Javascript Closures and Design Patterns

Nathan Lapinski

Excellent open-sourced resources

The You Don't Know JS titles are by far the best resources I have ever seen for learning the ins and outs of JS. And you can read them for free!
My experience has been that scope & closures and especially this & object prototypes are the topics that come up the most on JS interviews.

https://github.com/getify/You-Dont-Know-JS

This one is also really good if you are kind of new to the language. Again, completely free: http://eloquentjavascript.net/

Logistics

Topics

Topics

1. Scope

Topics

1. Scope
2. Closures

Topics

1. Scope
2. Closures
3. Prototypes

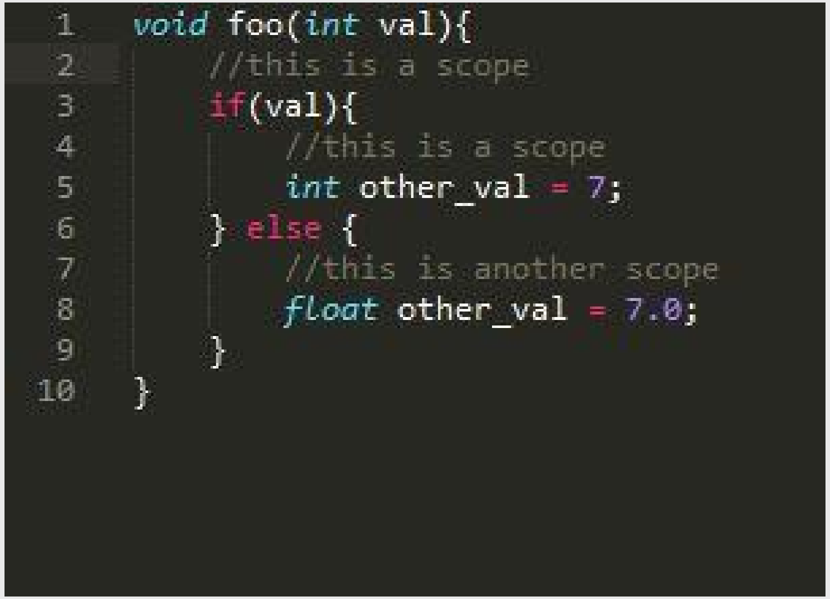
Topics

1. Scope
2. Closures
3. Prototypes
4. this

Scope

Scope is where to look for things

Scope

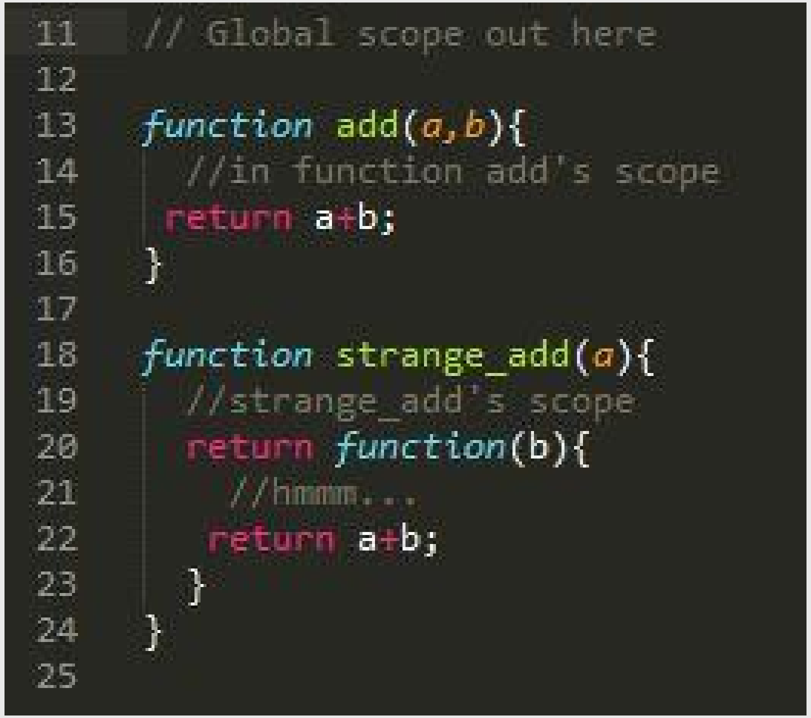


```
void foo(int val){
    //this is a scope
    if(val){
        //this is a scope
        int other_val = 7;
    } else {
        //this is another scope
        float other_val = 7.0;
    }
}
```

