

Patrones y principios de diseño



made with love by Redradix (www.redradix.com)

Principios de diseño

- SRP: Single Responsibility Principle
 - El código de una elemento ha de tener solo una razón para cambiar.
 - EL principio de diseño
 - También el complementario: cada responsabilidad ha de tener un único lugar en el código (D.R.Y.)

SRP

Es común ver cosas como esta:

```
$.ajax({ ... })
  .success(function() {
    cambioEnInterfaz();
    mostrarModal();
    if ($("#elemento").value() == "Ok") {
      /* ... */
    }
    globalSeHaGuardado = true;
})
.error(function() {
  // ...
});
```

SRP

O como esta:

```
var Widget = Class.extend({
    onClick: function() { ... },
    guardar: function() { ... },
    render: function() { ... },
    mostrarError: function() { ... }
});
```

SRP

```
var Widget = Model.extend({  
    guardar: function() { ... }  
});
```

```
var WidgetView = View.extend({  
    render: function() { ... }  
});
```

```
var WidgetController = Controller.extend({  
    onClick: function() { ... }  
});
```

```
var ErrorAlert = ModalWindow.extend({  
    mostrarError: function() { ... }  
});
```

SRP

Caso práctico: masonry.js

- <https://github.com/desandro/masonry/blob/master/jquery.masonry.js>
- en el método **_create** (línea 102)...

SRP

```
// sets up widget
_create : function( options ) {
    // [...]

    // get original styles in case we re-apply them in .destroy()
    var elemStyle = this.element[0].style;
    this.originalStyle = {
        // get height
        height: elemStyle.height || ''
    };
    // get other styles that will be overwritten
    // [...]

    s.isFluid = this.options.columnWidth && typeof this.options.columnWidth === 'function';

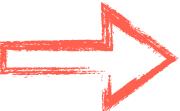
    // add masonry class first time around
    var instance = this;
    setTimeout( function() {
        instance.element.addClass('masonry');
    }, 0 );

    // bind resize method
    if ( this.options.isResizable ) {
        $(window).bind( 'smartresize.masonry', function() {
            instance.resize();
        });
    }
}
```

SRP

```
// sets up widget
_create : function( options ) {
  // [...]
  
  // get original styles in case we re-apply them in .destroy()
  var elemStyle = this.element[0].style;
  this.originalStyle = {
    // get height
    height: elemStyle.height || ''
  };
  // get other styles that will be overwritten
  // [...]

  s.isFluid = this.options.columnWidth && typeof this.options.columnWidth === 'function';

  
  // add masonry class first time around
  var instance = this;
  setTimeout( function() {
    instance.element.addClass('masonry');
  }, 0 );
  
  // bind resize method
  if ( this.options.isResizable ) {
    $(window).bind( 'smartresize.masonry', function() {
      instance.resize();
    });
  }
}
```

SRP

El resultado: caos!

- No hay un lugar claro para cada operación
- Es difícil entender qué hace cada línea
 - El “qué” está enterrado en el “cómo”
- Muy complicado de testear
- Difícil de reutilizar

SRP

Caso práctico: BrowserQuest

- [https://github.com/mozilla/BrowserQuest/blob/
master/client/js/chest.js](https://github.com/mozilla/BrowserQuest/blob/master/client/js/chest.js)

SRP

```
var Chest = Entity.extend({
    init: function(id, kind) {
        this._super(id, Types.Entities.CHEST);
    },

    getSpriteName: function() {
        return "chest";
    },

    isMoving: function() {
        return false;
    },

    open: function() {
        if(this.open_callback) {
            this.open_callback();
        }
    },

    onOpen: function(callback) {
        this.open_callback = callback;
    }
});
```

SRP

Caso práctico: BrowserQuest

- Todo el proyecto está muy bien estructurado
 - entity.js
 - character.js
 - animation.js
 - ...
- A pesar de ser muy grande, cada responsabilidad tiene su sitio

SRP

Caso práctico: Backbone.js

- <https://github.com/documentcloud/backbone/blob/master/backbone.js>
- en **Backbone.Model**, línea 179...

SRP

- Por un lado..
 - gestión de estado (**set, get**)
 - validación
 - formateo (**toJSON, escape**)
 - servidor (**fetch, save**)
- Por otro...
 - ✓ Delega los detalles a otros módulos (**Sync, Event**)
 - ✓ Bajo acoplamiento (“interfaces”)

SRP

“Una responsabilidad”...

- Subjetivo
- “Una sola razón para cambiar”...
 - “Para qué todo funcione bien”
 - Muy dependiente del nivel de abstracción
 - Y de cada módulo
- El exceso es tan malo como el defecto

Principios de diseño

- Tell, Don't Ask
 - “Dime lo que necesitas”
 - Claridad y expresividad
 - Encapsular las comprobaciones

Tell, Don't Ask

Los síntomas:

```
if (typeof(variables) === 'object' && !Array.isArray(variables)) {  
    variables = Object.keys(variables).map(function (k) {  
        var value = variables[k];  
  
        if (! (value instanceof tree.Value)) {  
            if (! (value instanceof tree.Expression)) {  
                value = new(tree.Expression)([value]);  
            }  
            value = new(tree.Value)([value]);  
        }  
        return new(tree.Rule)( '@' + k, value, false, 0);  
    });  
    frames = [new(tree.Ruleset)(null, variables)];  
}
```

Tell, Don't Ask

Los síntomas:

```
if (typeof(variables) === 'object' && !Array.isArray(variables)) {  
    variables = Object.keys(variables).map(function (k) {  
        var value = variables[k];  
  
        if (! (value instanceof tree.Value)) {  
            if (! (value instanceof tree.Expression)) {  
                value = new(tree.Expression)([value]);  
            }  
            value = new(tree.Value)([value]);  
        }  
        return new(tree.Rule)('@' + k, value, false, 0);  
    });  
    frames = [new(tree.Ruleset)(null, variables)];  
}
```

Tell, Don't Ask

Programación:

- Estructurada: adquiere info y toma decisiones
- OO: manda a los objetos hacer cosas

Tell, Don't Ask

El error:

1. Preguntar a un objeto sobre su estado
2. Tomar una decisión
3. Decirle lo que tiene que hacer

Tell, Don't Ask

El error:

1. Preguntar a un objeto sobre su estado
2. Tomar una decisión
3. Decirle lo que tiene que hacer

¡Probablemente ese código pertenece al objeto!

Tell, Don't Ask

```
if (usuario.primerLogin) {  
    usuario.mostrarMensajeBienvenida();  
} else {  
    usuario.mostrarSaludo();  
}
```

Tell, Don't Ask

```
var Usuario = Class.extend({
    saludar: function() {
        if (this.primerLogin) {
            this.mostrarMensajeBienvenida();
        } else {
            this.mostrarSaludo();
        }
    });
});
```

// y después...

 usuario.saludar();

Tell, Don't Ask

```
function comprobarTimeout(respuesta) {  
    if ((Date.now() - respuesta.start) > 10000) {  
        respuesta.notificarTimeout();  
    }  
}
```

Tell, Don't Ask

```
var Respuesta = Class.extend({
    comprobarTimeout: function() {
        if ((Date.now() - this.start) > 10000) {
            this.notificarTimeout();
        }
    }
});  
  
// y después...  
  
respuesta.comprobarTimeout();
```

Tell, Don't Ask

```
var elementos = miColeccion.getItems();
for (var i=0; i<elementos.length; i++) {
    var elemento = elementos[i];
    console.log(elemento.nombre);
}
```

Tell, Don't Ask

```
miColeccion.forEach(function(e) {  
    console.log(e.nombre);  
});
```

Tell, Don't Ask

```
var elemento = new Elemento("holá", 12);  
var lista = miColección.getItems();  
lista.addElementAt(elemento.getOrder(), elemento);
```

Tell, Don't Ask

```
var elemento = new Elemento("holá", 12);  
miColeccion.add(elemento);
```

Tell, Don't Ask

Es decir:

- Los datos y las operaciones sobre esos datos deben estar en el mismo sitio (objeto)
- Encapsular, desacoplar
- “Command/Query Separation”
 - Consulta información
 - Da una orden y deja al objeto decidir
 - Pero no las mezcles!

Tell, Don't Ask

Ventajas:

- ✓ Más robusto (menor acoplamiento)
- ✓ Menor tendencia a repetir lógica
- ✓ Mejor estructurado

Inconvenientes:

- Miles de métodos de 2 o 3 líneas
- “Ruido” en las clases

Principios de diseño

- S.O.L.I.D.
 - Single Responsibility
 - Open-Closed
 - Liskov Substitution
 - Interface Segregation
 - Development Dependency Inversion

S.O.L.I.D.

Open-Closed

- “Un elemento ha de estar abierto a la extensión pero cerrado a la modificación”
 - Abierto a la extensión: poder ser adaptado a las (futuras) necesidades de la aplicación
 - Cerrado a la modificación: que la adaptación no implique modificar su código

Open-Closed

```
var Lenguas = { Castellano: 0,  
                Ingles: 1 };  
  
var Persona = Class.extend({  
    init: function(lengua) { this.lengua = lengua; },  
    saludar: function(lengua) {  
        if (this.lengua == Lenguas.Castellano) {  
            alert("Hola!");  
        } else if (this.lengua == Lenguas.Ingles) {  
            alert("Hello!");  
        }  
    }  
});  
  
new Persona(Lenguas.Castellano).saludar();
```

Open-Closed

```
var Persona = Class.extend({
    saludar: function() {
        alert(this.saludo);
    }
});

var Angloparlante = Persona.extend({
    init: function() { this.saludo = "Hello!"; }
});

var Hispanohablante = Persona.extend({
    init: function() { this.saludo = "Hola!"; }
});

new Hispanohablante().saludar();
```

Open-Closed

Pretende:

- Promover el uso de abstracciones
- Código modular y flexible ante el cambio
- Evitar un torrente de cambios en cascada!

Open-Closed

Pretende:

- Promover el uso de abstracciones
- Código modular y flexible ante el cambio
- Evitar un torrente de cambios en cascada!

Es decir:

- Especificar y respetar interfaces

Open-Closed

```
var Canvas = Class.extend({
    render: function(figura) {
        if (figura instanceof Triangulo) {
            // ...
        } else if (figura instanceof Cuadrado) {
            // ...
        }
    }
});
```

Open-Closed

```
var Canvas = Class.extend({
    render: function(figura) {
        figura.draw(this);
    }
});
```

```
var Triangulo = Figura.extend({
    draw: function(canvas) { ... }
});
```

```
var Cuadrado = Figura.extend({
    draw: function(canvas) { ... }
});
```

Open-Closed

Caso práctico: three.js

- [https://github.com/mrdoob/three.js/blob/master/src/
renderers/CanvasRenderer.js](https://github.com/mrdoob/three.js/blob/master/src/renderers/CanvasRenderer.js)
- método **render**, línea 225

Open-Closed

```
if ( element instanceof THREE.RenderableParticle ) {  
    // ...  
}  
} else if ( element instanceof THREE.RenderableLine ) {  
    // ...  
}  
} else if ( element instanceof THREE.RenderableFace3 ) {  
    // ...  
}  
} else if ( element instanceof THREE.RenderableFace4 ) {  
    // ...  
}
```

Open-Closed

Caso práctico: jasmine.js

- <https://github.com/pivotal/jasmine/blob/master/src/core/Reporter.js>

Open-Closed

```
/** No-op base class for Jasmine reporters.  
 *  
 * @constructor  
 */  
jasmine.Reporter = function() {  
};  
  
//noinspection JSUnusedLocalSymbols  
jasmine.Reporter.prototype.reportRunnerStarting = function(runner) {  
};  
  
//noinspection JSUnusedLocalSymbols  
jasmine.Reporter.prototype.reportRunnerResults = function(runner) {  
};  
  
//...
```

S.O.L.I.D.

Sustitución de Liskov

- “Un objeto debe ser substituible por instancias de sus subclases”
 - Si **B** es subclase de **A**
 - Y **b** es una instancia de **B**
 - Se debería poder usar **b** allí donde se espere un objeto de clase **A**

Sustitución de Liskov

```
var Animal = Class.extend({  
    caminar: function() { /*...*/ },  
    comer: function() { /* .. */ }  
});
```

Sustitución de Liskov

```
var Animal = Class.extend({  
    caminar: function() { /*...*/ },  
    comer: function() { /* ... */ }  
});
```

```
var Serpiente = Animal.extend({  
    /* ... */  
});
```

```
var s = new Serpiente();  
s.caminar();
```



Sustitución de Liskov

Pretende:

- Promover la reutilización segura de código
- Mantener una semántica coherente
- Si “B es un A”, entonces “B ha de comportarse como A”

S.O.L.I.D.

