Object Oriented jQuery with MooTools

by Ryan Florence

$(document).ready(function(){

```
$('#my-form').submit(function(event){
  event.preventDefault();

  $.post($(this).attr('action'), $(this).serialize(),
   function(data){
     $('#container').html(data);
     document.forms.myForm.reset();
   });

});
```

});

$(document).ready(function(){

```
$('#spinner').hide();

$('#my-form').submit(function(event){
  event.preventDefault();
  $(this).css({'opacity': 0.5});
  $('#spinner').fadeIn().css({'background-position': 0 + 'px ' + 0 + 'px'});
  var loopCount = 0;
  var frames = 10;
  var frameWidth = 64;
  var computeX = function(){
    loopCount++;
    loopCount = (loopCount == (frames)) ? 0 : loopCount;
    return -loopCount * frameWidth;
  };
  var step = function(){
    $('#spinner').css({'background-position': computeX() + 'px ' + '0px' });
  };
  var interval = setInterval(step, 100);
  $('input').blur();
  $.post($(this).attr('action'), $(this).serialize(), function(data){
    clearInterval(interval);
    $('#container').html(data);
    $('#my-form').css({'opacity': 1});
    document.forms.myForm.reset();
    $('#spinner').fadeOut();
  });
});
```

});

$(document).ready(function(){

```
$('#spinner').hide();

$('#my-form').submit(function(event){
  event.preventDefault();
  $(this).css({'opacity': 0.5});
  $('#spinner').fadeIn().css({'background-position': 0 + 'px ' + 0 + 'px'});
  var loopCount =
  var frames = 10;
  var frameWidth = 64;
  var computeX = function(){
    loopCount++;
    loopCount = (loopCount ==
    return -loopCount *
  };
  var step = function(){
    $('#spinner').css(
  };
  var interval = setInterval(step, 100);
  $('input').blur();
  $.post($(this).attr('action'), $(this).serialize(), function(data){
    clearInterval(interval);
    $('#container').html(data);
    $('#my-form').css({'opacity': 1});
    document.forms.myForm.reset();
    $('#spinner').fadeOut();
  });
});
```

- Super fast
- Easy to ‘test’ because the app is the test

});

$(document).ready(function(){

```
var user;
var data;
var zoom = 'out';
var notesToggle = true;
var gridStick = false;
var view = 'both';
var viewTile = false;
var showNotes = false;
var showFavorites = true;
var showRest = true;
var grid;
var wrap;
var grip;
var gridWidth;
var mySlider;
var scroll;
function zoomInOut(){
  var size;
  var src;
  if(zoom == 'out') {
    zoom = 'in';
    size = 'large';
    src = 'out';
  } else {
    zoom = 'out';
    size = 'medium';
    src = 'in';
  }
  var req = new Request.HTML({
    method: 'get',
```

});

$(document).ready(function(){

```
// click a thumbnail
$('#span.frame').bind('click',function(event){
  scroll.toElement(this);
  $('#span.frame.current').removeClass('current');
  this.addClass('current');
  if(this.hasClass('favorite')) {
    $('#favorites img').set('src','../../../assets/remove_from_favorites.png');
    $('#favorites').set('rel','remove');
  } else {
    $('#favorites img').set('src','../../../assets/add_to_favorites.png');
    $('#favorites').set('rel','add');
  }
  var imageData = this.get('rel');
  if(view != 'tile') loadImage(imageData);
}).bind('dblclick',function(){
  var imageData = this.get('rel');
  if(view == 'tile') {
    loadImage(imageData);
    viewToggle('image');
   $('#toggles button.active').removeClass('active');
    $('#view_image').addClass('active');
  }
});


$('#zoom_btn').bind('click',zoomInOut);

});
```

});

$(document).ready(function(){

```
// click a thumbnail
$('#span.frame').bind('click',function(event){
  scroll.toElement(this);
  $('#span.frame.current').removeClass('current');
  this.addClass('current');
  if(this.hasClass('favorite')) {
    $('#favorites img').set('src','../../../assets/remove_from_favorites.png');
    $('#favorites').set('rel','remove');
  } else {
    $('#favorites img').set('src','../../../assets/add_to_favorites.png');
    $('#favorites').set('rel','add');
  }
  var imageData = this.get('rel');
  if(view != 'tile') loadImage(imageData);
}).bind('dblclick',function(){
  var imageData = this.get('rel');
  if(view == 'tile') {
    loadImage(imageData);
    viewToggle('image');
    $('#toggles button.active').removeClass('active');
    $('#view_image').addClass('active');
  }
});

$('#zoom_btn').bind('click',zoomOut);

});
```

