# jQuery Plugin Design Patterns

“Deep into JS” Meetup

## Constructor Function

```
function Person() {}
function Person(name, surname) {
	this.name = name;
	this.surname = surname;
}
```

```
function Person(name, surname) {
    this.name = name;
    this.surname = surname;
    this.getName = function() {
        // privileged method: useful if you need private method, attributes
        return this.name;
    }
}
```

```
function Person(name, surname) {
      this.name = name;
      this.surname = surname;
}
Person.prototype.getName = function() {
        return this.name;
}
```

```
var robb = new Person(‘Robert’, ‘Casanova’);
console.log(robb.getName());
```

## Object.create()

```
var robb = {
    name: ‘Robert’,
    surname: ‘Casanova’,
    getName: function() {
        return this.name;
    }
}
```

```
var gigi = Object.create(robb);
gigi.name = ‘Giorgio’;
gigi.surname = ‘Moroder’;

console.log(gigi.getName())
```

## Plugin jQuery

## Simplest jQuery plugin ever

```
$.fn[‘pluginName’] = function() {
    this.each(function(){
        $(this).doSomething();
    });
}
```

## LightWeight Start Pattern

## Immediately Invoked Function Expression (IIFE)

```
;(function($, window,document, undefined ){
    //plugin goes here
})(jQuery, window, document)
```

## Defaults

```
var pluginName = “mioPlugin”,
        defaults = {
                defaultProperty: ‘defaultValue’
        }
```

## Constructor Function

```
function Plugin(el, options) {
    this.el  = el;
    this.$el = $(el);
    this.options = $.extend({},defaults, options);

    this._defaults = defaults;
    this._name = pluginName;

      this.init();
}
```

## Methods

```
Plugin.prototype.init = function() {
        //the initial logic goes here
}
Plugin.prototype.someMethod = function() {
    …
}
```

## Plugin Magic

```
$.fn[pluginName] = function(options) {
        return this.each(function(){
              if(!$.data(this, pluginName)) {
                  $.data(this, pluginName, new Plugin(this, options));
              }
        });
}
```

## Example

```
$(‘#elem’).pluginName({
    defaultProperty: ‘value’
});
```

## DOM-to-Object Bridge Pattern

## Object

```
var myObject = {
    init: function (options,elem) {
        …
    }
}
```

## Init Function

```
init: function (options,elem) {
    this.options = $.extend({}, this.options,options);
    this.el = elem;
    this.$el = $(elem);

    this._build();

    return this;
}
```

## Default options

```
options: {
  defaultOption: ‘defaultValue’
}
```

## Methods

```
_build: function() {
     this.$el.html(‘inizialize html here’);
},
publicMethod: function() {
     this.$el.doSomething();
}
```

## Plugin Magic

```
$.plugin = function(name,object) {
    $.fn[name] = function(options) {
        return this.each(function(){
            if(!$.data(this,name)) {
                $.data(this,name, Object.create(myObject).init(options,this)
            }
        })
    }
}
$.plugin(“pluginName”, myObject )
```

## Example

```
$(‘#elem’).pluginName({
    defaultProperty: ‘value’
});

var plugin = $(‘#elem’).data(‘pluginName’);
plugin.publicMethod();
```