```

file.js

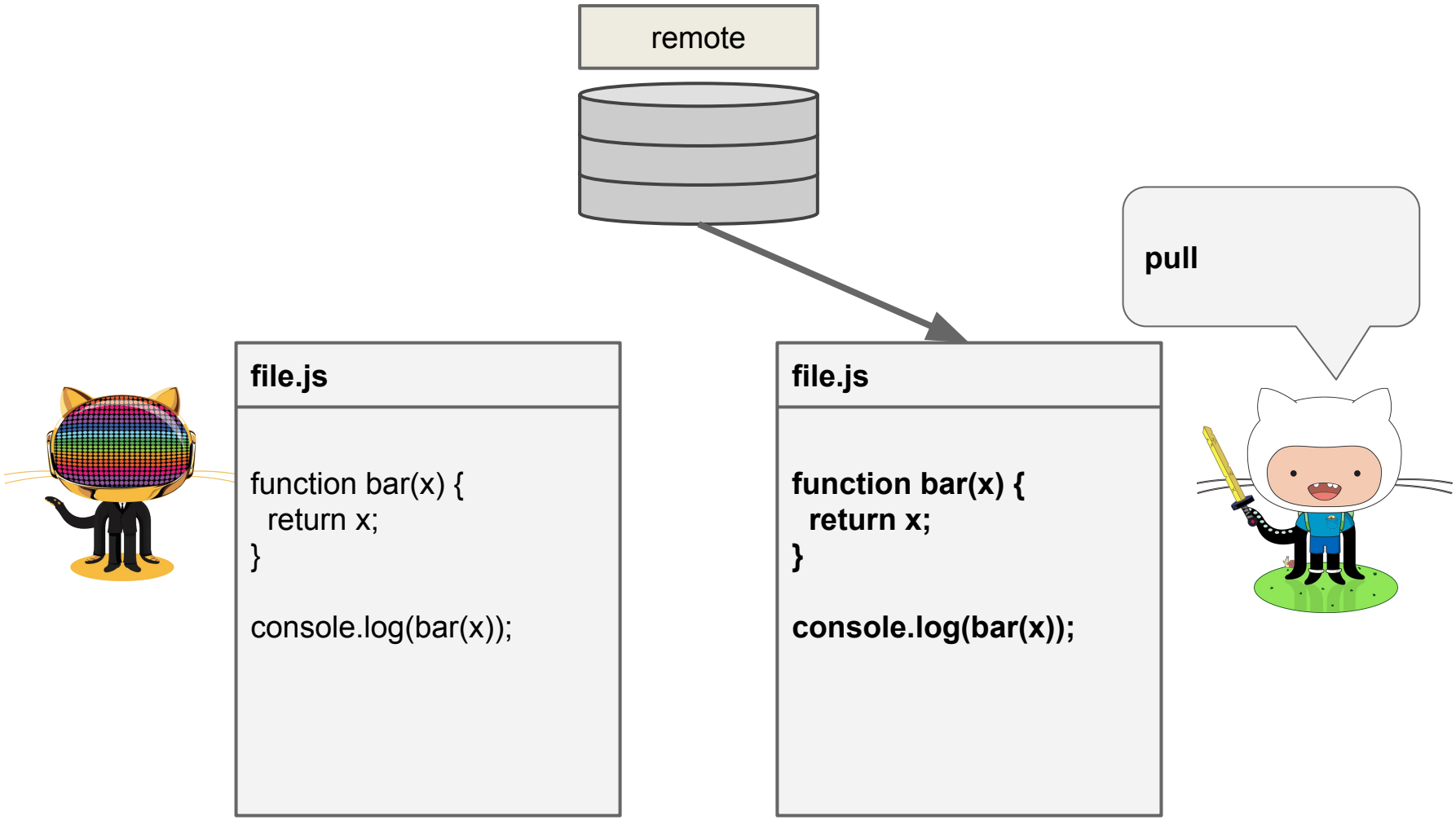
```
function foo(x) {
  return x;
}