- Way hard to maintain
- Application or even page specific
- can’t reuse in the app
- can’t reuse on your next project

});

The solution: Objects

- Objects; stateful objects
- Some people think of these as “Classes”
- This is how JavaScript itself solves these problems

The solution: Objects

- Objects; stateful objects
- Some people think of these as “Classes”
- This is how JavaScript itself solves these problems

```
$(document)
```

The solution: Objects

- Objects; stateful objects
- Some people think of these as “Classes”
- This is how JavaScript itself solves these problems

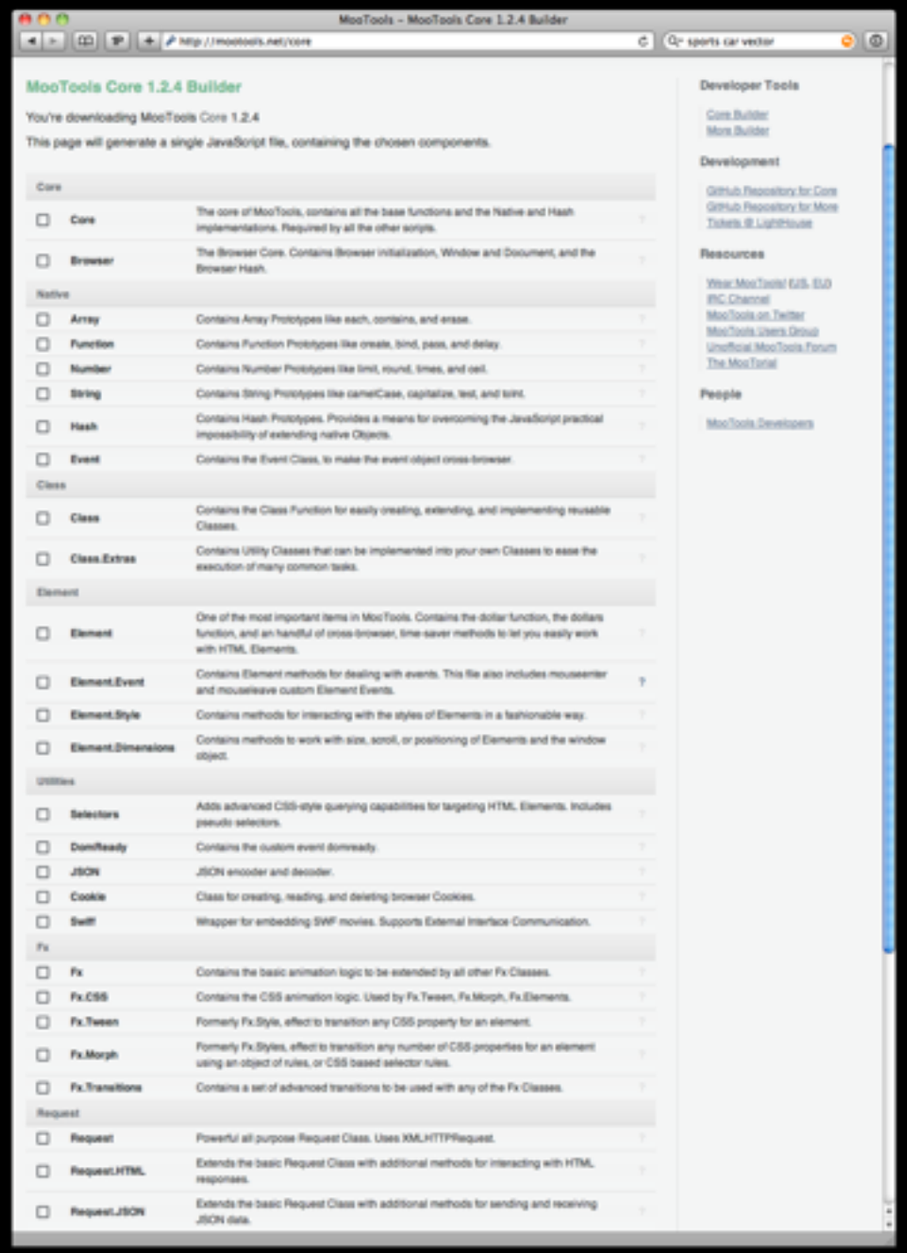
$(document)