Scope

Javascript uses functions to define scope

Scope

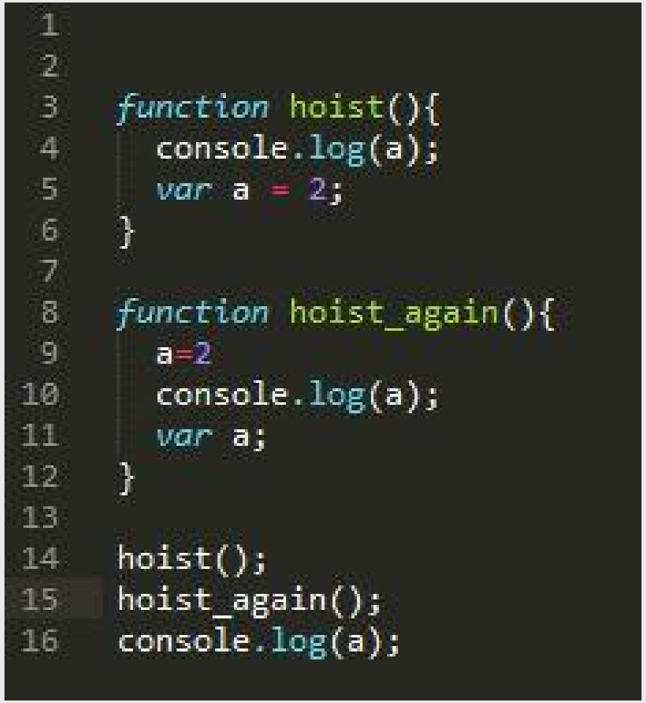


```
// Global scope out here

function add(a,b){
  //in function add's scope
 return a+b;
}

function strange_add(a){
  //strange_add's scope
  return function(b){
    //hmmm...
   return a+b;
  }
}

```

Scope

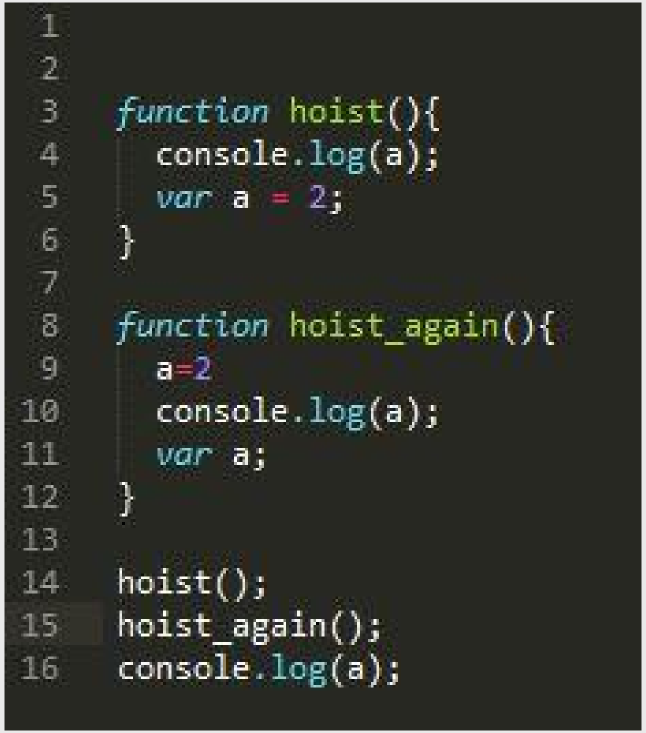
Things can get kind of strange though

Hoisting



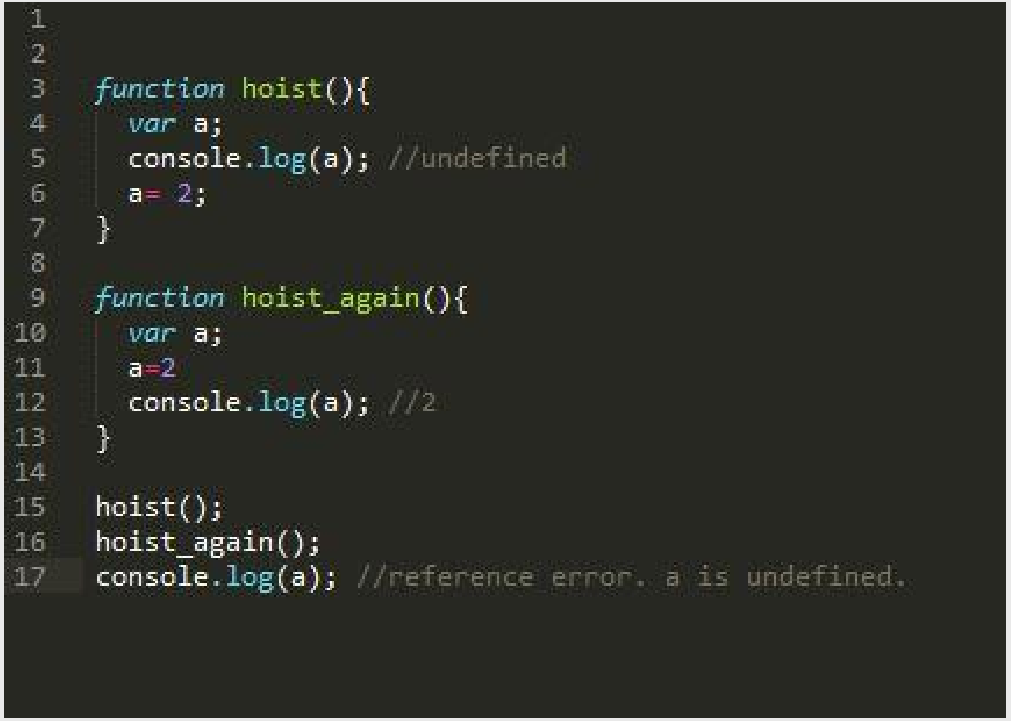
```


function hoist(){
  console.log(a);
  var a = 2;
}

function hoist_again(){
  a=2
  console.log(a);
  var a;
}

hoist();
hoist_again();
console.log(a);
```