Segregación de la Intefaz

- “muchas interfaces cliente específicas son mejores que una interfaz de propósito general”
 - No obligues a un cliente a depender de interfaces que no necesita
 - Polución de interfaz

Segregación de la interfaz

```
var Modal = Class.extend({  
    show: function() { ... },  
    hide: function() { ... }  
});
```

Segregación de la interfaz

```
var Modal = Class.extend({  
    show: function() { ... },  
    hide: function() { ... }  
});
```

```
var Timer = Class.extend({  
    setTimer: function(time) { ... },  
    startTimer: function() { ... },  
    onTimeout: function() { ... }  
});
```

Segregación de la interfaz

```
var Modal = Class.extend({
    show: function() { ... },
    hide: function() { ... }
});

var Timer = Class.extend({
    setTimer: function(time) { ... },
    startTimer: function() { ... },
    onTimeout: function() { ... }
});

var TimedModal = Modal.extend({
    init: function() {
        this.setTimer(100);
        this.onTimeout = this.hide;
        this.show();
        this.startTimer();
    }
});
```

Segregación de la interfaz

```
var Timer = Class.extend({
    setTimer: function(time) { ... },
    startTimer: function() { ... },
    onTimeout: function() { ... }
});
```



```
var Modal = Timer.extend({
    show: function() { ... },
    hide: function() { ... }
});
```

```
var TimedModal = Modal.extend({
    init: function() {
        this.setTimer(100);
        this.onTimeout = this.hide;
        this.show();
        this.startTimer();
    }
});
```

Segregación de la interfaz

```
var WidgetView = Class.extend({
  init: function(cont) {
    var cbind = curry(bind, cont);
    $("#E1").click(cbind(cont.onE1Click));
  }
});
```

Segregación de la interfaz

```
var WidgetView = Class.extend({
  init: function(cont) {
    var cbind = curry(bind, cont);
    $("#E1").click(cbind(cont.onE1Click));
    $("#E2").click(cbind(cont.onE2Click));
    $("#E3").click(cbind(cont.onE3Click));
    $("#E4").click(cbind(cont.onE4Click));
  }
});
```

Segregación de la interfaz

```
var WidgetView = Class.extend({
  init: function(cont) {
    var cbind = curry(bind, cont);
    $("#E1").click(cbind(cont.onE1Click));
    $("#E2").click(cbind(cont.onE2Click));
    $("#E3").click(cbind(cont.onE3Click));
    $("#E4").click(cbind(cont.onE4Click));
    $("#save").click(cbind(cont.save));
    $("#reset").click(cbind(cont.reset));
    $("#validate").click(cbind(cont.validate));
    $("#next-page").click(cbind(cont.getNextPage));
    // ...
  }
});
```

Segregación de la interfaz

```
var WidgetView = Class.extend({
  init: function(viewCont, cont, pagCont) {
    var vbind = curry(bind, cont),
        cbind = curry(bind, cont),
        pbind = curry(bind, pagCont);
    $("#E1").click(vbind(viewCont.onE1Click));
    $("#E2").click(vbind(viewCont.onE2Click));
    $("#E3").click(vbind(viewCont.onE3Click));
    $("#E4").click(vbind(viewCont.onE4Click));
    $("#save").click(cbind(cont.save));
    $("#reset").click(cbind(cont.reset));
    $("#validate").click(cbind(cont.validate));
    $("#next-page").click(pbind(pagCont.getNextPage));
    // ...
  }
});
```

Segregación de la interfaz

```
var WidgetView = Class.extend({
    init: function(viewCont, cont, pagCont) {
        var vbind = curry(bind, cont),
            cbind = curry(bind, cont),
            pbind = curry(bind, pagCont);
        $("#E1").click(vbind(viewCont.onE1Click));
        $("#E2").click(vbind(viewCont.onE2Click));
        $("#E3").click(vbind(viewCont.onE3Click));
        $("#E4").click(vbind(viewCont.onE4Click));
        $("#save").click(cbind(cont.save));
        $("#reset").click(cbind(cont.reset));
        $("#validate").click(cbind(cont.validate));
        $("#next-page").click(pbind(pagCont.getNextPage));
        // ...
    }
});
```

S.O.L.I.D.

Dependency inversion

- “Depende de abstracciones. No dependas de cocreciones”
 - Entidades de alto nivel no deben depender de entidades de bajo nivel. Ambos deben depender de abstracciones.
 - Las abstracciones no deben depender de detalles. Los detalles deben depender de abstracciones.

Dependency Inversion

```
var Model = Class.extend({
  save: function() {
    var tbind = curry(bind, this),
        stop = bind(this.icon, this.icon.stop);
    $.post(this.url, this.getData())
      .success(tbind(this.saved))
      .error(tbind(this.saveFailed))
      .complete(stop);
  }
});
```

Dependency Inversion

```
var Model = Class.extend({
    save: function() {
        var tbind = curry(bind, this),
            stop = bind(this.icon, this.icon.stop);
        $().post(this.url, this.getData())
            .success(tbind(this.saved))
            .error(tbind(this.saveFailed))
            .complete(stop);
    }
});
```



Dependency Inversion

```
var Model = Class.extend({
  init: function(store) { this.store = store; } ←
  save: function() {
    var tbind = curry(bind, this),
        stop = bind(this.icon, this.icon.stop);
    this.store.save(
      this.data,
      tbind(this.saved),
      tbind(this.saveFailed),
      stop
    );
  }
});
```

```
var Store = Class.extend({
  save: function(data, success, error, complete) { }
});
```

Dependency Inversion

```
var ServerStore = Store.extend({
  save: function(data, success, error, complete) {
    $.post(this.url, data)
      .success(success)
      .error(error)
      .complete(complete);
  }
});
```

```
var db = {};
var MemStore = Store.extend({
  save: function(data, success, error, complete) {
    db[this.url] = data;
    complete();
    success();
  }
});
```

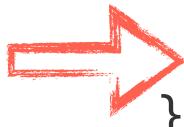
Dependency Inversion

Caso práctico: backbone.js

- <https://github.com/documentcloud/backbone/blob/master/backbone.js>
- `Backbone.Model#fetch`, línea 335

Dependency Inversion

```
fetch: function(options) {  
    options = options ? _.clone(options) : {};  
    var model = this;  
    var success = options.success;  
    options.success = function(resp, status, xhr) {  
        if (!model.set(model.parse(resp, xhr), options))  
            return false;  
        if (success) success(model, resp, options);  
    };  
    return this.sync('read', this, options);  
},
```



Patrones de organización

Patrones de organización

- Parámetros con nombre/por defecto
- Módulos y namespaces
- Control de acceso
- Mixins

Parámetros con nombre

```
function ajax(url, data, method,
              success, error, complete) {
  url || url = "/";
  data || data = {};
  method || method = "POST";
  // ...
}

ajax("/",  
      {},  
      "GET",  
      function(){ ... },  
      function() { ... },  
      function() { ... });
```

Parámetros con nombre

```
function ajax(options) {  
    var url = options.url || "/",  
        data = options.data || {},  
        method = options.method || "POST";  
    //...  
}  
  
ajax({data: [1, 2], complete: function() { ... }});
```

Parámetros por defecto

```
function ajax(options) {  
  var fn = function() {},  
      defaults = {url: "/", data: [], method: "POST",  
                  success: fn, error: fn, complete: fn};  
  options = merge(defaults, options);  
  // ...  
}  
  
ajax({data: [1, 2], complete: function() { ... }});
```

Parámetros por defecto

backbone.js:749

```
reset: function(models, options) {
  for (var i = 0, l = this.models.length; i < l; i++) {
    this._removeReference(this.models[i]);
  }
  this._reset();
  if (models) this.add(models, _.extend({silent: true}, options));
  if (!options || !options.silent) this.trigger('reset', this, options);
  return this;
},
```



Intermedio: merge

¿Cómo sería esa función `merge`?

Intermedio: merge

¿Cómo sería esa función `merge`?

```
function merge() {  
  var slice = Array.prototype.slice,  
      sources = slice.call(arguments),  
      target = {};  
  sources.forEach(function(source) {  
    for (var p in source) if (source.hasOwnProperty(p)) {  
      target[p] = source[p];  
    }  
  });  
  return target;  
}
```

Módulos y namespaces

- JavaScript no tiene concepto de namespace
- Todo tirado en objeto global
 - Mucha polución
 - Colisión de nombres
 - Difícil de navegar

Módulos y namespaces

- JavaScript no tiene concepto de namespace
- Todo tirado en objeto global
 - Mucha polución
 - Colisión de nombres
 - Difícil de navegar
- ¡Pero tenemos funciones!

Módulos y namespaces

```
function miHelper() {  
    // ...  
}  
  
var miVariableTemporal = 0;  
var estadoLocal = {};
```

Módulos y namespaces

```
function sandbox() {  
    function miHelper() {  
        // ...  
    }  
  
    var miVariableTemporal = 0;  
    var estadoLocal = {};  
}
```

Módulos y namespaces

```
(function sandbox() {
    function miHelper() {
        // ...
    }

    var miVariableTemporal = 0;
    var estadoLocal = {};
})()
```

Módulos y namespaces

```
(function sandbox() {  
    function miHelper() {  
        // ...  
    }  
  
    var miVariableTemporal = 0;  
    var estadoLocal = {};  
    }())
```



Módulos y namespaces

```
function miFuncionUtil() {  
    // ...  
}
```

```
function miGranMetodo() {  
    // ...  
}
```

```
function miEstupendoHelper() {  
    // ...  
}
```

Módulos y namespaces

```
function aux() { }
var state = "off";
```

```
function miFuncionUtil() {
    // ...
}
```

```
function miGranMetodo() {
    // ...
}
```

```
function miEstupendoHelper() {
    // ...
}
```

Módulos y namespaces

```
(function() {  
    function aux() { }  
    var state = "off";  
  
    function miFuncionUtil() { }  
  
    function miGranMetodo() { }  
  
    function miEstupendoHelper() { }  
}())
```

Módulos y namespaces

```
var Modulo = (function() {  
    function aux() { }  
    var state = "off";  
  
    function miFuncionUtil() { }  
  
    function miGranMetodo() { }  
  
    return {  
        miFuncionUtil: miFuncionUtil,  
        miGranMetodo: miGranMetodo  
    };  
}());
```

Módulos y namespaces

```
→ var Modulo = (function() {  
    function aux() { }  
    var state = "off";  
  
    function miFuncionUtil() { }  
  
    function miGranMetodo() { }  
  
    → return {  
        miFuncionUtil: miFuncionUtil,  
        miGranMetodo: miGranMetodo  
    };  
}());
```

Módulos y namespaces

```
Modulo.miFuncionUtil();
```

Módulos y namespaces

```
var Modulo = {};  
  
(function(Modulo) {  
    function aux() {}  
    var state = "off";  
  
    Modulo.miFuncionUtil = function() {}  
  
    Modulo.miGranMetodo = function() {}  
  
})(Modulo);
```

Módulos y namespaces

```
var Modulo = {};  
  
(function(Modulo) {  
    function aux() {}  
    var state = "off";  
    Modulo.miFuncionUtil = function() {}  
}(Modulo));  
  
(function(Modulo) {  
    Modulo.miGranMetodo = function() {}  
}(Modulo));
```

Módulos y namespaces

```
var Modulo = (function(Modulo) {  
    function aux() { }  
    var state = "off";  
    Modulo.miFuncionUtil = function() { }  
    return Modulo;  
}(Modulo || {}));
```

```
var Modulo = (function(Modulo) {  
    Modulo.miGranMetodo = function() { }  
    return Modulo;  
}(Modulo || {}));
```

Módulos y namespaces

```
→ var Modulo = (function(Modulo) {  
    function aux() { }  
    var state = "off";  
    Modulo.miFuncionUtil = function() { }  
    return Modulo;  
}(Modulo || {})); ←
```

```
var Modulo = (function(Modulo) {  
    Modulo.miGranMetodo = function() { }  
    return Modulo;  
}(Modulo || {}));
```

Módulos y namespaces

Una truco más sofisticado:

- Tenemos un módulo
- Al que añadimos propiedades en varios ficheros
- Queremos compartir cierta información entre ficheros
- Pero que no sea accesible una vez terminada la carga

Módulos y namespaces

```
var Modulo = (function(Modulo) {  
    _private.password = "1234";  
}(Modulo || {}));
```

```
var Modulo = (function(Modulo) {  
    Modulo.login = function(pass) {  
        return pass == _private.password;  
    };  
}(Modulo || {}));
```

```
Modulo._private; // undefined
```

Módulos y namespaces

```
var Modulo = (function (Modulo) {
    var _private = Modulo._private = (Modulo._private || {}),
        _seal = Modulo._seal = (Modulo._seal || function () {
            delete Modulo._private;
            delete Modulo._seal;
        });
    // acceso permanente a _private y _seal
    return Modulo;
})(Modulo || {});
```

Módulos y namespaces

```
var Modulo = (function (Modulo) {  
    var _private = Modulo._private;  
  
    // acceso a _private  
  
}(Modulo || {}));
```

Módulos y namespaces

```
Modulo._seal();
```

Intermedio: mejorar klass.js

Convierte klass.js en un módulo Class

Módulos y namespaces

Submódulos: muy fácil!

```
var MiLibreria = {};  
  
MiLibreria.eventos = (function(eventos) {  
    eventos.on = function() { };  
    eventos.off = function() { };  
    return eventos;  
}(MiLibreria.eventos));
```

Módulos y namespaces

Según crece la aplicación...

Módulos y namespaces

Según crece la aplicación...

```
var MiLibreria = MiLibreria || {};
```

Módulos y namespaces

Según crece la aplicación...

```
var MiLibreria = MiLibreria || {};
MiLibreria.widgets = MiLibreria.widgets || {};
```

Módulos y namespaces

Según crece la aplicación...

```
var MiLibreria = MiLibreria || {};
MiLibreria.widgets = MiLibreria.widgets || {};
MiLibreria.widgets.buttons = MiLibreria.widgets.buttons || {};

MiLibreria.widgets.buttons.actionButtons = (function(buttons) {
    buttons.ok = new Widget({ ... });
    buttons.cancel = new Widget({ ... });
})(MiLibreria.widgets.buttons.actionButtons || {}));
```

Módulos y namespaces

Namespaces, pero más cómodos:

```
MiLib.namespace('widgets.buttons.actionButtons', function(my) {  
    my.ok = new Widget({ ... });  
    my.cancel = new Widget({ ... });  
});
```

Intermedio: namespace

¿Como sería la función MiLib.namespace?

Intermedio: namespace

¿Como sería la función MiLib.namespace?

```
var MiLib = (function(my) {  
    my.namespace = function(string, sandbox) {  
        // ???  
    };  
    return my;  
}(MiLib || {}));
```

Intermedio: namespace

¿Como sería la función MiLib.namespace?

```
var MiLib = (function(my) {
    my.namespace = function(string, sandbox) {
        var spaces = string.split('.'),
            root = my,
            space;
        while (space = spaces.shift()) {
            root = root[space] || (root[space] = {});
        }
        return sandbox(root);
    };
    return my;
}(MiLib || {}));
```

Mixins

- Otra forma de reutilizar código
- Sin las limitaciones de la herencia
- Para código de propósito general
- Algo similar a herencia múltiple

Mixins

```
var Mixin = function() {};
Mixin.prototype = {
  inspect: function() {
    var output = [];
    for(key in this) {
      output.push(key + ': ' + this[key]);
    }
    return output.join(', ');
  }
};
```

Mixins

```
var Persona = Class.extend({
    init: function(nombre) {
        this.nombre = nombre;
    }
});

augment(Persona, Mixin);

var pepito = new Persona("Pepito");
pepito.inspect();
```

Mixins

```
function augment(target, source) {  
    for (var prop in source.prototype) {  
        if(!target.prototype[prop]) {  
            target.prototype[prop] = source.prototype[prop];  
        }  
    }  
}
```

Intermedio: Mejores Mixins

Lo podemos hacer mejor!