Duplication != Awesome

```
// an object of styles
var styles = {
  color: '#333',
  height: 500
};

// and animation
var animation = {
  left: 100,
  top: 200
};
```

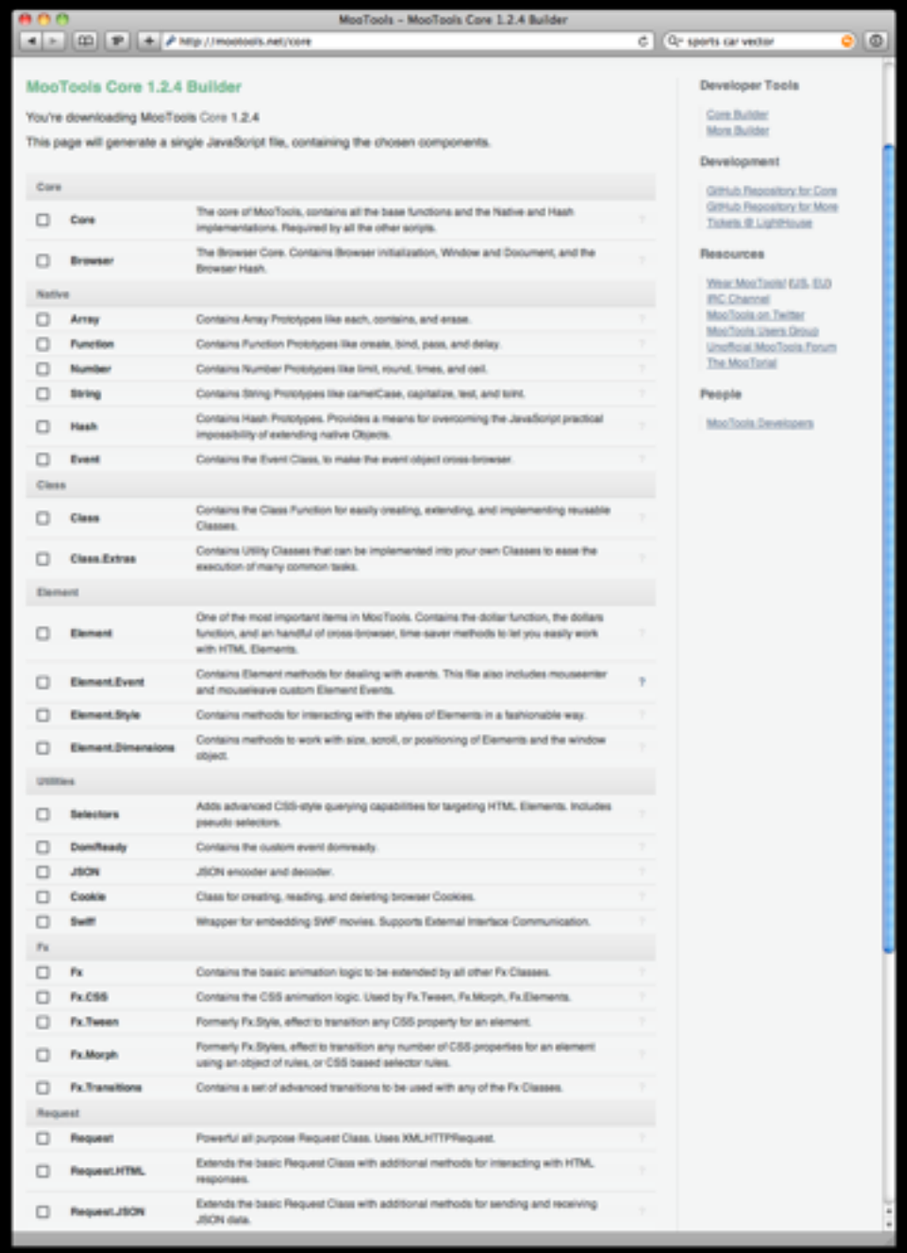
```
// jQuery
$('#elementId')
  .css(styles)
  .animate(animation)
;

// MooTools
$('elementId')
  .setStyles(styles)
  .morph(animation)
;
```

