# EXAMPLE 49: OVERRIDING PROPERTIES USING PROTOTYPES - TWO METHODS

# EXAMPLE 49: OVERRIDING PROPERTIES USING PROTOTYPES - TWO METHODS

THERE ARE 2 WAYS TO OVERRIDE AN OBJECT PROPERTY

- OVERRIDE FOR A SPECIFIC OBJECT ONLY

- OVERRIDE FOR ALL OBJECTS BUILT WITH A PARTICULAR CONSTRUCTOR

# EXAMPLE 49: OVERRIDING PROPERTIES USING PROTOTYPES - TWO

## THERE ARE 2 WAYS TO OVERRIDE AN OBJECT PROPERTY

- OVERRIDE FOR A SPECIFIC OBJECT ONLY
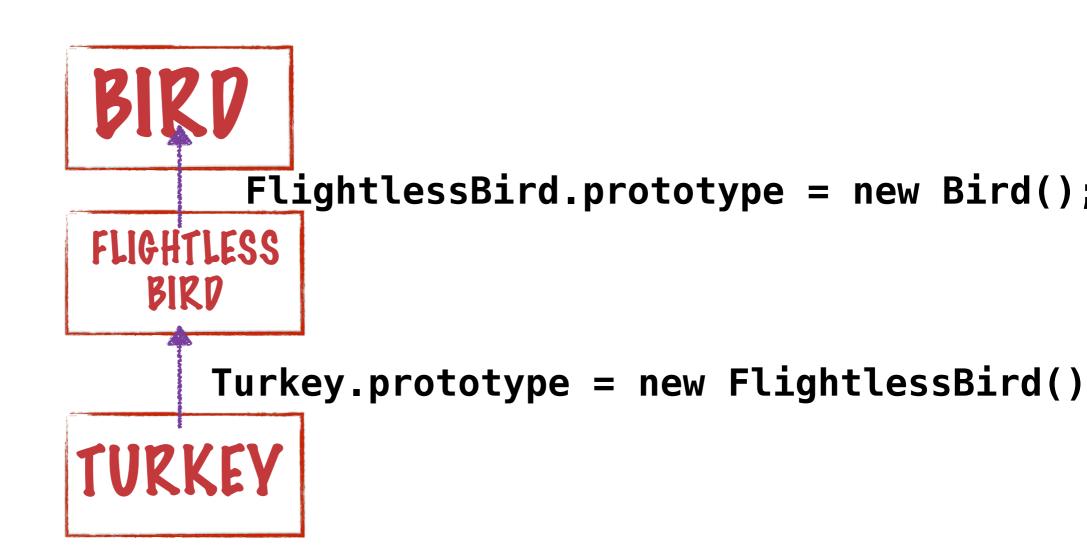- OVERRIDE FOR ALL OBJECTS BUILT WITH A PARTICULAR CONSTRUCTOR

# AN INHERITANCE HIERARCHY USING CHAINED PROTOTYPES

3 OBJECT
CONSTRUCTO

**BIRD**

**FLIGHTLESS BIRD**

**TURKEY**

`FlightlessBird.prototype = new Bird();`

`Turkey.prototype = new FlightlessBird()`

BIRD

FLIGHTLESS
BIRD

TURKEY

FlightlessBird.prototype = new Bird()

Turkey.prototype = new FlightlessBird()

```
function Bird() {
  this.sayHello = function() {
    console.log("I am a bird, of type = " +
this.name);
  }
  this.fly = function() {
    console.log("I am flying!!");
  }
}
```

```
function FlightlessBird() {
 this.fly = function() {
    console.log("I can't fly, even though
  }
}
```

```
function Turkey(){
  this.name = "Turkey";
}
```

BIRD

FLIGHTLESS
BIRD

```
function Bird() {
  this.sayHello = function() {
    console.log("I am a bird, of type = " +
this.name);
  }

  this.fly = function() {
    console.log("I am flying!!");
  }
}

FlightlessBird.prototype = new Bird();
function FlightlessBird() {

  this.fly = function() {
    console.log("I can't fly, even though
  }

}

Turkey.prototype = new FlightlessBird()
function Turkey() {
  this.name
```

THE PROPERTY `fly` APPEARS BOTH IN THE `Bird` OBJECT AND IN THE `FlightlessBird` OBECT - SO IF THIS IS CALLED ON A `Turkey` OR, THE `FlightlessBird` VERSION WILL BE CALLED.

# NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON A SPECIFIC `Turkey`

```
this.fly = function() {
  console.log("I am flying!!");
}
```

```
this.fly = function() {
  console.log("I can't fly, even though
}
```

```
function() {
  console.log("I am not quite flightless, I
can fly a few meters in a pinch");
};
```

**TURKEY**

NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON A SPECIFIC `Turkey`

NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON A SPECIFIC Turkey

# NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON A SPECIFIC `Turkey`

```javascript
var turkey1 =new Turkey();
turkey1.fly = function() {
  console.log("I am not quite flightless, I can fly a fe
meters in a pinch");
};
  turkey1.fly();
```

# NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON A SPECIFIC `Turkey`

```
var turkey1 =new Turkey();
turkey1.fly = function() {
  console.log("I am not quite flightless, I can fly a few
meters in a pinch");
};
turkey1.fly();
```

## SET THE `fly` PROPERTY ON A SPECIFIC `Turkey`

```
Instantiate an object and call fly – the FlightlessBird version will be called
I can't fly, even though I'm a bird :-(
Now, override the fly method for this specific object, and call the fly property – the specific verison will be called
I am not quite flightless, I can fly a few meters in a pinch
```

# NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON A SPECIFIC `Turkey`

```javascript
var turkey1 =new Turkey();
turkey1.fly = function() {
  console.log("I am not quite flightless, I can fly a fe
meters in a pinch");
};
  turkey1.fly();
```

## BUT IF YOU CREATE ANY OTHER TURKEY OBJECTS, IT IS STILL THE `FlightlessBird` VERSION THAT WILL BE CALLED

BUT IF YOU CREATE ANY OTHER TURKEY OBJECTS, IT IS STILL THE `FlightlessBird` VERSION THAT WILL BE CALLED

```
var turkey2 = new Turkey()
turkey2.fly();
```

```
But if you call fly on any other objects, its still the FlightlessBird version that will be called
I can't fly, even though I'm a bird :-(
```

NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON `Turkey` OBJECTS

# NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON `Turkey` OBJECTS

```
Turkey.prototype.fly = function() {
    console.log("I am not quite flightless, I can fly a few
meters in a pinch");
};
```

## SET THE `fly` PROPERTY ON THE `prototype` OF THE OBJECT CONSTRUCTOR FOR `Turkey`

# NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON `Turkey` OBJECTS

```
Turkey.prototype.fly = function() {
    console.log("I am not quite flightless, I can fly a few
meters in a pinch");
  };
```

## SET THE `fly` PROPERTY ON THE `prototype` OF THE OBJECT CONSTRUCTOR FOR `Turkey`

# NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON `Turkey` OBJECTS

```
Turkey.prototype.fly = function() {
    console.log("I am not quite flightless, I can fly a few
meters in a pinch");
};
```

## SET THE `fly` PROPERTY ON THE `prototype` OF THE OBJECT CONSTRUCTOR

# REMEMBER THAT `prototype` IS A PROPERTY OF ALL OBJECT CONSTRUCTORS

# NOW, SAY WE WOULD LIKE TO OVERRIDE THE PROPERTY `fly` ON `Turkey` OBJECTS

```
Turkey.prototype.fly = function() {
    console.log("I am not quite flightless, I can fly a few
meters in a pinch");
};
```

REMEMBER THAT `prototype` IS A PROPERTY

SET THE `fly` PROPERTY ON THE `prototype` OF THE OBJECT CONSTRUCTOR

# NOW, ALL `Turkey` OBJECTS CREATED AFTER THIS WILL USE THIS SPECIFIC VERSION!

# NOW, ALL `Turkey` OBJECTS CREATED AFTER THIS WILL USE THIS SPECIFIC VERSION!

```javascript
Turkey.prototype.fly = function() {
    console.log("I am not quite flightless, I can fly a few meters in a pinch");
};

var turkey3 = new Turkey();
turkey3.fly();
```

# NOW, ALL `Turkey` OBJECTS CREATED AFTER THIS WILL USE THIS SPECIFIC VERSION!

```
Turkey.prototype.fly = function() {
    console.log("I am not quite flightless, I can fly a few
meters in a pinch");
};


var turkey3 = new Turkey();
turkey3.fly();
```

I am not quite flightless, I can fly a few meters in a pinch

# SO - REMEMBER!

## THERE ARE 2 WAYS TO OVERRIDE AN OBJECT PROPERTY

- OVERRIDE FOR A **SPECIFIC OBJECT** ONLY

- OVERRIDE FOR **ALL OBJECTS** BUILT WITH A PARTICULAR CONSTRUCTOR