Программирование на JavaScript

Лекция 4
Наследование

Hexlet University

Конструктор

```
var alex = new Human();
```

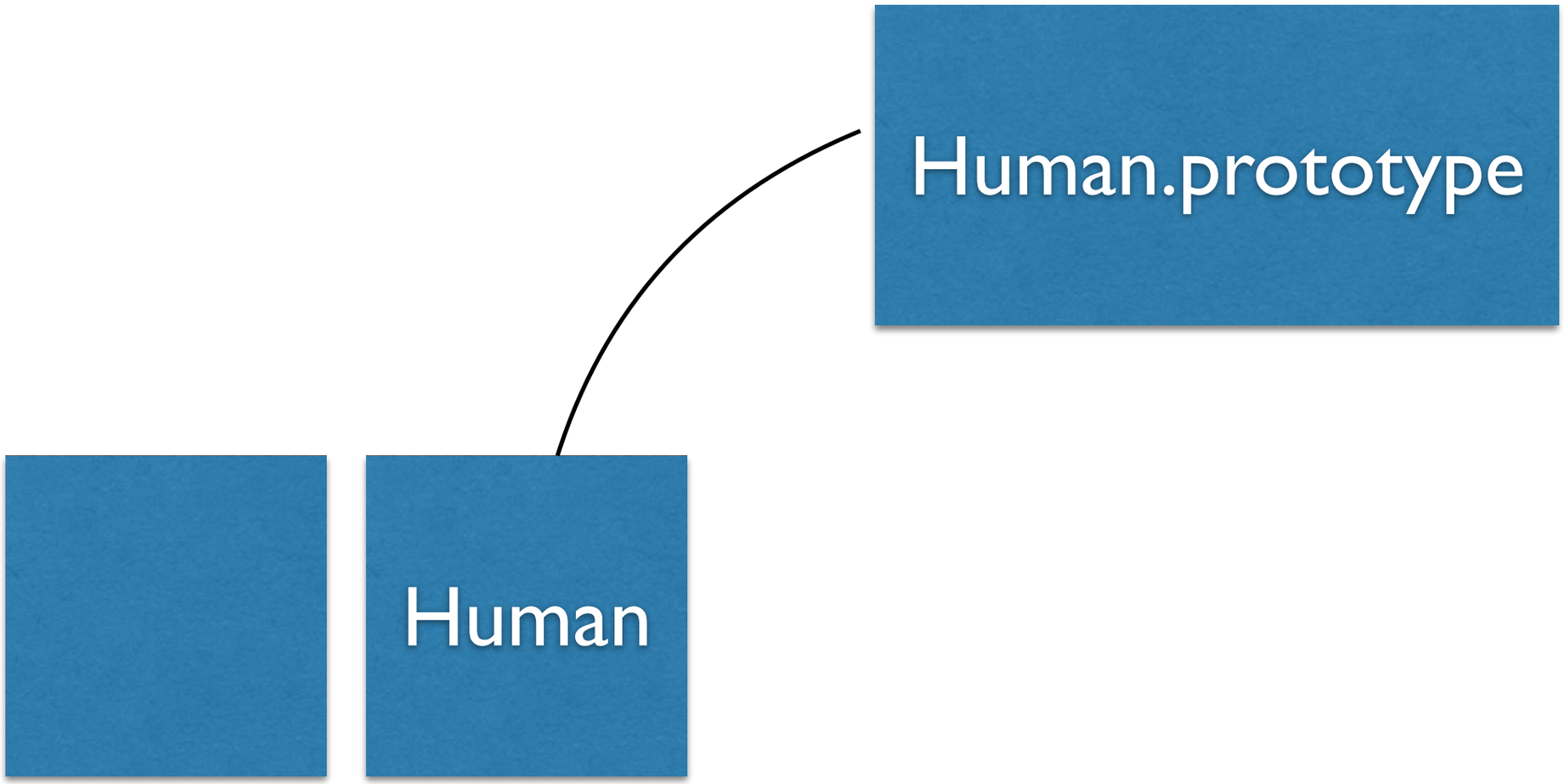
