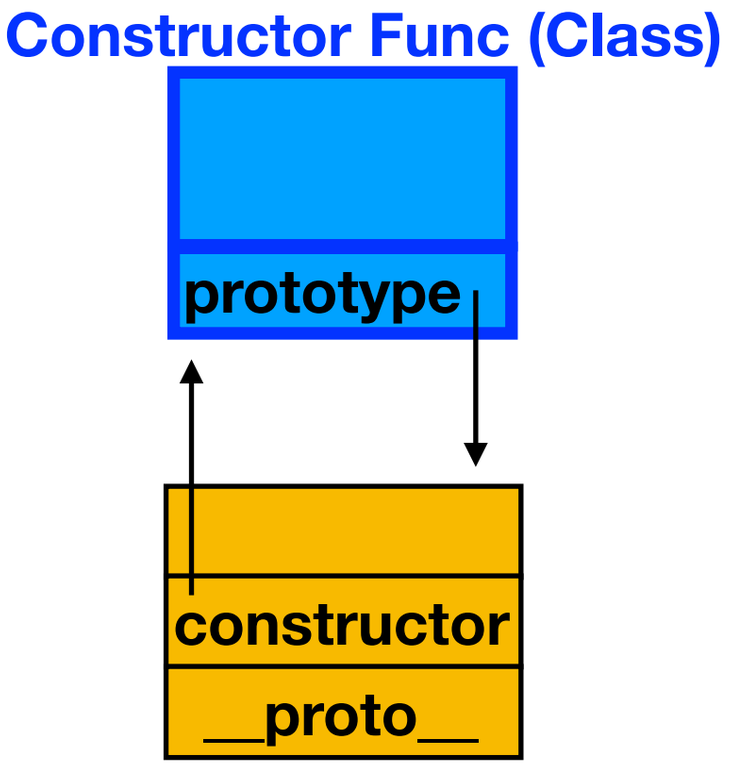
Prototypal Inheritance

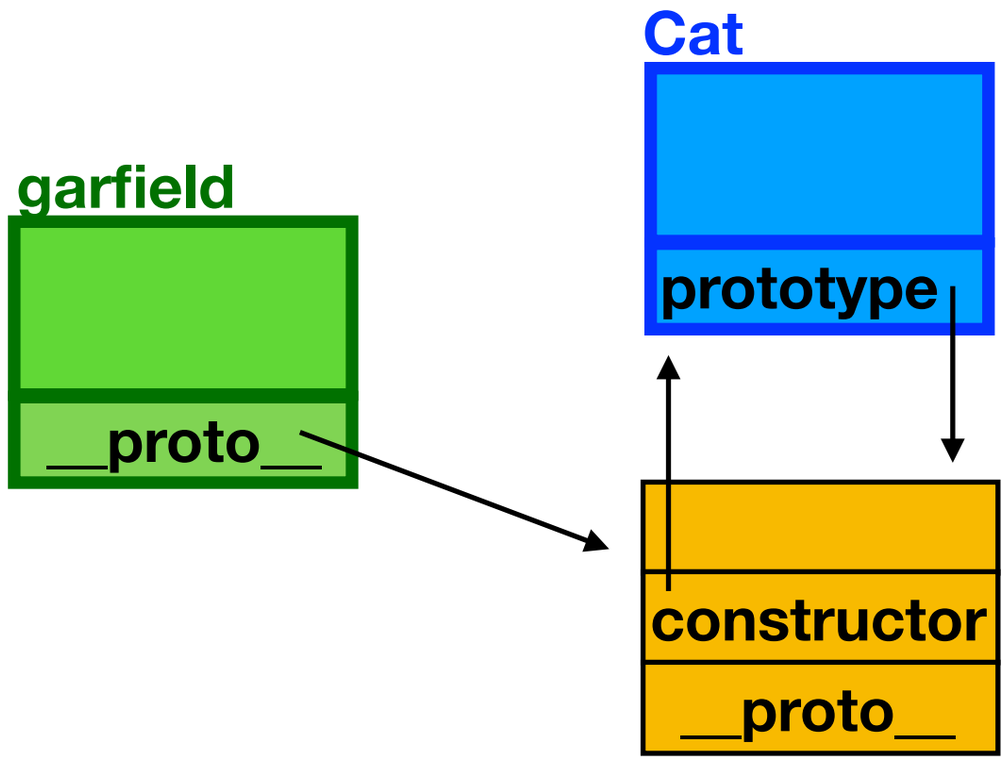
Constructor Functions & Prototypes

The blue object is a constructor function.
Every constructor function has a pointer to its prototype.
Every prototype object has a pointer back to the constructor.



