On Contracts and Sandboxes for JavaScript

Matthias Keil, Peter Thiemann
University of Freiburg, Germany

August 6, 2015
Darmstadt, Germany

89.8 %

of all web sites use JavaScript[1]

[1] according to `http://w3techs.com/`, status of July 2015

Motivation

89.8 %

of all web sites use JavaScript[1]

- Most important client-side language for web sites
- Web-developers rely on third-party libraries
  - e.g. for calendars, maps, social networks

[1] according to `http://w3techs.com/`, status of July 2015

Situation of a Web-developer

Situation of a Web-developer

Situation of a Web-developer

Situation of a Web-developer

Situation of a Web-developer

Situation of a Web-developer

JavaScript issues

- Dynamic programming language
  - Code is accumulated by dynamic loading
  - e.g. eval, mashups

JavaScript issues

- Dynamic programming language
  - Code is accumulated by dynamic loading
  - e.g. eval, mashups
- JavaScript has no security awareness
  - No namespace or encapsulation management
  - Global scope for variables/ functions
  - All scripts have the same authority

JavaScript issues

- Dynamic programming language
  - Code is accumulated by dynamic loading
  - e.g. eval, mashups
- JavaScript has no security awareness
  - No namespace or encapsulation management
  - Global scope for variables/ functions
  - All scripts have the same authority

Problems

1. Side effects may cause unexpected behavior
2. Program understanding and maintenance is difficult
3. Libraries may get access to sensitive data
4. User code may be prone to injection attacks

Key challenges of present research

- All-or-nothing choice when including code
- Isolation guarantees noninterference
- Some scripts must have access the application state or are allowed to change it

Goals

1. Manage untrusted JavaScript Code
2. Control the use of data by included scripts
3. Reason about effects of included scripts

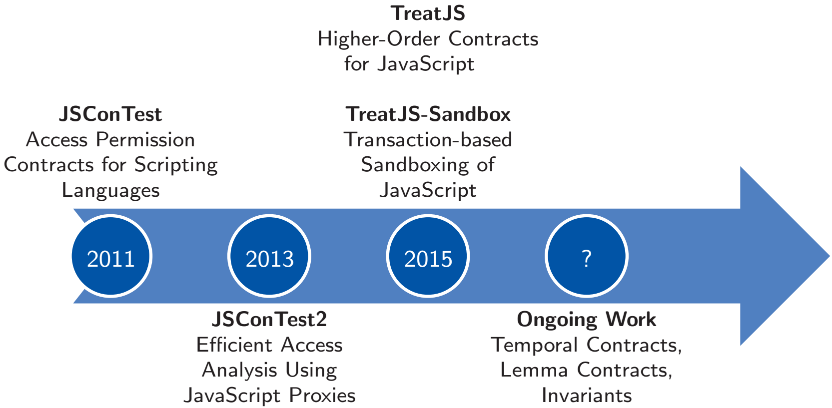
Language-embedded Systems

Shortcomings

- Static verifiers are imprecise because of JavaScript's dynamic features or need to restrict JavaScript's dynamic features
- Interpreter modifications guarantee full observability but need to be implemented in all existing engines

- Implemented as a library in JavaScript
- Library can easily be included in existing projects
- All aspects are accessible thought an API
- No source code transformation or change in the JavaScript run-time system is required

Timeline


TreatJS
Higher-Order Contracts for JavaScript
JSConTest
Access Permission Contracts for Scripting Languages
TreatJS-Sandbox
Transaction-based Sandboxing of JavaScript
2011
2013
2015
?
JSConTest2
Efficient Access Analysis Using JavaScript Proxies
Ongoing Work
Temporal Contracts, Lemma Contracts, Invariants

JSConTest

Access Permission Contracts for Scripting Languages

JSConTest

- Investigate effects of unfamiliar function
- Type and effect contracts with run-time checking
- Summarizes observed access traces to a concise description
- Effect contracts specifying allowed access paths

Type and effect contracts

```
/*c (obj, obj) -> any with [x.b,y.a] */
function f(x, y) {
  y.a = 1;
  y.b = 2; ✗ violation
}
```

Shortcomings of JSConTest

- Implemented by an offline code transformation
  - Partial interposition (dynamic code, *eval*, **with**, ...)
  - Tied to a particular version of JavaScript
  - Transformation hard to maintain
- Special contract syntax
  - Requires a special JavaScript parser
- Efficiency issues
  - Naive representation of access paths
  - Wastes memory and impedes scalability