MooTools



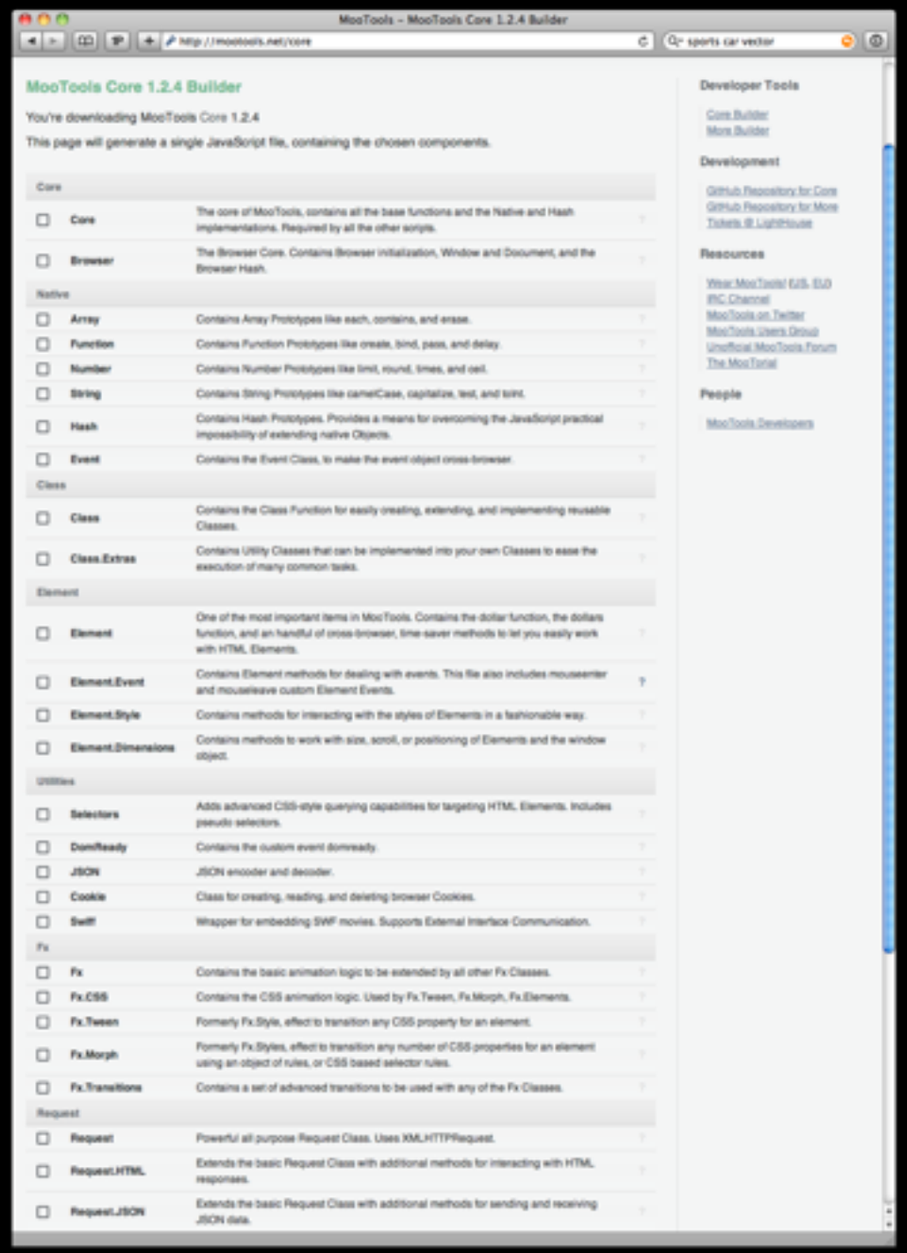
- Core
- Native
- Class
- Element
- Utilities
- Fx
- Request