Constructor Functions & Prototypes

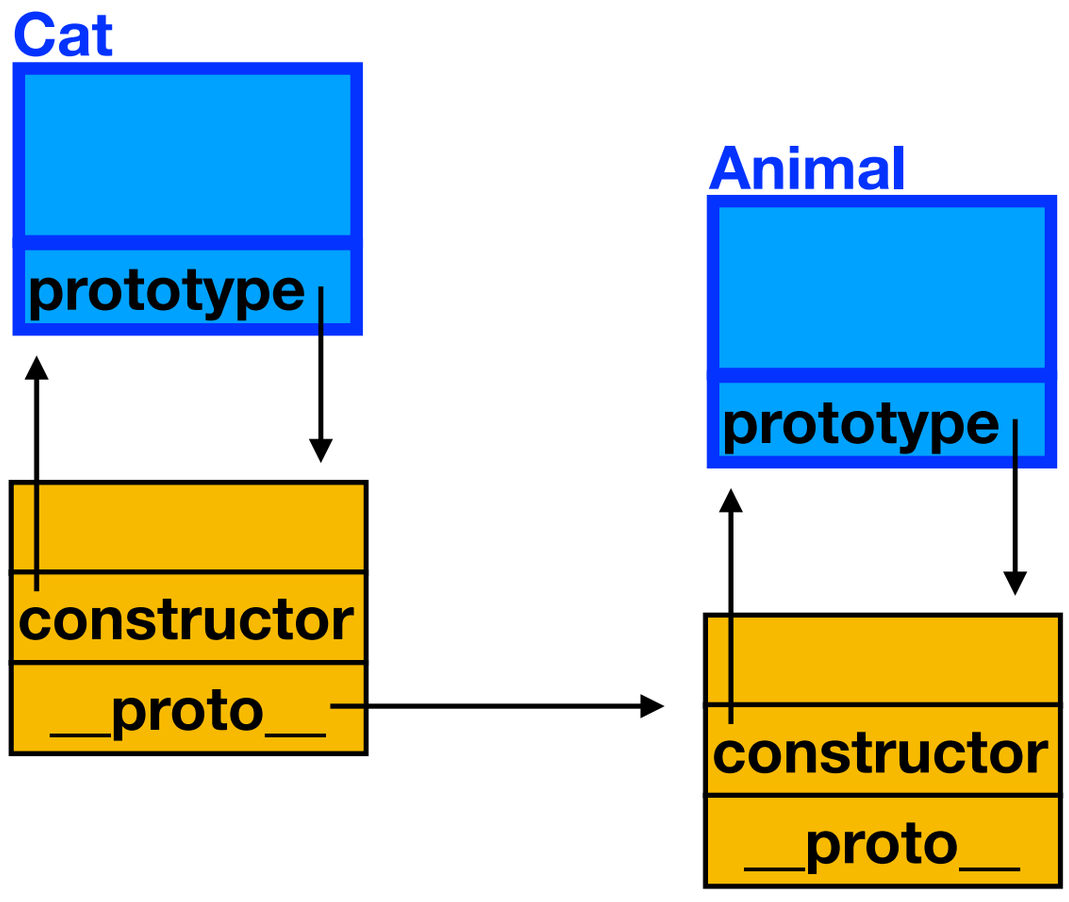
Let's say our constructor function creates Cat instances.



const garfield = new Cat();
creates a new Cat instance
whose dunder proto (__proto__) points to the Cat prototype

Constructor Functions & Prototypes

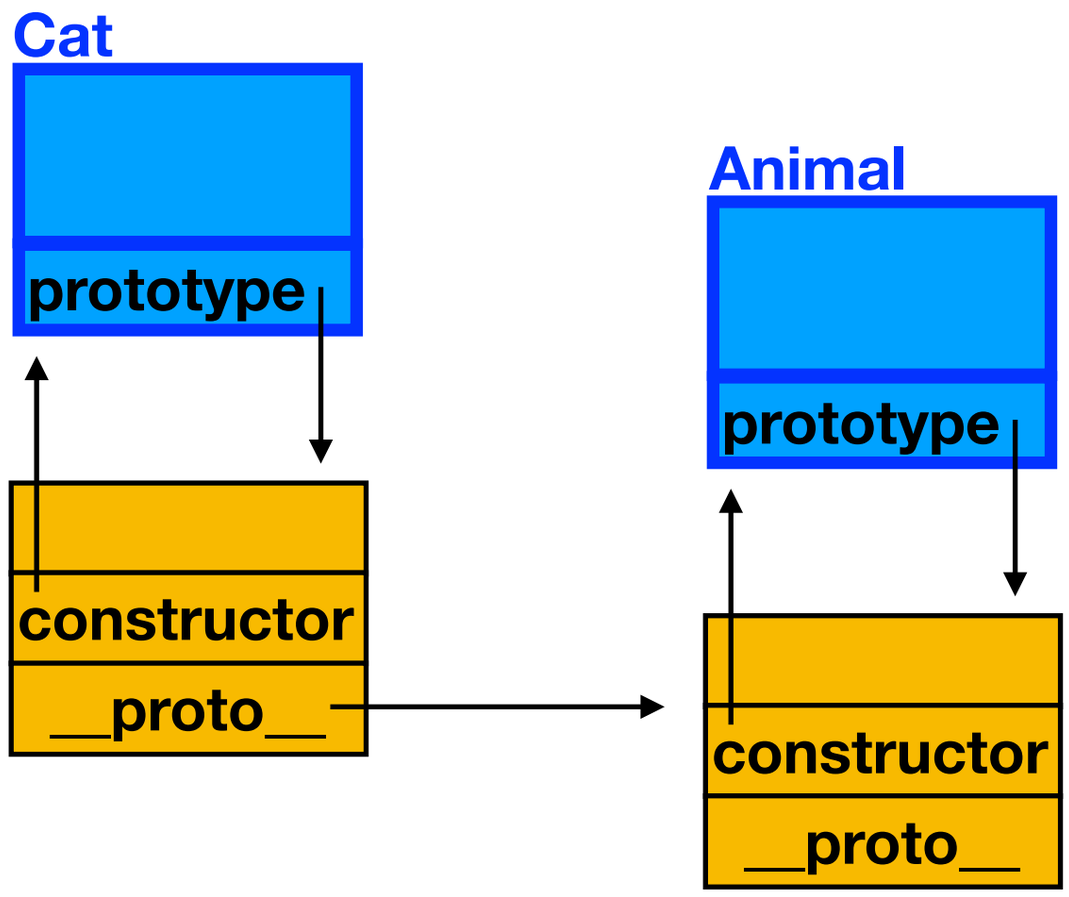
Say we want to have our Cat inherit from Animal.
That means we want the Cat prototype __proto__
to point to the prototype of Animal.