Hoisting



```


function hoist(){
    console.log(a);
    var a = 2;
}

function hoist_again(){
    a=2
    console.log(a);
    var a;
}

hoist();
hoist_again();
console.log(a);
```

=



```


function hoist(){
    var a;
    console.log(a); //undefined
    a= 2;
}

function hoist_again(){
    var a;
    a=2
    console.log(a); //2
}

hoist();
hoist_again();
console.log(a); //reference error. a is undefined.
```

Quiz Time

Is Javascript compiled or interpreted?

Quiz Time

Yep, it’s compiled

V8

Yep, it’s compiled

V8

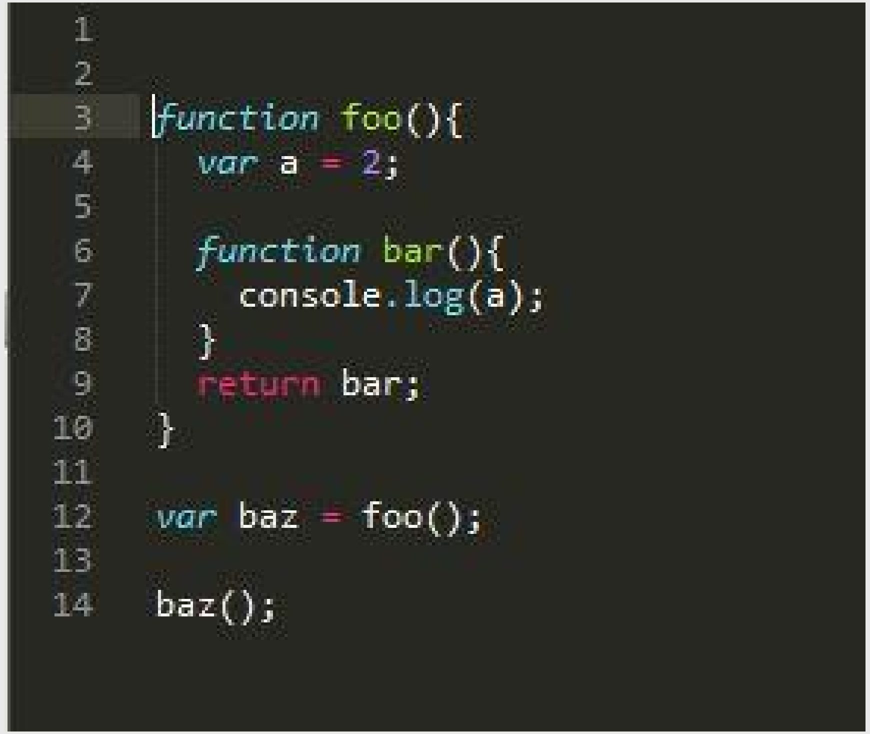
Yep, it’s compiled

http://www.html5rocks.com/en/tutorials/speed/v8/

Scope Closure

Let’s talk closures

Scope Closure

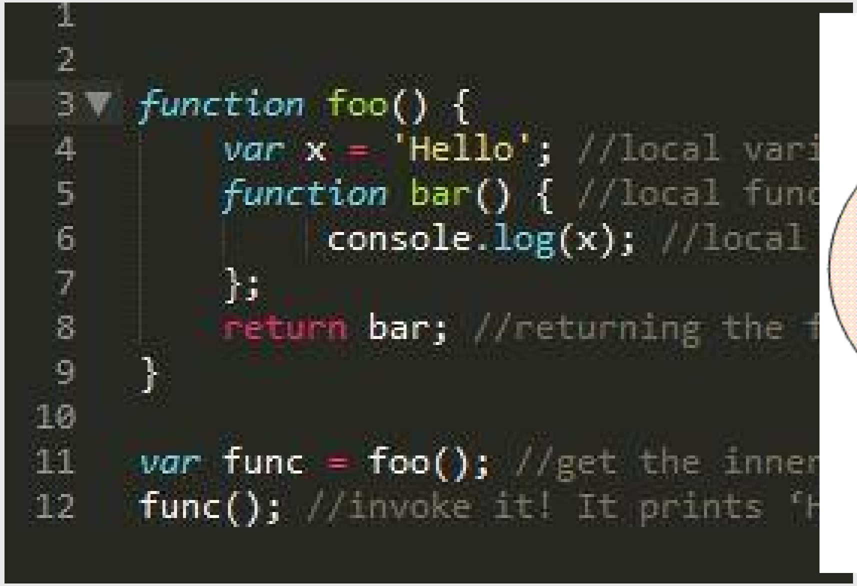


```


function foo(){
  var a = 2;

  function bar(){
    console.log(a);
  }
  return bar;
}

var baz = foo();

baz();
```