MooTools



- Core
- Native
- Class
- Element
- Utilities
- Fx
- Request

MooTools



- Core
- Native
- Class
- Element
- Utilities
- Fx
- Request

MooTools

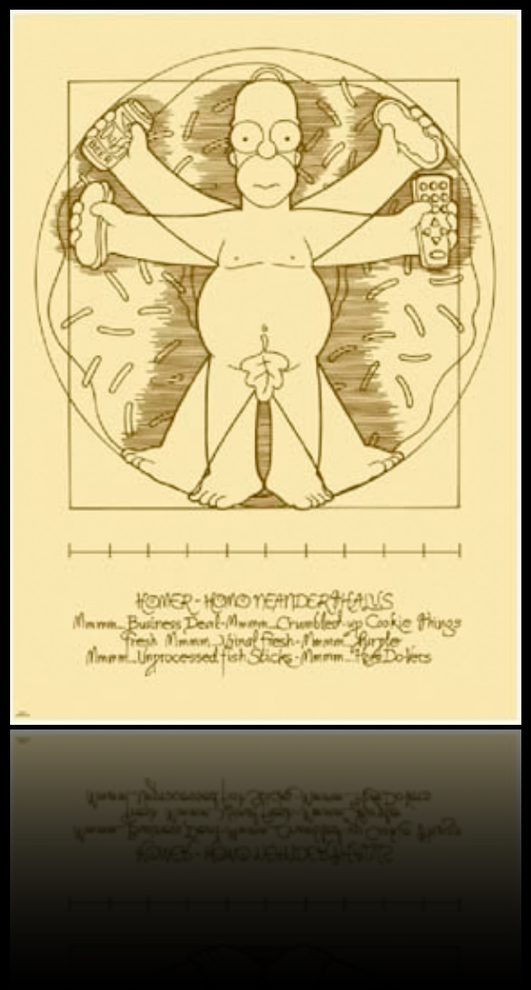
8k!

- Core
- Native
- Class
- Element
- Utilities
- Fx
- Request

MooTools Class

MooTools Class

MooTools Class




```
var Human = new Class({
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

MooTools Class

```
var Human = new Class({
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});

var bob = new Human();
//bob.energy === 1

bob.eat();
//bob.energy === 2
```

Prototypes and inheritance

Prototypes and inheritance

prototype

```
var humanBase = {
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
};
```

Prototypes and inheritance

prototype

```
var humanBase = {
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
};
```

constructor

```
var Human = new Class(humanBase);
```

Prototypes and inheritance

prototype

```
var humanBase = {
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
};
```

constructor

```
var Human = new Class(humanBase);
```

instance

```
var bob = new Human();
```

Prototypes and inheritance

```
bob.energy
```

prototype

```
var humanBase = {
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
};
```

constructor

```
var Human = new Class(humanBase);
```

instance

```
var bob = new Human();
```

Prototypes and inheritance

```
bob.energy == bob.prototype.energy
```


prototype

```
var humanBase = {
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
};
```

constructor

```
var Human = new Class(humanBase);
```

instance

```
var bob = new Human();
```

Prototypes and inheritance

```
bob.energy == bob.prototype.energy
bob.foo  == undefined
```

prototype

```
var humanBase = {
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
};
```

constructor

```
var Human = new Class(humanBase);
```

instance

```
var bob = new Human();
```

Prototypes and inheritance

```
bob.energy == bob.prototype.energy
bob.foo  == undefined
↳ bob.prototype.foo == undefined
```

prototype

```
var humanBase = {
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
};
```

constructor

```
var Human = new Class(humanBase);
```

instance

```
var bob = new Human();
```

Prototypes and inheritance

```
bob.energy == bob.prototype.energy
bob.foo  == undefined
↳ bob.prototype.foo == undefined
bob.foo = 'bar';
```

prototype

```
var humanBase = {
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
};
```

constructor

```
var Human = new Class(humanBase);
```

instance

```
var bob = new Human();
```

Prototypes and inheritance

```
bob.energy == bob.prototype.energy
bob.foo  == undefined
↳ bob.prototype.foo == undefined

bob.foo = 'bar';

bob.eat();
// assigns bob's own bob.energy
// prototype no longer checked
```

prototype

```
var humanBase = {
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
};
```

constructor

```
var Human = new Class(humanBase);
```

instance

```
var bob = new Human();
```

Extending a Class

Extending a Class

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

Extending a Class

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

```
var Ninja = new Class({
  Extends: Human,
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  },
  energy: 100,
  attack: function(target){
    this.energy = this.energy - 5;
    target.isAlive = false;
  }
});
```

Extending a Class

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

```
var Ninja = new Class({
  Extends: Human,
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  },
  energy: 100,
  attack: function(target){
    this.energy = this.energy - 5;
    target.isAlive = false;
  }
});
```

Extending a Class

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

```
var Ninja = new Class({
  Extends: Human,
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  },
  energy: 100,
  attack: function(target){
    this.energy = this.energy - 5;
    target.isAlive = false;
  }
});
```

Extending a Class