console.log(foo(x));
```



Git: конфликты



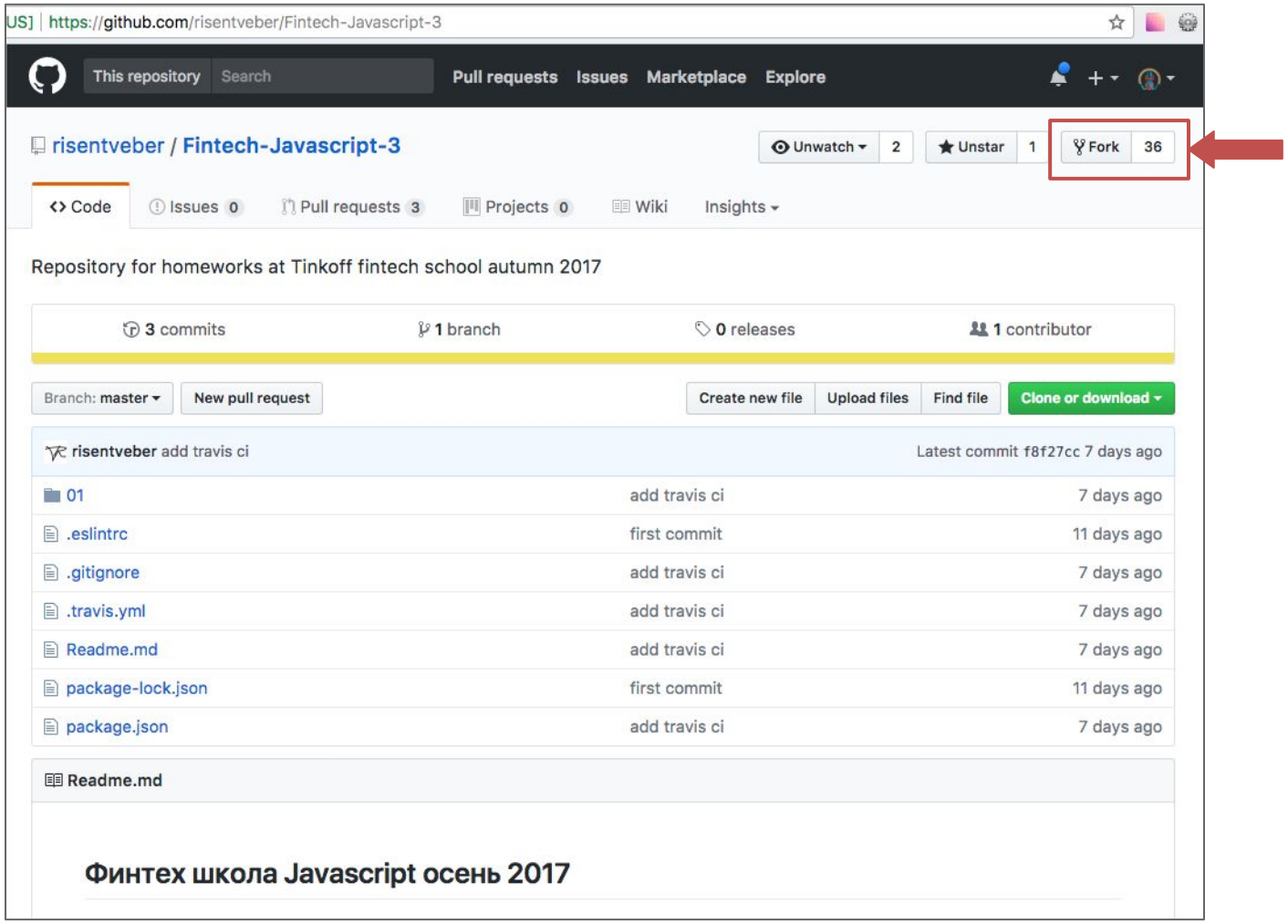
Git: конфликты



РАБОТА С GIT НА КУРСЕ

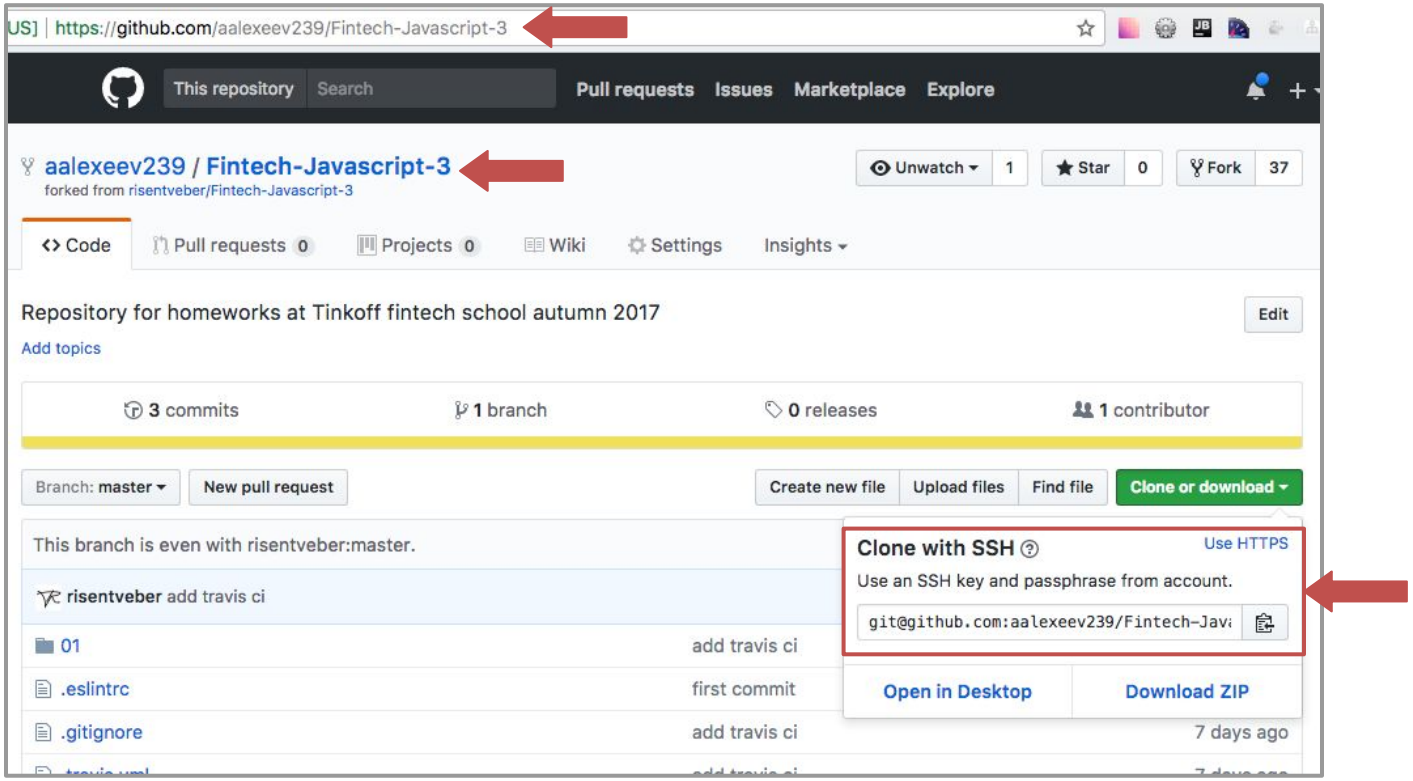
Сделайте форк

https://github.com/risentveber/Fintech-Javascript-3



Склонируйте свой форк