How can we do this?

Solution #1

Explicitly set Cat's __proto__ to point to Animal's prototype



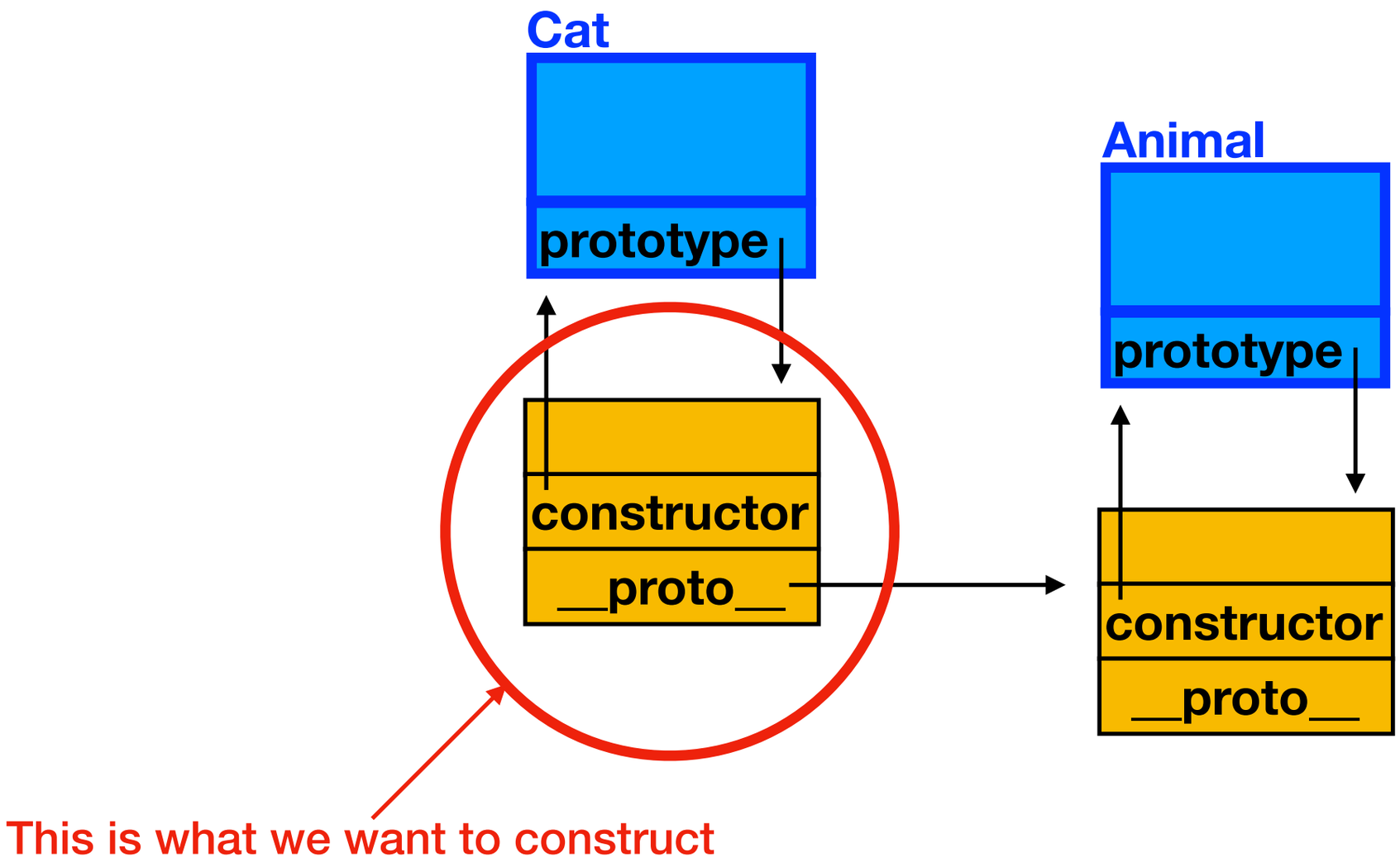
Cat.prototype.__proto__ = Animal.prototype