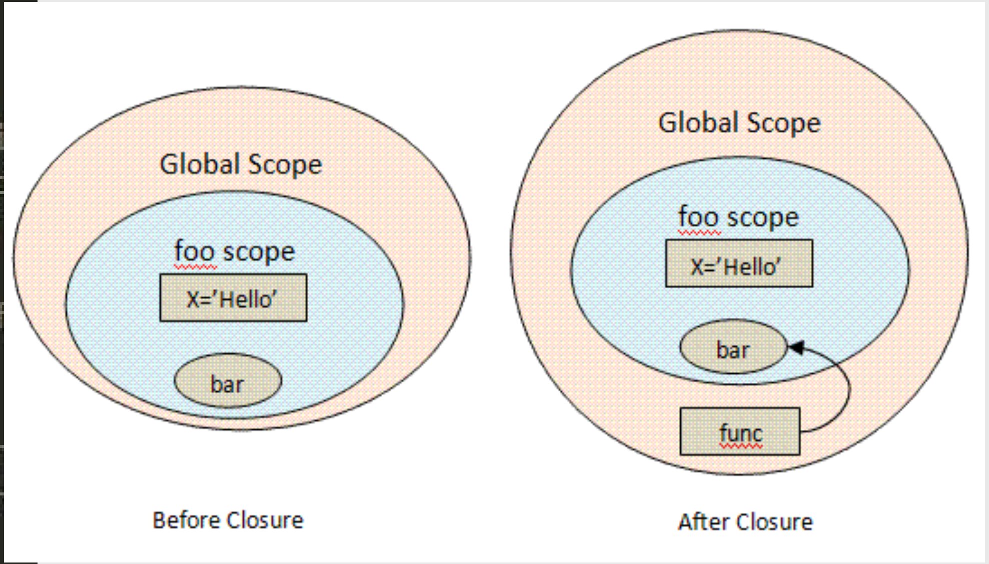
Scope Closure

what is a closure?

Scope Closure



```
function foo() {
        var x = 'Hello'; //local vari
        function bar() { //local func
                console.log(x); //local
        };
        return bar; //returning the f
}

var func = foo(); //get the inner
func(); //invoke it! It prints 'H
```



