Object oriented Javascript

We're building a quiz game with users

Some of our users

```
Name: Will
Score: 3

Name: Tim
Score: 6

Functionality
+ Ability to increase score
```

What would be the best way to store this data?

Objects - store functions with their associated data!

```
var user1 = {
  name: "Will",
  score: 3,
  increaseScore: function() {
    user1.score++;
  }
};

user1.increaseScore(); //user1.score => 4
```

Creating user2

This time we use the 'dot notation'

```
var user2 = {}; //create an empty object

user2.name = "Tim"; //assign properties to that object
user2.score = 6;
user2.increaseScore = function() {
  user2.score++;
};
```

Creating user3

This time we use the Object.create approach to create the user

```
var user3 = Object.create(null);

user3.name = "Alex";
user3.score = 9;
user3.increaseScore = function() {
  user3.score++;
};
```

Our code is getting repetitive, we're breaking our DRY principle

And suppose we have millions of users!

What could we do?

Solution 1. Generate using a function

```
var user = function(name, score) {
  var newUser = {};
  newUser.name = name;
  newUser.score = score;
  newUser.increaseScore = function() {
    newUser.score++;
  };
  return newUser;
};

//later

var me = user("Will", 3);
var you = user("Tim", 5);
```

Problems:

Each time we create a new user we make space in our computer's memory for all our data and functions. But our functions are just copies

A better way?

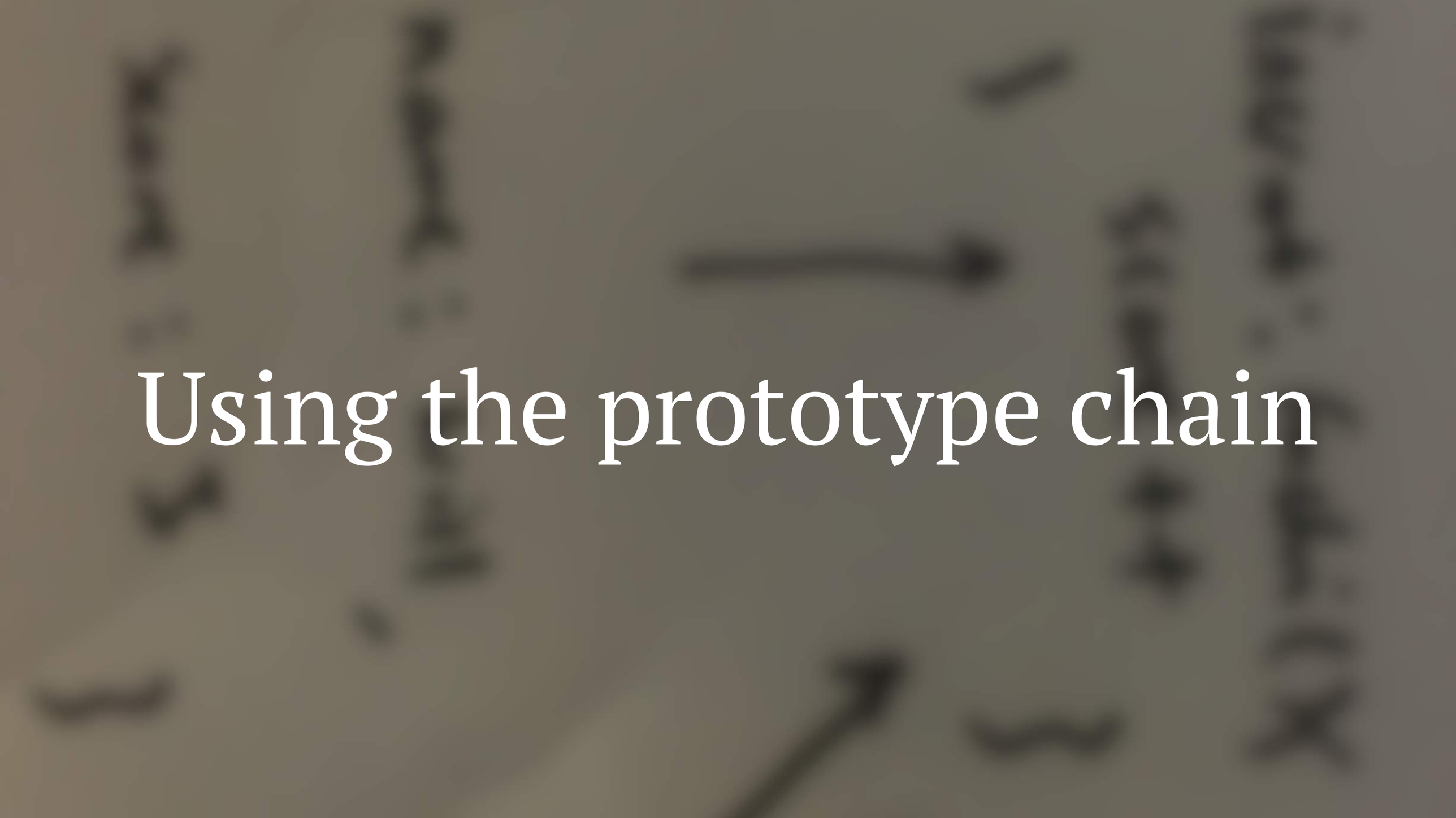
Benefits:

It's simple!

Solution 2:

Store the `iterate` function in just one object and have the interpreter, if it doesn't find the function on `user1`, look up to that object to check if it's there

How to make this link?