JSConTest2

Efficient Access Analysis Using JavaScript Proxies

JSConTest2

Redesign and reimplementation of JSConTest based on JavaScript proxies

Advantages

- Full interposition for the full language
  - Including dynamically loaded code and `eval`
- Safe for future language extensions
  - No transformation to maintain
- Runs faster in less memory
  - Efficient representation of access paths
  - Incremental path matching
- Maintenance is simplified
  - No custom syntax for contracts

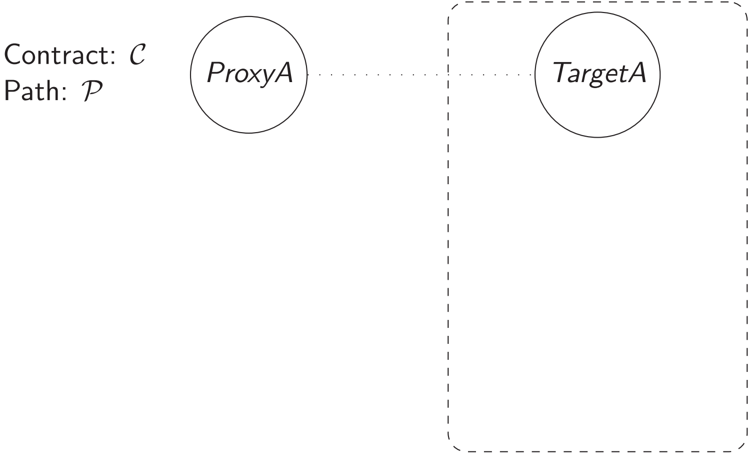
JSConTest2

Contracts on Objects

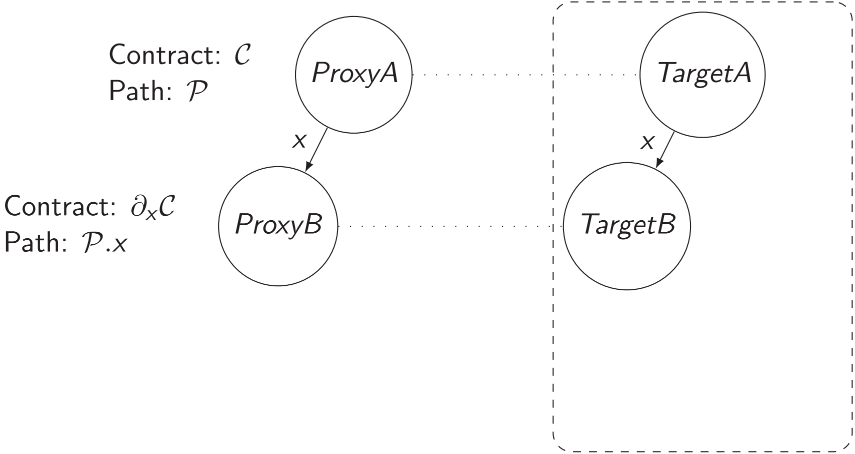
$\textbf{var}\ obj = APC.permit('(a.?+b*)', \{a:\{b:5\}, b:\{b:11\}\});$
$a = obj.a;\ //\ APC.permit('?', \{b:5\});$
$a.b = 3;$

- *APC* encapsulates JSConTest2
- *permit* wraps an object with a permission. Arguments:
  1. Permission encoded in a string
  2. Object that is protected by the permission
- Contract specifies permitted access paths
  - Last property is readable/ writeable
  - Prefix is read-only
  - Not addressed properties are neither readable nor writeable
  - Read-only paths possible (@ denotes a non-existing property)