Problem: Slow, deprecated; DO NOT DO THIS

See MDN docs on __proto__

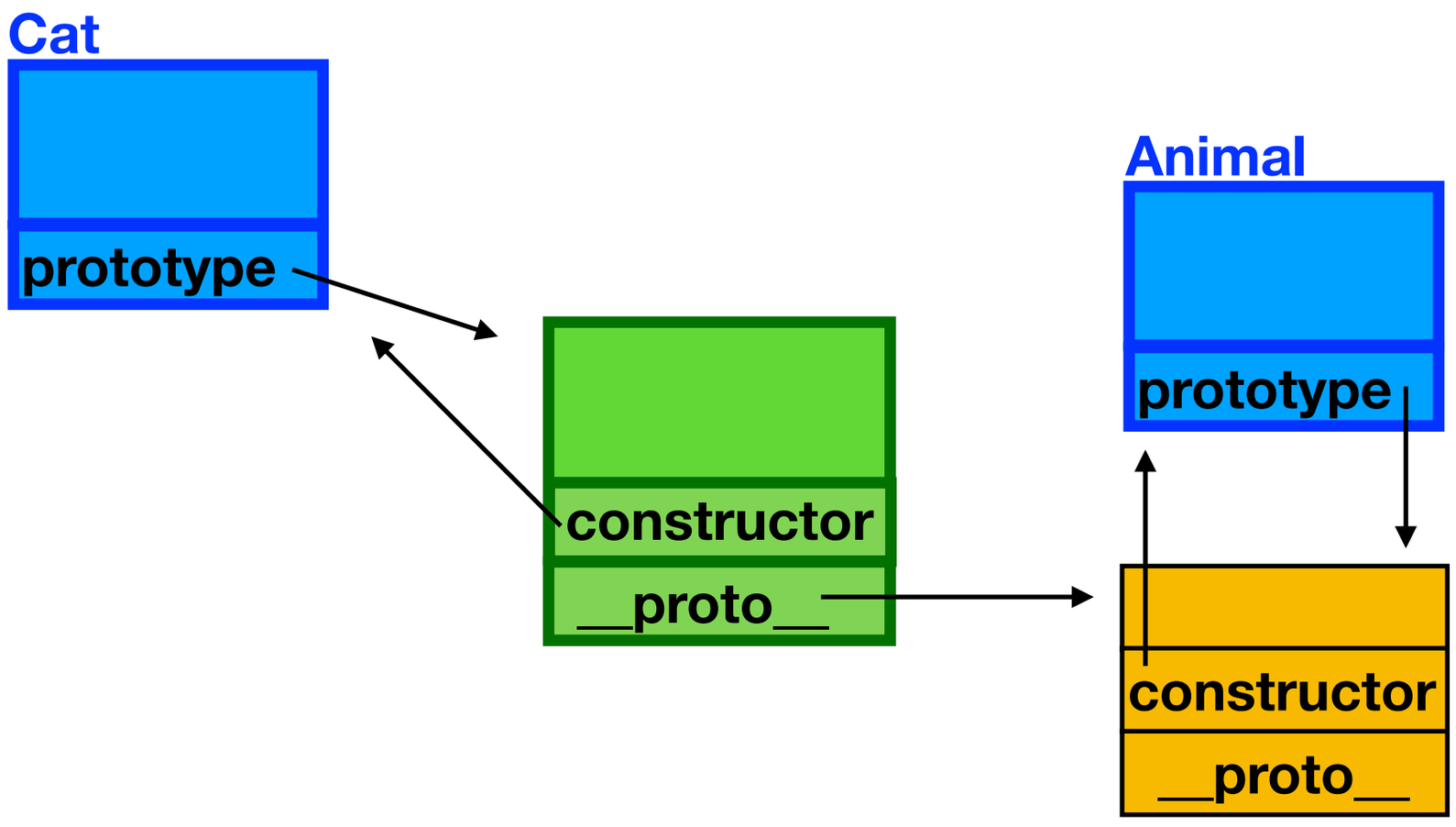
New Strategy

Construct an object whose __proto__ points to the prototype of Animal and assign it as Cat's prototype.