Modules

Angular services, module pattern

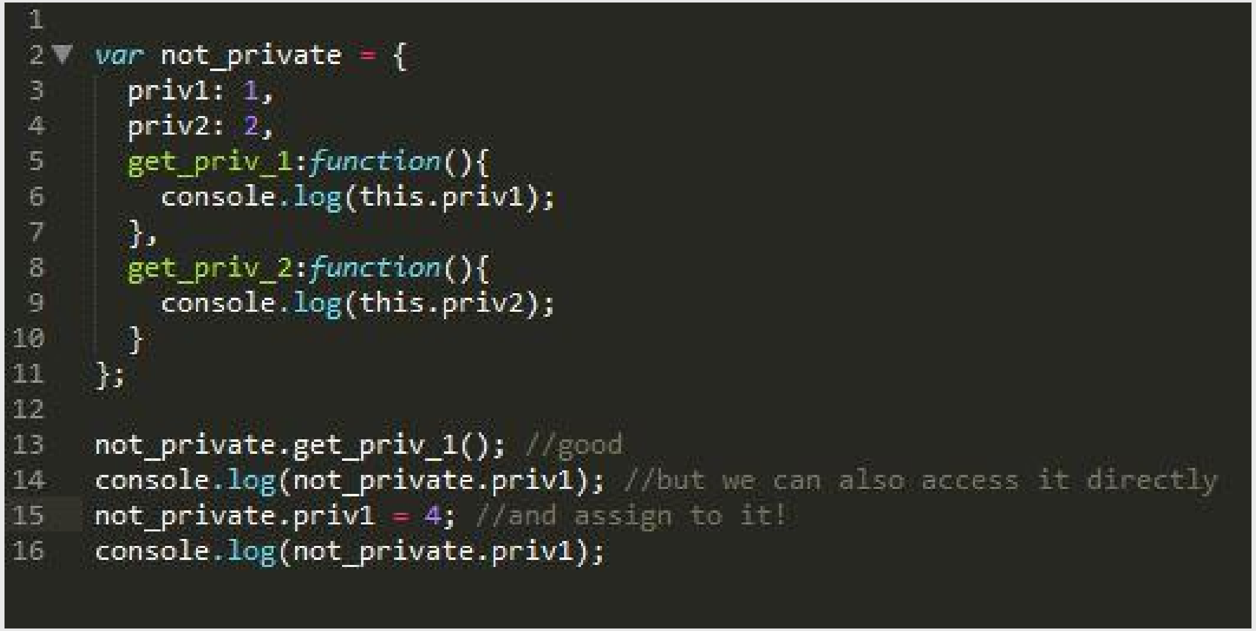
Javascript Module

Javascript doesn’t have “private” data

Javascript Module

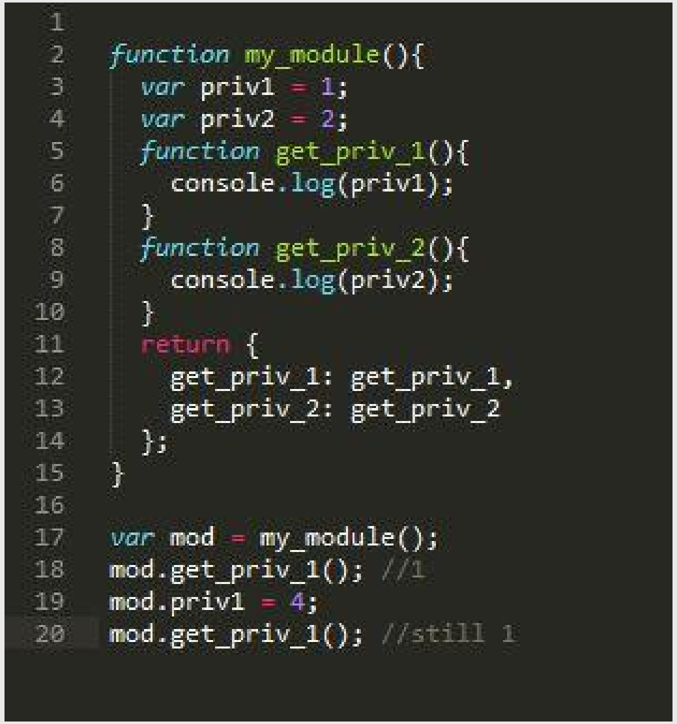
So use scope to hide data, and export an object that closes over it.

Javascript Module



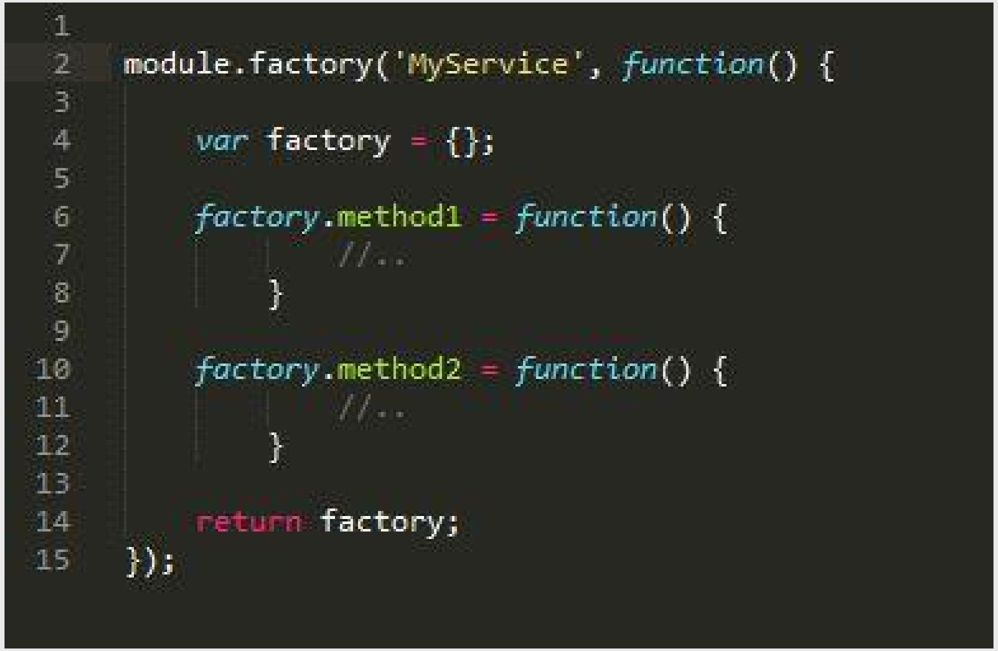
```javascript

var not_private = {
    priv1: 1,
    priv2: 2,
    get_priv_1:function(){
      console.log(this.priv1);
    },
    get_priv_2:function(){
      console.log(this.priv2);
    }
};

not_private.get_priv_1(); //good
console.log(not_private.priv1); //but we can also access it directly
not_private.priv1 = 4; //and assign to it!
console.log(not_private.priv1);
```

Javascript Module



```
function my_module(){
  var priv1 = 1;
  var priv2 = 2;
  function get_priv_1(){
    console.log(priv1);
  }
  function get_priv_2(){
    console.log(priv2);
  }
  return {
    get_priv_1: get_priv_1,
    get_priv_2: get_priv_2
  };
}

var mod = my_module();
mod.get_priv_1(); //1
mod.priv1 = 4;
mod.get_priv_1(); //still 1
```

Angular Service



```

module.factory('MyService', function() {

    var factory = {};

    factory.method1 = function() {
            //..
        }

    factory.method2 = function() {
            //..
        }

    return factory;
});
```