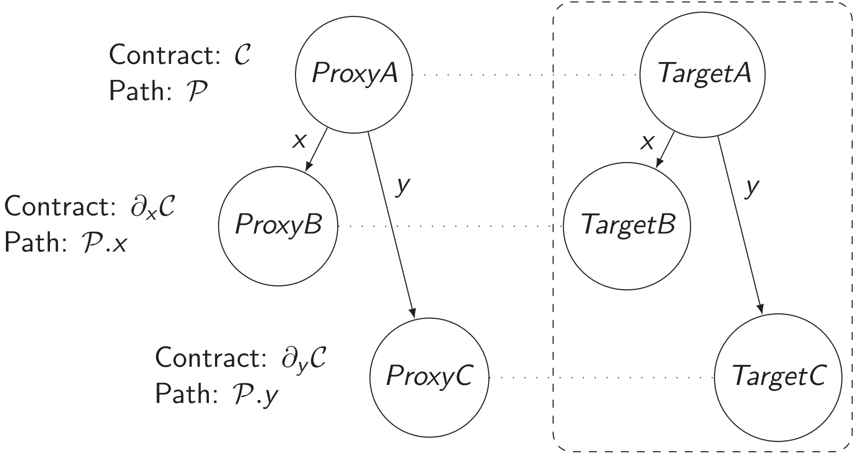
Proxy Membrane



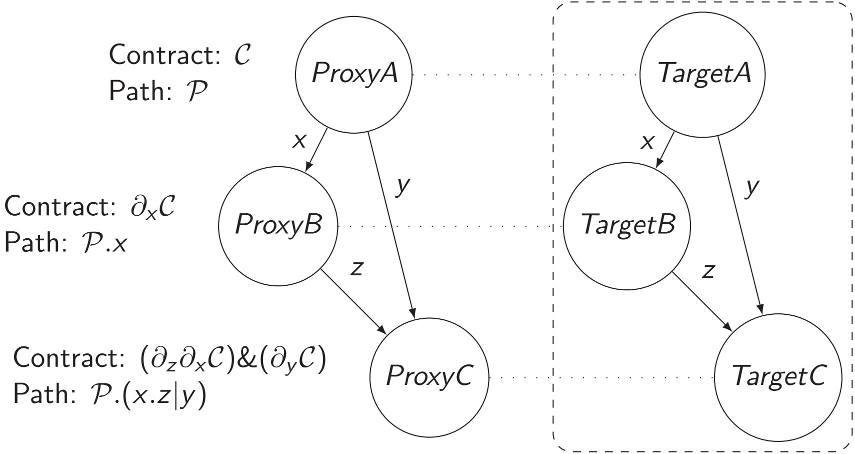
Proxy Membrane



Proxy Membrane



Proxy Membrane



The JSConTest2 Approach

- Implementation based on the JavaScript Proxy API
- Shortcomings of previous, translation-based implementation avoided

- Full interposition of contracted objects
  - Proxy intercepts all operations
  - Proxy-handler contains a contract and a path set
  - Forwards the operation or signals a violation
- Returned object contains the remaining contract (*Membrane*)

- Access contracts are regular expressions on literals
  - Each literal defines a property access
  - The language defines a set of permitted access paths

TreatJS

Higher-Order Contracts for JavaScript

Introduction

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

- Specifies the interface of a software component
- Pre- and postconditions

- Standard abstractions for higher-order-contracts (base, function, and dependent contracts) [Findler,Felleisen'02]
- Systematic blame calculation
- Side-effect free contract execution
- Contract constructors generalize dependent contracts

Base Contract [Findler,Felleisen'02]

- *Base Contracts* are built from predicates
- Specified by a plain JavaScript function

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

assert(1, _Number_); ✓
assert('a', _Number_); ✗ blame the subject
```

- Subject v gets blamed for *Base Contract* $\mathcal{B}$ iff: $\mathcal{B}(v) \neq \mathit{true}$

Function Contract [Findler,Felleisen'02]

```
// Number × Number → Number
function plus (x, y) {
  return (x + y);
}
```

Function Contract [Findler,Felleisen'02]

```
// Number × Number → Number
function plus (x, y) {
  return (x + y);
}

var plus = assert(plus, Contract.Function([_Number_,
    _Number_], _Number_));
```

Function Contract [Findler,Felleisen'02]

// *Number* × *Number* → *Number*
function *plus* (*x*, *y*) {
  return (*x* + *y*);
}

plus('a', 'a'); ✗ *blame the context*

- Context gets *blamed* for $\mathcal{C} \to \mathcal{C}'$ iff:
  Argument x gets *blamed* for $\mathcal{C}$ (as subject)

Function Contract [Findler,Felleisen'02]

```
// Number × Number → Number
function plusBroken (x, y) {
  return (x>0 && y>0) ? (x + y) : 'Error';
}
```

plusBroken(0, 1); ✗ *blame the subject*

- Subject f gets *blamed* for $\mathcal{C} \rightarrow \mathcal{C}'$ iff:
  $\neg$ (Context gets *blamed* $\mathcal{C}$) $\wedge$ ($f(x)$ gets *blamed* $\mathcal{C}'$)

TreatJS

New!

Overloaded Operator

- Function *plus* works for strings, too
- Requires to model overloading and multiple inheritances