- Mejor integración con klass.js
- Callbacks de inclusión (estilo ruby)

Intermedio: Mejores Mixins

```
var StaticModule = {  
  propiedadDeClase: "Soy una propiedad de clase",  
  included: function(klass) { console.log("Incluido!"); }  
};  
  
var InstanceModule = {  
  diHola: function() { alert("HOLA!"); },  
  mixed: function(klass) { console.log("Mezclado!"); }  
};  
  
var MiClase = Class.extend({  
  init: function() { }  
});  
  
MiClase.include(StaticModule);  
MiClase.mixin(InstanceModule);
```

Patrones de creación de objetos

Factoría

Delegar la creación de un objeto

- Elegir el constructor dinámicamente
- Procesos de construcción complejos
- Desacoplar detalles de implementación

Factoría

```
var locales = {
    es: {header: {title: "Mi Título"}},
    en: {header: {title: "My Title"}}
};

var I18n = Class.extend({
    translate: function(path) {
        var position = locales[this.locale],
            path = path.split('.'),
            currentPath;
        while (currentPath = path.shift()) {
            position = position[currentPath];
        }
        return position;
    }
});

var english = new I18n();
english.locale = "en";
english.translate('header.title'); // "My
```

Factoría

```
var GlobalConfig = {locale: "es"};
```

Factoría

```
var GlobalConfig = {locale: "es"};  
  
I18n.withCurrentLocale = function() {  
    var instance = new this;  
    instance.locale = GlobalConfig.locale;  
    return instance;  
};
```

Factoría

```
var i18n = I18n.withCurrentLocale();
alert(i18n.translate('header.title'));
```

Factoría

```
var Events = Class.extend({  
    on: function(event, cb) { },  
    off: function(event, cb) { }  
});
```

```
var IEEvents = Events.extend({  
    on: function(event, cb) { },  
    off: function(event, cb) { }  
});
```

```
Events.getInstance = function() {  
    if (checkForIExplorer()) {  
        return new IEEvents();  
    } else {  
        return new Events();  
    }  
};
```

Factoría

Controlar las instancias

```
var Enemigo = (function() {
    var enemigos = [];
    var Enemigo = Class.extend({ /*...*/ });
    Enemigo.crear = function() {
        var instance;
        if (enemigos.length < 5) {
            instance = new Enemigo();
            enemigos.push(instance);
            return instance;
        } else {
            throw new Error("No puede haber más!");
        }
    }
    return Enemigo;
}());
```

Factoría

```
var Recurso = (function() {
    var libres = [];
    var Recurso = Class.extend({
        liberar: function() { libres.push(this); }
    });
    Recurso.crear = function() {
        if (libres.length > 0) {
            return libres.pop();
        } else {
            return new Recurso();
        }
    }
    return Recurso;
}());
```

Factoría

¿Cuándo usar factorías?

- La construcción de un objeto es compleja
- Seleccionar el constructor adecuado según entorno
- Necesitamos controlar el instanciado

Singleton

Clase con una única instancia

- Un tipo peculiar de Factoría
- Cuando no tiene sentido más de una instancia

Intermedio: I18n.js

Librería de internacionalización

```
var Config = {locale: 'en'};

I18n.addTranslation('en', {header: {title: "My Title"}});
I18n.addTranslation('es', {header: {title: "Mi Título"}});

var i18n = I18n.withCurrentLocale();
alert(i18n.translate('header.title'));

// Singleton!
console.log(i18n === I18n.withCurrentLocale())
```

Patrones de abstracción

Patrones de abstracción

- Iteradores
- Decorador
- Fachada
- Estrategia
- Inyección de dependencias
- Proxy

Iteradores

Recorrer una colección

- Sin revelar detalles de implementación
- Mayor control sobre la iteración

Iteradores

```
var ListadoAlumnos = Class.extend({  
    init: function() { this.alumnos = []; },  
    add: function(nombre, ciudad) {  
        this.alumnos.push({  
            nombre: nombre,  
            ciudad: ciudad  
        });  
    }  
});
```

Iteradores

```
var ListadoAlumnos = Class.extend({  
    init: function() { this.alumnos = []; },  
    add: function(nombre, ciudad) {  
        this.alumnos.push({  
            nombre: nombre,  
            ciudad: ciudad  
        });  
    }  
});  
  
var lista = new ListadoAlumnos();  
lista.add("Gonzalo", "Madrid");  
lista.add("Gerardo", "Madrid");  
lista.add("Guzman", "Valencia");
```

Iteradores

```
var lista = new ListadoAlumnos();
lista.add("Gonzalo", "Madrid");
lista.add("Gerardo", "Madrid");
lista.add("Guzman", "Valencia");
```

```
var alumnos = lista.alumnos;
for (var i=0; i<alumnos.length; i++) {
    if (alumnos[i].ciudad == "Madrid") {
        console.log(alumnos[i].nombre);
    }
}
```

Iteradores

```
var lista = new ListadoAlumnos();
lista.add("Gonzalo", "Madrid");
lista.add("Gerardo", "Madrid");
lista.add("Guzman", "Valencia");
```

```
var alumnos = lista.alumnos;
for (var i=0; i<alumnos.length; i++) {
    if (alumnos[i].ciudad == "Madrid") {
        console.log(alumnos[i].nombre);
    }
}
```

Iteradores

```
var ListadoAlumnos = Class.extend({
    init: function() { this.alumnos = []; },
    add: function(nombre, ciudad) {
        this.alumnos.push({ nombre: nombre, ciudad: ciudad });
    },
    forEachIn: function(ciudad, fn) {
        for (var i=0; i<this.alumnos.length; i++)
            if (this.alumnos[i].ciudad == ciudad) {
                fn(this.alumnos[i].nombre);
            }
    }
});
```



Iteradores

```
var lista = new ListadoAlumnos();
lista.add("Gonzalo", "Madrid");
lista.add("Gerardo", "Madrid");
lista.add("Guzman", "Valencia");

lista.forEachIn("Madrid", function(n) {
    console.log(n);
});
```

Iteradores

```
var ListadoAlumnos = Class.extend({  
    init: function() { this.alumnos = []; },  
    add: function(nombre, ciudad) {  
        this.alumnos.push({ nombre: nombre, ciudad: ciudad });  
    },  
    next: function() {  
        this.index = this.index || 0;  
        return this.alumnos[this.index++];  
    }  
});
```

Iteradores

```
var lista = new ListadoAlumnos();
lista.add("Gonzalo", "Madrid");
lista.add("Gerardo", "Madrid");
lista.add("Guzman", "Valencia");
```

```
var alumno;
while(alumno = lista.next()) {
    alert(alumno.nombre);
}
```

Iteradores

```
var Iterator = Class.extend({
    init: function(collection) {
        this.col = collection;
        this.pos = 0;
    },
    next: function() {
        return this.col[this.pos++];
    }
});

var ListadoAlumnos = Class.extend({
    init: function() { this.alumnos = []; },
    add: function(nombre, ciudad) {
        this.alumnos.push({ nombre: nombre, ciudad: ciudad });
    },
    getIterator: function() {
        return new Iterator(this.alumnos);
    }
});
```

Iteradores

```
var lista = new ListadoAlumnos();
lista.add("Gonzalo", "Madrid");
lista.add("Gerardo", "Madrid");
lista.add("Guzman", "Valencia");

var alumno, iter = lista.getIterator();
while(alumno = iter.next()) {
    if (alumno.ciudad == "Madrid") {
        console.log(alumno.nombre);
    }
}
```

Iteradores

```
var Fibonacci = Class.extend({
  init: function() {
    this.a = 0;
    this.b = 1;
  },
  next: function() {
    var next = this.a;
    this.a = this.b;
    this.b = this.b + next;
    return next;
  }
});
```

```
var iter = new Fibonacci();
for (var i=10; i--;) {
  console.log(iter.next());
}
```

Decorador

Añadir funcionalidad dinámicamente

- Evitar subclases innecesarias
- Modificar comportamientos “en vivo”
- Manteniendo la interfaz/contrato! (transparente)
- Muy apropiada para JS

Decorador

```
var Producto = Class.extend({
    init: function(precio) {
        this.precio = precio;
    },
    getPrecio: function() {
        return this.precio;
    }
});
```

Decorador

```
var Producto = Class.extend({  
    init: function(precio) {  
        this.precio = precio;  
    },  
    getPrecio: function() {  
        return this.precio;  
    }  
});
```

```
var ProductoConIVA = Producto.extend({  
    init: function(precio) {  
        this._super(precio);  
    },  
    getPrecio: function() {  
        return this._super() * 1.21;  
    }  
});
```

Decorador

```
var ProductoConDescuento = Producto.extend({
    init: function(precio) {
        this._super(precio);
    },
    getPrecio: function() {
        return this._super() * 0.90;
    }
});

var ProductoConIVAConDescuento = ProductoConIVA.extend({
    init: function(precio) {
        this._super(precio);
    },
    getPrecio: function() {
        return this._super() * 0.90;
    }
});
```

Decorador

```
var ProductoTecnologico = Producto.extend({  
});  
  
var ProductoTecnologicoConDescuento =  
    ProductoConDescuento.extend({  
});  
  
var ProductoTecnologicoConIVA = ProductoConIVA.extend({  
});  
  
var ProductoTecnologicoConIVAConDescuento =  
    ProductoConIVAConDescuento.extend({  
});
```

Decorador

```
var Producto = Class.extend({
    init: function(precio) {
        this.precio = precio;
    },
    getPrecio: function() {
        return this.precio;
    }
});
```

Decorador

```
var Producto = Class.extend({
    init: function(precio) {
        this.precio = precio;
    },
    getPrecio: function() {
        return this.precio;
    }
});
```

```
var ProductoConIVA = Class.extend({
    init: function(producto) {
        this.producto = producto;
    },
    getPrecio: function() {
        return this.producto.getPrecio() * 1.21;
    }
});
```

Decorador

```
var ProductoConDescuento = Class.extend({
    init: function(producto) {
        this.producto = producto;
    },
    getPrecio: function() {
        return this.producto.getPrecio() * 0.90;
    }
});
```

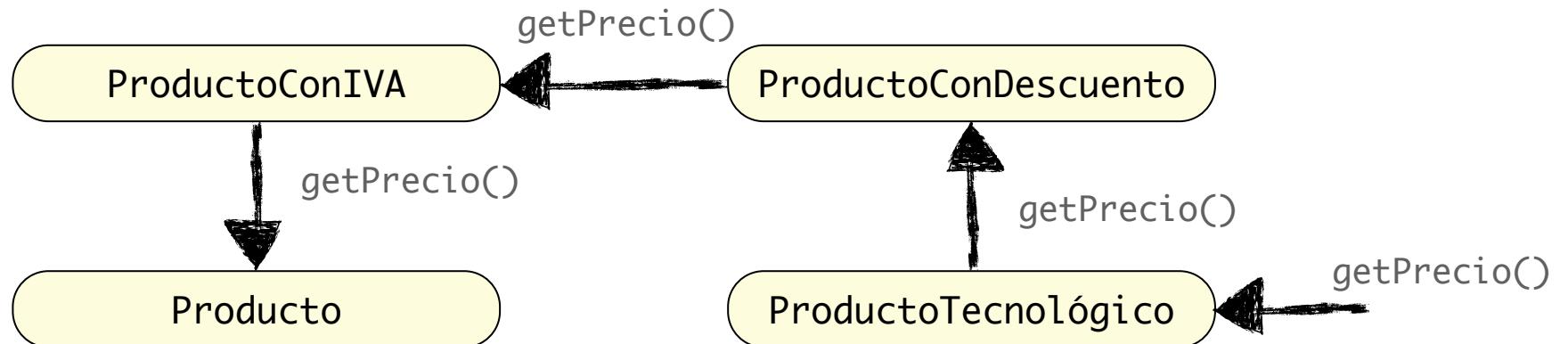
```
var ProductoTecnologico = Class.extend({
    init: function(producto) {
        this.producto = producto;
    },
    getPrecio: function() {
        return this.producto.getPrecio() * 0.90;
    }
});
```

Decorador

```
var producto = new Producto(20);
producto = new ProductoConIVA(producto);
producto = new ProductoConDescuento(producto);
producto = new ProductoTecnologico(producto);
producto.getPrecio(); // 19.602
```

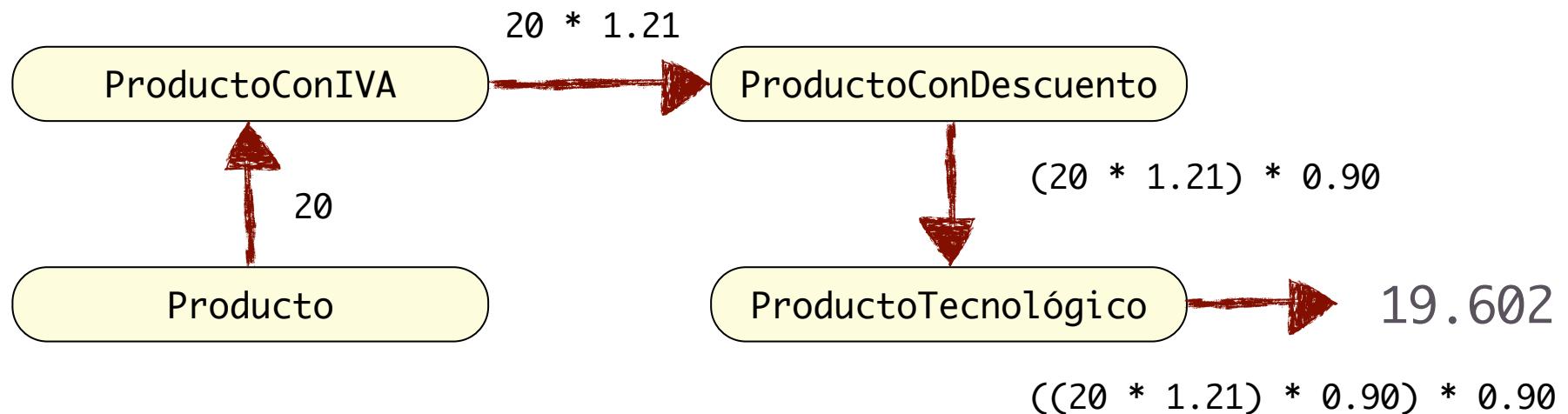
Decorador

```
var producto = new Producto(20);
producto = new ProductoConIVA(producto);
producto = new ProductoConDescuento(producto);
producto = new ProductoTecnológico(producto);
producto.getPrecio(); // 19.602
```



Decorador

```
var producto = new Producto(20);
producto = new ProductoConIVA(producto);
producto = new ProductoConDescuento(producto);
producto = new ProductoTecnologico(producto);
producto.getPrecio(); // 19.602
```



Decorador

Cuando usar decoradores:

- Modificar el comportamiento manteniendo interfaz
- Añadir post- o preproceso a la salida/entrada de un objeto
 - Serialización
 - Interfaz
- Dentro de factorías
- Intervenir las llamadas a métodos

Fachada

Simplificar el interfaz y desacoplar al cliente

- Patrón muy simple
- El mejor amigo del programador JavaScript
- Casi todas las librerías son fachadas
- Comodidad

Fachada

Ejemplo por excelencia: jQuery

- `$(“selector”)`
- `$(“selector”).click(...)`
- `$.ajax`
- ...

