Metaprogramming & JavaScript Object Proxies
What is *metaprogramming*?

Writing programs that manipulate other programs.
JavaScript Proxies

Special metaprogramming features are proposed for ECMAScript 6, code-named ECMAScript Harmony.

Proposed By:

Tom Van Cutsem

Mark Miller
Proxies: Design Principles for Robust Object-oriented Intercession APIs

Abstract: Proxies are a powerful approach to implement meta-objects in object-oriented languages without having to resort to metacircular interpretation. We introduce such a meta-level API based on proxies for Javascript...
Metaprogramming terms

• Reflection
  – **Introspection**: examine a program
  – **Self-modification**: modify a program

• **Intercession**: redefine the semantics of operations.

• Reflection is fairly common. Intercession is more unusual.
Introspection

Ability to examine the structure of a program. In JavaScript:

```
"x" in o;
for (prop in o){ ... }
```

Property lookup

Property enumeration
Self-modification

The ability to modify the structure of a program.

In JavaScript:

```javascript
o["x"]; // computed property
o.y = 42; // add a new property
delete o.x; // remove a property
o["m"].apply(o,[42]);
    // reflected method call
```
Until recently, JavaScript did not support intercession.

JavaScript proxies are intended to fix that.

But first a little history...
Metaobject Protocol History

- Common Lisp was developed before object-oriented languages were popular.
- Many libraries were created with non-standard object-oriented systems.
- The Common Lisp Object System (CLOS) was developed, but what could be done about pre-existing object-oriented libraries?
The Devil’s Choice

1. Rewrite libraries for CLOS?
   – Huge # of libraries, so not feasible to rewrite them all.

2. Make complex API?
   – Might makes API difficult to understand.
   – Systems had conflicting features…
   – …But were essentially doing the same things.
Gregor Kiczales chose option 3: Keep a Simple API, but modify object behavior to fit with different systems. *Metaobject protocols were born...*
JavaScript Object Proxies
Intercession API
WARNING!

The Proxies API is in transition. We will review the old API, since that is more widely available (including in Node.js)
Proxy design

• Proxy: A special object that has special *traps* that define its behavior.
  – Trap: method that intercepts an operation.

• Handler: The meta-object that specifies the details of the trap; the handler itself is usually a normal object.
Using proxies in Node.js

• Node.js supports proxies, but has them disabled by default.
• To run node with proxies, type:
  
  ```bash
  $ node --harmony-proxies prog.js
  ```
Two kinds of proxies

• Objects are defined with:
  Proxy.create(handler, proto);

• Functions (with extra traps) are defined with:
  Proxy.createFunction(handler, callTrap, constructTrap);

• Proxies do not exist for primitive values.

• Note: This API is in transition.
What kind of things do we want to do to an object?
Sample handler (incomplete)

var incompleteHandler = {
    get: function (myProxy, name) {
        console.log('Property ' + name + ' accessed.');
        return 1;
    }
};
Creating a new proxy object

```javascript
var p = Proxy.create(
    incompleteHandler);

var q = p.hello;

p.goodbye = "What happens here?";
```

Prints "Property hello accessed" and returns 1.

The set trap has not been specified, so this causes an error.
No-op forwarding proxy
(in-class)

This proxy wraps an object, intercepting all of its methods, and doing nothing else.

It serves as a useful base proxy.
Another use case for proxies

• We want to share a reference to an object, but do not want it to be modified.
  – Reference to the DOM, for instance

• We can modify the forwarding handler to provide this behavior:
function roHandler(obj) {
    return {
        delete: function(name) {
            return obj[name];
        },
        set: function(rcvr,name,val) {
            return true;
        },
        ...
    }
}
Aspect-oriented programming (AOP)

• Some code is not well organized by objects.
• For the canonical example, logging statements tend to be littered throughout code.
  – If you want to change your logging framework, you have to change massive amounts of code.
  – In AOP terms, logging is a *cross-cutting concern*.
• We can use proxies to build a *tracing API* and address this issue.
Proxies Lab

Modify the forwarding API to log all actions taken on an object.

With this proxy, you can debug an application by wrapping an object in a proxy and tracing its behavior.