```
var Ninja = new Class({
  Extends: Human,
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  },
  energy: 100,
  attack: function(target){
    this.energy = this.energy - 5;
    target.isAlive = false;
  }
});
```

Extending a Class

```
var Ninja = new Class({
  Extends: Human,
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  },
  energy: 100,
  attack: function(target){
    this.energy = this.energy - 5;
    target.isAlive = false;
  }
});
```

```
var bob = new Human('Bob', 50);

var ryu = new Ninja(
  'good',
  'Ryu Hayabusa',
  'unkown'
);
ryu.isAlive // true
ryu.name // 'Ryu Hayabusa'

ryu.side = 'evil';
ryu.attack(bob);
```

Mixins (Implements)

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

```
var Warrior = new Class({
  energy: 100,
  kills: 0,
  attack: function(target){
    if (target.energy < this.energy) {
      target.isAlive = false;
      this.kills++;
    }
    this.energy = this.energy - 5;
  }
});
```

```
var Ninja = new Class({
  Extends: Human,
  Implements: [Warrior],
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  }
});
```

```
var Samurai = new Class({
  Extends: Human,
  Implements: [Warrior],
  side: 'good',
  energy: 1000
});
```

Mixins (Implements)

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

```
var Warrior = new Class({
  energy: 100,
  kills: 0,
  attack: function(target){
    if (target.energy < this.energy) {
      target.isAlive = false;
      this.kills++;
    }
    this.energy = this.energy - 5;
  }
});
```

```
var Ninja = new Class({
  Extends: Human,
  Implements: [Warrior],
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  }
});
```

```
var Samurai = new Class({
  Extends: Human,
  Implements: [Warrior],
  side: 'good',
  energy: 1000
});
```

Mixins (Implements)

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

```
var Warrior = new Class({
  energy: 100,
  kills: 0,
  attack: function(target){
    if (target.energy < this.energy) {
      target.isAlive = false;
      this.kills++;
    }
    this.energy = this.energy - 5;
  }
});
```

```
var Ninja = new Class({
  Extends: Human,
  Implements: [Warrior],
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  }
});
```

```
var Samurai = new Class({
  Extends: Human,
  Implements: [Warrior],
  side: 'good',
  energy: 1000
});
```

Mixins (Implements)

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});


var Warrior = new Class({
  energy: 100,
  kills: 0,
  attack: function(target){
    if (target.energy < this.energy) {
      target.isAlive = false;
      this.kills++;
    }
    this.energy = this.energy - 5;
  }
});
```

```
var Ninja = new Class({
  Extends: Human,
  Implements: [Warrior],
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  }
});


var Samurai = new Class({
  Extends: Human,
  Implements: [Warrior],
  side: 'good',
  energy: 1000
});
```

Mixins (Implements)

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

```
var Warrior = new Class({
  energy: 100,
  kills: 0,
  attack: function(target){
    if (target.energy < this.energy) {
      target.isAlive = false;
      this.kills++;
    }
    this.energy = this.energy - 5;
  }
});
```

```
var Ninja = new Class({
  Extends: Human,
  Implements: [Warrior],
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  }
});
```

```
var Samurai = new Class({
  Extends: Human,
  Implements: [Warrior],
  side: 'good',
  energy: 1000
});
```

Mixins (Implements)

```
var Human = new Class({
  initialize: function(name, age){
    this.name = name;
    this.age = age;
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
  }
});
```

```
var Warrior = new Class({
  energy: 100,
  kills: 0,
  attack: function(target){
    if (target.energy < this.energy) {
      target.isAlive = false;
      this.kills++;
    }
    this.energy = this.energy - 5;
  }
});
```

```
var Ninja = new Class({
  Extends: Human,
  Implements: [Warrior],
  initialize: function(side, name, age){
    this.side = side;
    this.parent(name, age);
  }
});
```

```
var Samurai = new Class({
  Extends: Human,
  Implements: [Warrior],
  side: 'good',
  energy: 1000
});
```

Mixins (Implements)

```
var jack = new Samurai('Jack', 29);
jack.attack(ryu);
if(jack.energy == 0) jack.eat();
jack.kills // 1
jack.isAlive // true
```

Options & Events

```
var Human = new Class({
  Implements: [Options, Events],
  options: {
    name: 'no use for a name',
    age: 28
  },
  initialize: function(options){
    this.setOptions(options);
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
    this.fireEvent('eat', this.energy);
  }
});
```