Fachada

Nuestra librería klass.js

- Una capa sobre los mecanismos nativos
- Simplifica el “interfaz”
- Mucho más cómodo que hacerlo a mano

Fachada

```
var DOMNode = Class.extend({
    init: function(tag) {
        this.el = document.createElement(tag);
    },
    setClass: function(klass) {
        this.el.className += (" " + klass);
    },
    setId: function(id) { this.el.id = id; },
    getElement: function() { return this.el; }
});
```

```
var node = new DOMNode('div');
node.setClass("big");
node.setClass("red");
node.setId("my-button");
document.body.appendChild(node.getElement());
```

Fachada

```
DOMNode.create = function(tag, classes, id) {  
    var node = new this(tag);  
    node.setClass(classes.join(" "));  
    node.setId(id);  
    return node.getElement();  
}  
var el = DOMNode.create("div", ["big", "red"], "my-button");  
document.body.appendChild(el);
```

Fachada

¿Cuándo usar fachadas?

- Clases muy potentes y complejas
 - Simplificar los usos más comunes
- Operaciones con mucha preparación
 - XMLHttpRequest
- Código repetitivo
- Para cambios grandes del código

Estrategia

Seleccionar algoritmos dinámicamente

- Dada una familia de algoritmos
- Encapsulados bajo un mismo interfaz
- Elegir el más apropiado

Estrategia

```
var validate = (function() {
  var validators = {
    email: function(value) {
      return /^[([^\<>()\\.,;:\\s@"]+([^.][^\<>()\\.,;:\\s@"]+)*)(\".+")@([([0-9]{1,3}\\.){3}[0-9]{1,3}\.){3}[a-zA-Z\-\_0-9]+\.)+[a-zA-Z]{2,})$/ .test(value);
    },
    number: function(value) {
      return /^d+$/ .test(value);
    }
  };
  return function(data) {
    var validation = validators[data.type];
    return validation && validation(data.value);
  };
}());

validate({type: "email", value: "eliasagc@gmail.com"});
```



Estrategia

```
var MoodConsole = Class.extend({
    init: function() {
        this.state = "normal";
        this.filters = {
            normal: new DummyFilter(),
            relajado: new RelaxedFilter(),
            cabreado: new AngryFilter()
        };
    },
    log: function(msg) {
        msg = this.filters[this.state].filter(msg);
        console.log(msg);
    },
    molestar: function() { this.state = "cabreado"; },
    masajear: function() { this.state = "relajado"; },
    dejarEnPaz: function() { this.state = "normal"; }
});
```



Estrategia

```
var DummyFilter = Class.extend({
    filter: function(s) { return s; }
});

var AngryFilter = DummyFilter.extend({
    filter: function(s) { return s.toUpperCase() + "!!"; }
});

var RelaxedFilter = DummyFilter.extend({
    filter: function(s) { return "..." + s + "..."; }
});

var moodConsole = new MoodConsole();
moodConsole.log("hola!");
moodConsole.molestar();
moodConsole.log("grrr...");
moodConsole.masajear();
moodConsole.log("gracias");
```



Estrategia

¿Cuándo usar estrategias?

- Una misma acción puede tener varios significados
 - Según un estado variable (selector de contexto)
 - click, touch, una botón “Guardar”, etc..
- Validaciones
- En general: cuando hay que elegir un algoritmo diferente para cada caso

Inyección de dependencias

Selección dinámica de un componente

- Según el principio de Inversión de Dependencias
- El cliente consume siguiendo un interfaz
- La clase concreta se selecciona en tiempo de ejecución

Inyección de dependencias

```
var View = Class.extend({
    init: function(renderer, model) {
        this.renderer = renderer;
        this.model = model;
    },
    render: function() {
        this.renderer.render(this.template, this.model);
    }
});
```



```
var HtmlRenderer = Class.extend({
    render: function() { ... }
});
```

```
var XMLRenderer = Class.extend({
    render: function() { ... }
});
```

Inyección de dependencias

```
var miVista = new View(new XMLRenderer('fast'), {});
```

Proxy

Un objeto hace de interfaz de otro objeto o recurso

- Entre el cliente y el servicio
- Controlar y optimizar el acceso
- El Proxy implementa exactamente el mismo interfaz
 - No es un decorador (no añade funcionalidad)
 - No es una fachada (no simplifica)

Proxy

Tres tipos de proxy:

- Virtual Proxy
 - Sustituye al objeto real mientras carga
 - Optimización
- Protection Proxy
 - Controla el acceso
 - Seguridad
- Remote Proxy
 - Reflejo de un recurso remoto
 - Abstracción

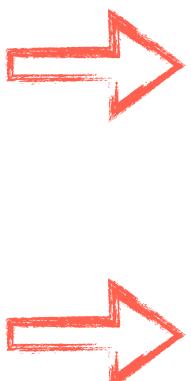
Virtual Proxy

```
var Recurso = Class.extend({
    init: function() {
        operacionMuyCostosa();
    },
    read: function() { ... },
    write: function() { ... }
});
```

Virtual Proxy

```
var Recurso = Class.extend({  
    init: function() {  
        operacionMuyCostosa();  
    },  
    read: function() { ... },  
    write: function() { ... }  
});
```

```
var RecursoProxy = Recurso.extend({  
    init: function() { this.super_init = this._super; },  
    read: function() {  
        this.super_init();  
        this._super();  
    },  
    write: function() {  
        this.super_init();  
        this._super();  
    }  
});
```



Virtual Proxy

```
var Recurso = Class.extend({
    init: function() {
        operacionMuyCostosa();
        if (this.onLoad) this.onLoad();
    },
    write: function() { ... }
});

var RecursoProxy = Recurso.extend({
    init: function() {
        this.pending = [];
        this.loaded = false;
        this._super();
    },
    write: function() {
        if (loaded) { this._super(); }
        else { pending.push({op: 'write', params: arguments}); }
    }
});
```



Remote Proxy

```
var User = Class.extend({
    init: function() { /*...*/ },
    getName: function() { return this.name; }
});
```

```
User.find = function(id, cb) {
    $.get('/user/' + id, function(data) {
        var user = new User(data);
        cb(user);
    });
};
```

```
User.find(12, function(user) {
    console.log(user.getName());
});
```

Proxy

¿Cuándo usar un Proxy?

- Inicialización perezosa
- “Sustituto” que anote mientras el sujeto real arranca
- Cuando el sujeto real es remoto
- Cache!

Patrones de interacción

Patrones de interacción

- Pub/Sub u Observador
- Mediator
- Comandos y Cadena de Responsabilidad
- “Hydra”

Observador

En vez de preguntar, escucha lo que sucede

- Muy común en JavaScript (eventos)
- Reaccionar ante cambios de estado o sucesos
- Dos entidades
 - Publicador
 - Suscriptor(es)

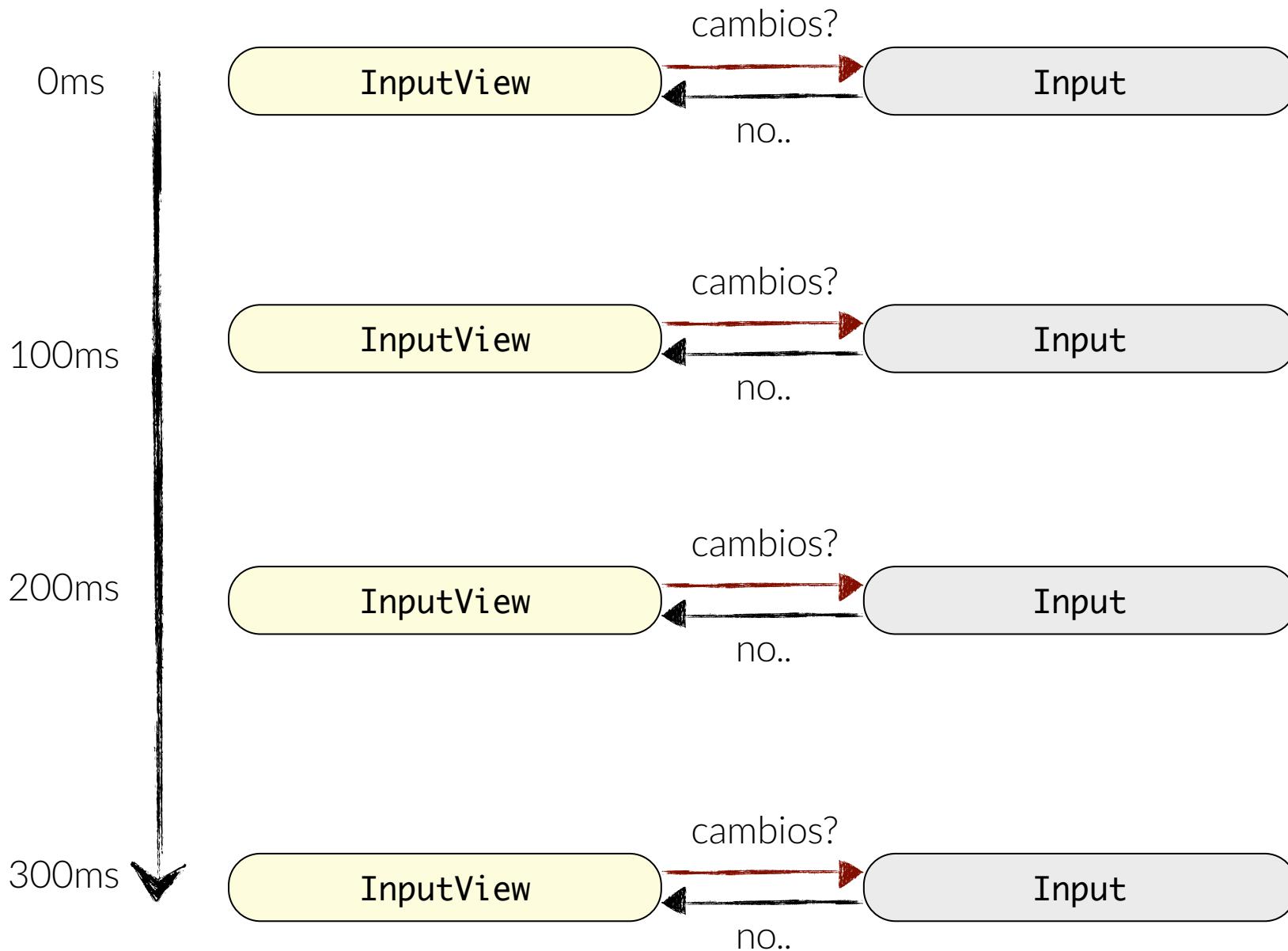
Observador

```
var Input = Class.extend({
    getValue: function() { return this.value; }
});

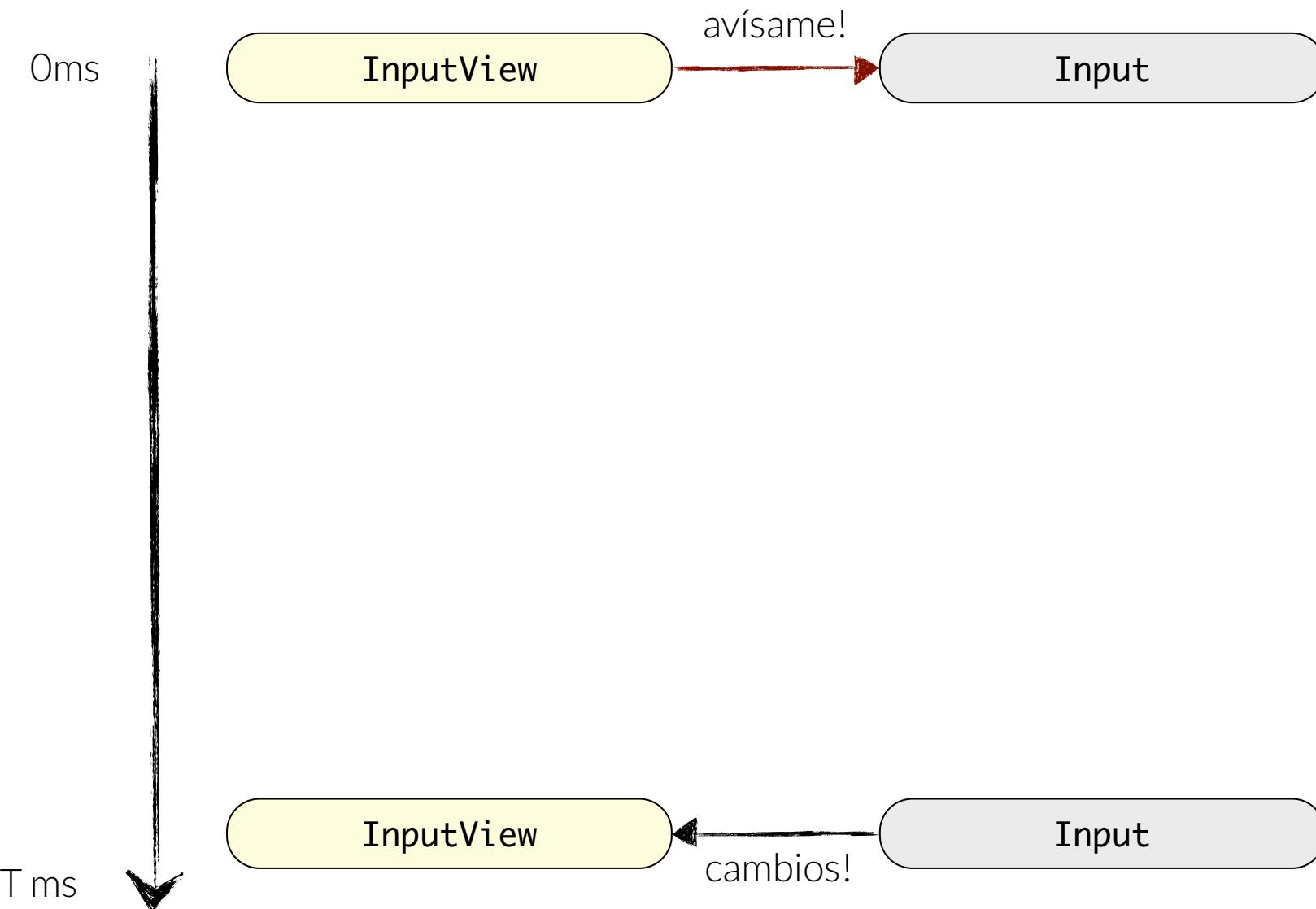
var InputView = Class.extend({
    init: function(input) {
        this.input = input;
        this.value = this.input.getValue();
        setInterval(bind(this, this.onTimer), 100);
    },
    onTimer: function() {
        var newValue = this.input.getValue();
        if (newValue !== this.value) {
            this.value = newValue;
            console.log("CHANGED: " + this.value);
        }
    }
});
```



Observador



Observador



Observador

```
var Input = Class.extend({
    init: function() {
        this.observers = [];
    },
    getValue: function() {
        return this.value;
    },
    setValue: function(v) {
        this.value = v;
        this.publish(v);
    },
    onChange: function(cb) {
        this.observers.push(cb);
    },
    publish: function(cb) {
        this.observers.forEach(function(o) { o(cb); });
    }
});
```

Observador

```
var Input = Class.extend({
    init: function() {
        this.observers = [];
    },
    getValue: function() {
        return this.value;
    },
    setValue: function(v) {
        this.value = v;
        this.publish(v);
    },
    onChange: function(cb) {
        this.observers.push(cb);
    },
    publish: function(cb) {
        this.observers.forEach(function(o) { o(cb); });
    }
});
```



Observador

```
var InputView = Class.extend({
  init: function(input) {
    this.input = input;
    this.input.onChange(bind(this, this.update));
  },
  update: function(newValue) {
    console.log("CHANGED: " + newValue);
  }
});
```

Observador

```
var input = new Input();  
  
var observer1 = new InputView(input);  
var observer2 = new InputView(input);  
var observer3 = new InputView(input);  
  
input.setValue("No te duermas!");
```

Intermedio: Observable

¡Podemos hacerlo mejor!

```
Input.mixin(Observable);
```

```
var InputView = Class.extend({
  init: function(input) {
    this.input.subscribe('change', bind(this, this.update));
    this.input.subscribe('invalid', bind(this, this.invalid));
  },
  update: function(newValue) {
    console.log("CHANGED: " + newValue);
  },
  invalid: function(value) {
    console.log("El valor " + value + " es inválido!");
  }
});
```

Intermedio: Observable

```
var Observable = {  
  mixed: function(klass) {  
    // ???  
  },  
  subscribe: function(event, callback) {  
    // ???  
  },  
  unsubscribe: function(event, callback) {  
    // ???  
  },  
  publish: function(event) {  
    // ???  
  }  
};
```

Observador

¿Cuándo utilizar Observador?