Конструктор

функция может быть конструктором

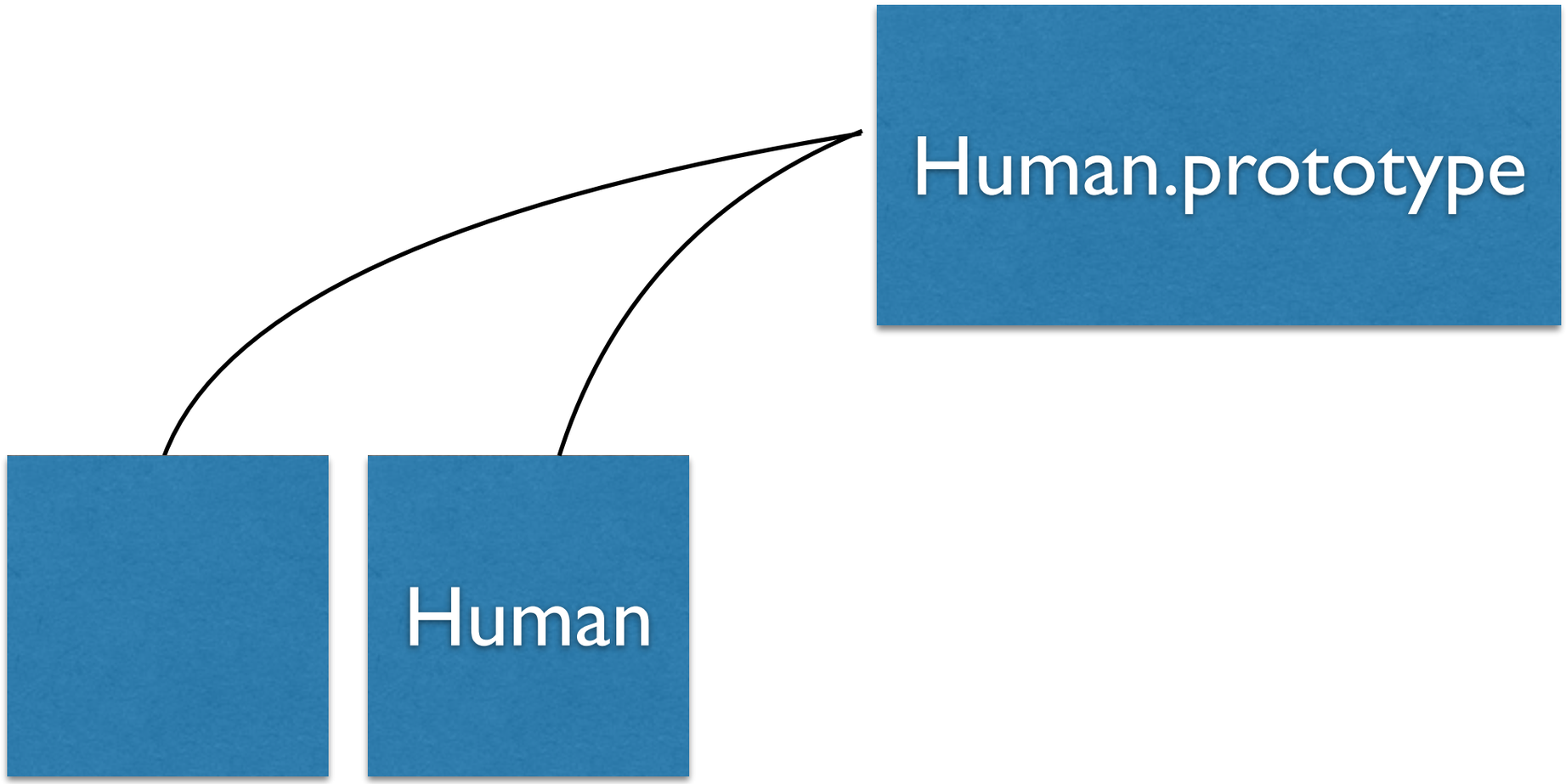
```
var alex = new Human();
```

```
var alex = new Human();
```