Options & Events

```
var Human = new Class({
  Implements: [Options, Events],
  options: {
    name: 'no use for a name',
    age: 28
  },
  initialize: function(options){
    this.setOptions(options);
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
    this.fireEvent('eat', this.energy);
  }
});
```

Options & Events

```
var Human = new Class({
  Implements: [Options, Events],
  options: {
    name: 'no use for a name',
    age: 28
  },
  initialize: function(options){
    this.setOptions(options);
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
    this.fireEvent('eat', this.energy);
  }
});
```

```
var bob = new Human({
  name: 'Bob Ross',
  age: 50
});
```

Options & Events

```
var Human = new Class({
  Implements: [Options, Events],
  options: {
    name: 'no use for a name',
    age: 28
  },
  initialize: function(options){
    this.setOptions(options);
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
    this.fireEvent('eat', this.energy);
  }
});
```

```
var bob = new Human({
  name: 'Bob Ross',
  age: 50
});

bob.options.name; // Bob Ross
```

Options & Events

```
var Human = new Class({
  Implements: [Options, Events],
  options: {
    name: 'no use for a name',
    age: 28
  },
  initialize: function(options){
    this.setOptions(options);
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
    this.fireEvent('eat', this.energy);
  }
});
```

```
var bob = new Human({
  name: 'Bob Ross',
  age: 50
});

bob.options.name; // Bob Ross
```

Options & Events

```
var Human = new Class({
  Implements: [Options, Events],
  options: {
    name: 'no use for a name',
    age: 28
  },
  initialize: function(options){
    this.setOptions(options);
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
    this.fireEvent('eat', this.energy);
  }
});
```

```
var bob = new Human({
  name: 'Bob Ross',
  age: 50
});

bob.options.name; // Bob Ross

bob.addEvent('eat',function(energy){
  // do something else
});
```

Options & Events

```
var Human = new Class({
  Implements: [Options, Events],
  options: {
    name: 'no use for a name',
    age: 28
  },
  initialize: function(options){
    this.setOptions(options);
  },
  isAlive: true,
  energy: 1,
  eat: function(){
    this.energy++;
    this.fireEvent('eat', this.energy);
  }
});
```

```
var bob = new Human({
  name: 'Bob Ross',
  age: 50
});

bob.options.name; // Bob Ross

bob.addEvent('eat',function(energy){
  // do something else
});

var joe = new Human({
  name: 'Joe Jonson',
  age: 28,
  onEat: function(energy){
    // burp ...
  }
});
```

Hey, What About


```
var jack = new Samurai('Jack', 29);
jack.attack(ryu);
if(jack.energy == 0) jack.eat();
jack.kills // 1
jack.isAlive // true
```

jQuery is designed to change the way that you write javascript

Hey, What About jQuery

```
var jack = new Samurai('Jack', 29);
jack.attack(ryu);
if(jack.energy == 0) jack.eat();
jack.kills // 1
jack.isAlive // true
```

jQuery is designed to change the way that you write javascript

```
$('#jack').samurai(); // new Samurai('#jack')
$('#jack').samurai('eat') // jack.eat()
$('#jack').samurai('isAlive'); // jack.isAlive
$('#jack').samurai('kills', 2); // jack.kills = 2
$('#jack').samurai('attack', bob).animate().css();
$('#jack').samurai('addEvent', fn);
```

How is it done?

```
var Samurai = new Class({
    Extends: Human,
    Implements: [Warrior],
    jQuery: 'samurai',
    side: 'good',
    energy: 1000
});
```

How is it done?

```
Class.Mutators.jQuery = function(name){
  var self = this;
  jQuery.fn[name] = function(arg){
    var instance = this.data(name);
    if ($type(arg) == 'string'){
      var prop = instance[arg];
      if ($type(prop) == 'function'){
        var returns = prop.apply(instance, Array.slice(arguments, 1));
        return (returns == instance) ? this : returns;
      } else if (arguments.length == 1){
        return prop;
      }
      instance[arg] = arguments[1];
    } else {
      if (instance) return instance;
      this.data(name, new self(this.selector, arg));
    }
    return this;
  };
};
```

Demo

http://ryanflorence.com/mootools-jquery/demo

Thanks!

Ryan Florence
http://ryanflorence.com
rpflorence@gmail.com

http://bit.ly/9EVcvK How to Write a MooTools Class

http://bit.ly/ca3k4i Writing Flexible Classes