Solution #2

Assign Cat's prototype to a new instance of Animal

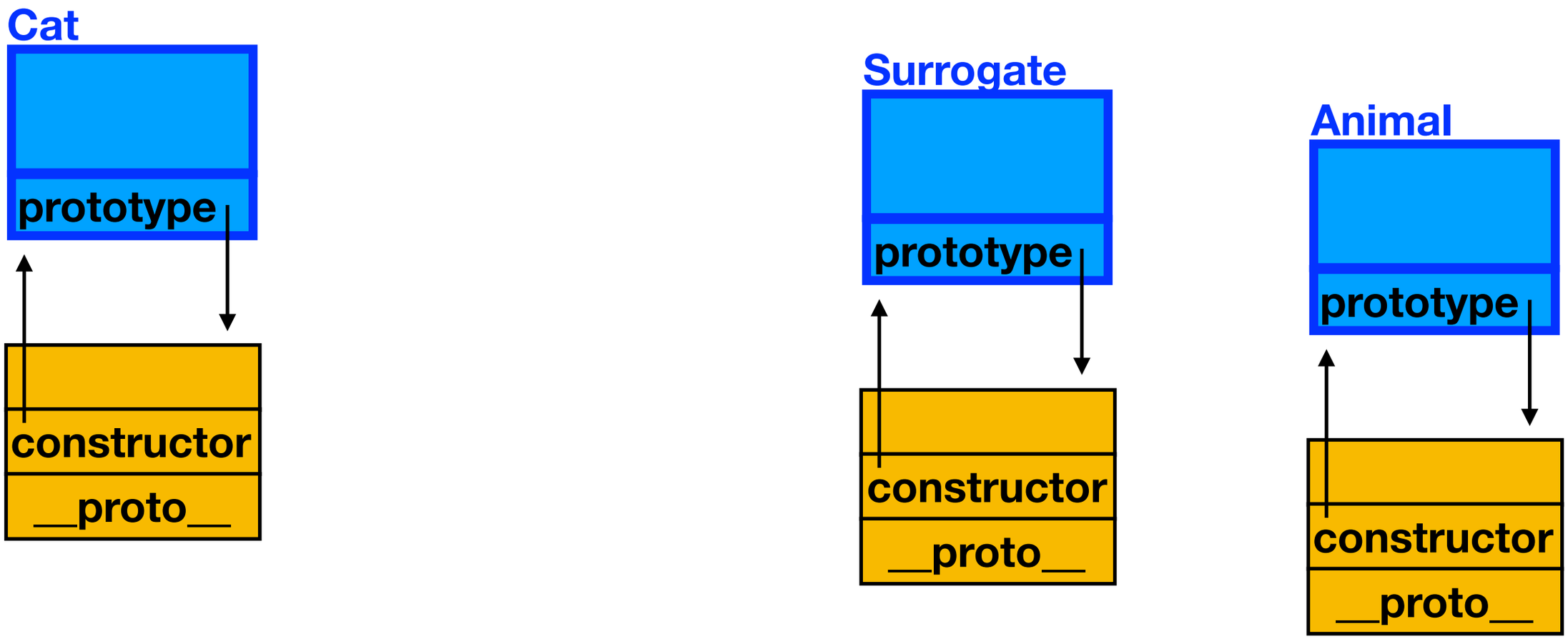



```
Cat.prototype = new Animal ();
Cat.prototype.constructor = Cat;
```

Problem: Animal() might be expensive to run

Solution #3

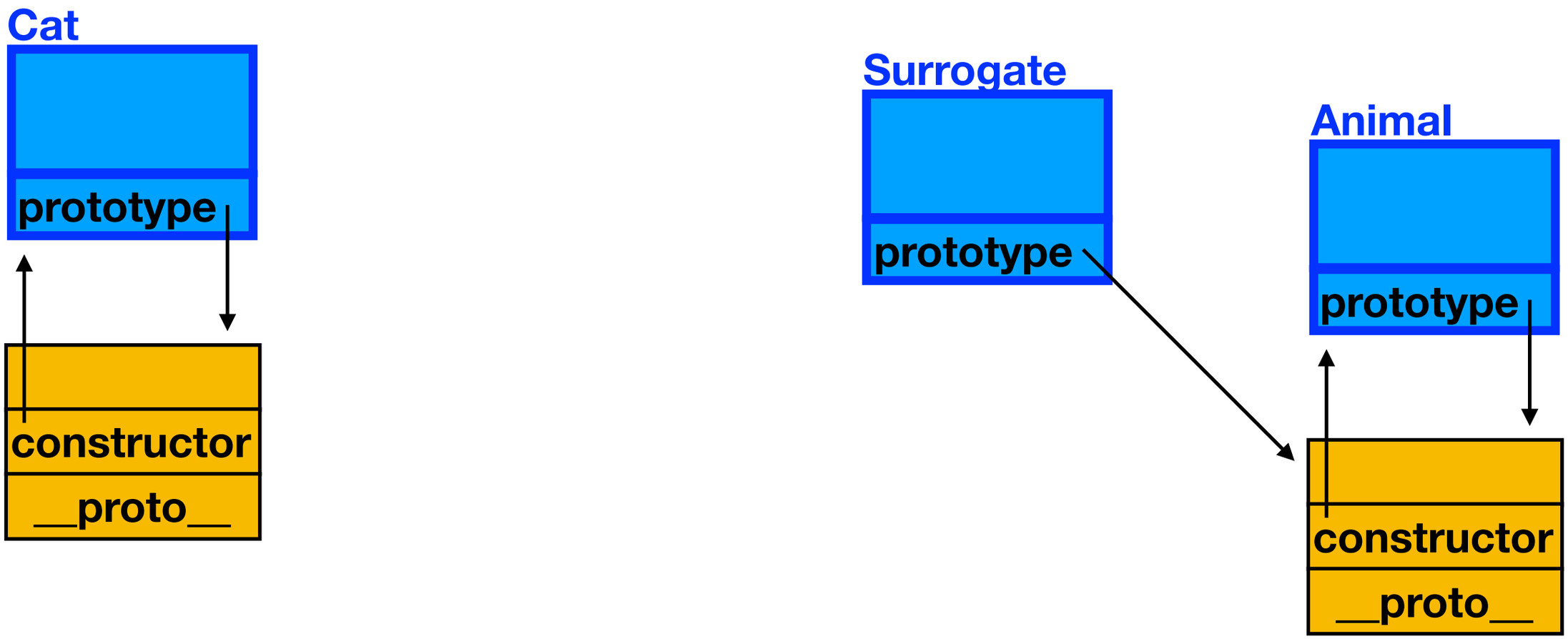
Use Surrogate




```
function Surrogate() {};
```

Solution #3

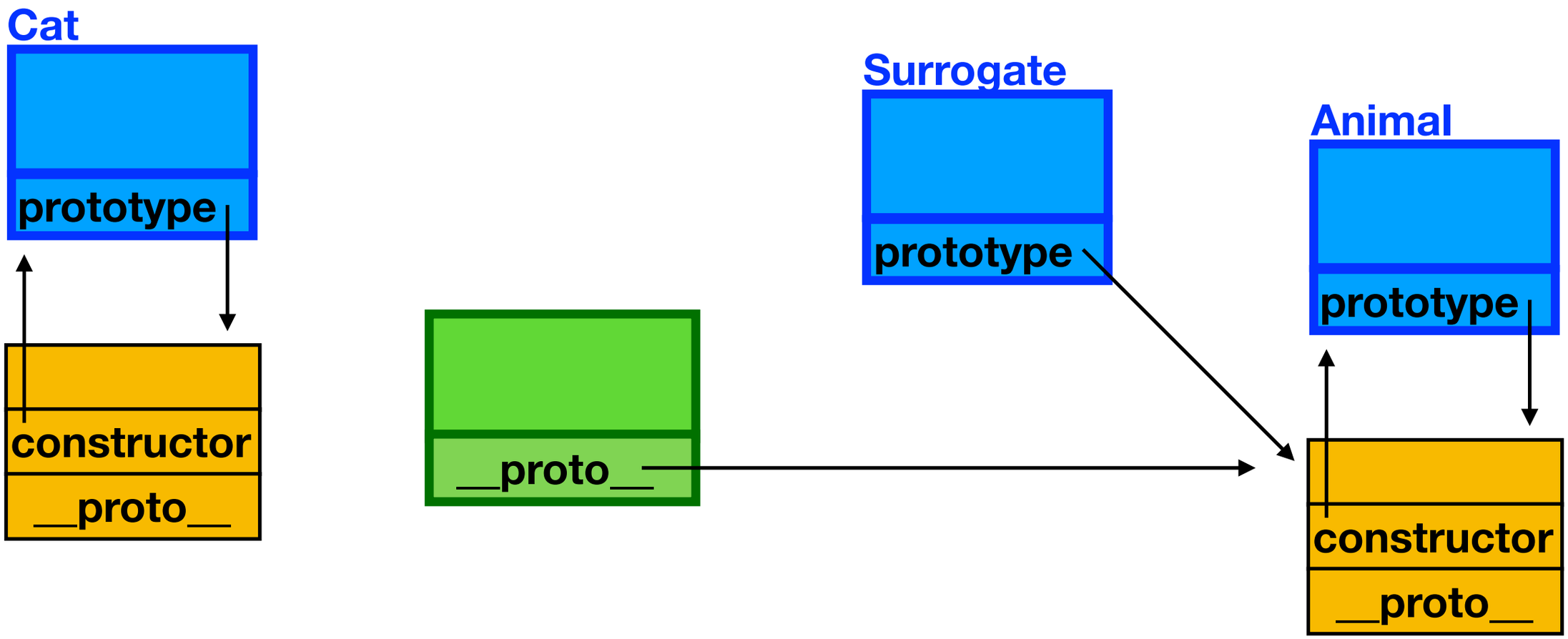
Use Surrogate




```
function Surrogate() {};
Surrogate.prototype = Animal.prototype;
```