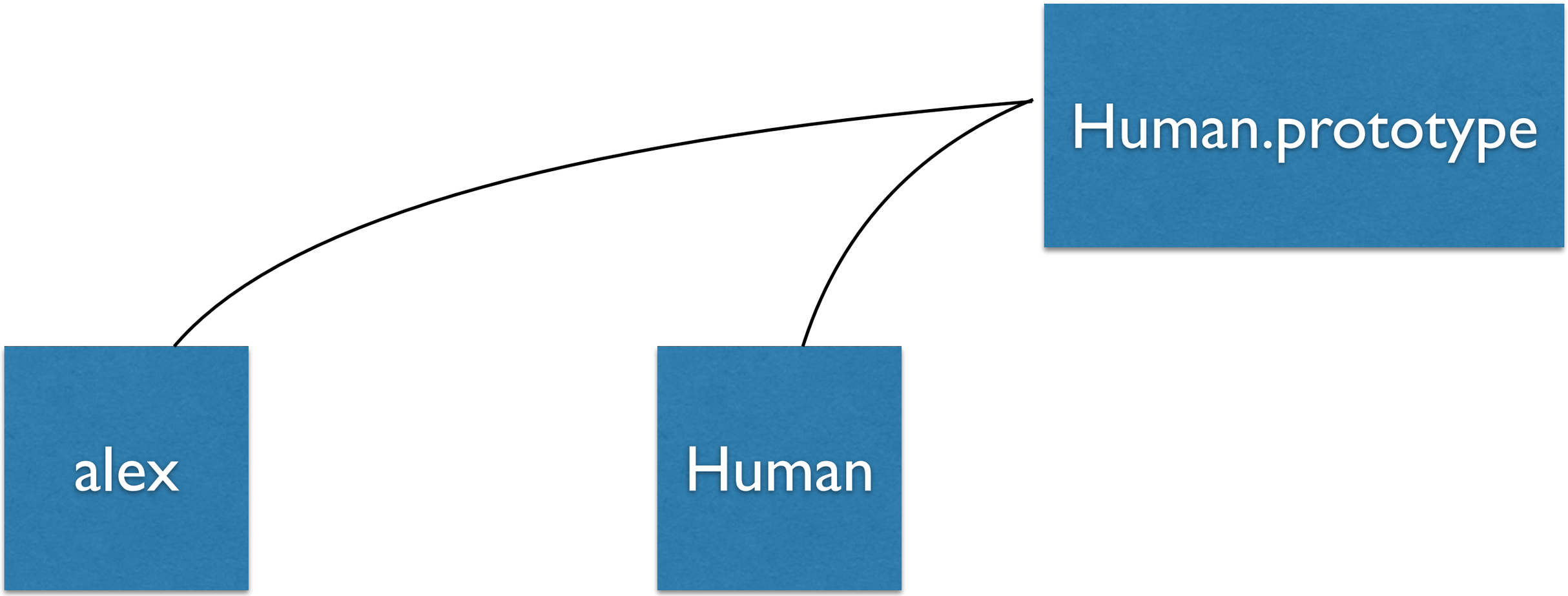
```
var alex = new Human();
```



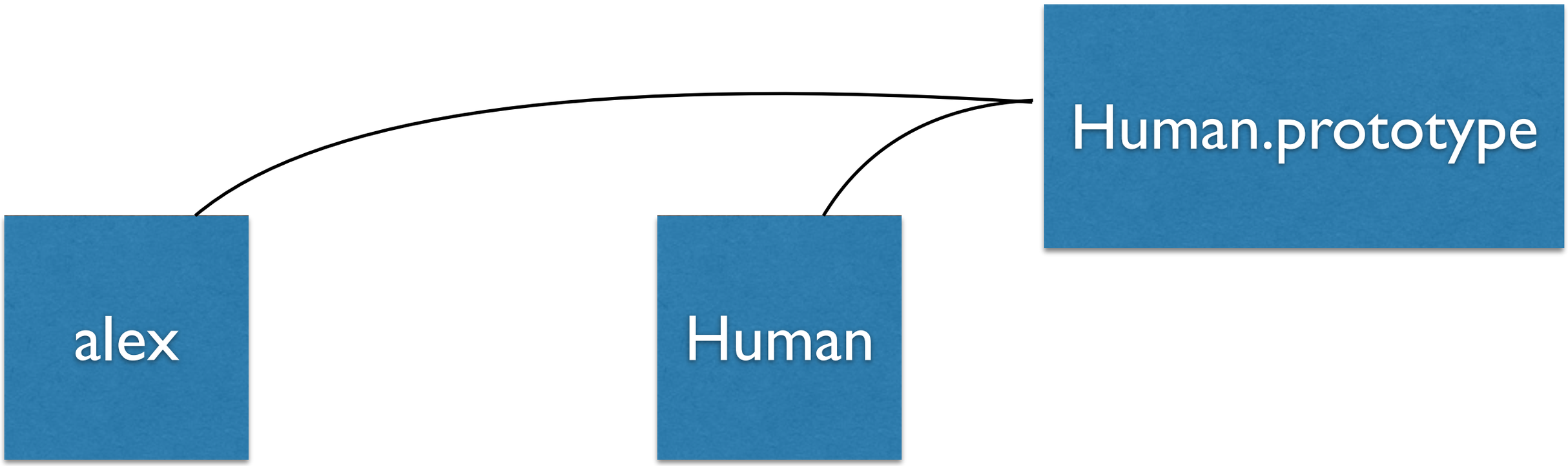
```
var alex = new Human();
```



```
var alex = new Human();
```



```
var alex = new Human();
```

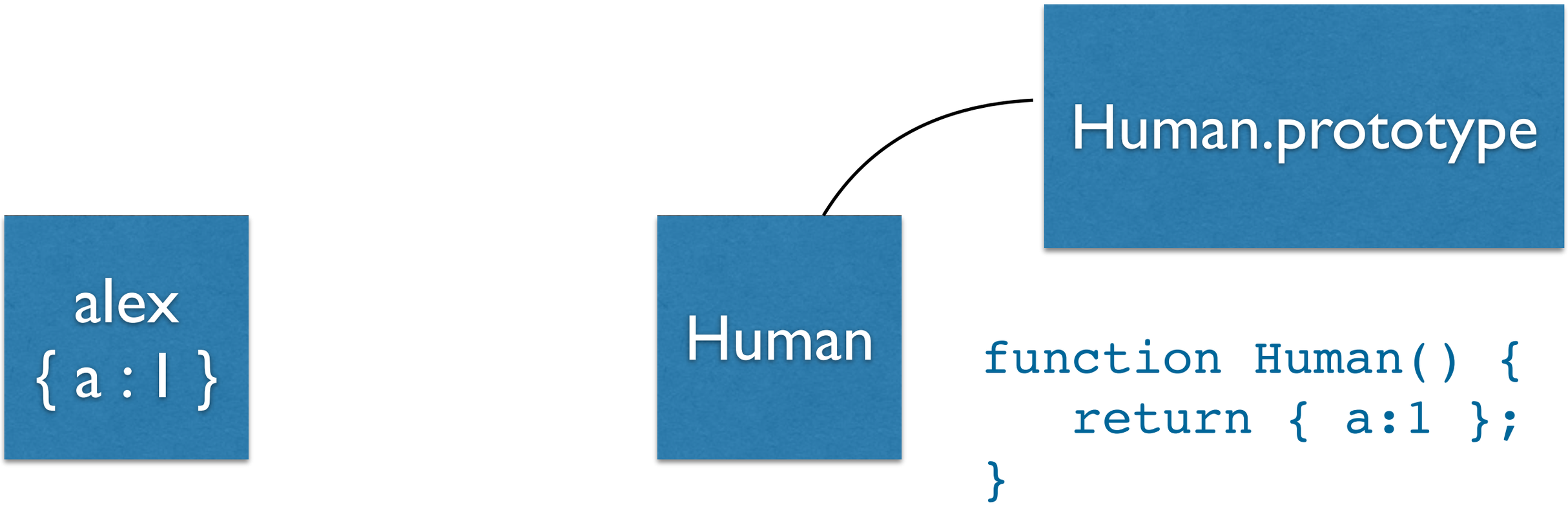


Это работает если функция Human
ничего не возвращает

или

возвращает примитивный тип (не объект)

```
var alex = new Human();
```



Если функция Human
возвращает объект,
то он и будет возвращен
без указания нового прототипа