Javascript

Class vs Prototype

Javascript

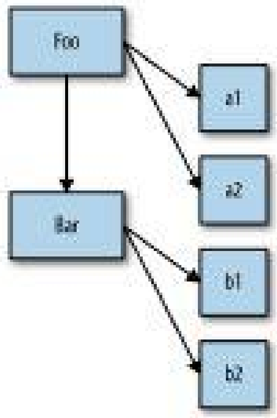
Class vs Prototype

(Inheritance vs. Delegation)

Java,C++,etc

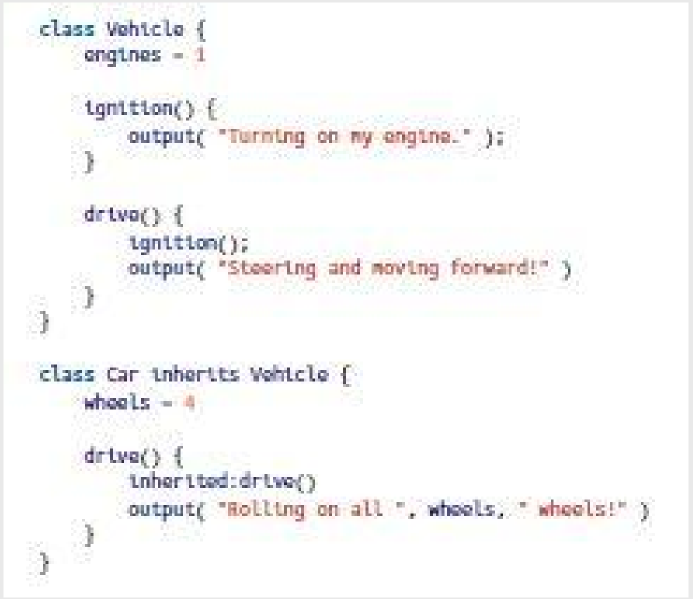
Inheritance means having an abstract base class and specializing with derived classes

Java,C++,etc



As you can see, the arrows move from left to right, and from top to bottom, which indicates the copy operations that occur, both conceptually and physically.

Java,C++,etc



```
class Vehicle {
    engines = 1

    ignition() {
        output( "Turning on my engine." );
    }

    drive() {
        ignition();
        output( "Steering and moving forward!" )
    }
}

class Car inherits Vehicle {
    wheels = 4

    drive() {
        inherited:drive()
        output( "Rolling on all ", wheels, " wheels!" )
    }
}
```