Solution #3

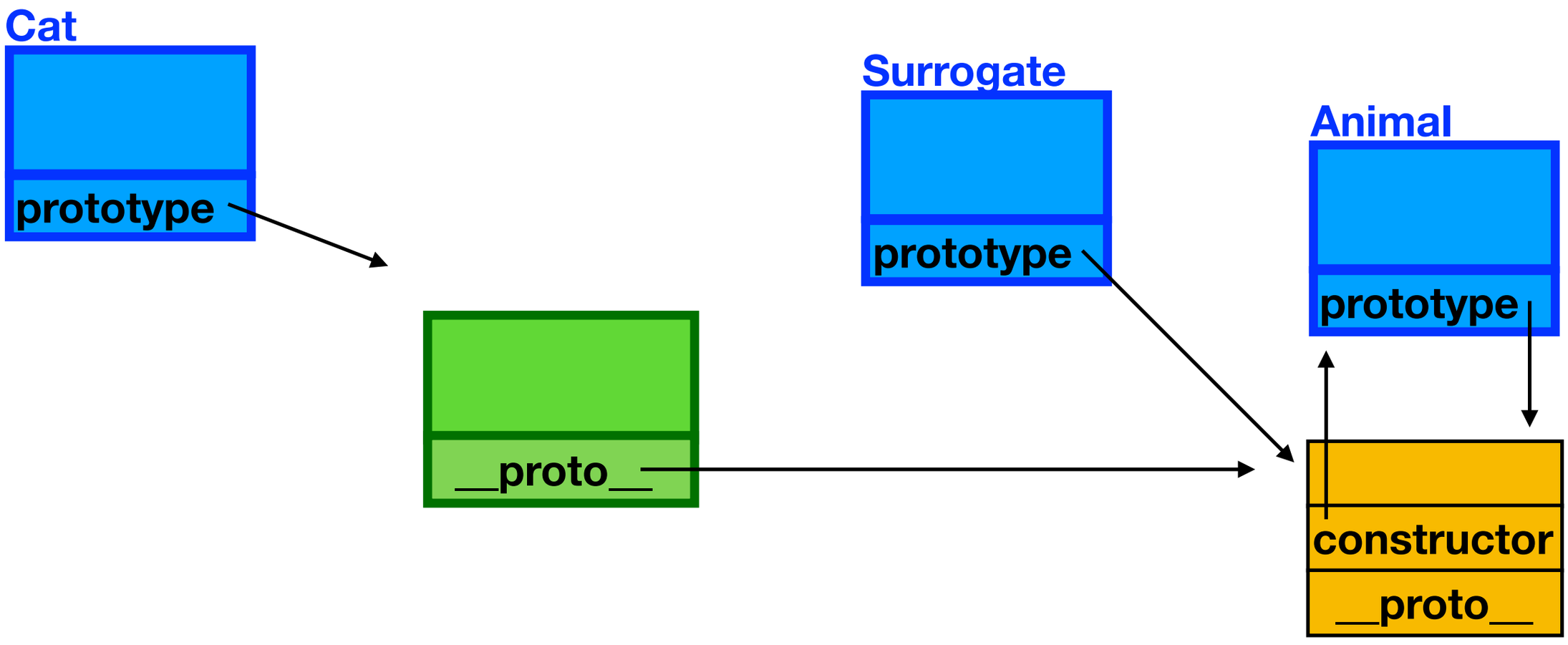
Use Surrogate




```
function Surrogate() {};
Surrogate.prototype = Animal.prototype;
                new Surrogate ();
```

Solution #3

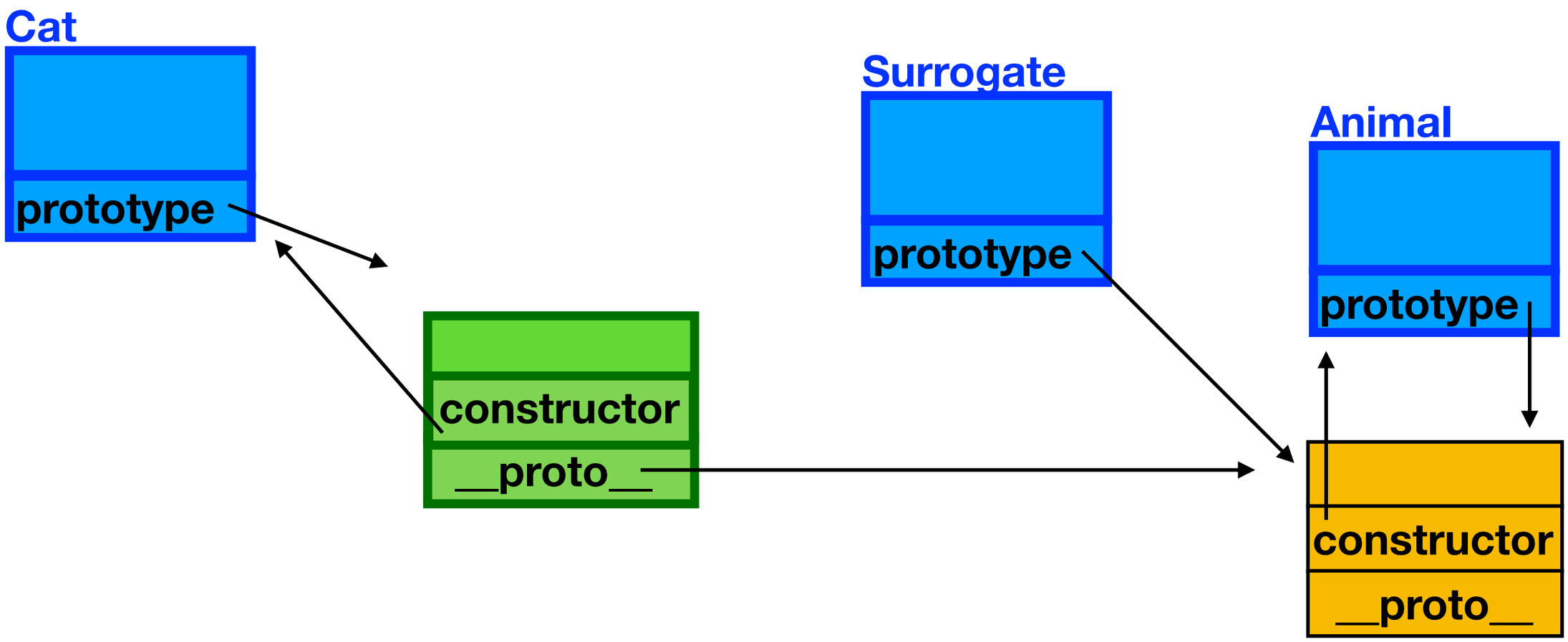
Use Surrogate




```
function Surrogate() {};
Surrogate.prototype = Animal.prototype;
Cat.prototype = new Surrogate ();
```