```
// Number × Number → Number
function plus (x, y) {
  return (x + y);
}

plus('a', 'a'); ✗ blame the context
```

Combinations of Contracts

- No support for arbitrary combination of contracts
- Racket supports `and/c` and `or/c`
- Attempt to extend conjunction and disjunction to higher-oder contracts

- `and/c` tests any contract
- no value fulfills Number and String at the same time

$(and/c\ (Number \times Number \rightarrow Number)\ (String \times String \rightarrow String))$

function *plus* (x, y) {
  return (x + y);
}

plus('a', 'a'); ✗ *blame the context*

Combinations of Contracts
or/c

- or/c checks first-order parts and fails unless exactly one (range) contract remains
- Work for disjoint base contracts
- No combination of higher-oder contracts
- No support for arbitrary combinations of contracts

$(or/c\ (Number \times Number \rightarrow Number)\ (String \times String \rightarrow String))$
function *plus* (*x*, *y*) {
  return ($x + y$);
}

plus('a', 'a'); ✓

- Support for arbitrary combination of contracts
  - Combination of base and function contracts
  - Combination of function contracts with a different arity
- Intersection and union contracts
- Boolean combination of contracts

Intersection Contract

```
// (Number × Number → Number) ∩ (String × String → String)
function plus (x, y) {
  return (x + y);
}
```

Intersection Contract

```
// (Number × Number → Number) ∩ (String × String → String)
function plus (x, y) {
  return (x + y);
}

var plus = assert(plus, Contract.Intersection(
  Contract.Function([_Number_, _Number_], _Number_)
  Contract.Function([_String_, _String_], _String_));
```

Intersection Contract

// (Number × Number → Number) ∩ (String × String → String)

```
function plus (x, y) {
  return (x + y);
}
```

plus(true, true); ✗ blame the context

- Context gets <u>*blamed*</u> for $\mathcal{C} \cap \mathcal{C}'$ iff:
  (Context gets *blamed* for $\mathcal{C}$) $\wedge$ (Context gets *blamed* for $\mathcal{C}'$)

Intersection Contract

```
// (Number × Number → Number) ∩ (String × String → String)
function plusBroken (x, y) {
  return (x>0 && y>0) ? (x + y) : 'Error';
}
```

plusBroken(0, 1); ✗ *blame the subject*

- Subject f gets *blamed* for $\mathcal{C} \cap \mathcal{C}'$ iff:
  (f gets *blamed* for $\mathcal{C}$) $\vee$ (f gets *blamed* for $\mathcal{C}'$)

Contract Assertion

- A failing contract must not signal a violation immediately
- Violation depends on combinations of failures in different sub-contracts

```
// (Number → Number) ∩ (String → String)
function addOne (x) {
  return (x + 1);
}

addOne('a');
```

Contract Assertion

- A failing contract must not signal a violation immediately
- Violation depends on combinations of failures in different sub-contracts

```
// (Number → Number) ∩ (String → String)
function addOne (x) {
  return (x + 1);
}

addOne('a');
```

Contract Assertion

- A failing contract must not signal a violation immediately
- Violation depends on combinations of failures in different sub-contracts

```
// (Number → Number) ∩ (String → String)
function addOne (x) {
  return (x + 1);
}
```

addOne('a'); ✓

Blame Calculation

- Contract assertion must connect each contract with the enclosing operations
- *Callback* implements a constraint and links each contracts to its next enclosing operation
- Reports a record containing two fields, *context* and *subject*
- Fields range over $\mathbb{B}_4 = \{\perp, \mathsf{f}, \mathsf{t}, \top\}$ [Belnap'1977]

Non-Interference

- No syntactic restrictions on predicates
- Problem: Contract may interfere with program execution
- Solution: Predicate evaluation takes place in a sandbox

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

var _Number_ = Contract.Base(isNumber);
```

Non-Interference

- No syntactic restrictions on predicates
- Problem: Contract may interfere with program execution
- Solution: Predicate evaluation takes place in a sandbox

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

var _Number_ = Contract.Base(isNumber);

assert(1, _Number_);
```

Sandbox

- All contracts guarantee noninterference
- Read-only access is safe

```
var _Array_ = Contract.Base(function (arg) {
  return (arg instanceof Array); ✗ access forbidden
});
```

Sandbox

- All contracts guarantee noninterference
- Read-only access is safe

var *_Array_* = **Contract**.**Base**(**function** *(arg)* {
  return *(arg instanceof OutsideArray);* ