

JavaScript и D3

Структуры данных JS

Объекты

```
01: var human = {  
02:   firstName: 'John',  
03:   secondName: 'Doe',  
04:   age: 25,  
05:   name: function () {  
06:     return this.firstName + ' ' + this.secondName;  
07:   },  
08:   greet: function () {  
09:     return 'Hello, I am ' + this.name();  
10:   }  
11: }
```

Объекты

```
01: var human = {  
02:   firstName: 'John',  
03:   secondName: 'Doe',  
04:   age: 25,  
05:   name: function () {  
06:     return this.firstName + ' ' + this.secondName;  
07:   },  
08:   greet: function () {  
09:     return 'Hello, I am ' + this.name();  
10:   }  
11: }
```



Ключ



Значение

Объекты

```
01: var object = {  
02:     4eburek-2016: null, нельзя  
03:  
04:     '4eburek-2016': null, можно  
05: }
```

Объекты

```
01: var human = {  
02:   firstName: 'John',  
03:   secondName: 'Doe',  
04:   age: 25,  
05:   name: function () {  
06:     return this.firstName + ' ' + this.secondName;  
07:   },  
08:   greet: function () {  
09:     return 'Hello, I am ' + this.name();  
10:   },  
11: }
```

Свойство

Метод

Объекты

```
01: var human = { ... }
02:
03: human.firstName // -> John Doe
04:
05: human['firstName']; // -> John Doe
06:
07: human.name; // -> function () { ... }
08:
09: human.name(); // -> 'John Doe'
10:
11: human.firstName = 'Jake';
12:
13: human.name(); // -> 'Jake Doe'
```

Объекты: контекст

```
01: var vector = {  
02:   x: 3,  
03:   y: 4,  
04:   r: function () {  
05:     return Math.sqrt(this.x * this.x + this.y * this.y);  
06:   }  
07: }  
08:  
09: vector.r(); // -> 5
```


Объекты: контекст

```
01: var vector = {  
02:   x: 3,  
03:   y: 4,  
04:   r: function () {  
05:     return Math.sqrt(this.x * this.x + this.y * this.y);  
06:   }  
07: }  
08:  
09: vector.r(); // -> 5
```

The diagram illustrates the execution of the `vector.r()` call. Red arrows show the resolution of `this.x` and `this.y` within the function `r` to the `x` and `y` properties of the `vector` object. The expressions `this.x * this.x` and `this.y * this.y` are circled in red, indicating they are the terms being summed under the square root.

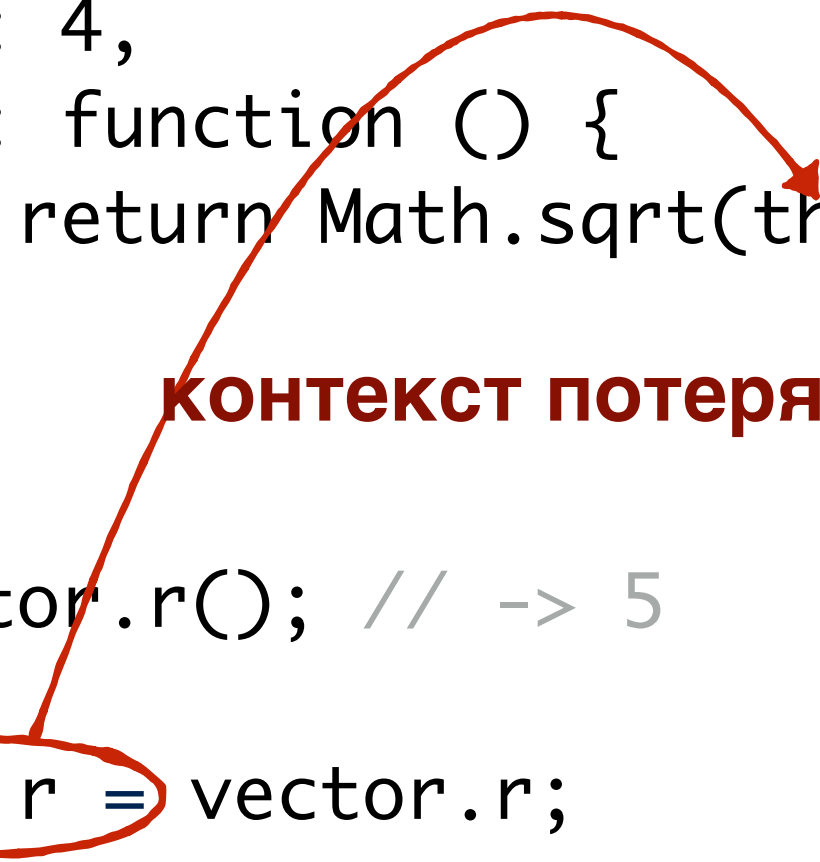
Объекты: контекст

```
01: var vector = {  
02:   x: 3,  
03:   y: 4,  
04:   r: function () {  
05:     return Math.sqrt(this.x * this.x + this.y * this.y);  
06:   }  
07: }  
08:  
09: vector.r(); // -> 5  
10:  
11: var r = vector.r;  
12:  
13: r(); // -> NaN
```

Объекты: контекст

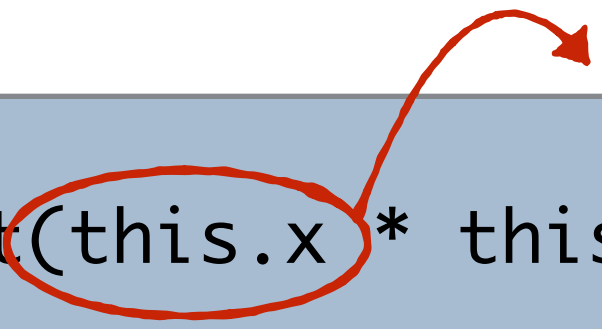
```
01: var vector = {  
02:   x: 3,  
03:   y: 4,  
04:   r: function () {  
05:     return Math.sqrt(this.x * this.x + this.y * this.y);  
06:   }  
07: }  
08:  
09: vector.r(); // -> 5  
10:  
11: var r = vector.r;  
12:  
13: r(); // -> NaN
```

контекст потерялся



Объекты: контекст

```
01: var vector = {  
02:   x: 3,  
03:   y: 4,  
04:   r: function () {  
05:     return Math.sqrt(this.x * this.x + this.y * this.y);  
06:   }  
07: }  
08:  
09: vector.r(); // -> 5  
10:  
11: var r = vector.r;  
12:  
13: r(); // -> NaN
```

A red arrow originates from the circled `this.x` in line 05 and points to the text "НОВЫЙ КОНТЕКСТ" (New Context).

НОВЫЙ КОНТЕКСТ

Объекты: контекст

```
01: var vector = {  
02:   x: 3,  
03:   y: 4,  
04:   r: function () {  
05:     return Math.sqrt(this.x * this.x + this.y * this.y);  
06:   }  
07: }  
08:  
09: vector.r(); // -> 5  
10:  
11: var r = vector.r;  
12:  
13: r(); // -> NaN
```

Глобальный контекст

Вызов в глобальном контексте

Объекты: контекст (решение)

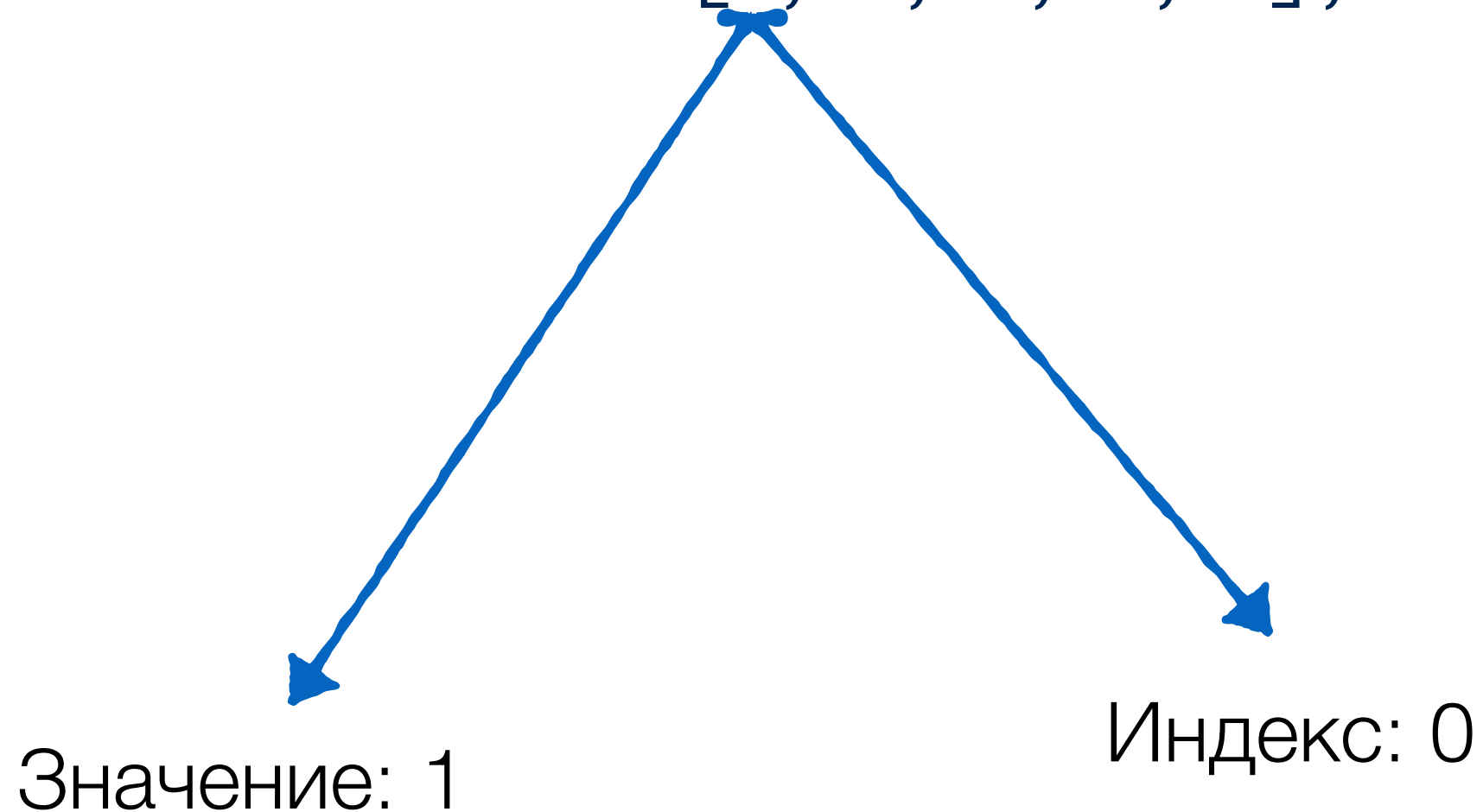
```
01: var vector = {  
02:   x: 3,  
03:   y: 4,  
04:   r: function () {  
05:     return Math.sqrt(this.x * this.x + this.y * this.y);  
06:   }  
07: }  
08:  
09: vector.r(); // -> 5  
10:  
11: var r = vector.rbind(vector); СВЯЗЫВАНИЕ КОНТЕКСТА  
12:  
13: r(); // -> NaN
```

Массивы

```
01: var numbers = [1, 2, 3, 4, 5];
```

Массивы

```
01: var numbers = [1, 2, 3, 4, 5];
```



Массивы

```
01: var numbers = [1, 2, 3, 4, 5];  
02:  
03: numbers[0]; // -> 1  
04:  
05: numbers.length; // -> 5  
06:  
07: numbers[4] = 'Hello';  
08:  
09: numbers; // -> [1, 2, 3, 4, 'Hello'];
```

Массивы

вообще-то это объекты

```
01: var numbers = [1, 2, 3, 4, 5];
```

```
02:
```

```
03: var numbers = {
```

```
04:     0: 1,
```

```
05:     1: 2,
```

```
06:     2: 3,
```

```
07:     3: 4,
```

```
08:     4: 5,
```

```
09:     length: 5
```

```
10: }
```

Массивы

вообще-то это объекты

```
01: var numbers = [1, 2, 3, 4, 5];
```

```
02:
```

```
03: var numbers = {
```

```
04:     0: 1,
```

```
05:     1: 2,
```

```
06:     2: 3,
```

```
07:     3: 4,
```

```
08:     4: 5,
```

```
09:     length: 5
```

```
10: }
```

- порядок (все ключи - числа)
- длина (есть ключ length)

Работа со структурами данных

- перебор
- фильтрация
- трансформация
- сортировка
- сворачивание

for ... in

```
01: var object = { ... };
```

```
02:
```

```
03: var copy = {};
```

```
03:
```

```
03: for (var property in object) {
```

```
04:     copy[property] = object[property];
```

```
05: }
```

for ... in

```
01: var object = { ... };
02:
03: var copy = {};
04:
05: for (var property in object) {
06:     if (object.hasOwnProperty(property)){
07:         copy[property] = object[property];
08:     }
09: }
```

forEach, map, filter

```
01: var numbers = [1, 2, 3, 4, 5];
02:
03: numbers.forEach(function (number) {
04:     console.log(number);
05: });
06:
07: numbers.map(function (number) {
08:     return number * 2;
09: });
10:
11: numbers.filter(function (number) {
12:     return number > 2;
13: });
```

reduce

```
01: var numbers = [1, 2, 3, 4, 5];
02:
03: var sum = numbers.reduce(function (sum, number) {
04:     return sum + number;
05: });
06:
07: var max = numbers.reduce(function (max, number) {
08:     return number > max ? number : max;
09: });
```


reduce =

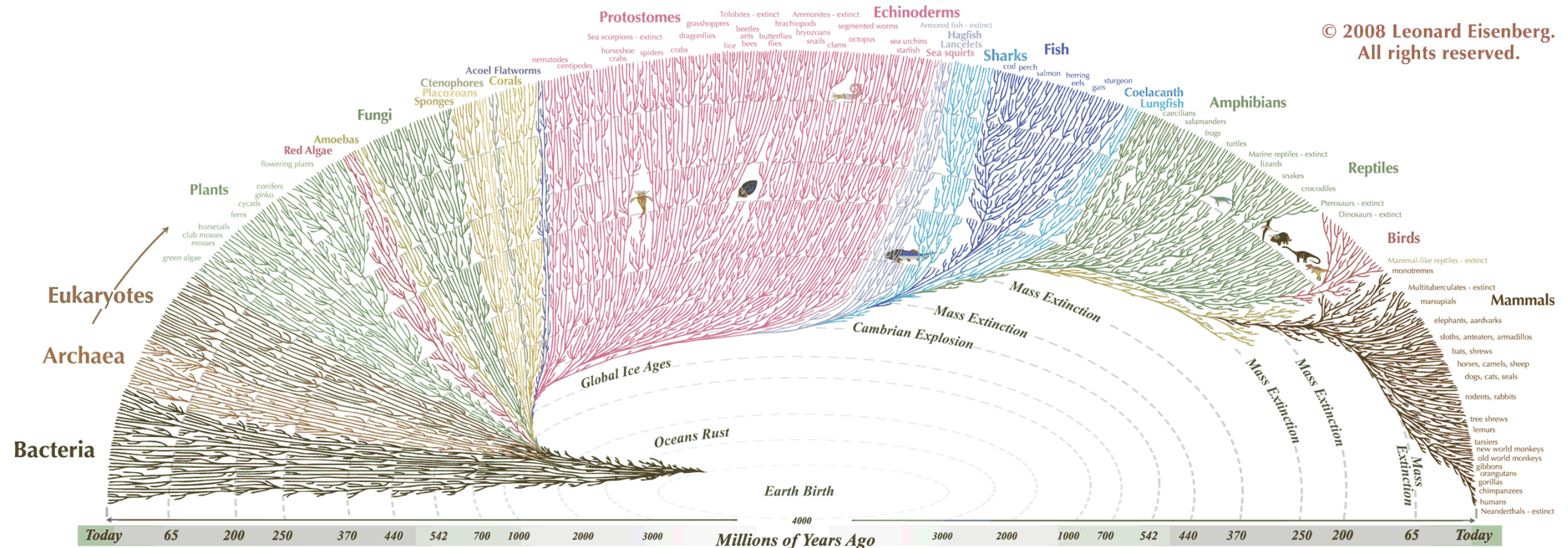


ООП

ну, или типа того



© 2008 Leonard Eisenberg.
All rights reserved.



All the major and many of the minor living branches of life are shown on this diagram, but only a few of those that have gone extinct are shown. Example: Dinosaurs - extinct



© 2008 Leonard Eisenberg. All rights reserved.
evogeneao.com

Абстрактный класс чего-то

abstract

Класс всех животных

animal

Класс млекопитающих

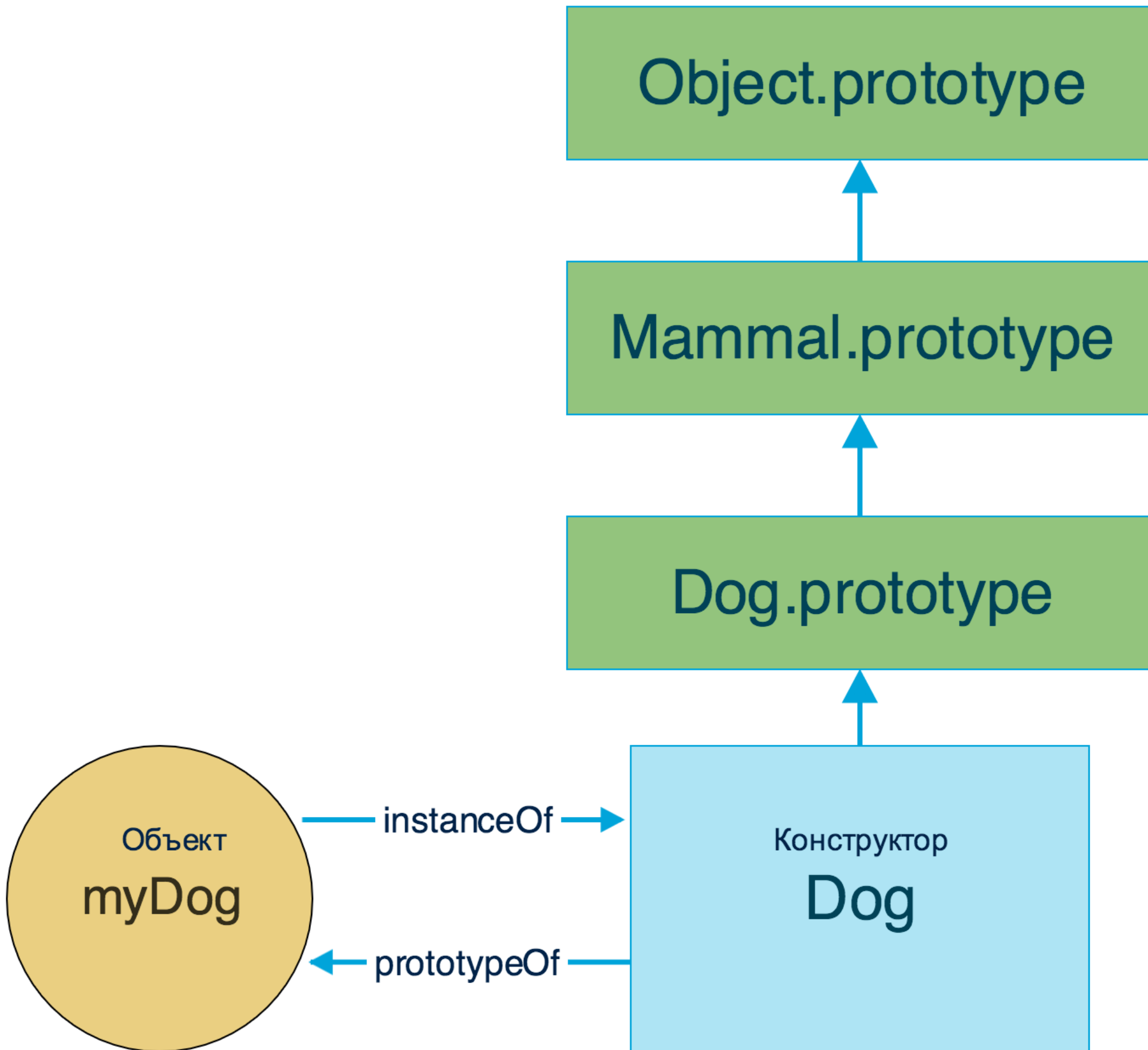
mammal

Класс коров

cow

Классы - это функции

```
01: function Mammal(name, legs){}
02:   this.type = 'mammal';
03:   this.name = name;
04:   this.legs = legs;
05: }
06:
07: Mammal.prototype.hello = function () {
08:   console.log('I am ' + this.type + ' ' +
09:     this.name);
10: }
11:
12: var pet = new Mammal('Bobik', 4);
13:
14: pet.hello();
```



DOM API

Все начинается с документа

document

```
document.getElementById('myId')
```

```
var selector = 'body ul a.active';  
document.querySelector(selector);
```

```
var selector = 'body ul a.active';
```

```
document.querySelector(selector);
```

ссылка на элемент в документе

jQuery

```
var $navLinks = $('body ul a.active');
```

jQuery

```
var $navLinks = $('body ul a.active');
```

Без jQuery

```
01: var items = document.querySelectorAll('ul li');
```

С jQuery

```
01: var $items = $('ul li');
```

Без jQuery


```
01: var items = document.querySelectorAll('ul li');
02:
03: var names = [].map.call(items, function(item){
04:     return item.innerHTML;
05: });
```

C jQuery

```
01: var $items = $('ul li');
02:
03: var names = $items.forEach(function(item){
04:     return item.html();
05: });
```


Без jQuery

```
01: var items = document.querySelectorAll('ul li');
02:
03: var names = [].map.call(items, function(item){
04:     return item.innerHTML;
05: });
```


 **боль и унижение**

С jQuery

```
01: var $items = $('ul li');
02:
03: var names = $items.forEach(function(item){
04:     return item.html();
05: });
```


Без jQuery

```
01: var items = document.querySelectorAll('ul li');
02:
03: var names = [].map.call(items, function(item){
04:     return item.innerHTML;
05: });
```

 **боль и унижение**

С jQuery

```
01: var $items = $('ul li');
02:
03: var names = $items.forEach(function(item){
04:     return item.html();
05: });
```

процветание и гармония 

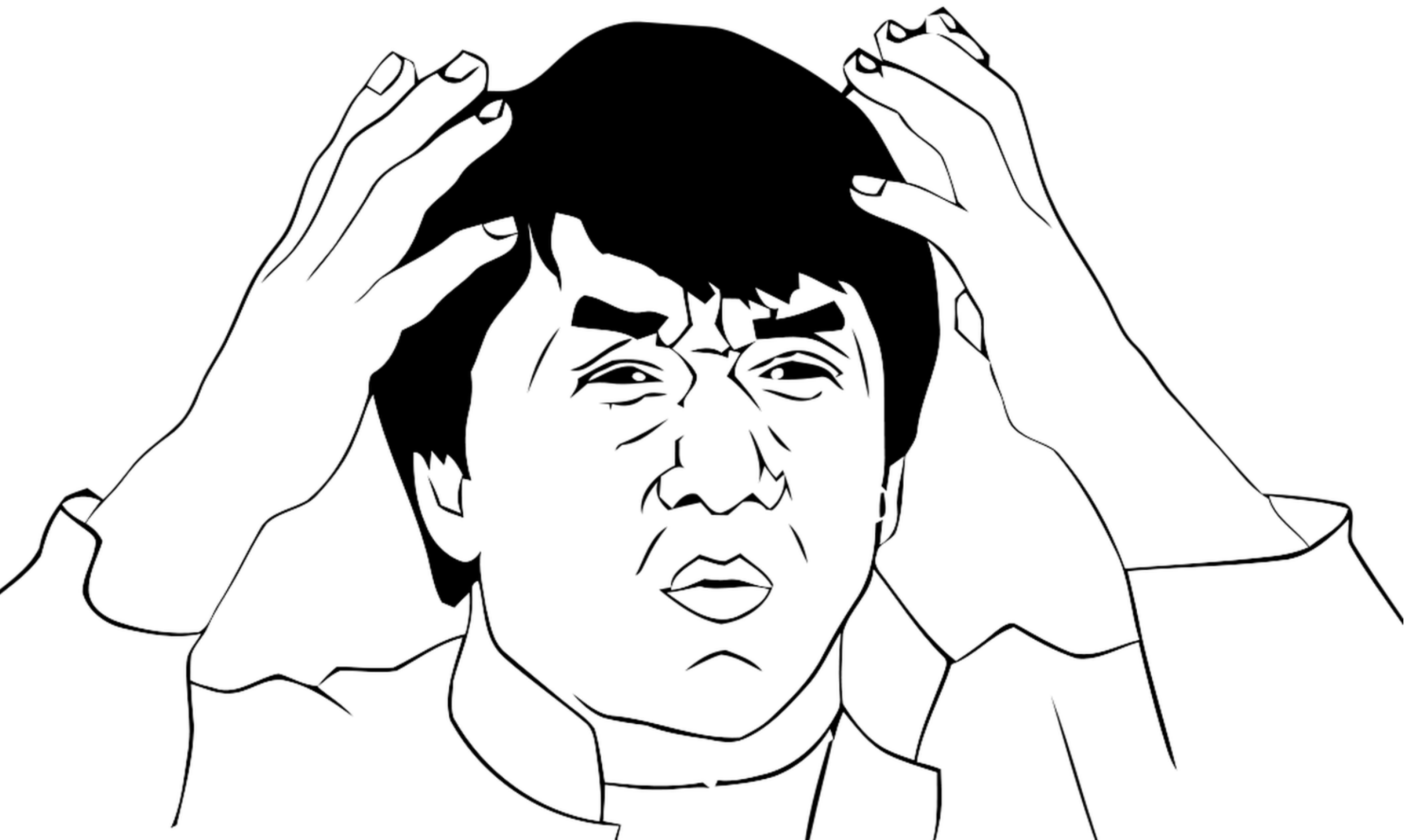


Асинхронность

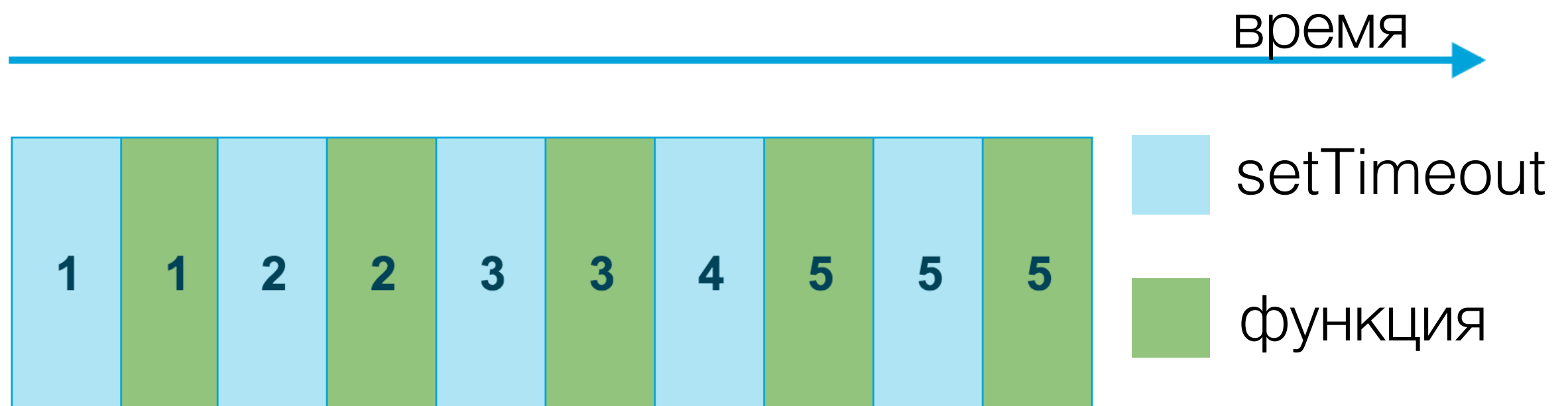
```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(i);  
    }, 0);  
}
```

```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(i);  
    }, 0);  
}
```

```
> for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(i);  
    }, 0)  
    }  
< 35  
5 5  
>
```



```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(i);  
    }, 0);  
}
```

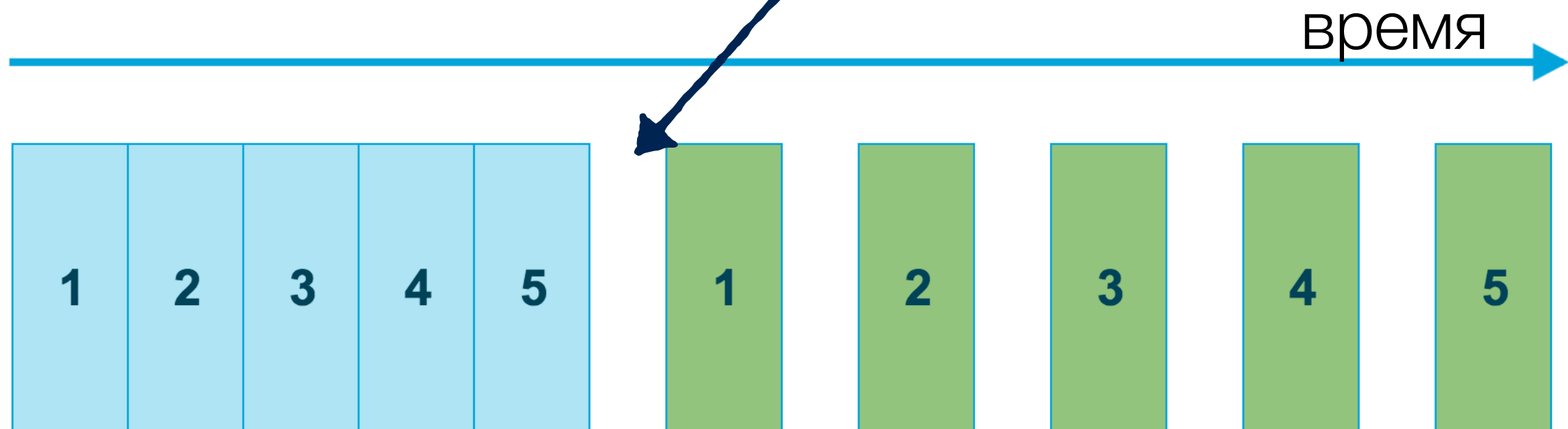


```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(i);  
    }, 0);  
}
```




```
for(var i = 0; i < 5; i++){  
  setTimeout(function(){  
    console.log(i);  
  }, 0);  
}
```

такт event-loop'a



```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(i);  
    }, 0);  
}
```

Блокирующее выполнение

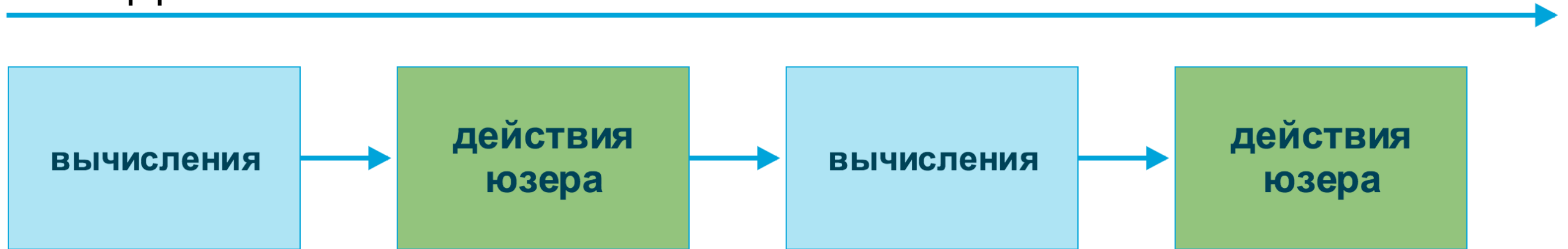
```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(i);  
    }, 0);  
}
```

Асинхронное выполнение

Асинхронность - это боль

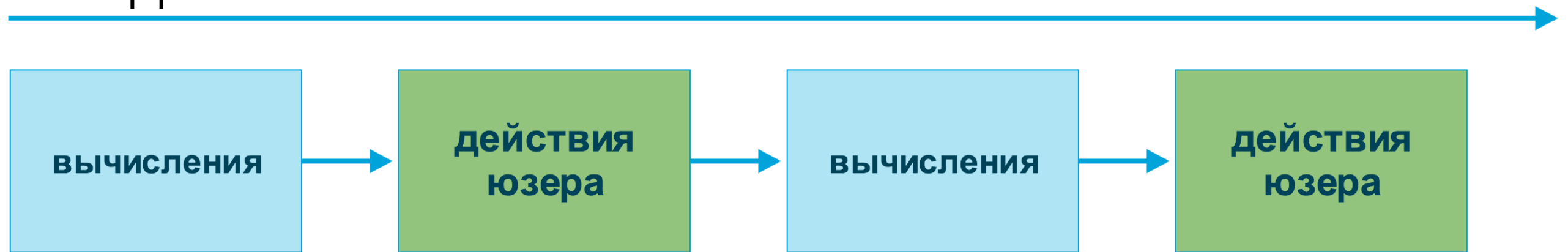
Асинхронность - это боль

Ожидание

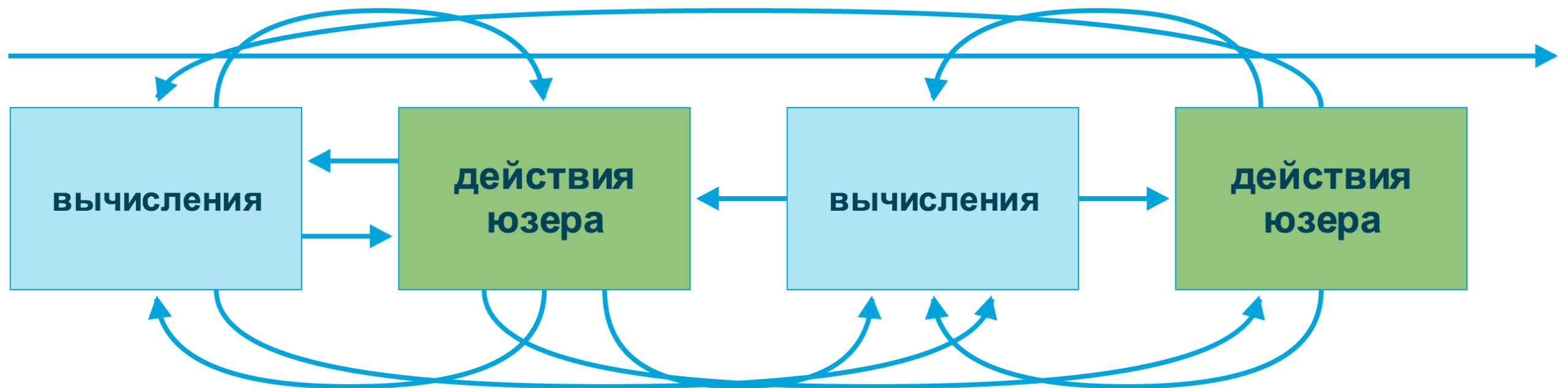


Асинхронность - это боль

Ожидание



Реальность



```
▶ for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(j);  
    }, 0);  
}
```

```
for(var i = 0; i < 5; i++){  
    ▶    setTimeout(function(){  
        console.log(j);  
    }, 0);  
}
```



```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(j);
```



```
    }, 0);
```

```
}
```

```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(j);  
    }, 0);  
}
```

```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(j);  
    }, 0);  
}
```

Решение: замыкания

```
for(var i = 0; i < 5; i++){  
    setTimeout(function(j){  
        return function(){  
            return console.log(j);  
        };  
    })(i), 0);  
}
```

Решение: связывание контекста

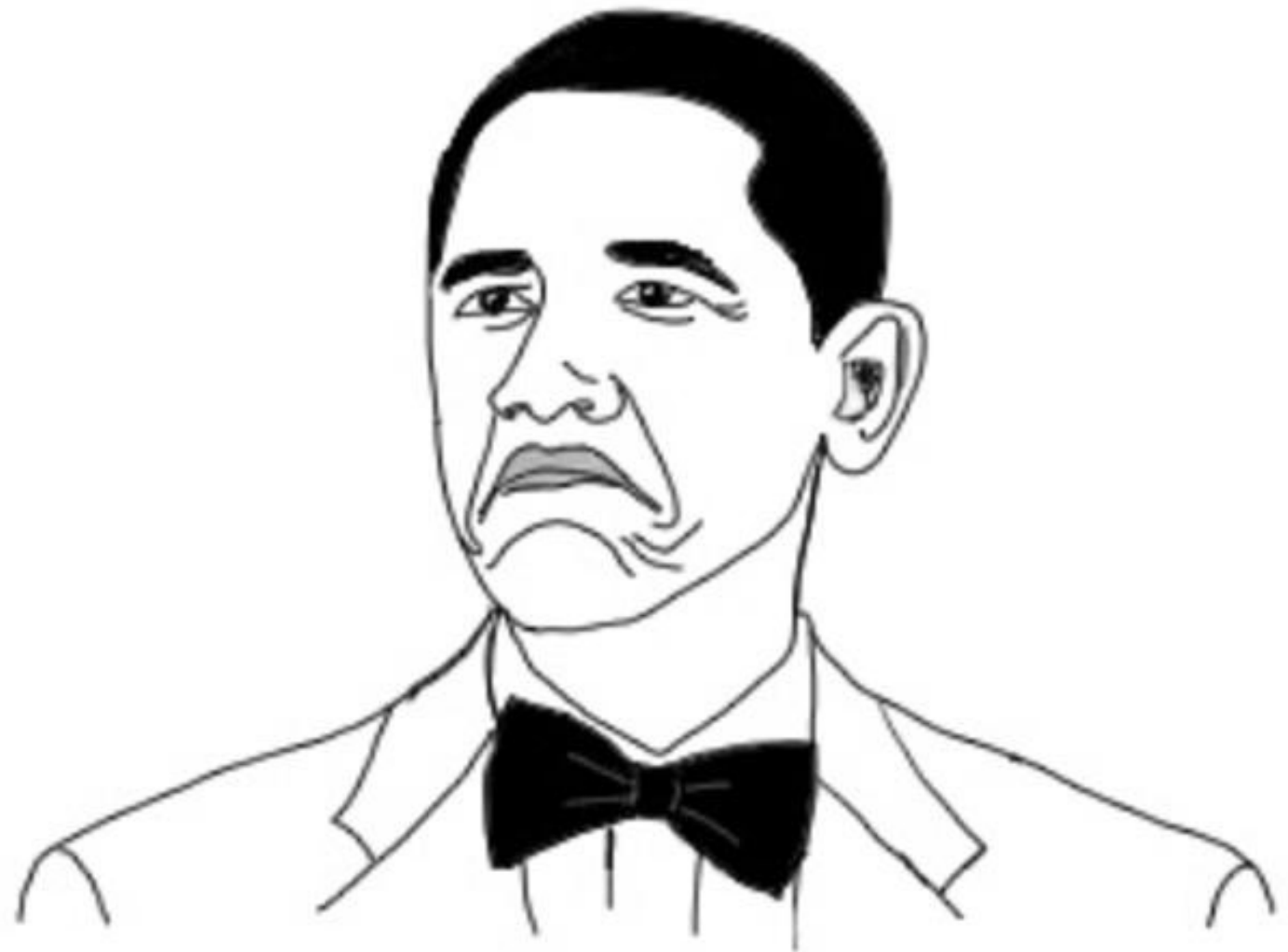
```
for(var i = 0; i < 5; i++){  
    setTimeout(function(){  
        console.log(this);  
    }.bind(i), 0);  
}
```

События

- Происходят во времени
- На них можно подписаться
- От них можно отписаться

События

- Происходят во времени
- На них можно подписаться
- От них можно отписаться



NOT BAD

События


```
01: var input = document.getElementById('text');  
02:  
03: input.addEventListener('keyup', function (event) {  
04:     console.log(event);  
05: });
```

Источник события



События

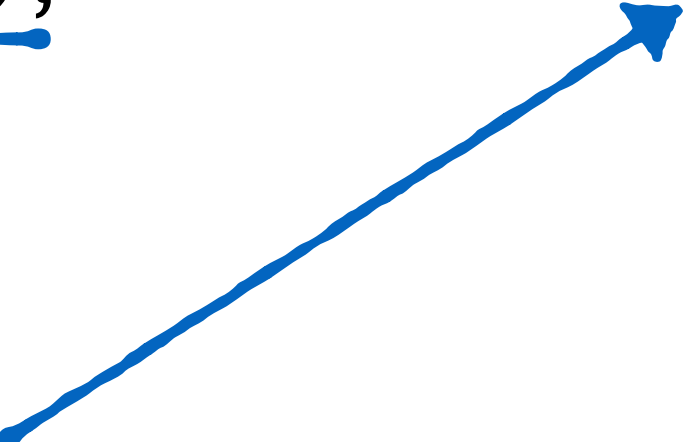
```
01: var input = document.getElementById('text');  
02:  
03: input.addEventListener('keyup', function (event) {  
04:     console.log(event);  
05: });
```



Название события

События

```
01: var input = document.getElementById('text');  
02:  
03: input.addEventListener('keyup', function (event) {  
04:   console.log(event);  
05: });
```



Обработчик события

События

- события ввода (мышь, клавиатура, тачскрин, микрофон)
- события окна (изменение размера, хэша)
- события браузера (загрузка, отрисовка)
- события сети (завершение запроса, веб-сокеты)
- пользовательские события

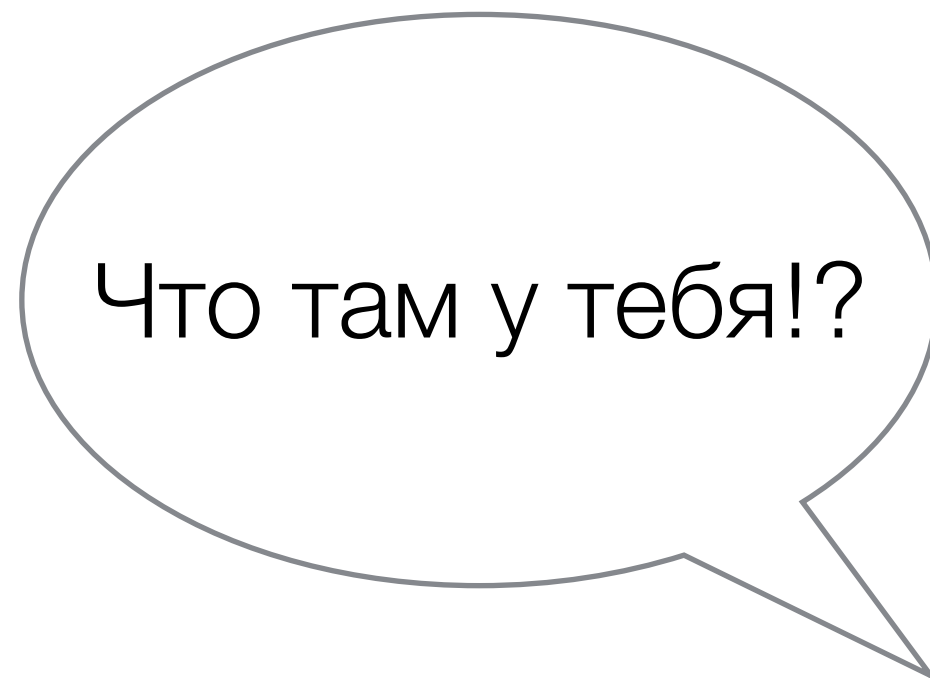
Pub/Sub

Publisher Subscriber



Pub/Sub

Publisher Subscriber



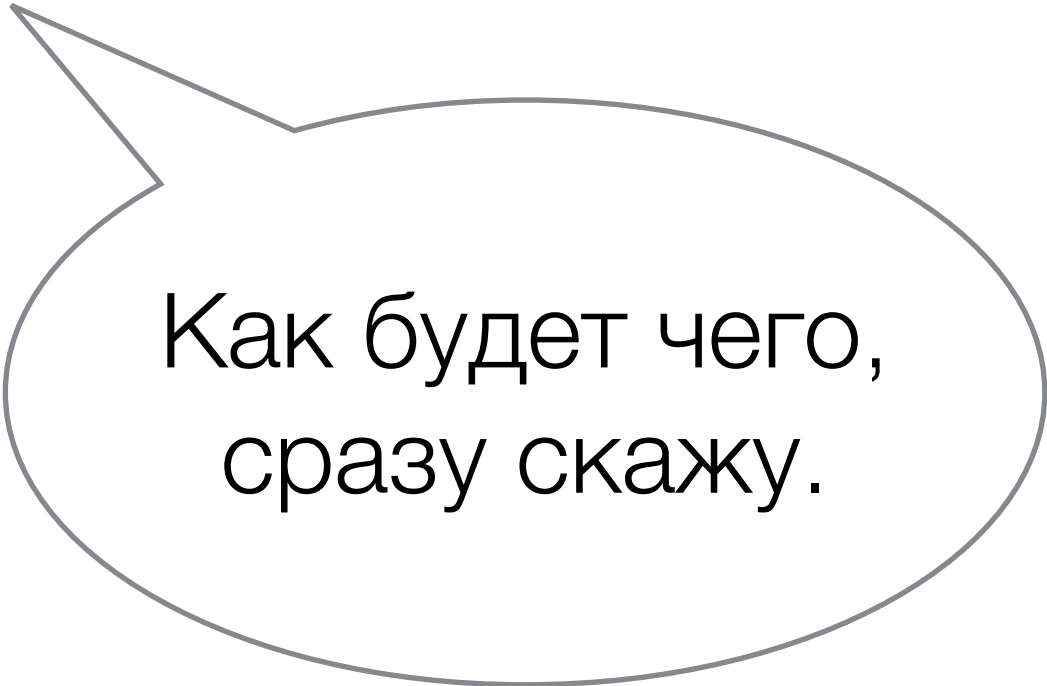
Pub/Sub

Publisher Subscriber



Pub/Sub

Publisher Subscriber



Как будет чего,
сразу скажу.