Solution #3

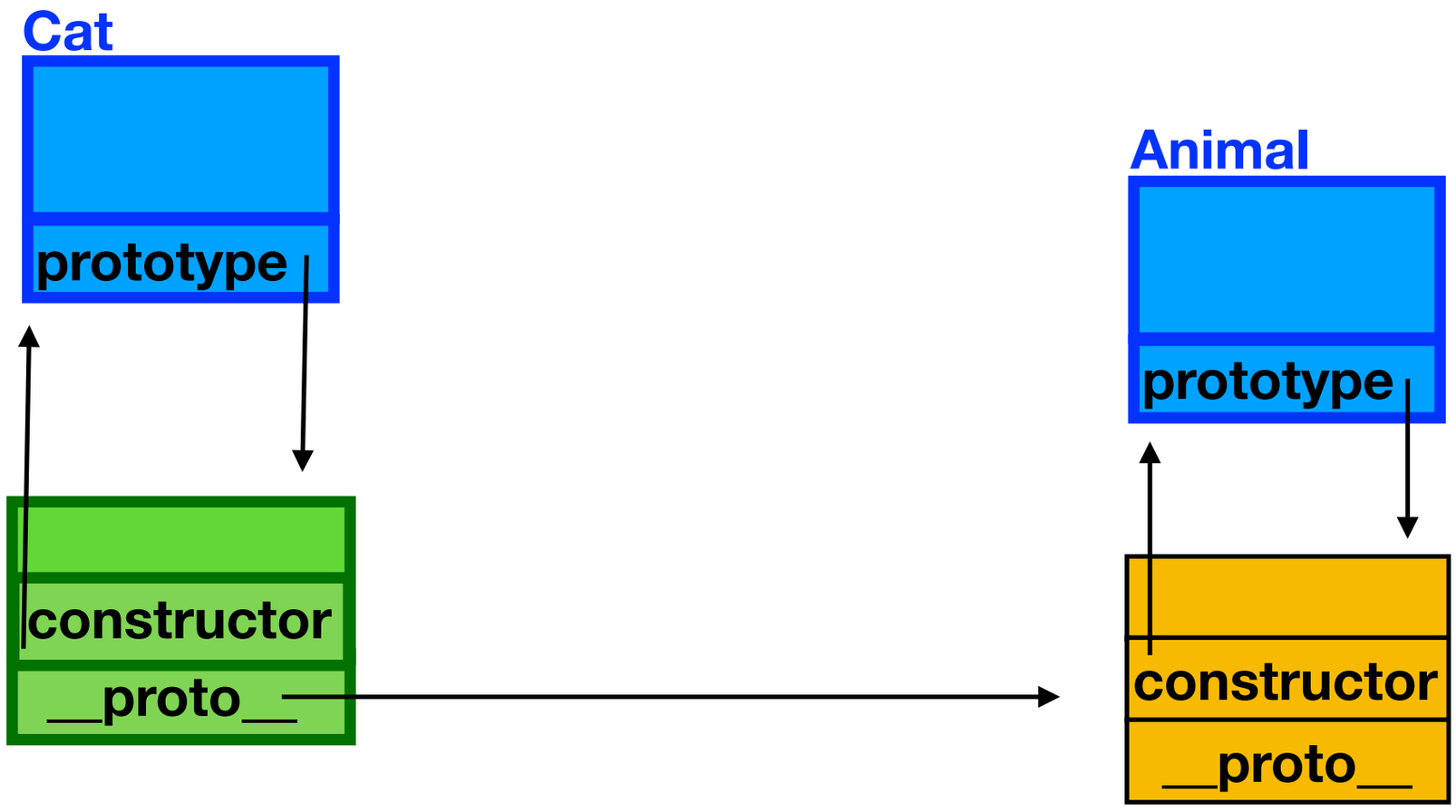
Use Surrogate




```
function Surrogate() {};
Surrogate.prototype = Animal.prototype;
Cat.prototype = new Surrogate ();
Cat.prototype.constructor = Cat;
```

Solution #3

Use Surrogate



SUCCESS!