- Se utiliza muy a menudo!
- Propagar cambios
- Reaccionar a sucesos
- Comunicación uno-a-muchos

Mediator

Un punto central de control del sistema

- Desacoplamiento a gran escala
- Sencillo y muy importante
- Dividir la lógica en dos niveles
 - Componente
 - Aplicación

Mediator

Home Connect Discover Me Search

Elias Alonso
View my profile page

135 TWEETS 160 FOLLOWING 55 FOLLOWERS

Compose new Tweet...

Who to follow · Refresh · View all

- Chad Fowler** @chadfowler
Followed by Enrique Garcia
[Follow](#)
- jQuery Style** @jQueryStyle
Followed by Pedro Martin
[Follow](#)
- Digg** @digg
Followed by Digg_Updates
[Follow](#)

Browse categories · Find friends

Trends · Change

Tweets

1 new Tweet

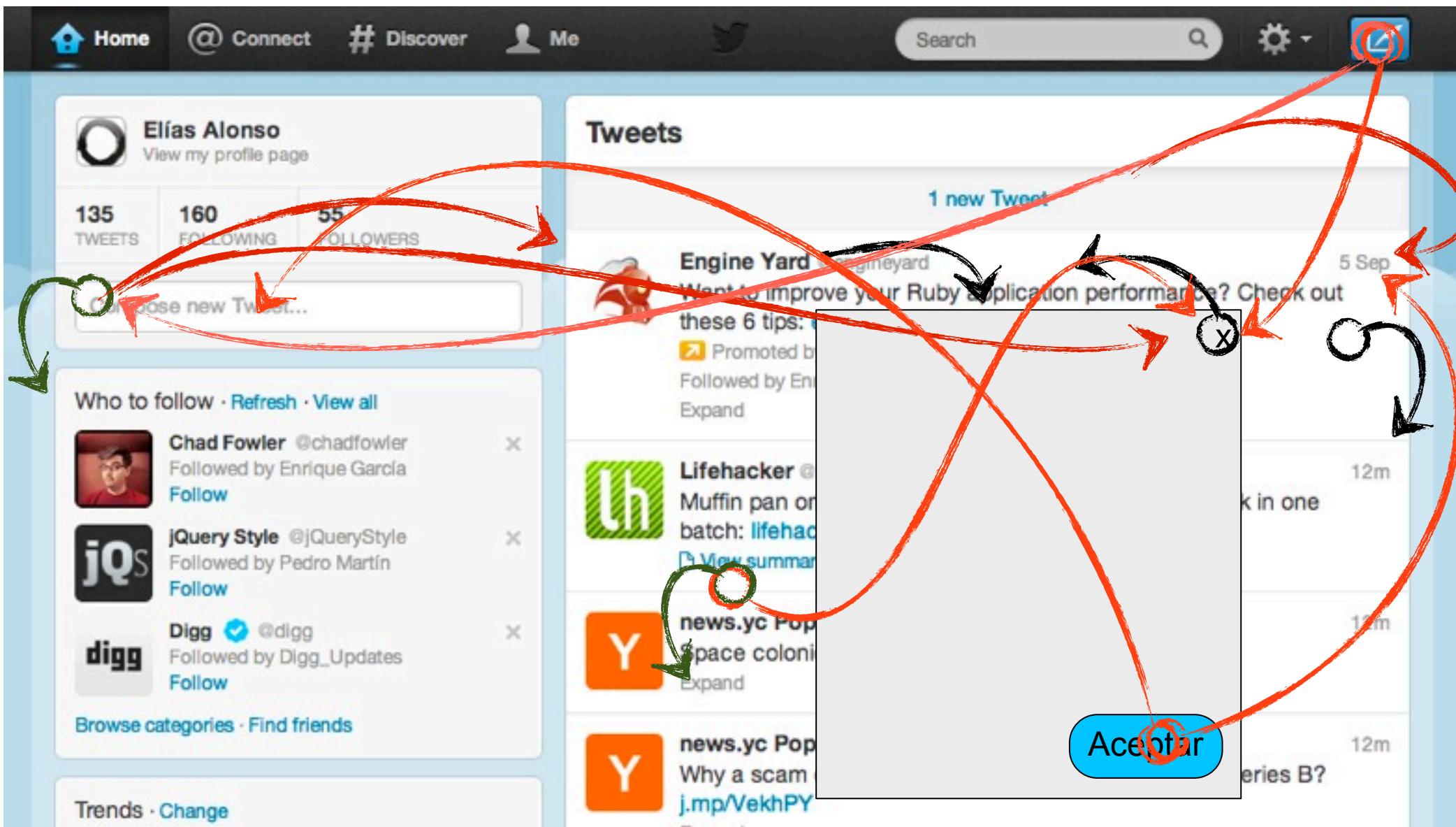
- Engine Yard** @engineyard 5 Sep
Want to improve your Ruby application performance? Check out these 6 tips: ey.io/RaHPoP
 Promoted by Engine Yard
Followed by Enrique Macip, dotemacs and Rendered Text.
[Expand](#)
- Lifehacker** @lifehacker 12m
Muffin pan omelets let you make breakfast for the week in one batch: lifehac.kr/IVYdp
[View summary](#)
- news.yc Popular** @newsycombinator 12m
Space colonies of the 1970s j.mp/pzsqnh
[Expand](#)
- news.yc Popular** @newsycombinator 12m
Why a scam company was able to raises \$76 Million Series B? j.mp/VekhPY

Mediator

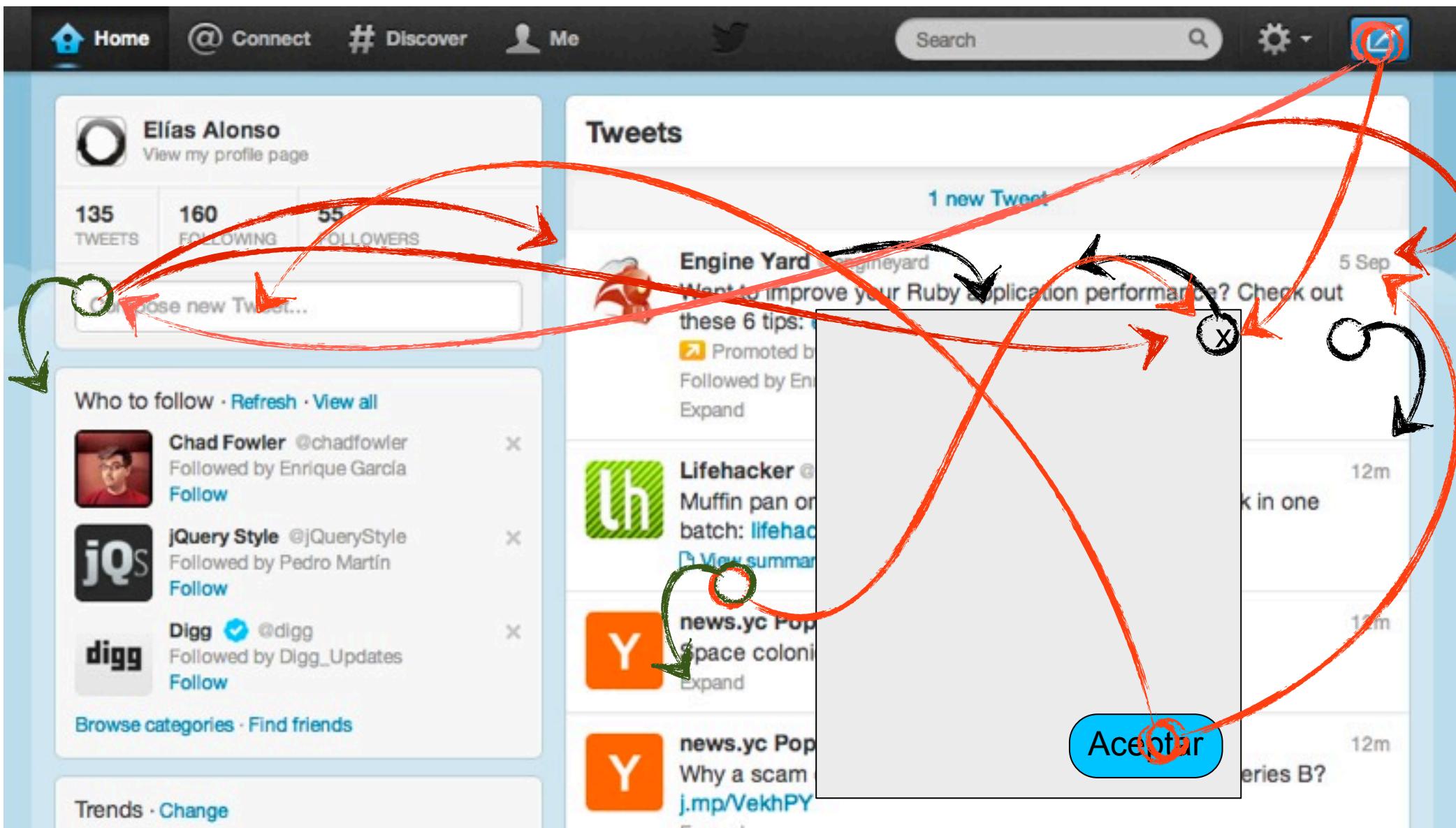
The screenshot shows a user profile page for "Elías Alonso" and a "Tweets" feed. Hand-drawn annotations include:

- A red circle highlights the "Compose new Tweet..." input field.
- Black arrows point from the "Compose new Tweet..." field to the "Tweets" feed area, indicating the flow of data from the user input to the application's main content.
- A large black box surrounds the "Tweets" feed area, representing the mediator component that manages interactions between the user interface and the data source.
- Handwritten text "1 new Tweet" is written above the tweet list.
- Annotations on the tweet from "Engine Yard" include:
 - A black arrow points from the "Compose new Tweet..." field to the tweet content.
 - A black arrow points from the tweet content back to the "Compose new Tweet..." field, forming a loop.
 - A red arrow points from the "Compose new Tweet..." field to the "X" icon at the top right of the tweet card.
 - A black arrow points from the "Compose new Tweet..." field to the "5 Sep" timestamp.
- Annotations on the tweet from "Lifehacker" include:
 - A black arrow points from the "Compose new Tweet..." field to the tweet content.
 - A black arrow points from the tweet content back to the "Compose new Tweet..." field, forming a loop.
 - A black arrow points from the "Compose new Tweet..." field to the "X" icon at the top right of the tweet card.
- Annotations on the tweet from "news.yc" include:
 - A black arrow points from the "Compose new Tweet..." field to the tweet content.
 - A black arrow points from the tweet content back to the "Compose new Tweet..." field, forming a loop.
 - A black arrow points from the "Compose new Tweet..." field to the "X" icon at the top right of the tweet card.