Наследование в JS

- Объект содержит свойства
- Объект содержит специальное свойство, указывающее на прототип объекта
- Объект может переопределять любое свойство прототипа
- Конструктор создает объект. Прототипом этого объекта будет прототип конструктора

super?

```
function Human(name) { this.name = name; }

Human.prototype.say = function(what) {
    console.log(this.name + ": " + what);
}
```

```
function Human(name) { this.name = name; }

Human.prototype.say = function(what) {
    console.log(this.name + ": " + what);
}


var alex = new Human("Alex");

alex.say("hi");     // "Alex: hi"


var jack = new Human("Jack");

jack.say("hi");     // "Jack: hi"
```

```
function Human(name) { this.name = name; }

Human.prototype.say = function(what) {
    console.log(this.name + ": " + what);
}


var alex = new Human("Alex");

alex.say("hi");     // "Alex: hi"


var jack = new Human("Jack");

jack.say("hi");     // "Jack: hi"
```

this == alex

this == jack

```
function Human(name) { this.name = name; }

Human.prototype.say = function(what) {
    console.log(this.name + ": " + what);
}
```

```
var alex = new Human("Alex");
alex.say("hi");     // "Alex: hi"
var jack = new Human("Jack");
jack.say("hi");     // "Jack: hi"

alex.say.apply(jack,["hi"]);   // "Jack: hi"
```

functionName.apply(thisArg, [params])

объект, который будет `this` в вызове

параметры вызовы функции `functionName`

можно использовать `arguments`

из лекции 3

arguments

- Дополнительный параметр, доступный во время вызова функции
- Доступ ко всем аргументам (а не только параметрам)
- Позволяет писать функции с неопределенным количеством аргументов

параметры

```
function myFuncName (x, y, z) {...};
myFuncName (1,2,3,4,5,6,7,8);
```

