Prototypal inheritance

Advanced JavaScript

1

tl;dr

- JavaScript doesn't handle traditional inheritance but it does have prototypal inheritance and that can be used to mimic it.
- Overriding can be done simply by setting the value in the object itself; you get it for free.

3

JavaScript does not support traditional inheritance

- In c#

```
public class Employee
{
  public decimal Salary { get; set; }
  public overrides string ToString() {
    return string.Format("{0}, {1}",
      this.Last, this.First);
  }
}
```

- But it does have these things called *prototypes*

4

prototype

/ˈprōdəˌtīp/

noun

1. the original or model on which something is based or formed.
2. someone or something that serves to illustrate the typical qualities of a class; model; exemplar. ie: She is the prototype of a student activist.
3. something analogous to another thing of a later period. ie: a Renaissance prototype of our modern public housing.
4. Biology. an archetype; a primitive form regarded as the basis of a group.

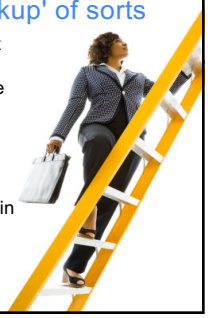
From Random House Dictionary via Dictionary.com

5

The prototype serves as a 'backup' of sorts

If we try to access a property which doesn't exist on the object itself,

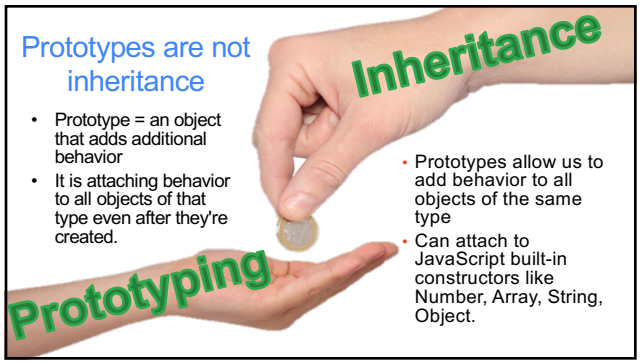
1. JavaScript looks in its constructor's prototype for that property.
2. If it is present, that property is reported
3. If not, the prototype's prototype is searched
4. ... and so on and so on and so on
5. ... until JavaScript reaches the top of the chain
6. Then it reports that property as 'undefined'



6

Prototypes are not inheritance

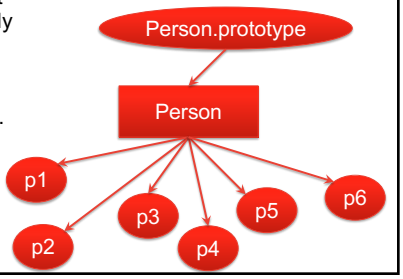
- Prototype = an object that adds additional behavior
- It is attaching behavior to all objects of that type even after they're created.



- Prototypes allow us to add behavior to all objects of the same type
- Can attach to JavaScript built-in constructors like Number, Array, String, Object.

7

- The object's prototype is never checked.
- Regular objects don't have a prototype, only Functions do.
- The object's constructor's prototype is checked.
- Thus you can add prototypes after instantiation and it is there.



8

The prototype on instances is read-only

```
var p = new Person();
console.log(p.firstname);
console.log(p.prototype.firstname);
p.firstname = "Joe";
p.prototype.firstname = "Joe";
p.__proto__.firstname = "Joe";
```

Person does not have a firstname, but its prototype does.

Works – pulls from the prototype

Does not work b/c objects don't have prototypes; Functions do

Works. Adds a firstname to p

Does not work. Can't set property 'firstname' of 'undefined'

Works, but really bad idea. Sets firstname for all instances. Breaks encapsulation

9

So you can see why this isn't exactly like traditional inheritance

10

ES2015 classes allow prototypal inheritance

- extends - Assigns a prototype
- super - Refers to the prototype

- Don't be tricked!
- These don't add any new capabilities. They're only more direct ways of working with prototypes.

11

extends registers a prototype

```
class Employee extends Person {
  constructor(first, last, salary) {
    super(first, last);
    this._salary = salary;
  }
}
const e = new Employee("Chris","Lee",75000);
console.log(e.doStuff()); // Chris Lee doing stuff
```

12

Overriding

13

Overriding is easy

Set a value (property) or a function (method) in either the constructor or on the object itself after instantiation

14

Overridding is easy ...

```
function SuperHero(){ }
SuperHero.prototype.fly = function() {
  // Most heroes can fly
};
function HumanSuperHero() { ... }
HumanSuperHero.prototype = new SuperHero();
var batman = new HumanSuperHero();
batman.fly();
// Human superheroes can't fly! Let's override
HumanSuperHero.prototype.fly = function () {
  console.warn('This hero cannot fly');
};
batman.fly();
```

Most SuperHeroes can fly.

A HumanSuperHero is a SuperHero

Batman can fly?!? That's not right.

Better. Batman can't fly now.

15

tl;dr

- JavaScript doesn't handle traditional inheritance but it does have prototypal inheritance and that can be used to mimic it.
- Overriding can be done simply by setting the value in the object itself; you get it for free.

17