



# Advanced Prototype

**Stuart Halloway**  
**[stu@thinkrelevance.com](mailto:stu@thinkrelevance.com)**

# Prototype, Beyond Ajax

- **functional programming**
- **object model**
- **DOM builder**
- **selectors**
- **templates**

# Functional Programming

# Function#methodize

```
function shout(something) {  
    return something.toUpperCase();  
}  
String.prototype.shout = shout.methodize();  
$("result").update("hello world".shout());
```

# Function#wrap

```
Array.prototype.inspect = Array.prototype.inspect.wrap(  
  function(proceed, options) {  
    var max = options && options.max;  
    if (max) {  
      return this.slice(0,max).inspect() + "..."  
    } else {  
      return proceed();  
    }  
  }  
)
```

# Function#curry

```
var result_updater = Element.update.curry("result");
new Range(1..10).each(function(n) {
  result_updater.curry("Count is " + n).delay(n);
})
```

# Function#delay

```
var result_updater = Element.update.curry("result");  
new Range(1..10).each(function(n) {  
  result_updater.curry("Count is " + n).delay(n);  
})
```

# Object Model

# Class#create

```
Mammal = Class.create({
  furry: true,
  initialize: Prototype.emptyFunction
});
Cat = Class.create(Mammal, {
  initialize: function(lives) {
    this.lives = lives;
  }
});
Dog = Class.create(Mammal, {barks: true});
m = new Mammal();
c = new Cat(9);
```

# String#interpolate

```
var ins = "<p>Mammal is furry? #{furry}</p>".interpolate(m);  
$(result).insert(ins);
```

```
ins = new Element("p", {style: "color: green;"});  
ins.update("Cat is furry? #{furry}".interpolate(c));  
$(result).insert(ins);
```

# Element#insert

```
var ins = "<p>Mammal is furry? #{furry}</p>".interpolate(m);  
$(result).insert(ins);
```

```
ins = new Element("p", {style: "color: green;"});  
ins.update("Cat is furry? #{furry}".interpolate(c));  
$(result).insert(ins);
```

# DOM Builder

```
var ins = "<p>Mammal is furry? #{furry}</p>".interpolate(m);  
$(result).insert(ins);
```

```
ins = new Element("p", {style: "color: green;"});  
ins.update("Cat is furry? #{furry}".interpolate(c));  
$(result).insert(ins);
```

# OO in Prototype

- **Classes are ~~language~~ library features**
- **Inheritance is a ~~language~~ library feature**
- **Classes and inheritance are flexible ideas**

# Selectors

# Selector Demo

```
<p>Selector</p>
<p><input id="selector" value=""></input></p>
<p>Script</p>
<p><textarea id="script" value="" cols="50" rows="10"></
textarea></p>
<button id="go">Go</button>
```

```
$("#go").observe("click", function() {
  var func = eval($("#script").value);
  $$($("#selector").value).each(func);
})
```

# Augmenting Built-in Classes

# Augmenting String

```
Object.extend(String.prototype, {
  stripTags: function() {
    return this.replace(/<\/?[^\>]+>/gi, '');
  },

  stripScripts: function() {
    return this.replace(new RegExp(Prototype.ScriptFragment,
                                'img'), '');
  },
  // ... snip ...
}
```

**Prototype 1.6 is**

**Ajax**

**+**

**General JavaScript**

# Resources and Samples

**Prototype:**  
[prototype.conio.net](http://prototype.conio.net)

**About Relevance:**  
[thinkrelevance.com](http://thinkrelevance.com)

**Sample Code:**  
[thinkrelevance.com](http://thinkrelevance.com)

**Rails Book Samples:**  
[www.pragmaticprogrammer.com/titles/fr\\_rails4java/](http://www.pragmaticprogrammer.com/titles/fr_rails4java/)