```
git clone git@github.com:aalexeev239/Fintech-Javascript-3.git
cd Fintech-Javascript-3/
```



Создайте ветку урока

```
git checkout -b lesson-01
```

Внесите изменения, закоммитьте их

git commit -am 'first task'

```
1  /**
2   * найдите минимум и максимум в любой строке
3   * @param  {string} string входная строка(числа отделены от других частей строки п
4   * @return {{min: number, max: number}} объект с минимумом и максимумом
5   * '1 и 6.45, -2, но 8, а затем 15, то есть 2.7 и -1028' => { min: -1028, max: 15
6   */
7  function getMinMax(string) {
8    return str
9      .split(' ')
10     .map(word => parseFloat(word))
11     .filter(word => !isNaN(word))
12     .reduce(({min, max}, current) => ({
13       min: Math.min(min, current),
14       max: Math.max(max, current)
15   }), {min: Infinity, max: -Infinity});
16 }
17
18 /* ================================================== */
19
```

Отправьте изменения

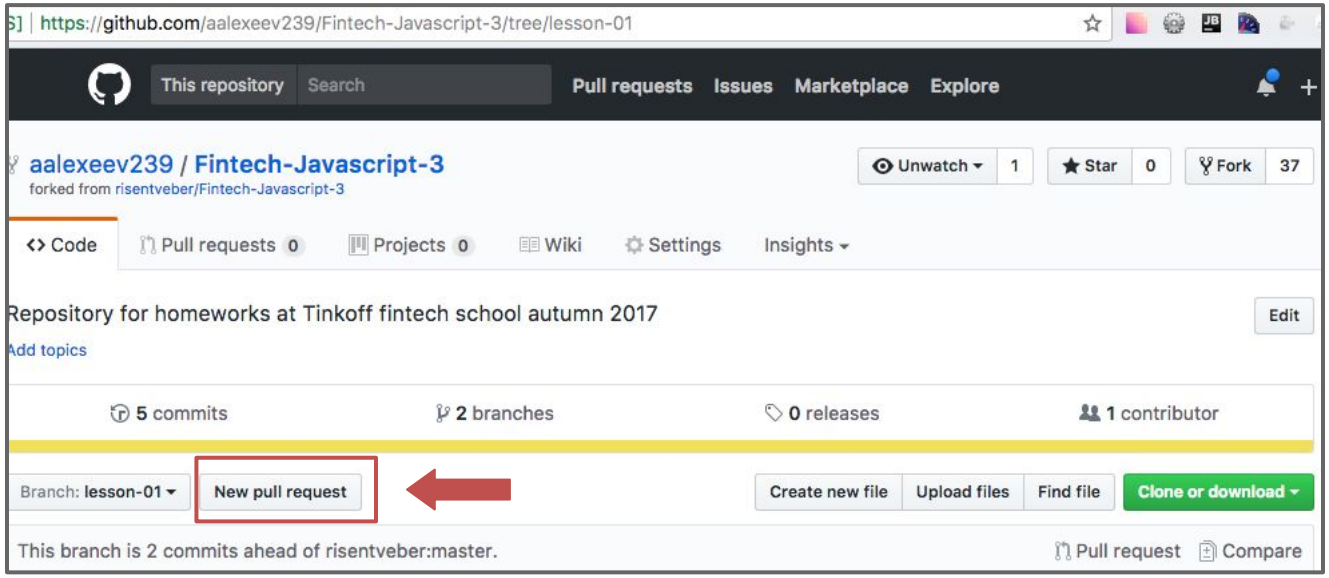
git push
fatal: The current branch lesson-01 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin lesson-01

git push --set-upstream origin lesson-01

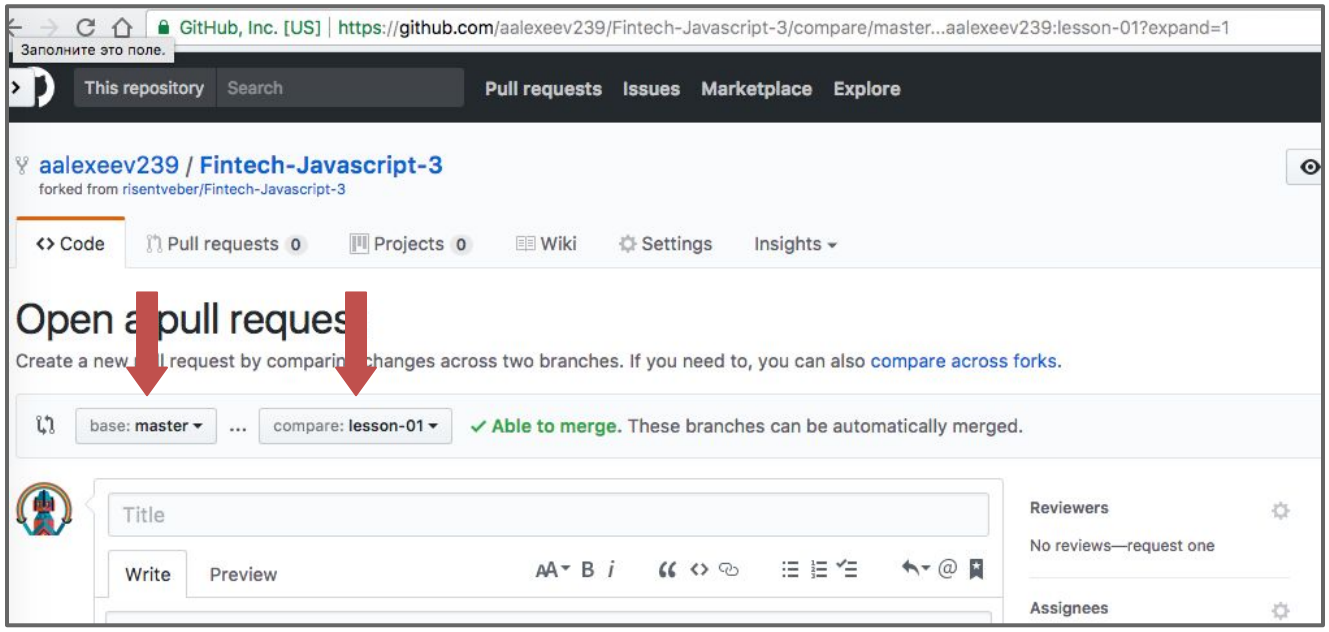
Создаем бранч на сервере через --set-upstream

Создайте ПР из ветки lesson-01



lesson-01→ ваш master

Нужно выбрать именно вашу ветку master



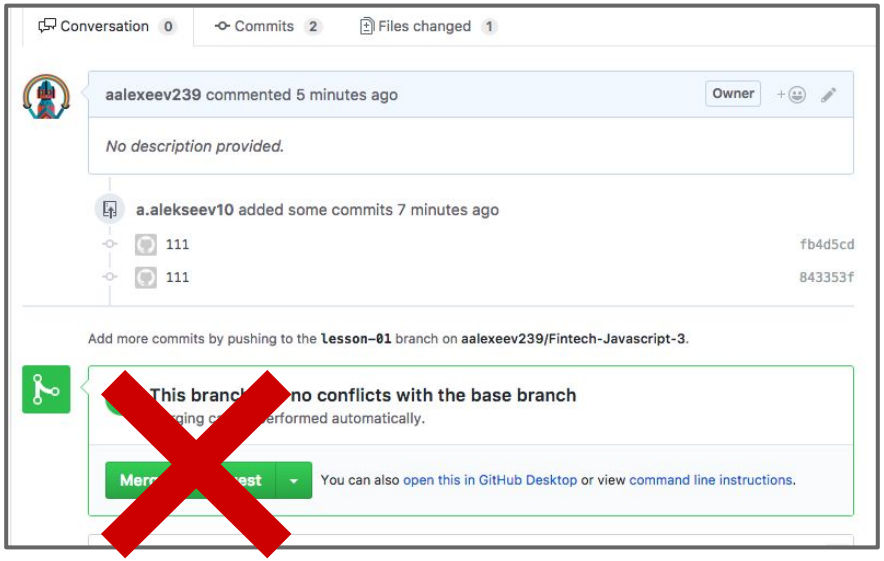
НЕ НАЖИМАЙТЕ MERGE

Для выполнения заданий ветки мержить не нужно.

Совсем.

Следующее задание будет делаться в отдельной ветке.

Дополнительные коммиты в ветку lesson-01 будут автоматически подключаться к этому ПР.



Получение изменений с основного репозитория

Добавьте основной репозиторий