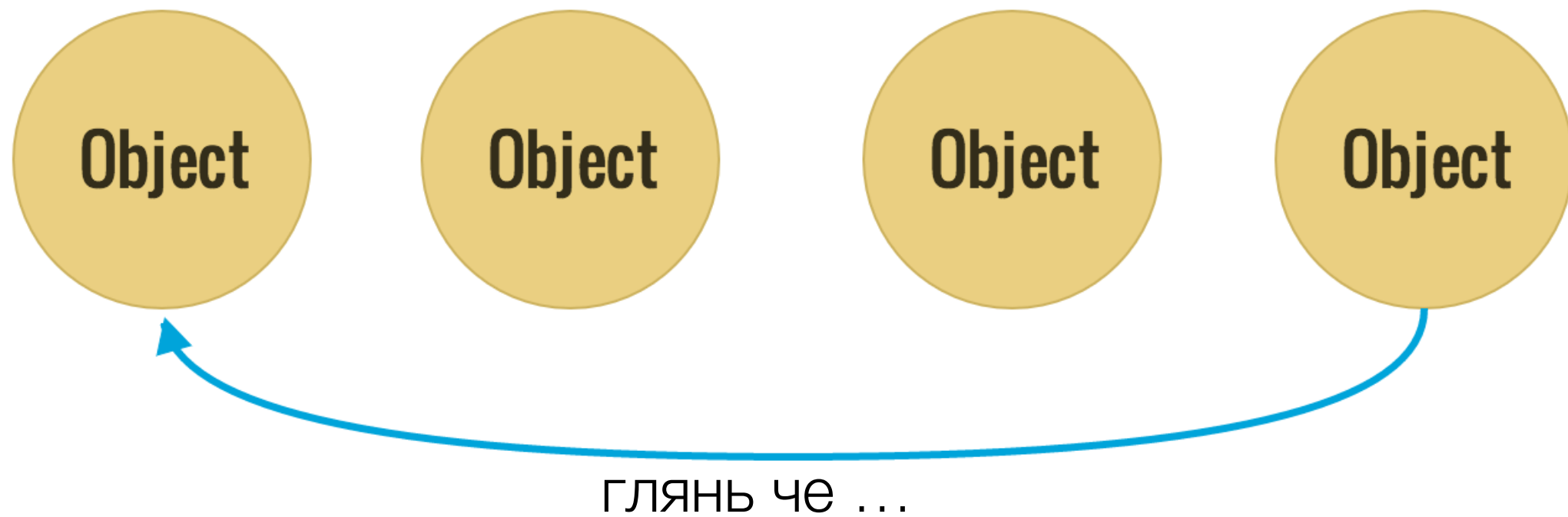
Pub/Sub

Publisher Subscriber



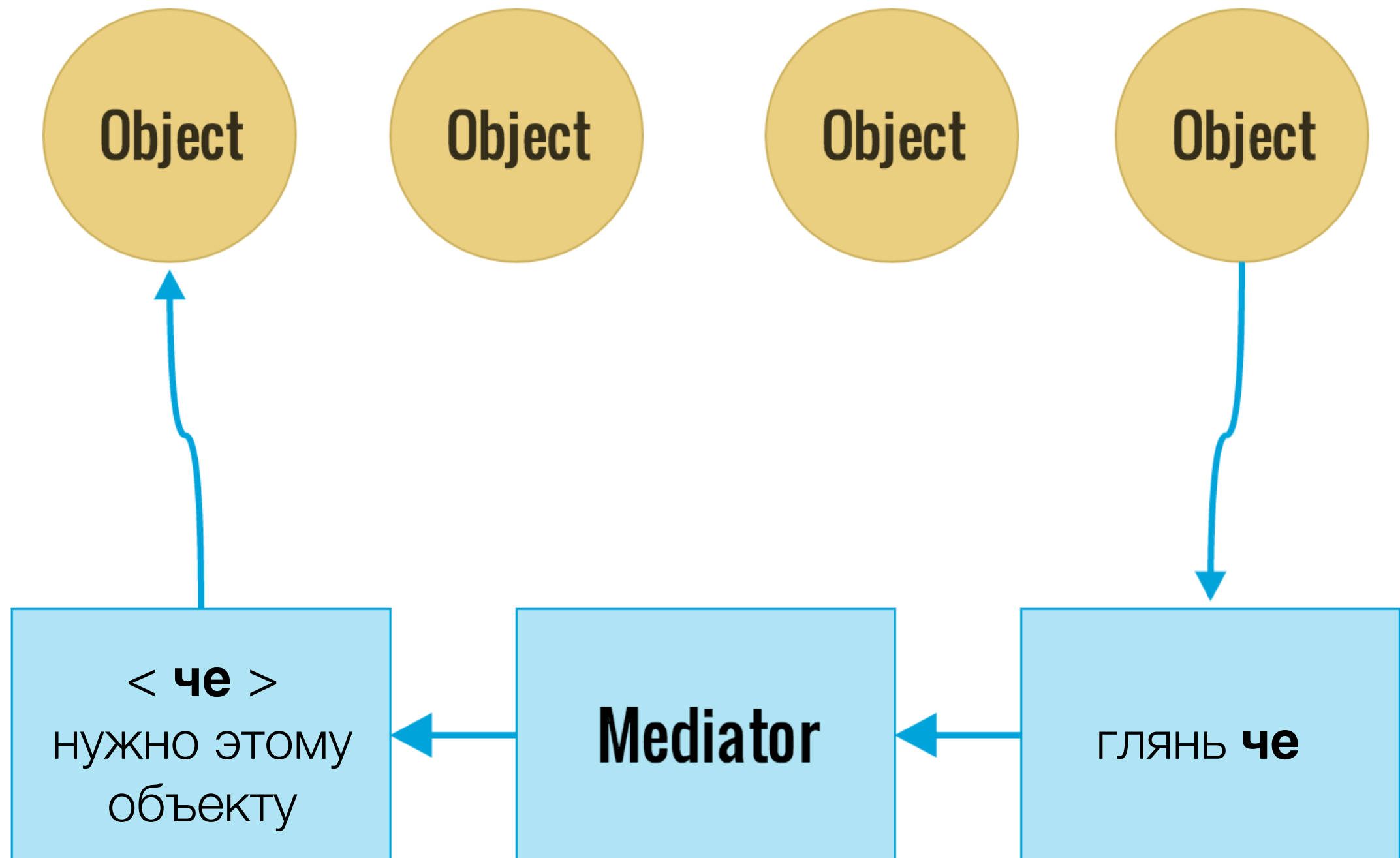
Pub/Sub

Publisher Subscriber



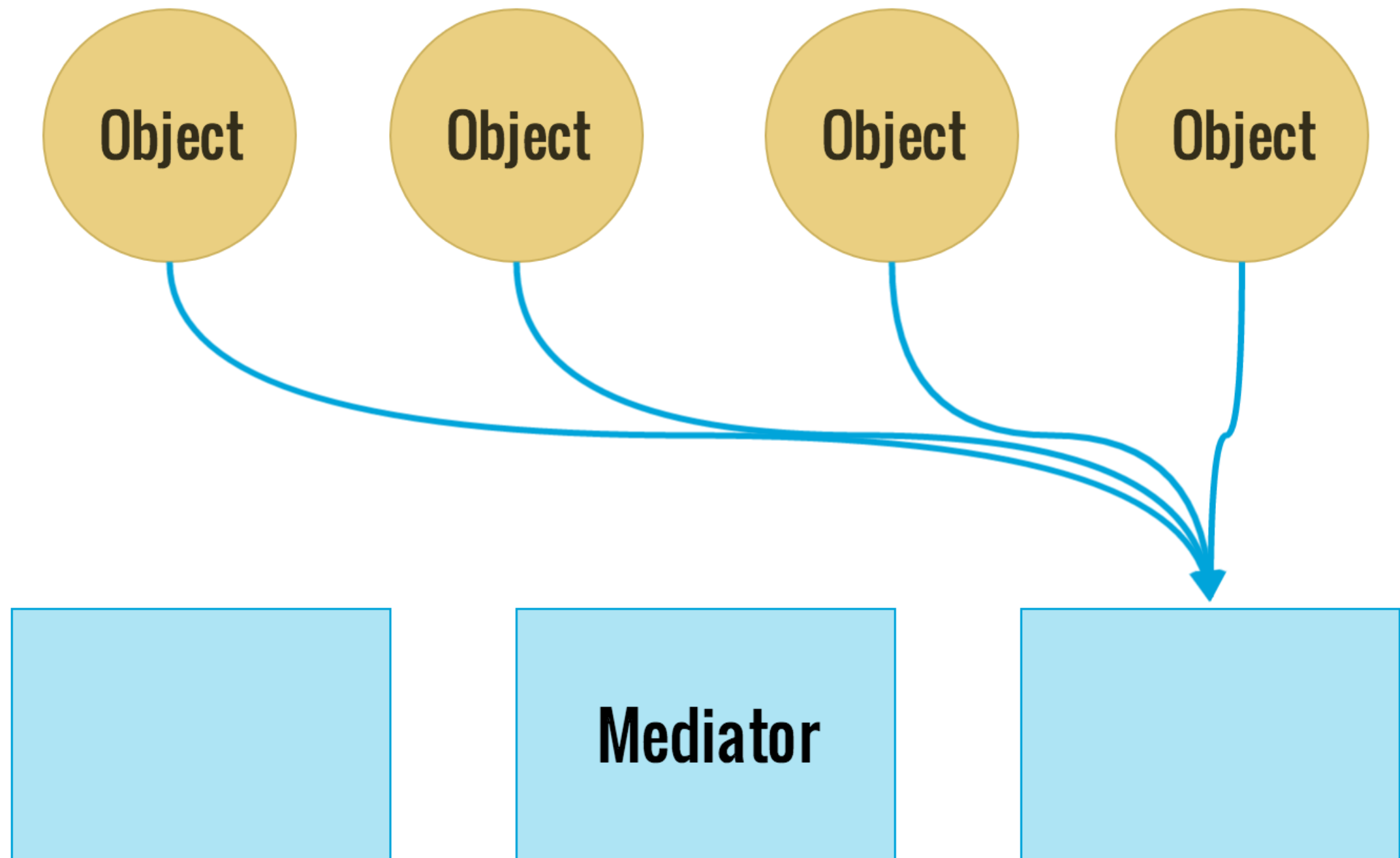
Pub/Sub

Publisher Subscriber



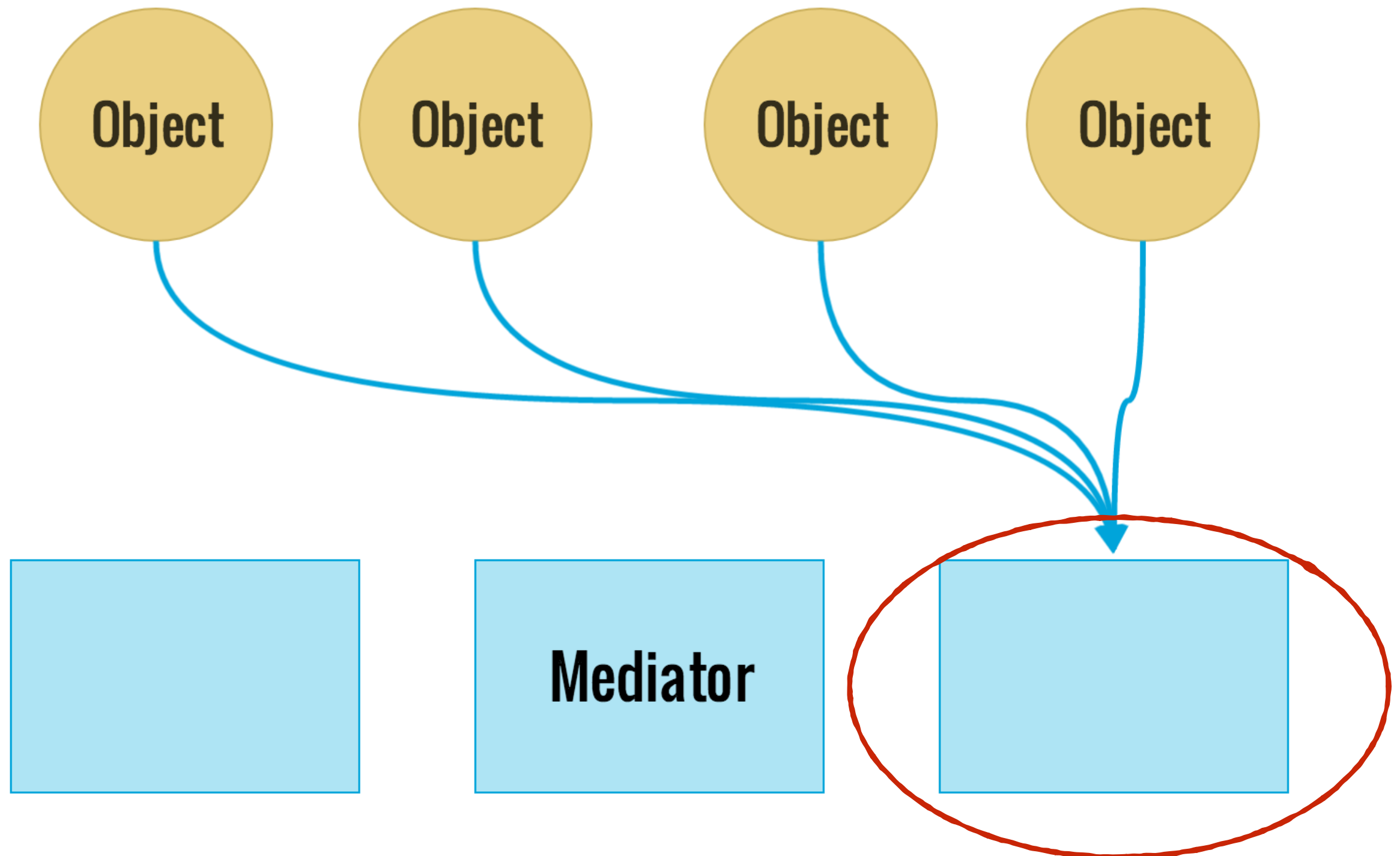
Pub/Sub

Publisher Subscriber



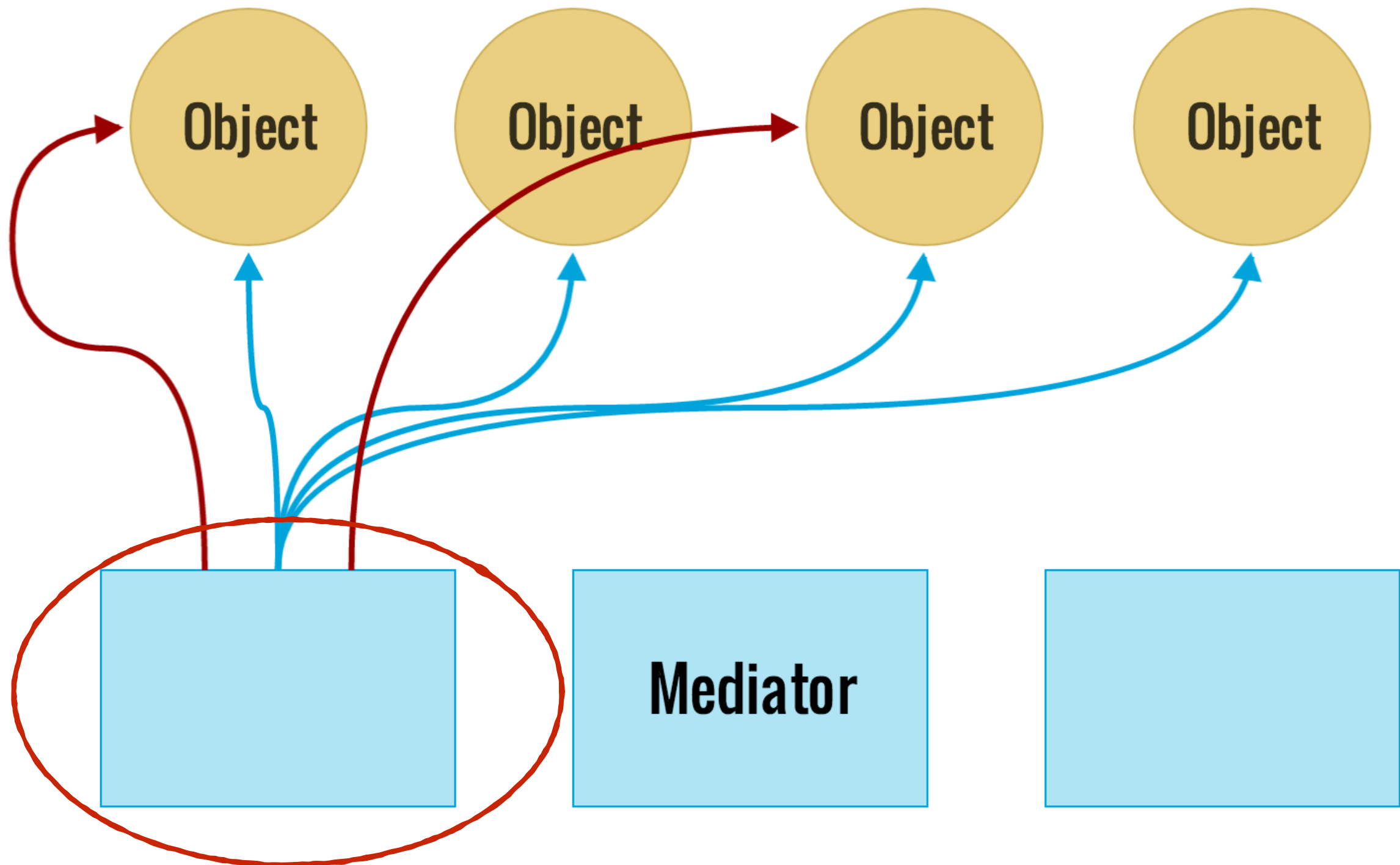
Pub/Sub

Publisher Subscriber



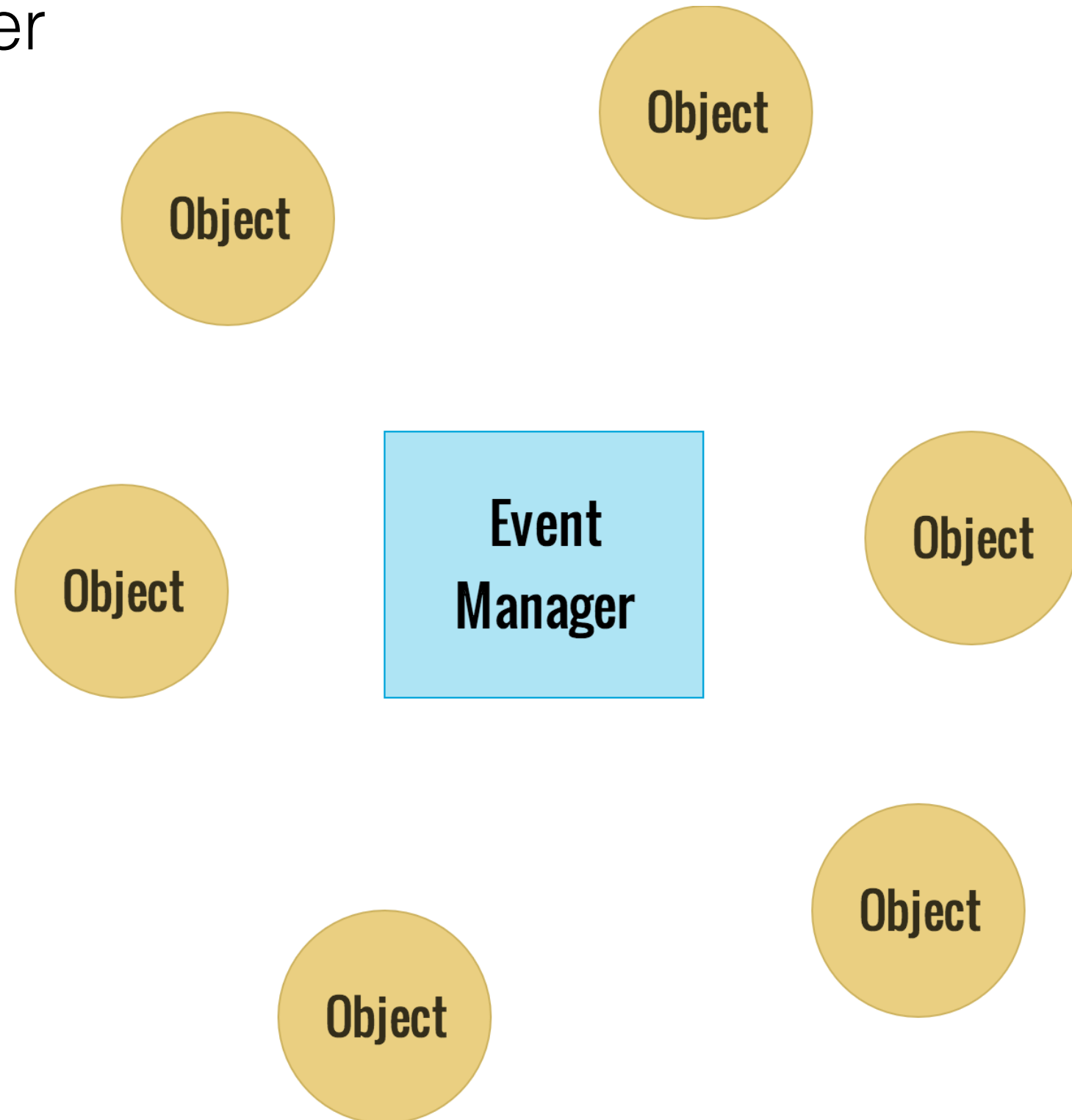
Pub/Sub

Publisher Subscriber



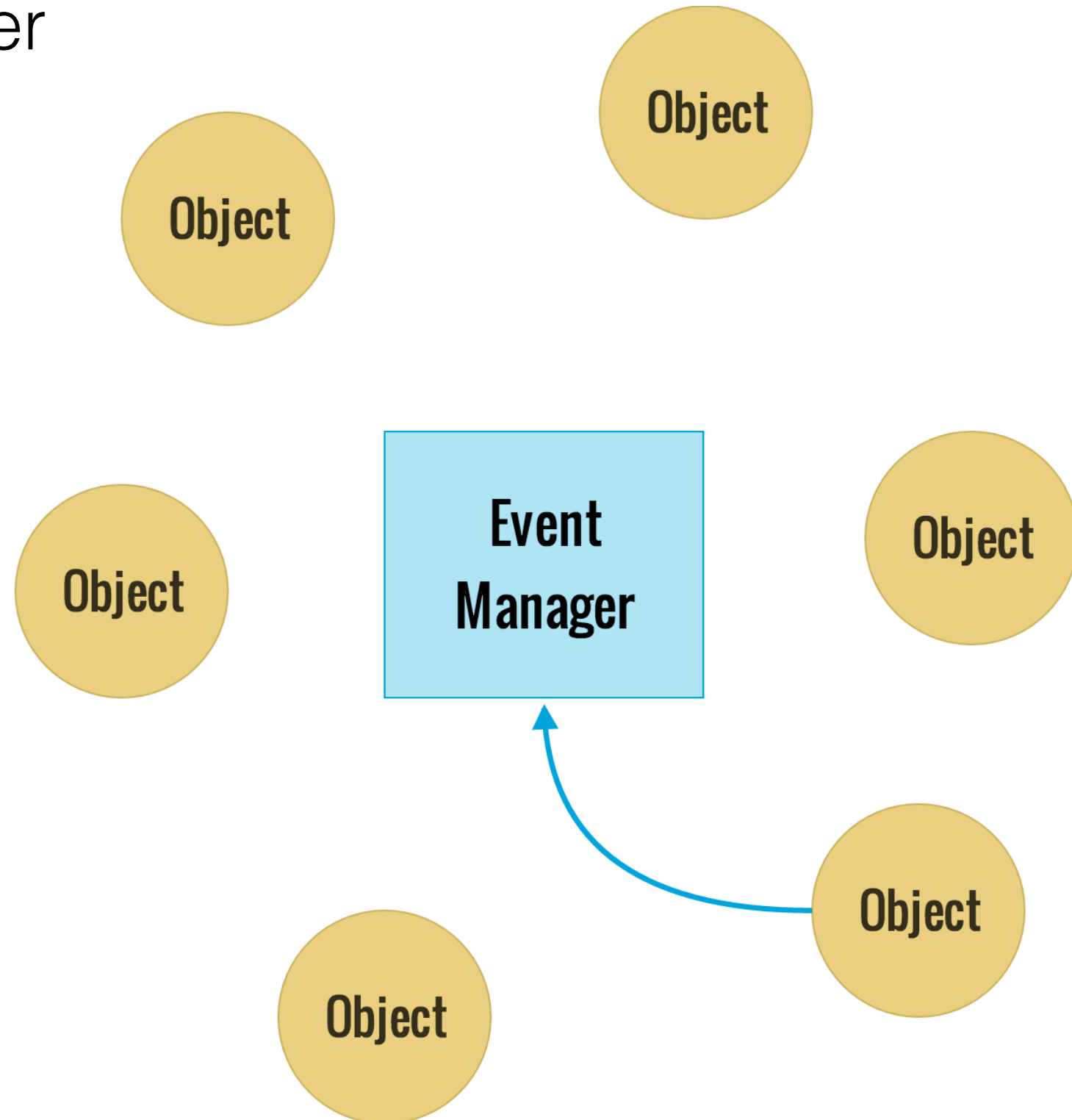
Pub/Sub

Publisher Subscriber



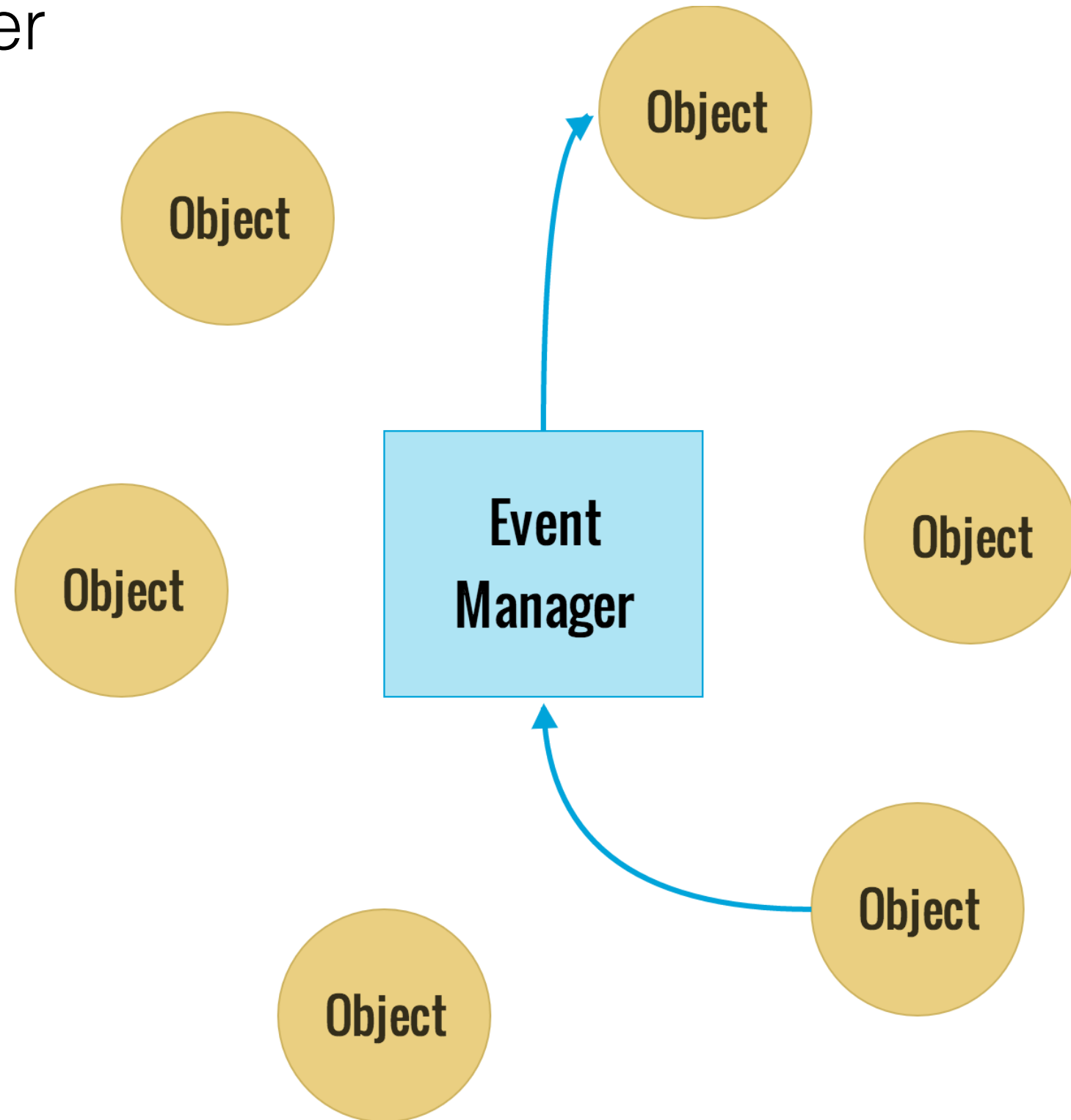
