CS 252:

Advanced Programming Language Principles

Metaprogramming & JavaScript Object Proxies

Prof. Tom Austin

San José State University

What is *metaprogramming*?

Writing programs
that manipulate
other programs.

JavaScript Proxies

Metaprogramming feature proposed for ECMAScript 6 (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

Ability to modify the structure of a program.

```
o["x"]; // computed property
o.y = 42; // add new property
delete o.x; // remove 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…

Common Lisp

- Developed before object-oriented languages were popular.
- Many libraries were created with non-standard OO systems.

Common Lisp Object System (CLOS)

- Became standard object-oriented system for Lisp
- *What could be done about pre-existing object-oriented libraries?*

The Devil's Choice

1. Rewrite libraries for CLOS?
   - huge # of libraries
   - infeasible to rewrite them all
2. Make complex API?
   - difficult API to understand.
   - Systems had conflicting features…
   - …But were essentially doing the same things.

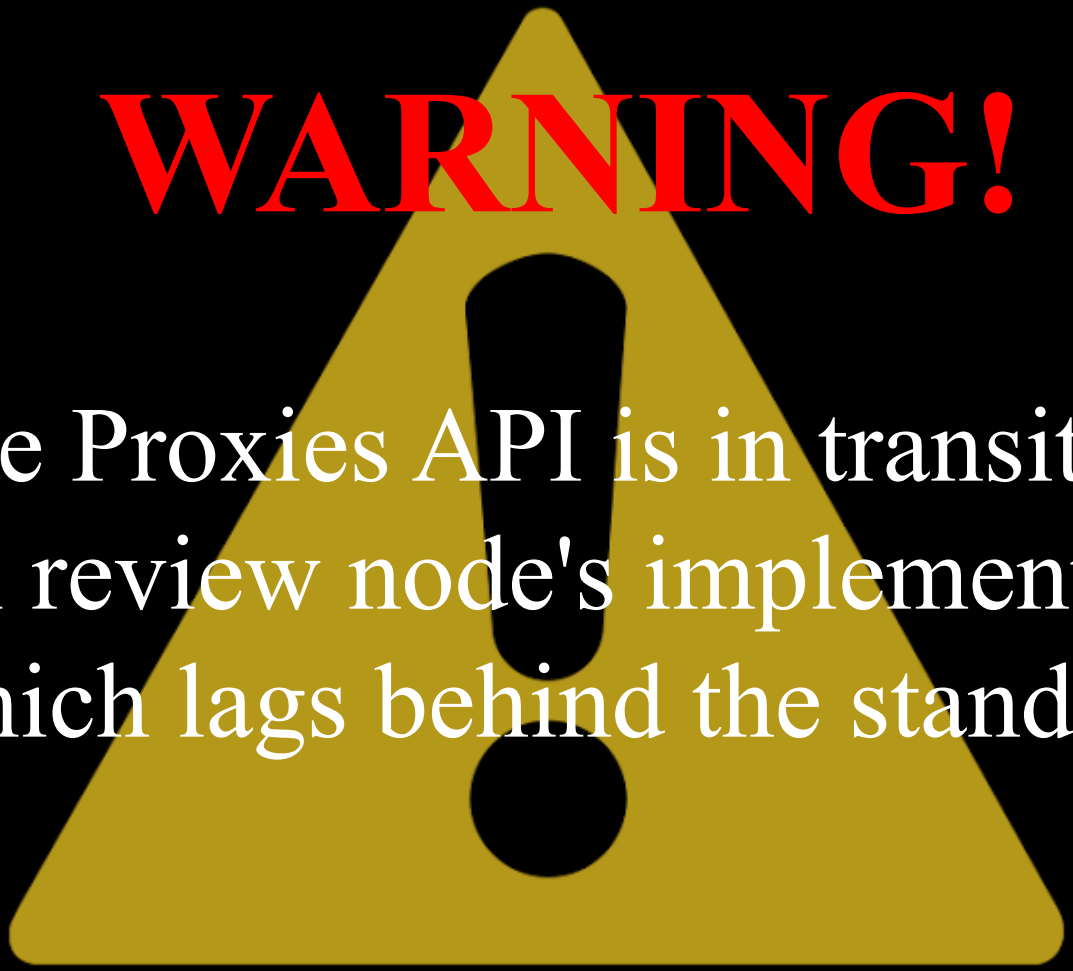
Gregor Kiczales chose option 3:

- Keep API simple.
- Modify object behavior to fit different systems.

Metaobject protocols were born…

JavaScript Object Proxies Intercession API

WARNING!



The Proxies API is in transition.
We'll review node's implementation,
which lags behind the standard.

Proxy and handler

The behavior of a *proxy* is determined by *traps* specified in its *handler*.

The metaobject

Proxy design

- Proxy: special object that has special *traps* that define its behavior.
  - Trap: method that intercepts an operation.
  - Available traps: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Old_Proxy_API
- 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 --harmony-proxies prog.js
```

Two kinds of proxies

- Objects are defined with:

```
Proxy.create(handler,proto);
```

- Functions (with extra traps) defined with:

```
Proxy.createFunction(
                handler,
                callTrap,
                constructTrap);
```

- Do not exist for primitive values.

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

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

var q = p.hello;

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

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)

Wraps an object, intercepting all of its methods, and does nothing else.

Another use case for proxies

- 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 not well organized by objects
  - *Cross-cutting concern*
- Canonical example: logging statements
  - littered throughout code
  - Swap out logger = massive code changes

Lab: Tracing API

- Use proxies to log all actions taken on an object
- Avoids having complexity of logging framework