On Contracts, Sandboxes, and Proxies for JavaScript

Matthias Keil, Peter Thiemann
University of Freiburg, Germany

Pörtschach am Wörthersee, Österreich
TreatJS

- Language embedded contract system for JavaScript
- Enforced by run-time monitoring

- Standard abstractions for higher-order-contracts (base, function, and dependent contracts) [Findler, Felleisen’02]

- Systematic blame calculation
- Contract constructors generalize dependent contracts
- Internal noninterference
Base Contracts are built from predicates

Specified by a plain JavaScript function

```javascript
function isNumber (arg) {
    return (typeof arg) === 'number';
}
var _Number_ = Contract.Base(isNumber);

assert(1, _Number_); ✓
assert('a', _Number_); ✗ blame the subject
```
// Number × Number → Number
function plus (x, y) {
    return (x + y);
}

var _Plus_ = Contract.Function([_Number_, _Number_], _Number_);

var plusNumber = assert (plus, _Plus_);

plusNumber(1, 1); ✓
plusNumber('a', 'a'); ✗ blame the context
JavaScript Proxies

```
handler.get(target, 'x', proxy);
handler.set(target, 'y', 1, proxy);
...
```

```
target['x'];
target['y']=1;
...
```

```
proxy.x;
proxy.y=1;
...
```

**Handler**

**Proxy**

**Target**

**Meta-Level**

**Base-Level**

Matthias Keil, Peter Thiemann
On Contracts, Sandboxes, and Proxies
October 5, 2015
Noninterference

The contract system should not interfere with the execution of application code.

- **External Noninterference** arises from the interaction of the contract system with the host program
  1. Exceptions
  2. Object equality

- **Internal Noninterference** arises from executing unrestricted code in predicates
Internal Noninterference

- No syntactic restrictions on predicates
- Predicates may try to write to data that is visible to the application
- **Solution**: Predicate evaluation takes place in a sandbox

### Faulty Predicate

```javascript
function isNumber (arg) {
    let type = (typeof arg); // access forbidden
    return type === 'number';
}
```

Matthias Keil, Peter Thiemann

On Contracts, Sandboxes, and Proxies

October 5, 2015 7 / 18
All contracts guarantee noninterference
Read-only access is safe

Predicate with Read-Access

```javascript
function isArray (arg) {
    return (arg instanceof Array); ✓
}
```
Predicate execution may violate contracts

**Solution:** Sandbox redefines the responsibility

```javascript
function addOne (arg) {
    return plusNumber(arg, '1'); // blame the contract
}
```
Faulty Predicate

```javascript
function isNumber (arg) {
    type = (typeof arg);
    return type === 'number';
}
```

1. Place a write protection on objects (e.g. `this`, `arg`)
2. Remove external bindings of functions (e.g. `type`)
Identity Preserving Membrane

ProxyA

ProxyB

ProxyC

TargetA

TargetB

TargetC

Matthias Keil, Peter Thiemann
Shadow Objects

handler.get(target, ’x’, proxy);
handler.set(target, ’y’, 1, proxy);

proxy.x;
proxy.y=1;
target[’x’];
target[’y’]=1;
Shadow Objects

```
handler.get(target, 'x', proxy);
handler.set(target, 'y', 1, proxy);
handler.get(target, 'y', proxy);
```

```
target['x'];
shadow['y'] = 1;
shadow['y'];
```

`proxy.x; proxy.y = 1; proxy.y;`
```javascript
var x = 1;

function f (){
    function g (y) {
        var z = 1;
        return x+y+z;
    }
}
```
var x = 1;

function f (){

  "function g (y) {
  var z = 1;
  return x+y+z;
  }"

}
Function Recompilation

```javascript
var x = 1;

with (sbxglobal)
{
    eval("function g (y) {
        var z = 1;
        return x+y+z;
    } ");
}
```

Matthias Keil, Peter Thiemann
On Contracts, Sandboxes, and Proxies
October 5, 2015 13 / 18
Function Recompilation

```javascript
var x = 1;

with(sbxglobal){
    function g (y) {
        var z = 1;
        return x+y+z;
    }
}
```
Conclusion

TreatJS:
- Language embedded, dynamic, higher-order contract system for full JavaScript
- Guarantees noninterference

Sandbox:
- Language embedded sandbox for full JavaScript
- Runs JavaScript code in isolation