```
git remote add upstream git@github.com:risentveber/Fintech-Javascript-3.git
```

Переключитесь на локальную ветку master

```
git checkout master
```

Скачайте изменения из master-ветки **upstream**

```
git pull upstream master
```

и запушьте в свой репозиторий

```
git push origin master
```

Создайте ветку для следующего задания

```
git checkout -b lesson-xx
```

ФУНКЦИИ

Глобальный объект

```
var a = 5;
console.log(window.a); // 5;
```

https://learn.javascript.ru/global-object

Проблема пространства имен

```
// A.js

function funcA () {
   console.log(helper());
}

function helper () {
   return 'A';
}
```

```
// B.js

function funcB () {
   console.log(helper());
}

function helper () {
   return 'B';
}
```

Проблема пространства имен

```
// A.js

function funcA () {
   console.log(helper());
}

function helper () {
   return 'A';
}
```

```
// B.js

function funcB () {
   console.log(helper());
}

function helper () {
   return 'B';
}
```

```
<script src="A.js"></script>
<script src="B.js"></script>
<script>
   funcA();
   funcB();
</script>
```

Проблема пространства имен

```
// A.js

function funcA () {
   console.log(helper());

   function helper () {
      return 'A';
   }
}
```

```
// B.js

function funcB () {
   console.log(helper());

   function helper () {
      return 'B';
   }
}
```

```
<script src="A.js"></script>
<script src="B.js"></script>
<script>
   funcA();
   funcB();
</script>
```

IIFE

Immediately-Invoked Function Expression