Javascript

Javascript uses prototype inheritance

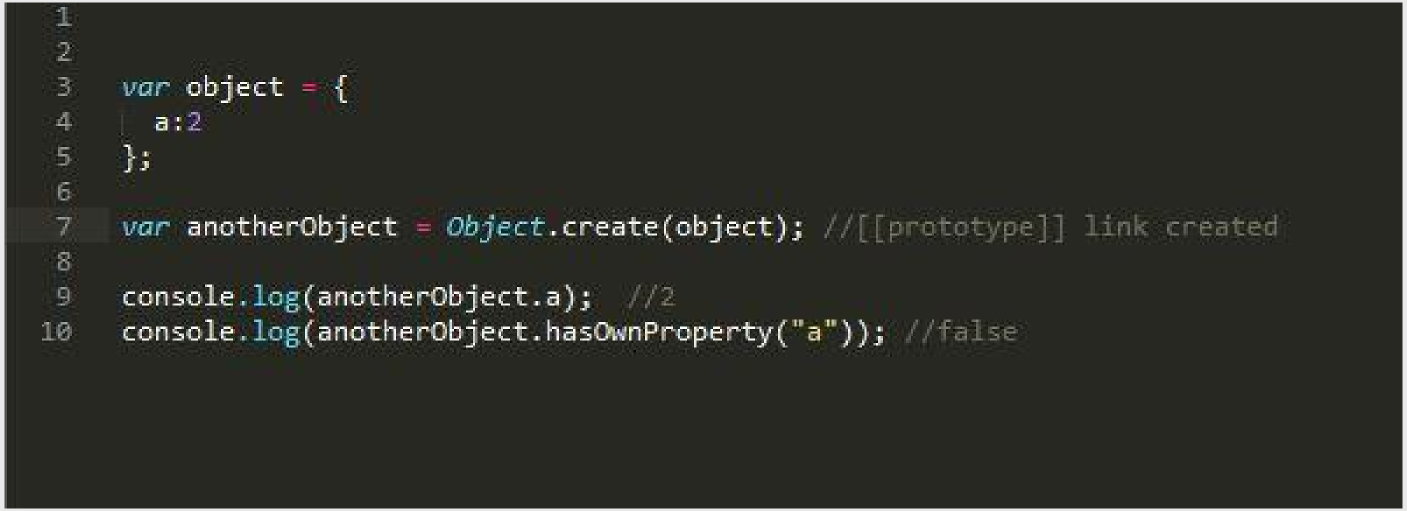
Javascript

it has no notion of a `class`

Javascript

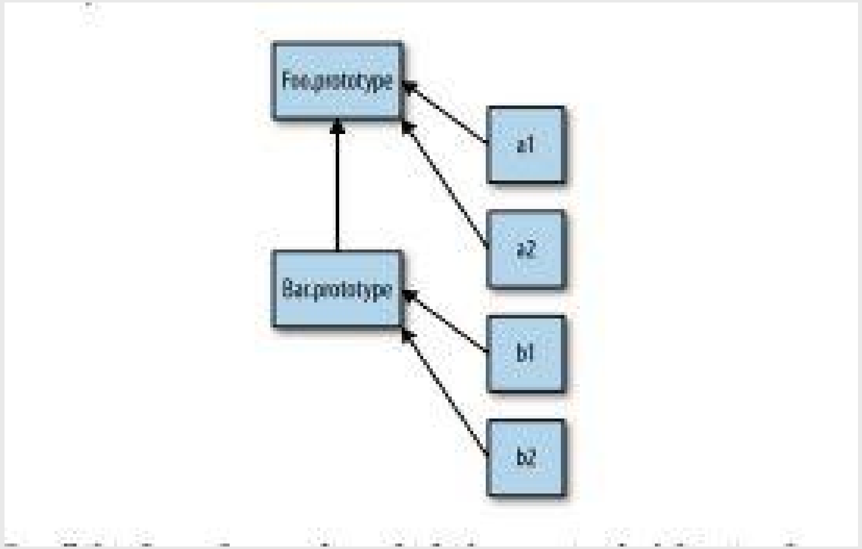
prototypes use delegation

[[prototype]]



```


var object = {
  a:2
};

var anotherObject = Object.create(object); //[[prototype]] link created

console.log(anotherObject.a);  //2
console.log(anotherObject.hasOwnProperty("a")); //false
```

[[prototype]]



this

Let’s talk about something else

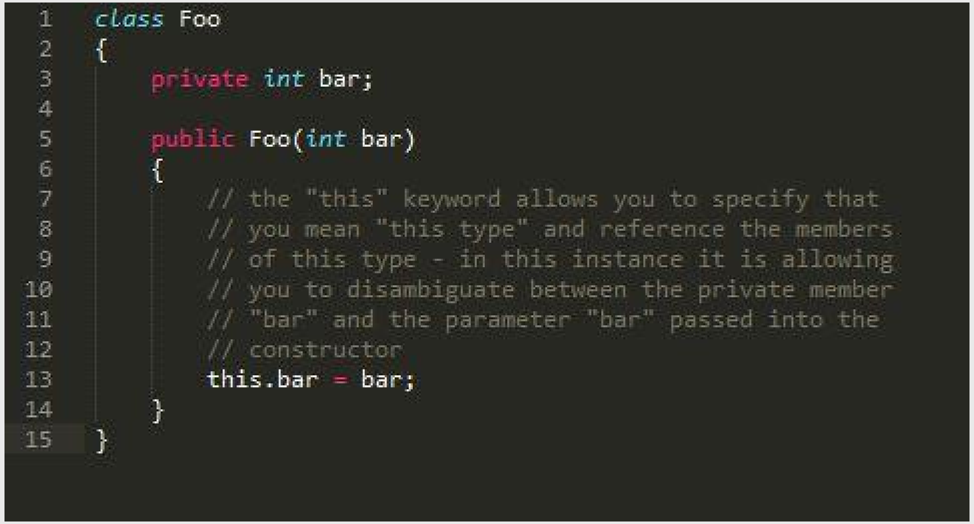
this

This always trips people up

Java,C++

this is bound at compile time

Java,C++



```
class Foo
{
    private int bar;

    public Foo(int bar)
    {
        // the "this" keyword allows you to specify that
        // you mean "this type" and reference the members
        // of this type - in this instance it is allowing
        // you to disambiguate between the private member
        // "bar" and the parameter "bar" passed into the
        // constructor
        this.bar = bar;
    }
}
```

Javascript

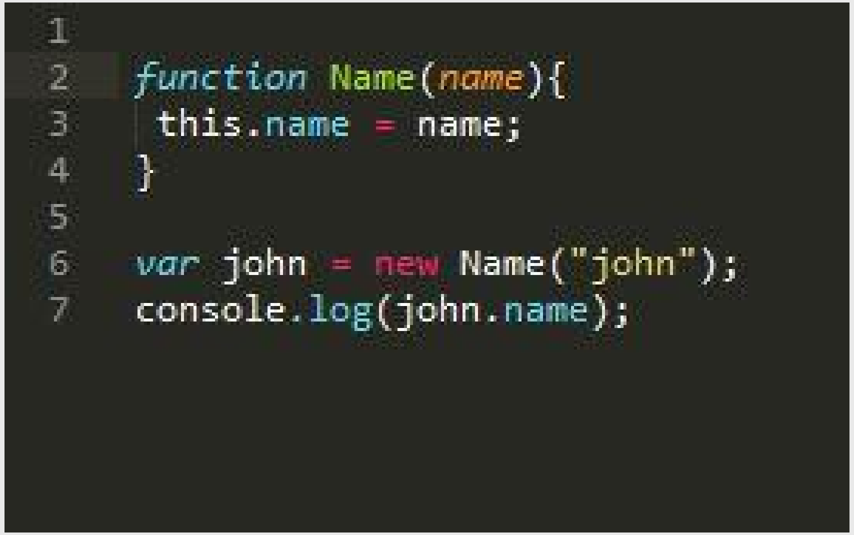
In JS, this is bound at run-time, and is determined entirely by the call site.