Using the prototype chain

```
functionStore = {
  increment: function(){ console.log("woo"); }
}

user1 = {
  name: "Will",
  score: 3
}

user1.name // name is a property of user1 object
user1.increment // Error! increment is not!
```

Link user1 and functionStore so the interpreter, on not finding .increment, makes sure to check up in functionStore where it would find it

Make the link with `Object.create()` technique

```
var user1 = Object.create(functionStore)
user1 // {}

user1.increment // function....
```

Interpreter doesn't find `.increment` on user1 and looks up the prototype chain to the next object and finds `.increment` 1 level up

Solution 2 in full

```
var user = function(name, score) {
  var newUser = Object.create(functionStore);
  newUser.name = name;
  newUser.score = score;
  return newUser;
};


functionStore = {};

functionStore.increment = function(){
  this.score++;
}

var user1 = user("Will", 3);
var user2 = user("Tim", 5);
```

Problem

No problems! It's beautiful

Maybe a little long-winded

```
var newUser = Object.create(functionStore);
...
return newUser
```

Write this every single time - but it's 6 words!

Super sophisticated but not standard

Solution 3

Introduce magic keyword new

```
var user1 = new user("Will", 3)
```

What happens when call user("Will", 3) normally?

When we call the generator function with new in front we automate 2 things

1. Create a new user object
2. return the new user object

We also have to name our functionStore generatorName.prototype

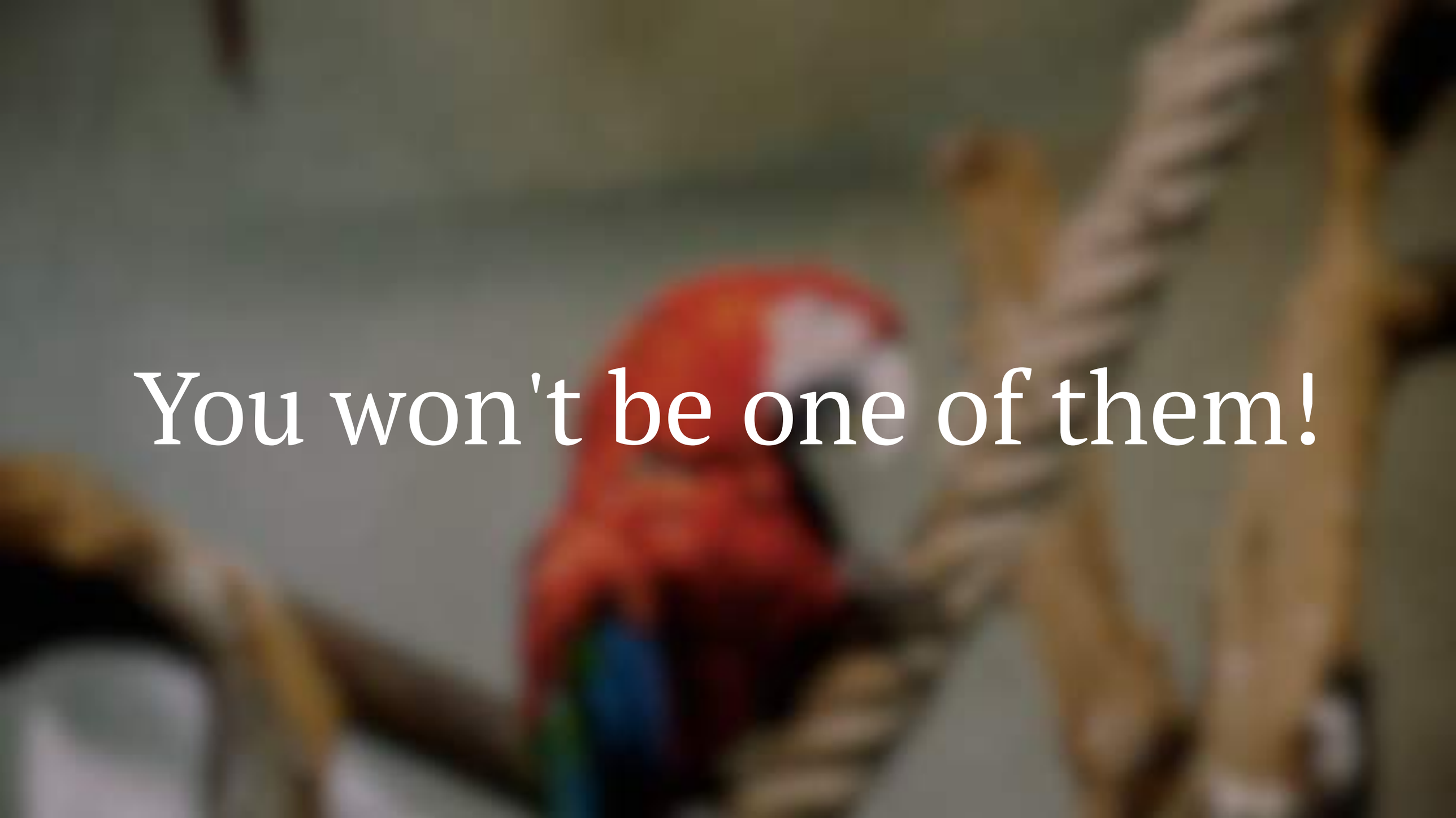
```
var user = function(name, score) {
 var newUser = Object.create(functionStore);
 newUser this.name = name;
 newUser this.score = score;
 return newUser;
};

functionStore user.prototype = {};
functionStore user.prototype.increment = function(){
 this.score++;
}

var user1 = new user("Will", 3);
```

Benefits:

— Very common in pro code
— Feels more like style of other languages
— A bit faster

Problems

— 95% of developers have no idea how it works and therefore fail interviews


You won't be one of them!