```
(function () {
  function funcA() {
    console.log(helper());
  }

  function helper() {
    return 'A';
  }
})();
```

http://getinstance.info/articles/javascript/immediately-invoked-function-expressions/

Строгий режим

– исправляет многие «вольности» js

– объявляется в начале файла или в начале функции

```
"use strict";

function abc() {
   a = 5;
}
```

http://frontender.info/why-use-strict-in-javascript-can-save-you-hours/
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Strict_mode

Псевдомассив Arguments

```
function hey() {
  for (var i = 0; i < arguments.length; i++) {
    console.log(`Эй, ${arguments[i]}!`);
  }
}

hey('Вася', 'Маша', 'Петя');
// Эй, Вася!
// Эй, Маша!
// Эй, Петя!
```

https://learn.javascript.ru/arguments-pseudoarray

ES6: ...spread

```
function heyYou(you, ...args) {
 console.log(`Эй, ${you}!`);
 console.log(`Эй, ${args.join(', ')}!`);
}

heyYou('Вася', 'Маша', 'Петя');
// Эй, Вася!
// Эй, Маша, Петя!
```

http://learn.javascript.ru/es-function

ES6: ...spread

```
const array = [1, 2, 3, 4];
const min = Math.min(...array);
const max = Math.max(...array);
```

ES6: => и arguments

```
const hello = () => console.log(`Здравствуйте, ${[...arguments].join(', ')}`);
hello('Вася', 'Маша', 'Петя');
// Uncaught ReferenceError: arguments is not defined
```

ES6: => и arguments

```
const hello = () => console.log(`Здравствуйте, ${[...arguments].join(', ')}`);
hello('Вася', 'Маша', 'Петя');
// Uncaught ReferenceError: arguments is not defined

const helloFixed = (...args)=>
console.log(`Здравствуйте, ${args.join(', ')}`);
// Здравствуйте, Вася, Маша, Петя!
```

ES6: => и arguments

```
function helloWrap() {
 const hello = () => console.log(`Здравствуйте, ${[...arguments].join(', ')}!`);

 hello();
}

helloWrap('Вася', 'Маша', 'Петя');
// Здравствуйте, Вася, Маша, Петя!
```

ES6: defaults/значения по умолчанию

```
function helloDefaults(user, greeting = 'Здравствуйте') {
 console.log(`${greeting}, ${user}!`);
}

helloDefaults('Вася', 'Эй'); // Эй, Вася!
helloDefaults('Василий'); // Здравствуйте, Василий!
```

ES6: деструктуризация аргументов

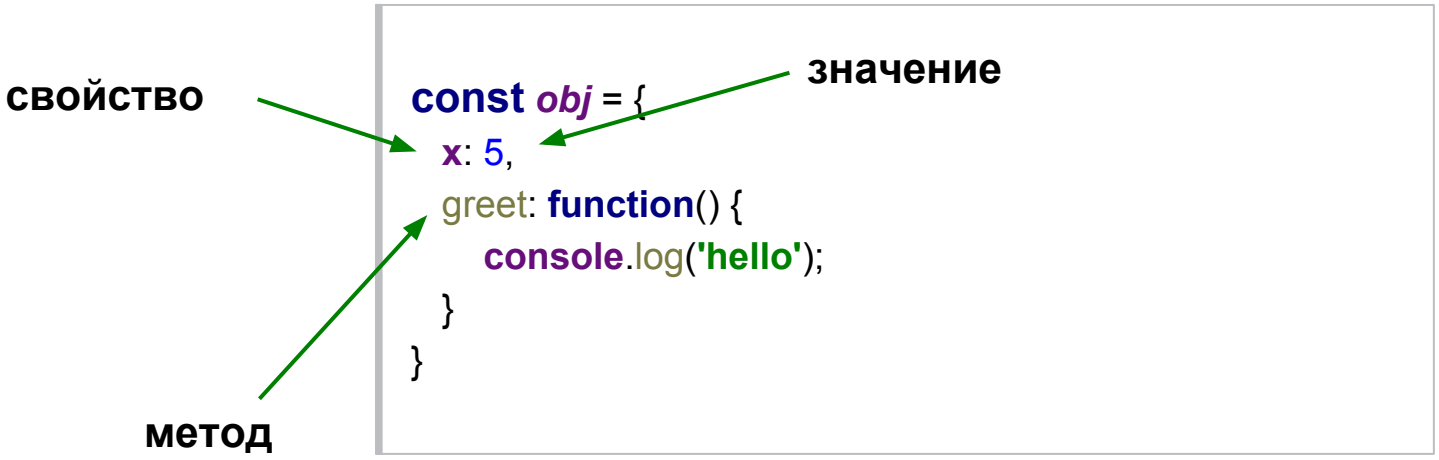
```
function helloObject({name, job}) {
 console.log(`Привет! Меня зовут ${name}, я ${job}.`);
}

helloObject({
 name: 'Олег',
 job: 'самый главный',
 officeAddress: 'Головинское ш. 5/1'
});

// Привет! Меня зовут Олег, я самый главный.
```

ОБЪЕКТЫ

Создание объектов

```
const obj1 = new Object();
const obj2 = {};
const obj3 = {
  x: 5
};
```

Объект


свойство
значение
метод
const obj = {
x: 5,
greet: function() {
console.log('hello');
}
}

Вложенные объекты