Javascript

In JS, this is bound at run-time, and is determined entirely by the call site.

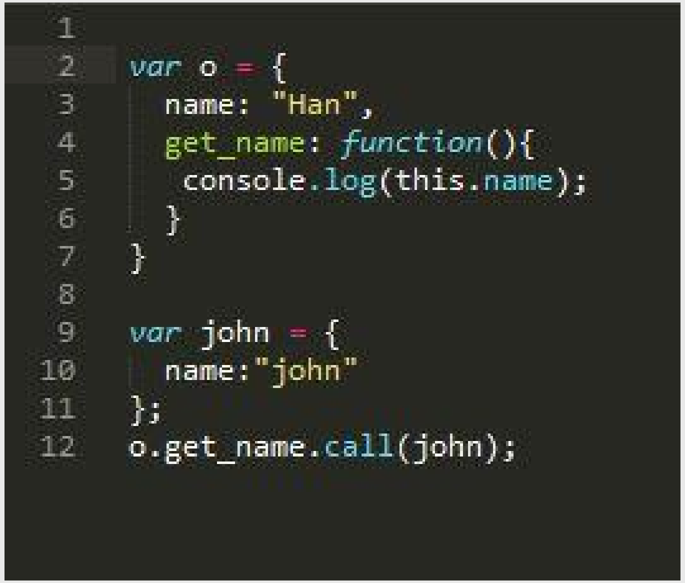
(4 scenarios)

Case #1: new



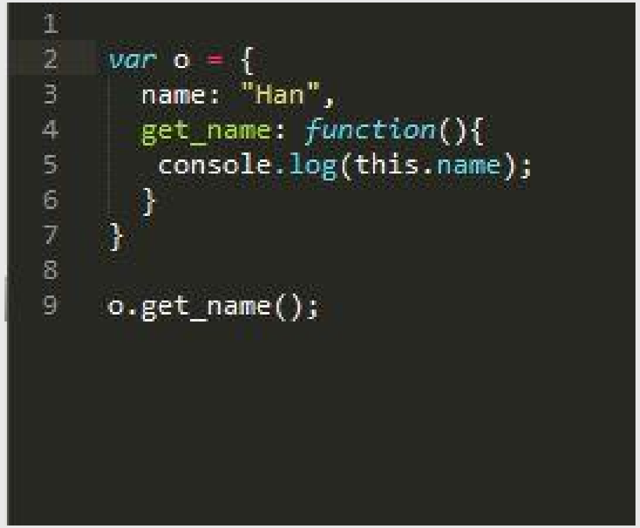
```

function Name(name){
 this.name = name;
}

var john = new Name("john");
console.log(john.name);
```

Case #2: explicit



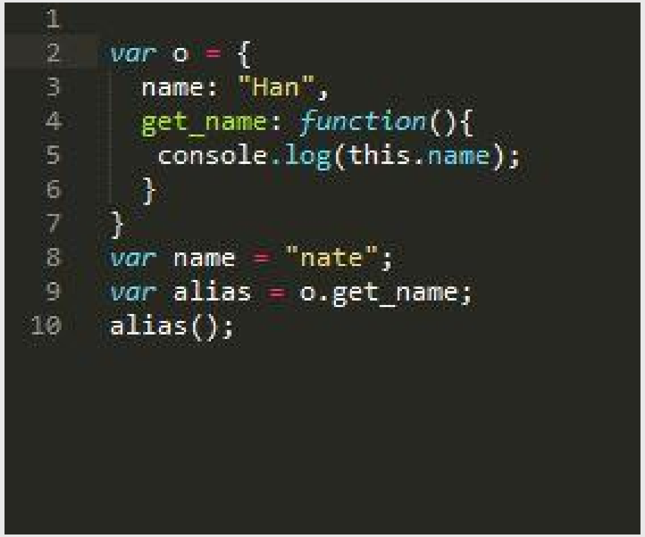
```javascript

var o = {
    name: "Han",
    get_name: function(){
     console.log(this.name);
    }
}

var john = {
    name:"john"
};
o.get_name.call(john);
```

Case #3: implicit



```

var o = {
  name: "Han",
  get_name: function(){
   console.log(this.name);
  }
}

o.get_name();
```

Case #4: default



```
var o = {
  name: "Han",
  get_name: function(){
   console.log(this.name);
  }
}
var name = "nate";
var alias = o.get_name;
alias();
```

JS

So yea…JS is kind of weird

JS

but it’s also everywhere!

JS

It’s best to understand the technology that you use

End

Thanks!