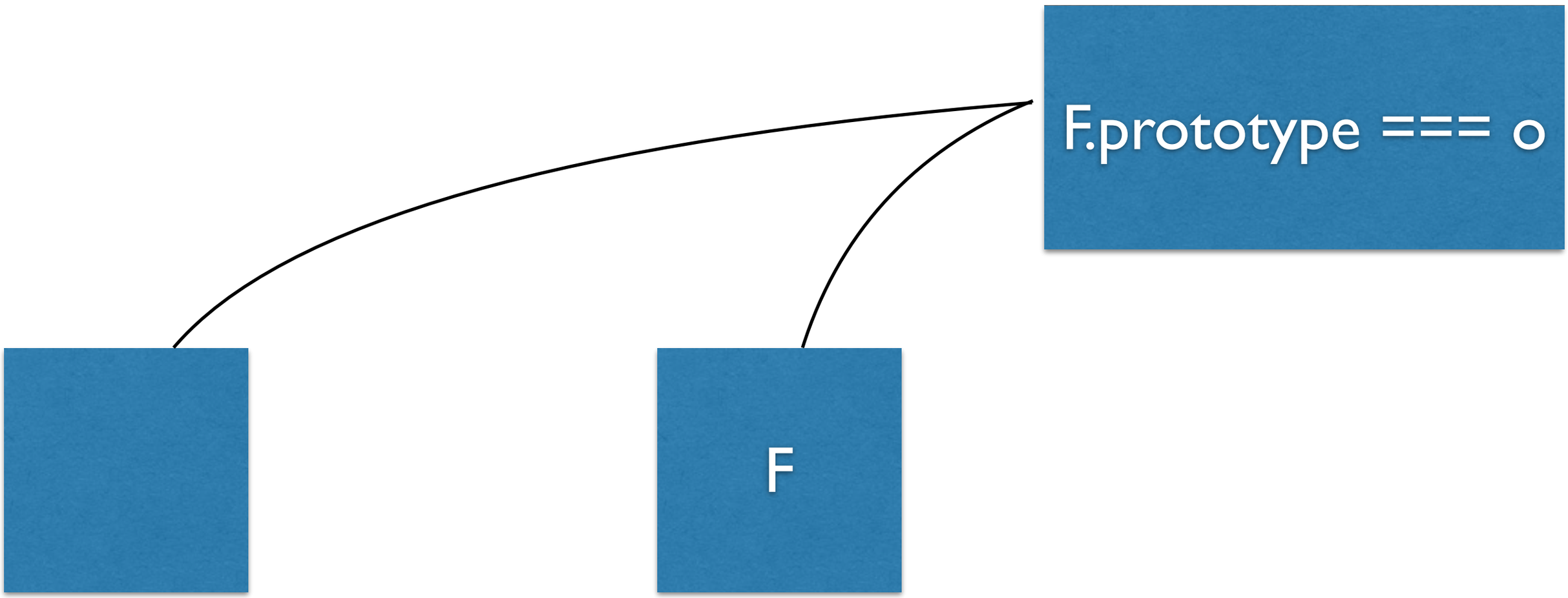
аргументы

Что предложил Дуглас Крокфорд

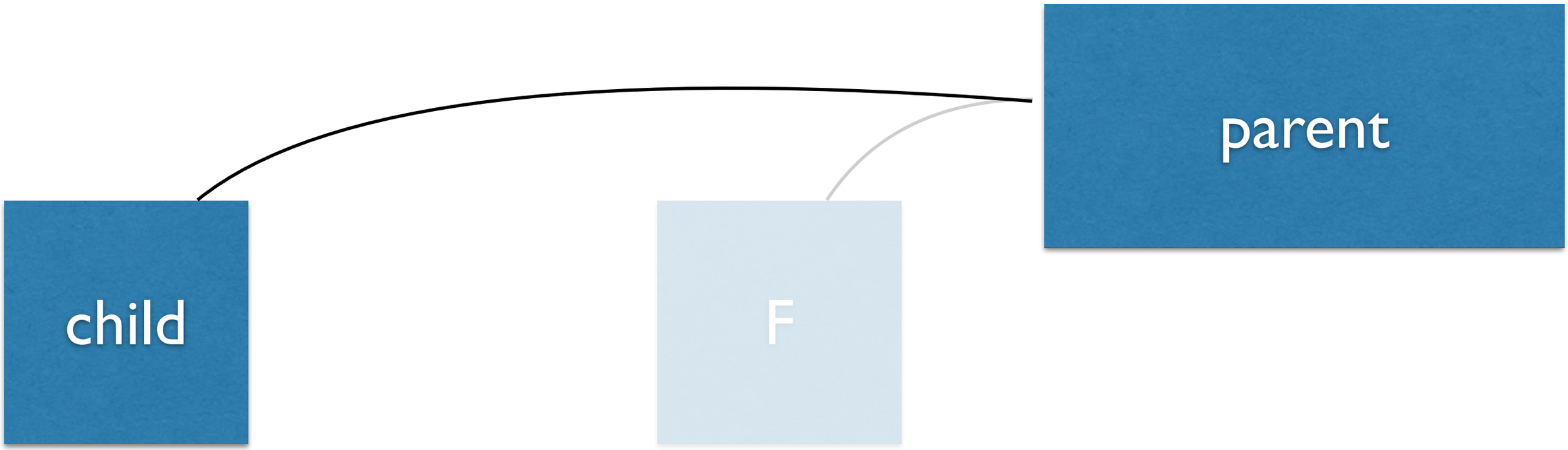
```
function object(o) {
    function F() {}
    F.prototype = o;
    return new F();
}
```

Что предложил Дуглас Крокфорд

```
function object(o) {
    function F() {}
    F.prototype = o;
    return new F();
}
```



```
function object(o) {
    function F() {}
    F.prototype = o;
    return new F();
}
```




```
var parent = { a : 1 };
var child = object(parent);

child.a;    // 1
```

Object.create()

ECMAScript 5
JavaScript 1.8.5