```
const user = {
   name: 'Вася',
   address: {
      city: 'Москва',
      street: 'Тверская'
   }
};
```

Чтение свойств

```
const user = {
   name: 'Вася',
   address: {
      city: 'Москва',
      street: 'Тверская'
   }
};

user.name; // Вася
user.address.city; // Москва
user['name'];
user['address']['city'];
```

Доступ к несуществующему ключу

```
const user = {
   name: 'Вася',
   address: {
      city: 'Москва',
      street: 'Тверская'
   }
};

user.name; // Вася
user.address.city; // Москва
user['name'];
user['address']['city'];
user.secondName; // undefined
```

Алярма! null и undefined

```
const user = null; // или undefined

console.log(user.name);

// Uncaught TypeError: Cannot read property 'name' of null
```

Доступ к несуществующему ключу

```
null.secondName; // Error
undefined.secondName; // Error

const secondName = user && user.secondName;
// Альтернативный вариант
const secondName = (user || {}).secondName;
```

ES6: короткие свойства

```
const name = 'Вася';
const address = {
 city: 'Москва',
 street: 'Тверская'
};

const user = {name, address};
// name: 'Вася',
// address: {
//   city: 'Москва',
//   street: 'Тверская'
// }
```

http://learn.javascript.ru/es-object

ES6: деструктуризация

```
const user = {
 name: 'Вася',
 address: {
   city: 'Москва',
   street: 'Тверская'
 }
};
const {name, address: {city}} = user;

console.log(name); // Вася
console.log(city); // Москва
```

http://learn.javascript.ru/destructuring

Проверка наличия свойств в объекте

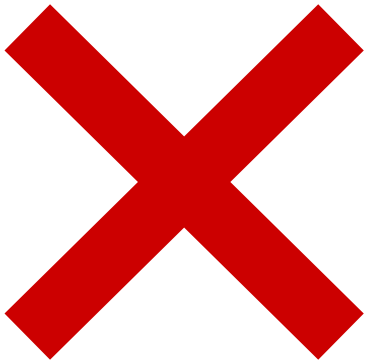
```
if ('secondName' in user) {/* … */}

if (user.hasOwnProperty('secondName')) {/* … */}
```

Запись свойств

```
const user = {
   name: 'Вася',
   address: {
      city: 'Москва',
      street: 'Тверская'
   }
};

user.name = 'Миша';
user.secondName = 'Боярский';
```

Запись свойств

```
let user = {
   name: 'Вася',
   address: {
      city: 'Москва',
      street: 'Тверская'
   }
};

user.name = 'Миша';
user = { secondName: 'Боярский' };
// создаст новый объект и перезапишет ссылку
```



Перебор свойств

```
const user = {
   name: 'Вася',
   address: {
      city: 'Москва',
      street: 'Тверская'
   }
};

for (let key in user) {
   console.log(key, user[key]);
}

// name Вася
// address Object {city: "Москва", street: "Тверская"}
```

Перебор свойств

```
const user = {
   name: 'Вася',
   address: {
      city: 'Москва',
      street: 'Тверская'
   }
};

Object.keys(user);
// ["name", "address"]
```

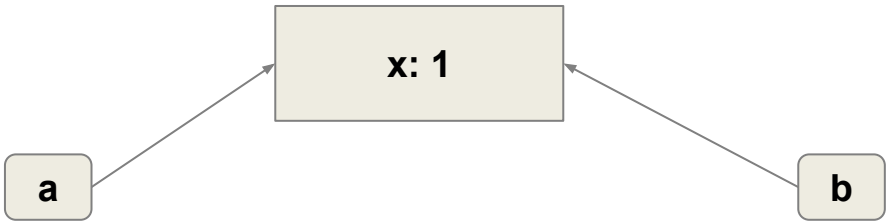
ССЫЛОЧНАЯ ПРИРОДА ОБЪЕКТОВ

Ссылки

```
const a = {x: 1};
const b = a;
b.x = 2;
a.x // ?
```

Ссылки

```
const a = {x: 1};
const b = a;
b.x = 2;
a.x // 2
```



Ссылки

```
const obj = {x: 1};

function func(o) {
  o = {x: 2};
}

func(obj);
obj.x; // ?
```

Ссылки

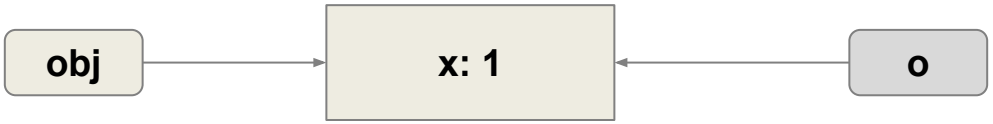
```
const obj = {x: 1};

function func(o) {
   o = {x: 2};
}

func(obj);
obj.x; // 1
```

Ссылки

```
const obj = {x: 1};

function func(o) {
  o.x = 2;
}

func(obj);
obj.x; // ?
```

Ссылки

```
const obj = {x: 1};

function func(o) {
   o.x = 2;
}

func(obj);
obj.x; // 2
```



THIS

Метод

```
const user = {
   uname: 'Вася',
   greet: function () {
       return 'Вася приветствует Вас!';
   }
};

console.log(user.greet());
```

Ссылка на объект

```
const user = {
   uname: 'Вася',
   greet: function () {
      return this.uname + ' приветствует Вас!';
   }
};

console.log(user.greet()); // Вася приветствует Вас!
```

Ссылка на объект

```
const user = {
   uname: 'Вася',
   greet: function () {
      return this.uname + ' приветствует Вас!';
   }
};

console.log(user.greet()); // Вася приветствует Вас!
user.uname = 'Петя';
console.log(user.greet()); // Петя приветствует Вас!
```

Контекст

```
const user = {
   uname: 'Вася',
   greet: function () {
      return this.uname + ' приветствует Вас!';
   }
};

const greet = user.greet;
console.log(greet()); // ?
```

Контекст

```
const user = {
   uname: 'Вася',
   greet: function () {
      return this.uname + ' приветствует Вас!';
   }
};

const greet = user.greet;
console.log(greet()); // undefined приветствует Вас!
```

Контекст

```
const user = {
   uname: 'Вася',
   greet: function () {
      return this.uname + ' приветствует Вас!';
   }
};

const greet = user.greet;
console.log(window.greet()); // undefined приветствует Вас!
// Контекст: window. window.uname === undefined
```

Call

```
const user = {
   uname: 'Вася'
};

function greet() {
   return this.uname + ' приветствует Вас!';
}

greet.call(user); // Вася приветствует Вас!
```

Call

```
const user = {
   uname: 'Вася'
};

function greet() {
   return this.uname + ' приветствует Вас!';
}

greet.call(user); // Вася приветствует Вас!
greet(); // undefined приветствует Вас!
```

Call

```
const user = {
   uname: 'Вася'
};

function greet() {
   return this.uname + ' приветствует Вас!';
}

greet.call(user); // Вася приветствует Вас!
greet(); // undefined приветствует Вас!
greet.call(window); // undefined приветствует Вас!
```

Call

```
const user = {
   uname: 'Вася',
   greet: function () {
       return this.uname + ' приветствует Вас!';
   }
};

user.greet();
user.greet.call(user);
```

Call

```
const user = {
   uname: 'Вася',
   greet: function () {
       return this.uname + ' приветствует Вас!';
   }
};

user.greet();
user.greet.call(user);
const greet = user.greet;
greet();
greet.call(window);
```

Call

```
const user = {
   uname: 'Вася',
   greet: function () {
       return this.uname + ' приветствует Вас!';
   }
};

user.greet();
user.greet.call(user);
const greet = user.greet;
greet();
greet.call(window);
user.greet.call(window);
```

Строгий режим

```
'use strict';

const user = {
   uname: 'Вася',
   greet: function () {
      return this.uname + ' приветствует Вас!';
   }
};

const greet = user.greet;
console.log(greet());
// Uncaught TypeError: Cannot read property 'uname' of undefined
```

Строгий режим

```
'use strict';

const user = {
   uname: 'Вася',
   greet: function () {
      return this.uname + ' приветствует Вас!';
   }
};

const greet = user.greet;
console.log(greet());
// Uncaught TypeError: Cannot read property 'uname' of undefined
// greet.call(undefined)
```

смена контекста

```
const user = {
   uname: 'Вася',
   greet: function () {
      return this.uname + ' приветствует Вас!';
   }
};

const greet = user.greet;

const anotherUser = {
   uname: 'Петя',
   greet: greet
};

console.log(anotherUser.greet()); // Петя приветствует Вас!
```

call / apply

```
const user = {
  /* ... */
  greetsSomebody: function(somebody, phrase) {
    return 'А тут ' + this.uname + ' и говорит: ' + somebody + ', ' + phrase + '!';
  }
};

const greetsSomebody = user.greetsSomebody;

console.log(greetsSomebody.call(user, 'Петя', 'здравствуй'));
console.log(greetsSomebody.apply(user, ['Петя', 'здравствуй']));
// А тут Вася и говорит: Петя, здравствуй!
```

https://learn.javascript.ru/call-apply

Применение: call

В JS есть массивы, а есть массивоподобные объекты — похожи по структуре на массив, но не имеют их методов.

```
function hey() {
  // for (let i = 0; i < arguments.length; i++) {
  //     console.log('Эй, ' + arguments[i]+ '!');
  // }
  const all = [].join.call(arguments, ' и '); // arguments.join – ошибка
  console.log('Эй, ' + all + '!');
}

hey('Вася', 'Маша', 'Петя'); // Эй, Вася и Маша и Петя!
```

Применение: apply

```
console.log(Math.max(2, 3, 9)); // 9

const randomArray = getRandomArray();

function getRandomArray() {
  const res = [];
  for(let i = Math.random() * 10 + 3; i > 0; i--) {
    res.push(Math.floor(Math.random() * 100));
  }
  return res;
}

console.log(randomArray); // [35, 17, 75, 78, 4, 79, 37]
console.log(Math.max.apply(null, randomArray)); // 79
```

bind

```
const user = {
   /* ... */
   greet: function () {
      return this.uname + ' приветствует Вас!';
   }
};

var greet = user.greet.bind(user);

console.log(anotherUser.greet()); // Вася приветствует Вас!
console.log(greet()); // Вася приветствует Вас!
```

Потеря контекста

```
const user = {
 uname: 'Вася',
 greetAsync: function () {
   setTimeout(function(){
     console.log(this.uname + ' приветствует Вас!');
   }, 1000);
 }
};

user.greetAsync();
// undefined приветствует Вас!
```

Потеря контекста

```
const user = {
 uname: 'Вася',
 greetAsync: function () {
   setTimeout(function(){
     console.log(this.uname + ' приветствует Вас!');
   }, 1000);
 }
};

user.greetAsync();
// undefined приветствует Вас!
```

1. Исполнить **fn** через *1000мс.*

Потеря контекста