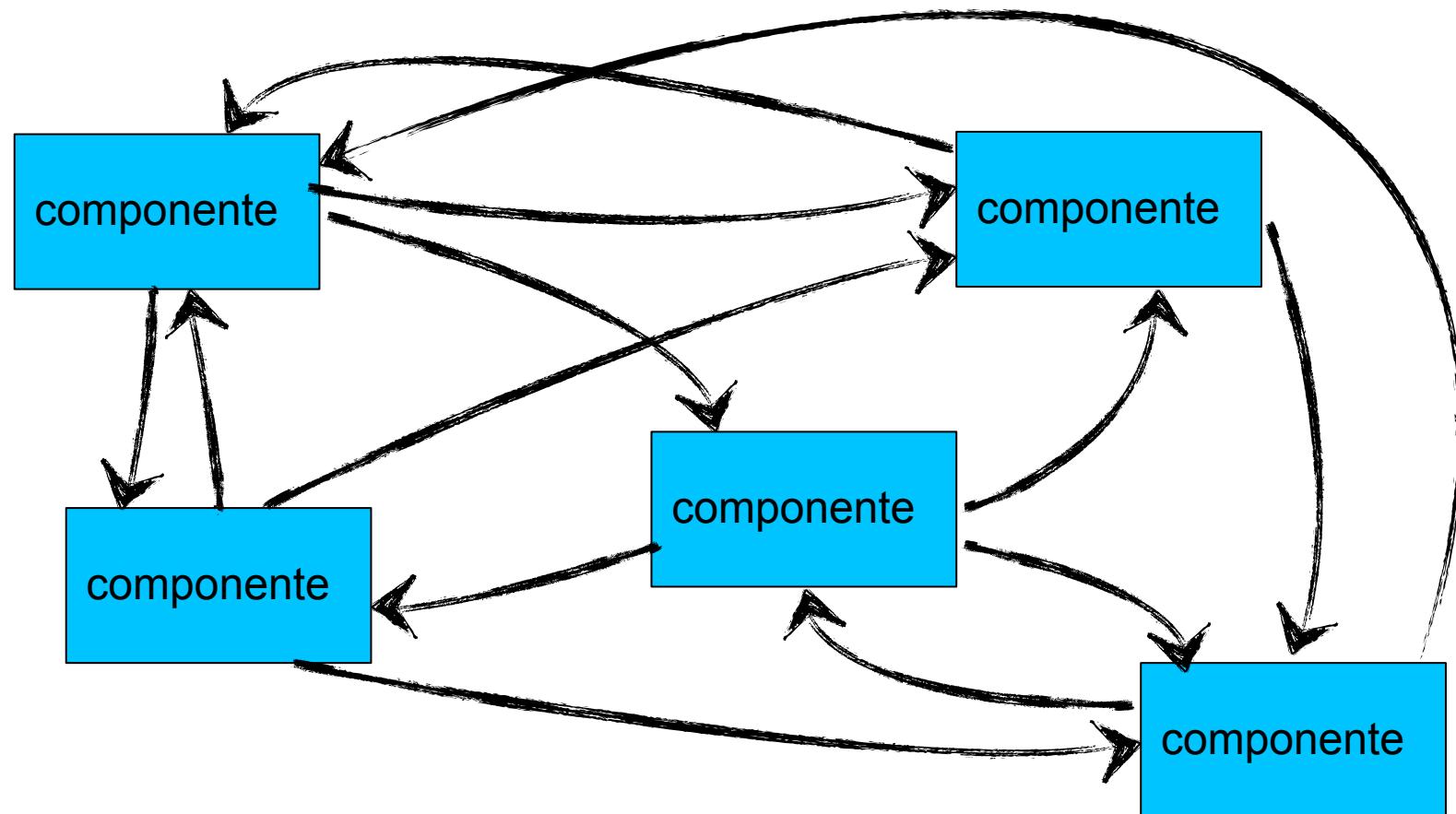
Mediator



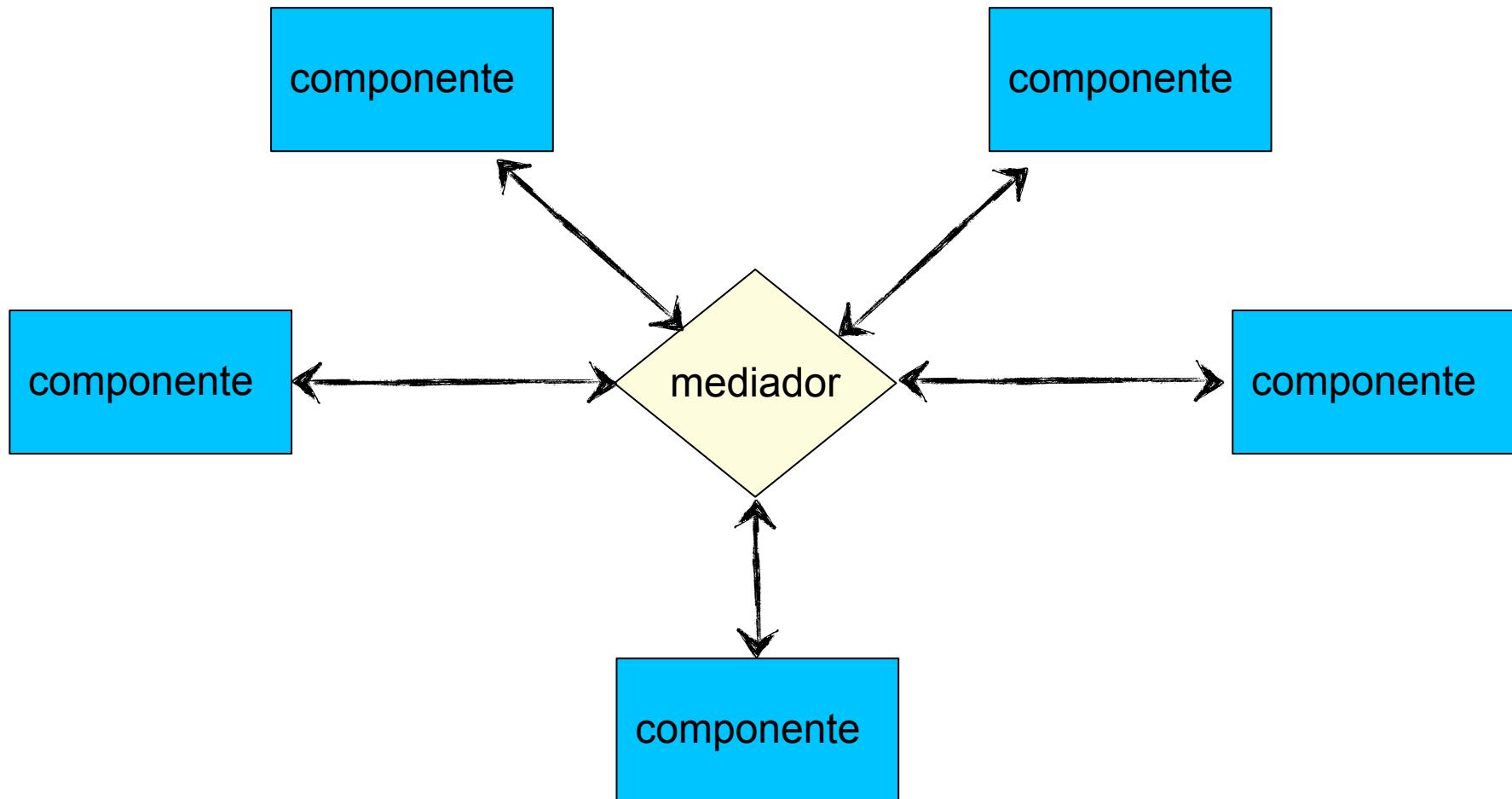
Mediator



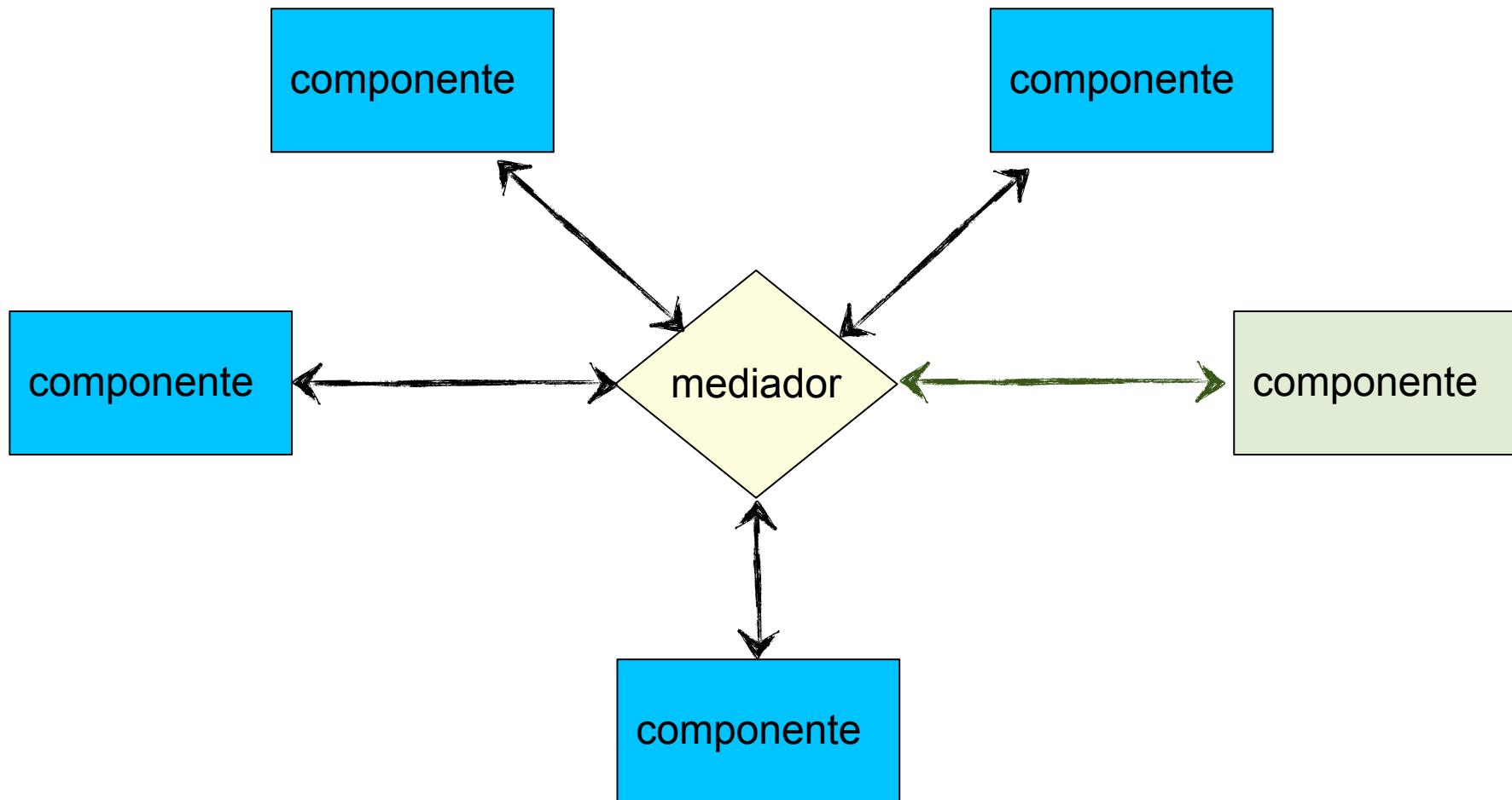
Mediator



Mediator



Mediator



Mediator

```
var Mediator = Class.extend({
    init: function(fn) {
        this.connections = {};
        if (fn) fn(this);
    },
    on: function(msg, cb) {
        this.connections[msg] = cb;
    },
    send: function(msg) {
        var args = [].slice.call(arguments, 1),
            action = this.connections[msg];
        if (action) action.apply({}, args);
    },
    addComponent: function(component) {
        component.setMediator(this);
    }
});
```

Intermedio: Mediator

- Mediador en acción:
 - tema2/mediador-1/index.html

Intermedio: Mediator

```
// Inicialización

$(function() {

    var mediator = new Mediator(),
        button1 = new Button('pulsador', '#button-1');

    mediator.addComponent(button1);
    mediator.on('pulsador:clicked', function() {
        alert("hola?");
    });

});
```

Intermedio: Mediator

```
// Inicialización

$(function() {

    var mediator = new Mediator(function(mediator) {
        var button1 = new Button('pulsador', '#button-1');

        mediator.addComponent(button1);
        mediator.on('pulsador:clicked', function() {
            alert("hola?");
        });
    });
});
```



Intermedio: Mediator

- Mediador en acción:
 - tema2/mediador-2/index.html

Intermedio: Mediator

```
mediator.on('ignore:on', function() {  
    enabled = false;  
});  
  
mediator.on('ignore:off', function() {  
    enabled = true;  
});  
  
mediator.on('inc:clicked', function() {  
    if (enabled) { count.increment(); }  
    display.update(count.getCurrentValue());  
});  
  
mediator.on('dec:clicked', function() {  
    if (enabled) { count.decrement(); }  
    display.update(count.getCurrentValue());  
});
```

Intermedio: Mediator

```
mediator.on('ignore:on', function() {  
    enabled = false;  
});
```

```
mediator.on('ignore:off', function() {  
    enabled = true;  
});
```

```
mediator.on('inc:clicked', function() {  
    if (enabled) { count.increment(); }  
    display.update(count.getCurrentValue());  
});
```

```
mediator.on('dec:clicked', function() {  
    if (enabled) { count.decrement(); }  
    display.update(count.getCurrentValue());  
});
```



Mediator

¿Cuándo usar un Mediador?

- Muchos componentes interactuando
- La interacción se puede separar del componente
- Es decir: coordinar diferentes widgets de la aplicación

Mediator

¿Cuándo usar un Mediador?

- Muchos componentes interactuando
- La interacción se puede separar del componente
- Es decir: coordinar diferentes widgets de la aplicación

¡Cuidado!

- El mediador tiende a convertirse en un cajón de sastre
- Una Gran Función que Todo lo Hace!

Comandos

Separar la preparación de una acción y su ejecución

- Flujos de ejecución no convencionales
- Tres actores: cliente, invocador, receptor
 - cliente: prepara o configura una acción
 - invocador: controla la ejecución
 - receptor: lleva a cabo la acción
- Generalizar comportamientos

Comandos

tema2/comandos-1/index.html

Comandos

```
var Command = Class.extend({
    init: function(msg) { this.msg = msg; },
    execute: function() { alert(this.msg); }
});

var ActionList = Class.extend({
    init: function(selector) { this.ul = $(selector); },
    append: function(name, command) {
        var li = $("<li>")
            .append($('<a/>', {html:name, href:'#'}))
            .click(bind(command, command.execute))
            .appendTo(this.ul);
    }
});

var list = new ActionList('#menu');
list.append('saludo', new Command("Hola!"));
list.append('achis!', new Command("Salud!"));
```

Comandos

```
var Command = Class.extend({  
    init: function(msg) { this.msg = msg; },  
    execute: function() { alert(this.msg); }  
});
```



```
var ActionList = Class.extend({  
    init: function(selector) { this.ul = $(selector); },  
    append: function(name, command) {  
        var li = $("<li>")  
            .append($('<a/>', {html:name, href:'#'}))  
            .click(bind(command, command.execute))  
            .appendTo(this.ul);  
    }  
});
```



```
var list = new ActionList('#menu');  
list.append('saludo', new Command("Hola!"));  
list.append('achis!', new Command("Salud!"));
```



Comandos

```
var AlertCommand = Class.extend({  
    init: function(msg) { this.msg = msg; },  
    execute: function() {  
        alert(this.msg);  
    }  
});
```



```
var LogCommand = Class.extend({  
    init: function(msg) { this.msg = msg; },  
    execute: function() {  
        console.log(this.msg);  
    }  
});
```



Comandos

Encapsular las acciones y controlar su ejecución

- Una idea muy potente
- Deshacer! (estados reversibles)
- Historial
- Control de cambios
- Sincronización de estados

Comandos

Un ejemplo sencillo de estado reversible:

- tema2/comandos-2/index.html

```
var Movement = Class.extend({
    init: function(amount, axis) {
        this.amount = amount;
        this.axis = axis;
    },
    execute: function(mosca) {
        mosca.move(this.amount, this.axis);
    },
    undo: function(mosca) {
        mosca.move(-this.amount, this.axis);
    }
});
```



Comandos

¿Cuándo usar Comandos?