Pub/Sub

Publisher Subscriber



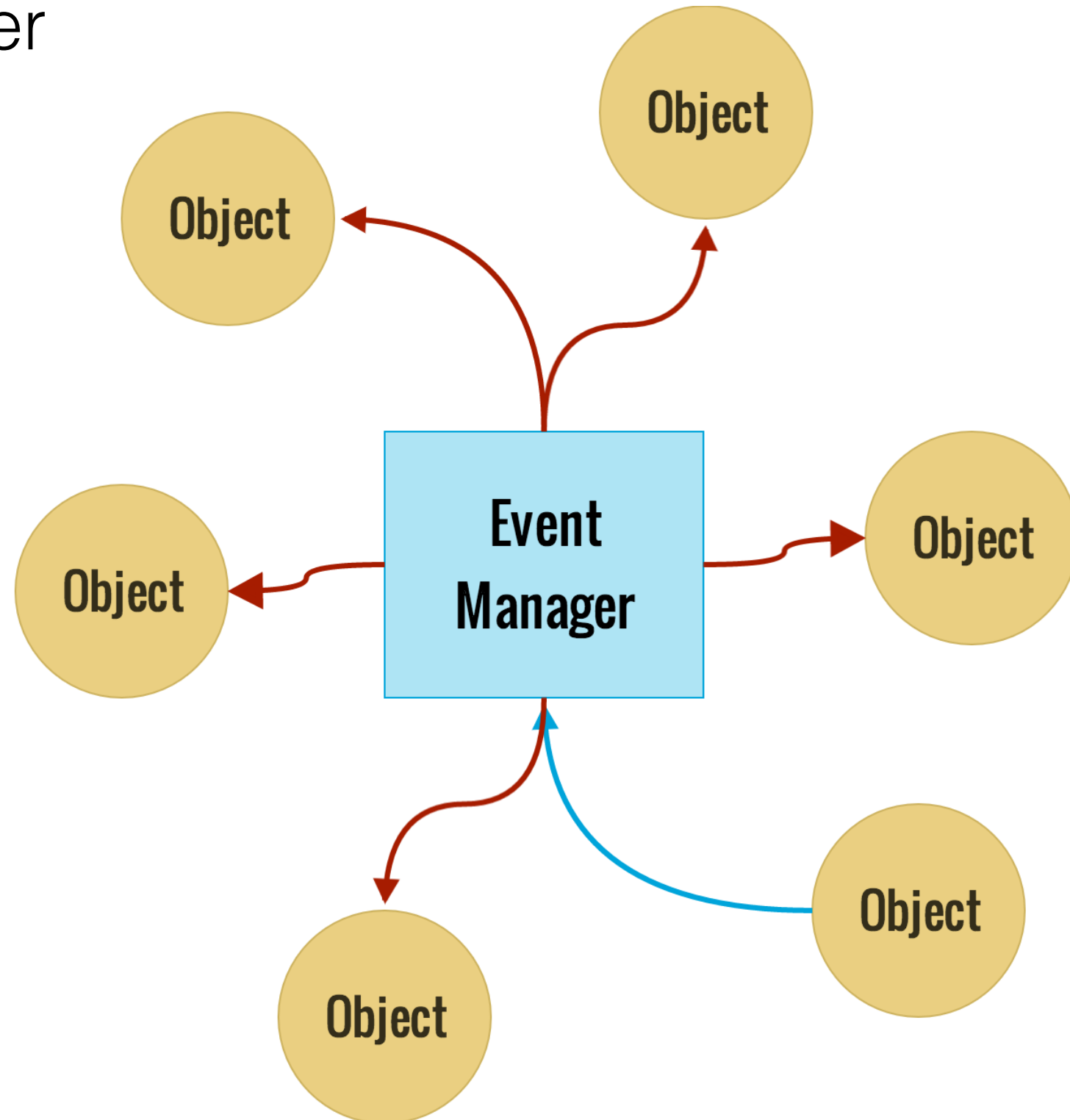
Pub/Sub

Publisher Subscriber

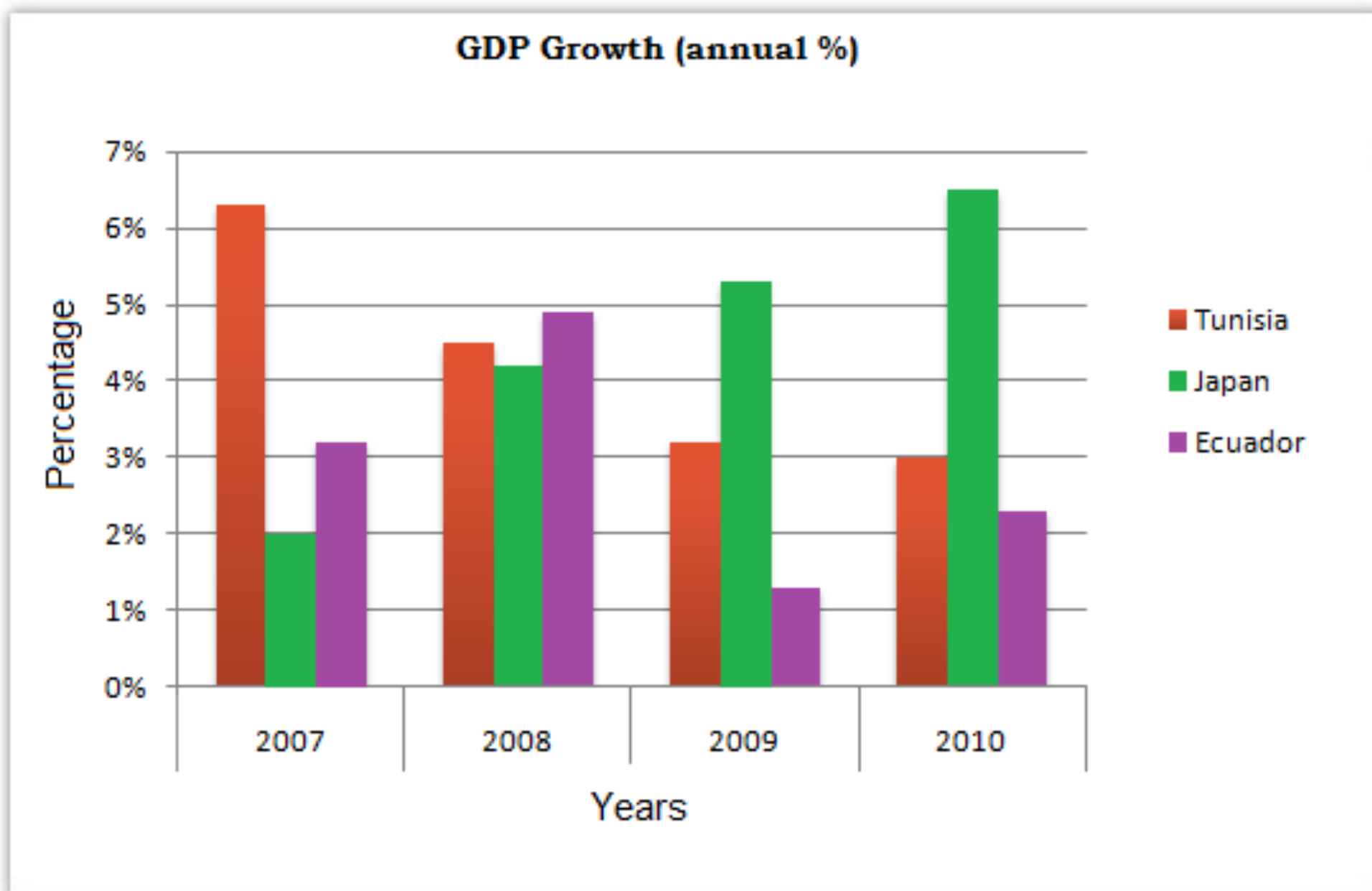


Pub/Sub

Publisher Subscriber



D3



ВСЕГО-ЛИШЬ

HTML

CSS

JS

SVG

DOM

```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```

```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```

```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```

```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```



```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```

```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```

```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```

```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```

```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```

```
01: d3.select('.container')
02:   .selectAll('div')
03:   .data(BIG_DATA)
04:   .enter()
05:   .append('div')
06:   .attr('class', 'bar')
07:   .style('width', function(d) {
08:     return d + 'px';
09:   })
10:   .text(function(d) {
11:     return d;
12:   })
13:   .exit();
```

```
var BIG_DATA = [16, 23, 42, 100, 20, 77, 22, 2];
```