```
const user = {
 uname: 'Вася',
 greetAsync: function () {
   setTimeout(function(){
     console.log(this.uname + ' приветствует Вас!');
   }, 1000);
 }
};

user.greetAsync();
// undefined приветствует Вас!
```

1. Исполнить **fn** через *1000мс.*
…
прошла 1000мс
…
*выполняем **fn***

Потеря контекста

```
(function()
 console.log(this.uname + ' приветствует Вас!');
})()
```

1. Исполнить **fn** через *1000мс.*
…
прошла 1000мс
…
*выполняем **fn***

Потеря контекста

```
const user = {
 uname: 'Вася',
 greetAsyncFixed: function () {
   setTimeout(() => {
     console.log(this.uname + ' приветствует Вас!');
   }, 1000);
 }
};

user.greetAsyncFixed();
// Вася приветствует Вас!
```

МАССИВЫ

Массив

```
const empty = [];
const arr = ['Вася', 'Петя', 'Миша'];

arr[0]; // Вася
arr[2]; // Миша
arr[8237]; // undefined
```

http://learn.javascript.ru/array

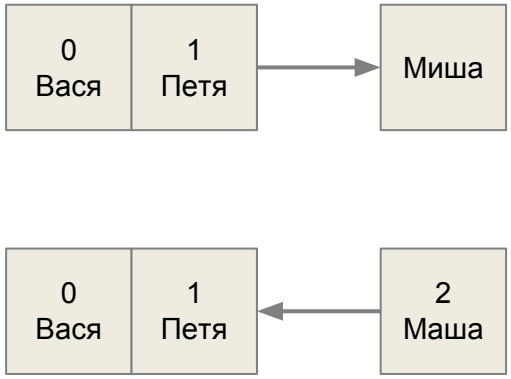
Деструктуризация массива

```
const arr = ['Вася', 'Петя', 'Миша'];

const [first, ...rest] = arr;
console.log(first); // Вася
console.log(rest); // ["Петя", "Миша"]

const [, second] = arr;
console.log(second); // Петя
```

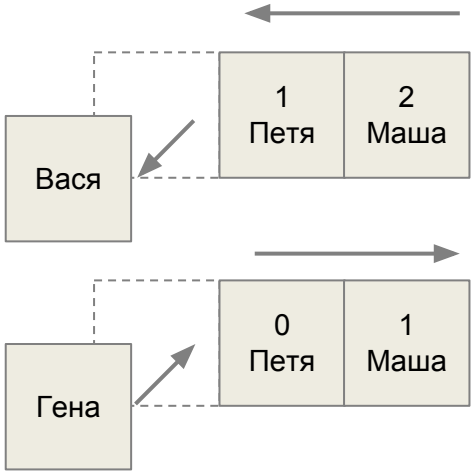
http://learn.javascript.ru/destructuring

push / pop; shift / unshift

```
const arr = ['Вася', 'Петя', 'Миша'];

arr.pop(); // Миша
arr; // ["Вася", "Петя"]

arr.push('Маша');
arr; // ["Вася", "Петя", "Маша"]
```




```
const arr = ['Вася', 'Петя', 'Маша'];

arr.shift(); // Вася
arr.unshift('Геннадий'); // 3
arr; // ["Геннадий", "Петя", "Маша"]
```



Длина

```
var arr = ['Вася', 'Петя', 'Миша'];

arr.length; // 3
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array
https://learn.javascript.ru/array#особенности-работы-length

Методы работы с массивами

http://learn.javascript.ru/array-methods

– concat
– slice
– splice
– reverse
– indexOf

Перебор

```
const arr = ['Вася', 'Петя', 'Миша'];

for (let i = 0; i < arr.length; i++) {
   console.log(arr[i]);
}

for (let i of arr) {
 console.log(i);
}
// Вася
// Петя
// Миша
```

ES6

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/for...of

итераторы http://learn.javascript.ru/iterator

forEach

```
const arr = ['Геннадий', 'Петя', 'Миша'];

arr.forEach(function(item, i, arr) {
  console.log(item);
});
// Геннадий
// Петя
// Маша
```

http://learn.javascript.ru/array-iteration

filter

```
const arr = ['Геннадий', 'Петя', 'Маша'];

const filteredArr = arr.filter(item => item.length === 4);

console.log(filteredArr); // ["Петя", "Маша"]
```

map

```
const arr = ['Геннадий', 'Петя', 'Миша'];

const lengths = arr.map(item => item.length);

console.log(lengths); // [8, 4, 4]
```

every / some

```
const arr = ['Геннадий', 'Петя', 'Миша'];

arr.every(item => item.length === 4); // false

arr.some(item => item.length === 4); // true
```

reduce

```
const lengths = [8, 4, 4];

lengths.reduce((total, current) => total + current); // 16
```

Chaining / цепочки

```
getMinMax = (str) => {
 return str
   .split(' ')
   .map(word => parseFloat(word))
   .filter(word => !isNaN(word))
   .reduce(({min, max}, current) => ({
       min: Math.min(min, current),
       max: Math.max(max, current)
   }), {min: Infinity, max: -Infinity});
};

console.log(getMinMax('1 и 6.45, -2, но 8, а затем 15, то есть 2.7 и -1028'));
```

Вопросы?