- Controlar la ejecución de acciones
- Separar la definición de la acción y su ejecución
- Es decir:
 - Intervienen actores que pueden fallar (servidor, usuario)
 - Menús, selectores y otros “contenedores de acciones”
 - Sincronizar cambios simultáneos (control de versiones)

Cadena de Responsabilidad

Cada eslabón decide actuar o delegar en el próximo

- Elegir dinámicamente quien responde a una petición
- Dividir una operación compleja en varios pasos
- Desacoplar la complejidad del cliente

Cadena de Responsabilidad

```
var AlmacenVehiculos = Class.extend({
    init: function(tipo) {
        this.tipo = tipo;
        this.vehiculos = [];
    },
    guardar: function(vehiculo) {
        if (vehiculo.getTipo() == this.tipo) {
            this.vehiculos.push(vehiculo);
        }
    }
});

var hangar = new AlmacenVehiculos('avion'),
    parking = new AlmacenVehiculos('coche'),
    puerto = new AlmacenVehiculos('barco');
```

Cadena de Responsabilidad

```
var Vehiculo = Class.extend({
    init: function(tipo) { this.tipo = tipo; },
    getTipo: function() { return this.tipo; },
    guardar: function() {
        hangar.guardar(this);
        puerto.guardar(this);
        parking.guardar(this);
    }
});
```

Cadena de Responsabilidad

```
var AlmacenVehiculos = Class.extend({  
    init: function(tipo, siguiente) {  
        this.tipo = tipo;  
        this.vehiculos = [];  
        this.sigiente = siguiente;  
    },  
    guardar: function(vehiculo) {  
        if (vehiculo.getTipo() == this.tipo) {  
            this.vehiculos.push(vehiculo);  
        } else if (siguiente.next) {  
            this.sigiente.guardar(vehiculo);  
        }  
    }  
});
```



Cadena de Responsabilidad

```
var hangar = new AlmacenVehiculos('avion'),  
    parking = new AlmacenVehiculos('coche', hangar),  
    puerto = new AlmacenVehiculos('barco', parking),  
    almacenes = new AlmacenVehiculos('', puerto);  
  
var Vehiculo = Class.extend({  
    init: function(tipo) { this.tipo = tipo; },  
    getTipo: function() { return this.tipo; },  
    guardar: function() {  
        almacenes.guardar(this);  
    }  
});
```



Cadena de Responsabilidad

¿Cuándo usar Cadenas de Reponsabilidad?

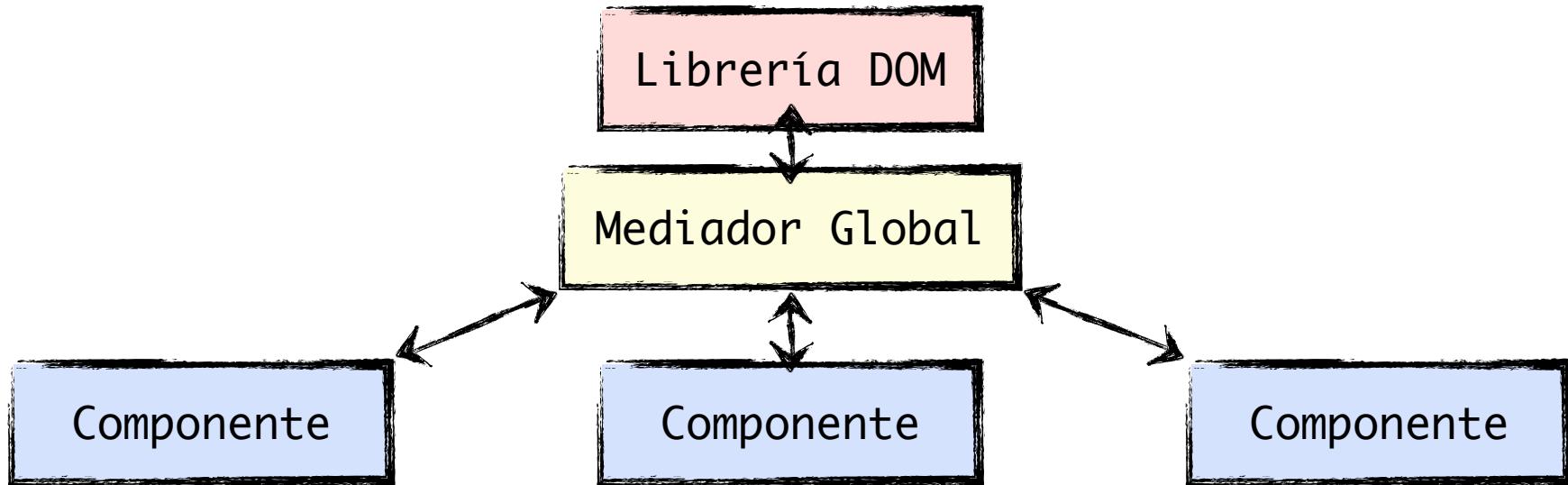
- Jerarquías de objetos
 - Los eventos del navegador
- Diferentes fuentes para responder a una petición
 - Ej: buscar por nombre, por fecha, por ciudad, ...
- El input puede afectar a varias entidades
 - Ej: Colisiones

Hydra

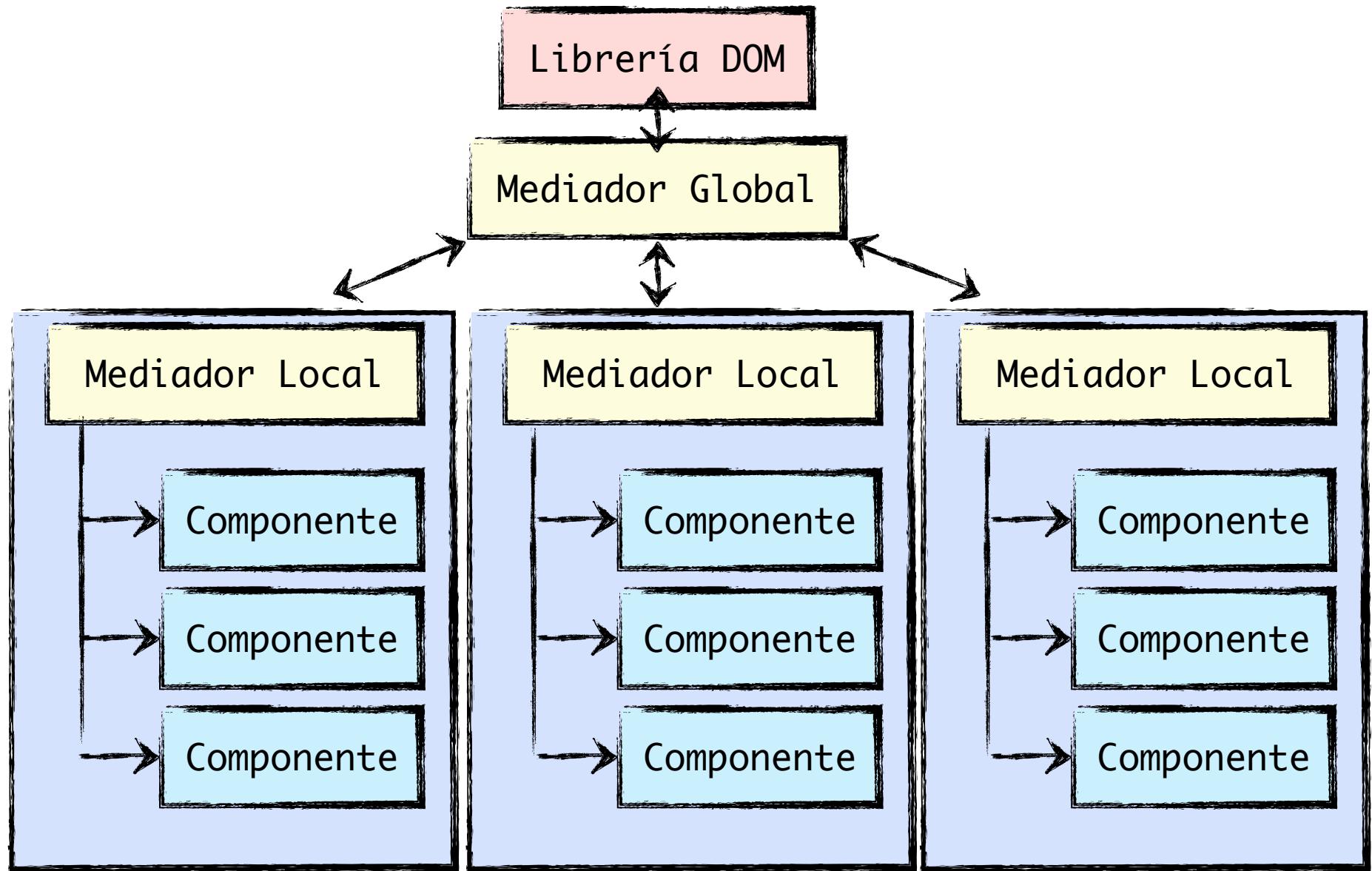
Árbol de comportamientos

- No existe
- Organización al más alto nivel
- Mediador + Comandos + Cadena de Responsabilidad
- Extremadamente escalable!
- Para aplicaciones muy grandes

Hydra



Hydra



Hydra

Librería DOM

Ajax + DOM

Mediador Global

Coordinación general

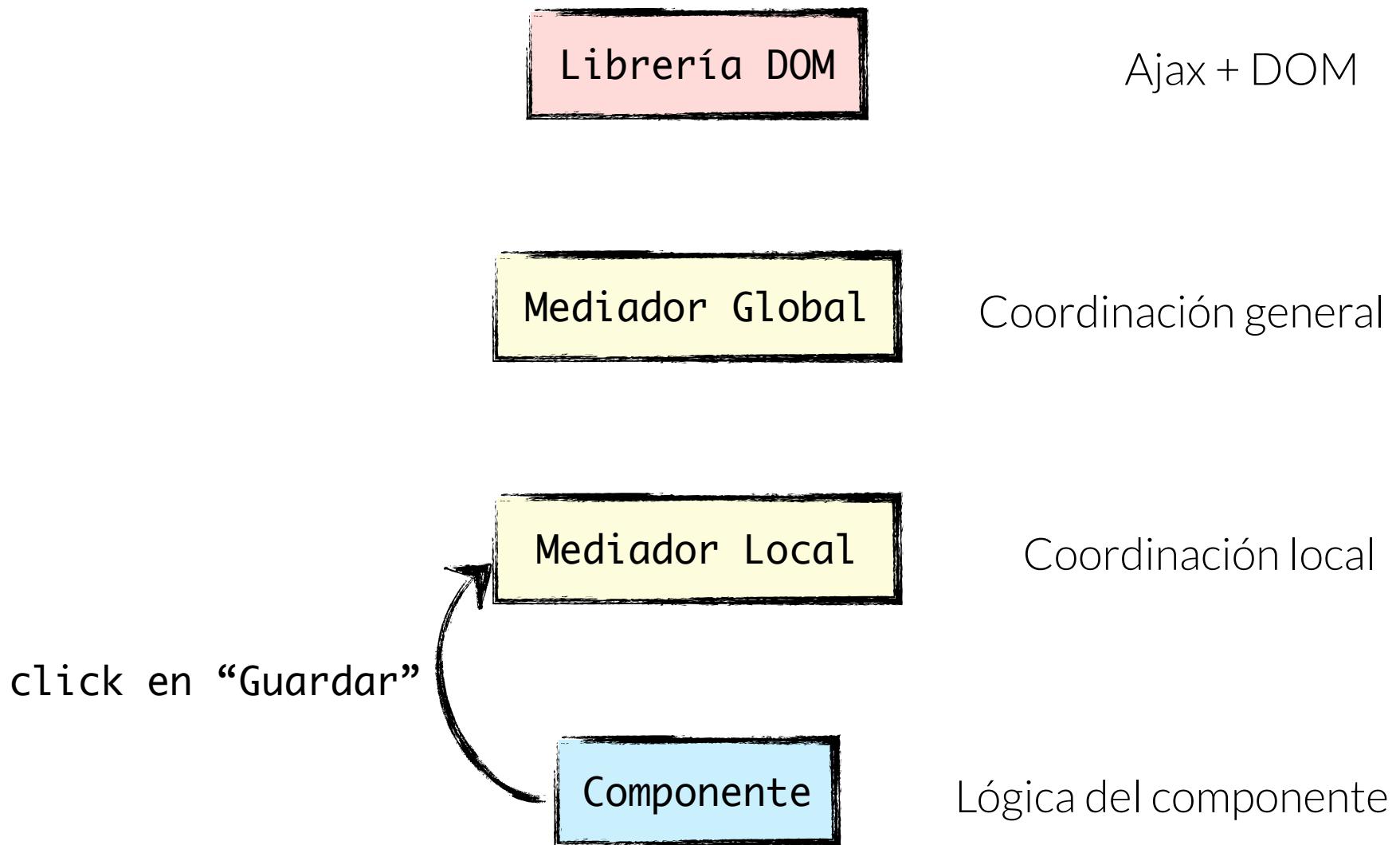
Mediador Local

Coordinación local

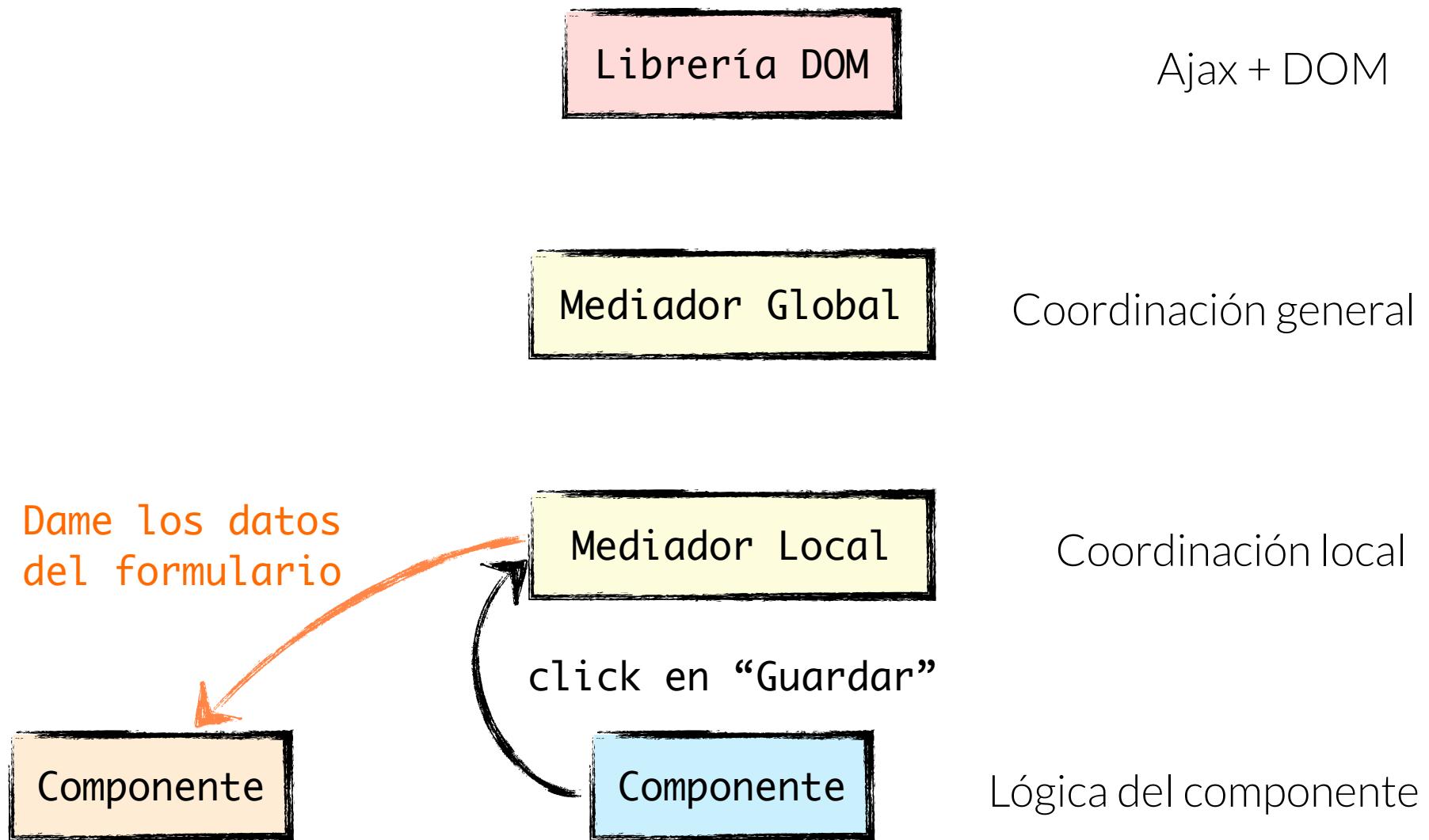
Componente

Lógica del componente

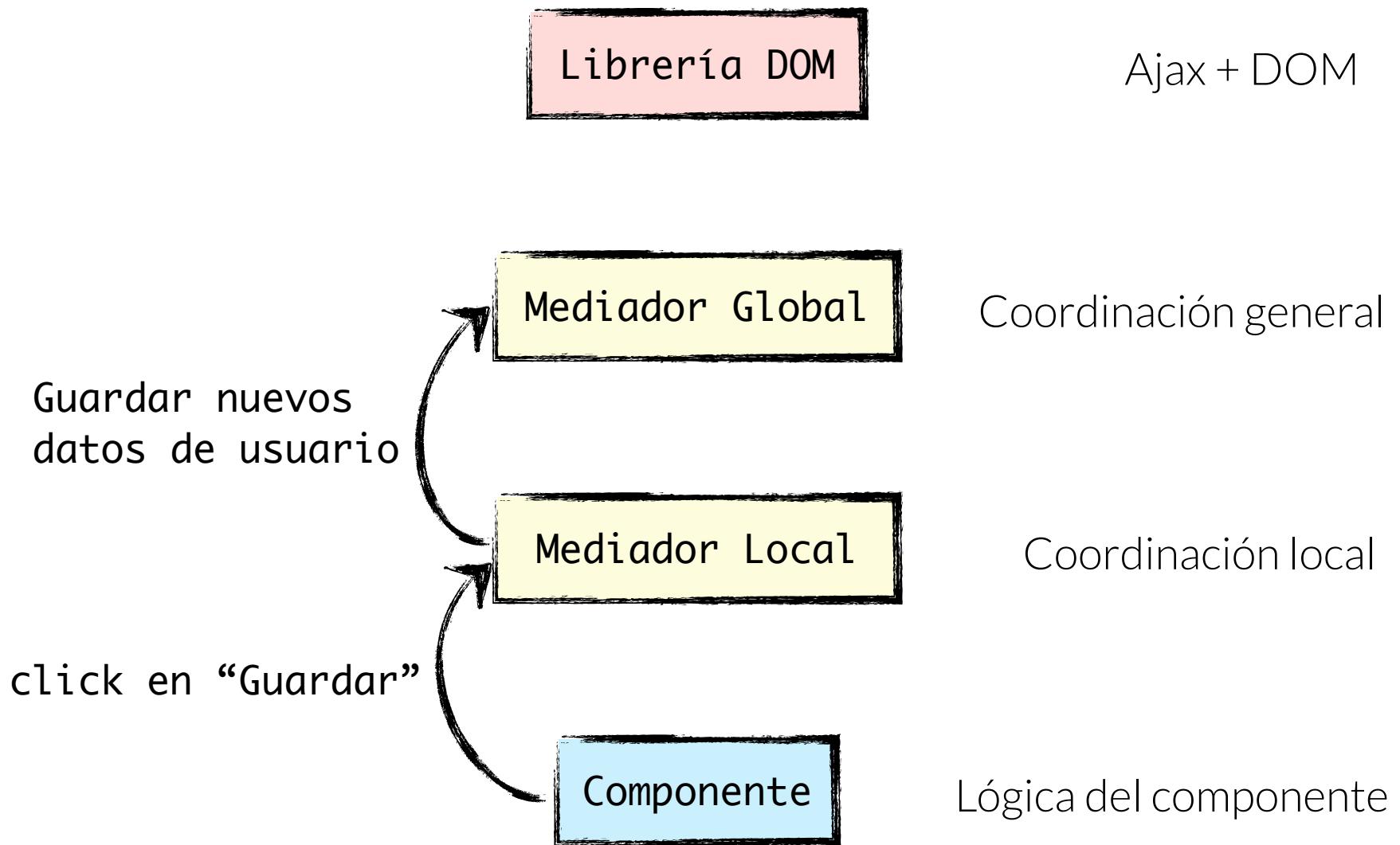
Hydra



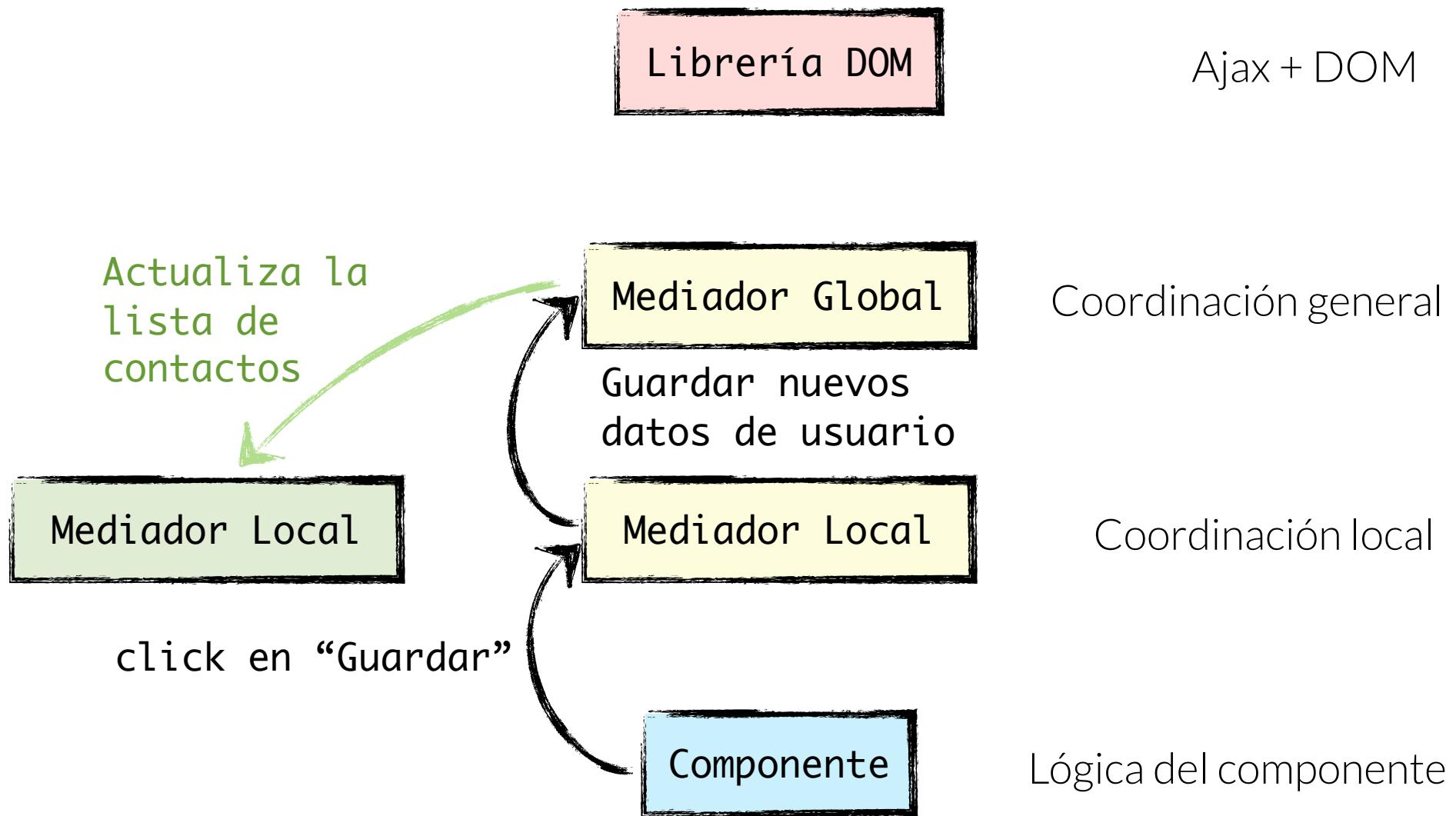
Hydra



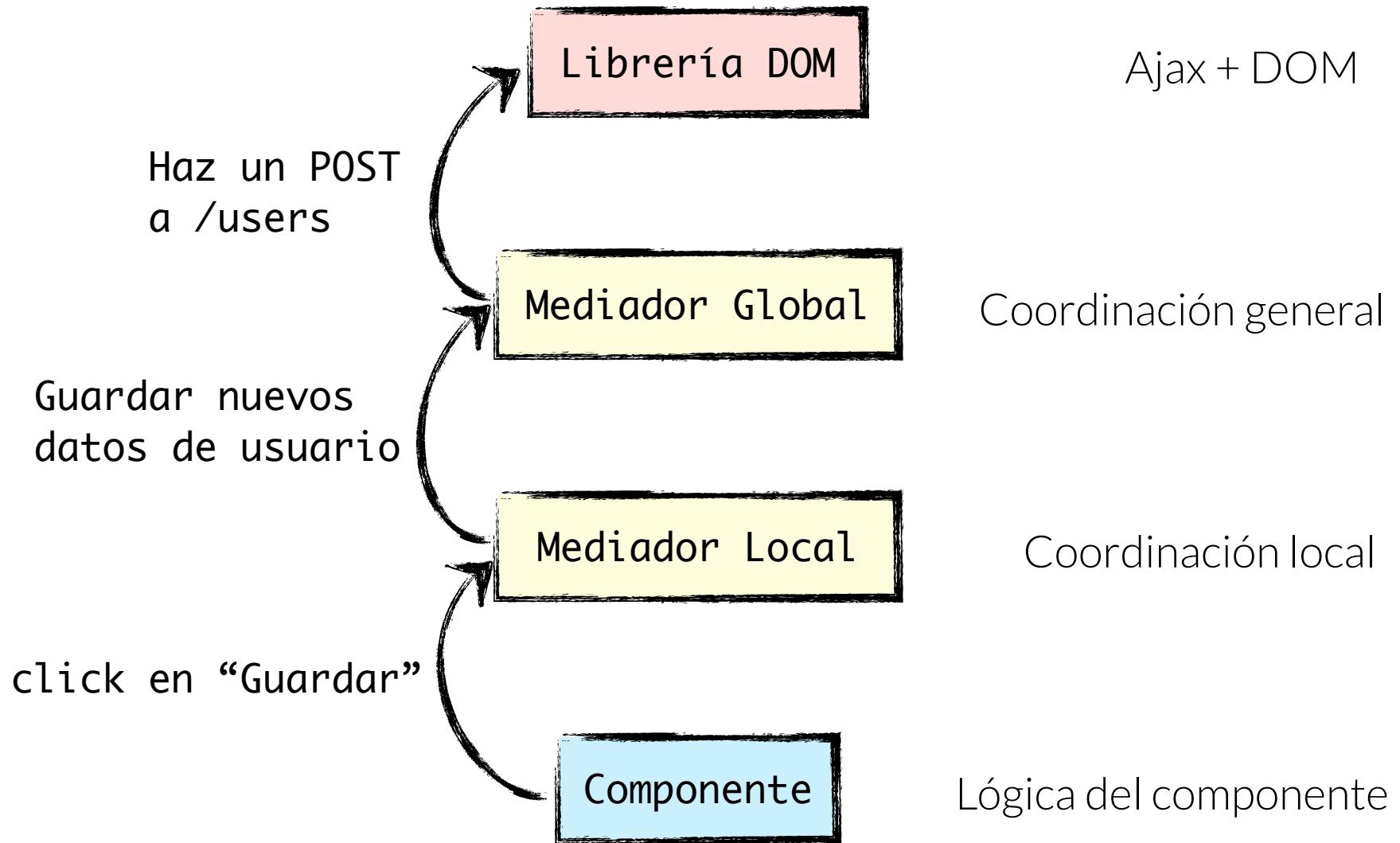
Hydra



Hydra



Hydra



Hydra

Separación por capas de significado

- Cada mediador gestiona lo que hay a su nivel
- El significado de las acciones va cobrando sentido
- La lógica de los componentes se limita a hablar con su mediador
- La coordinación es “recursiva”

Hydra

¿Cuándo utilizar Hydra?

- Aplicaciones grandes!
- Widgets complicados que se beneficien de mediación
- En general, en cuanto un mediador engorde demasiado

Hydra

¿Cuándo utilizar Hydra?

- Aplicaciones grandes!
- Widgets complicados que se beneficien de mediación
- En general, en cuanto un mediador engorde demasiado

¡Cuidado!

- El alto desacoplamiento hace complicado leer el código
- El significado TOTAL de cada acción está diseminado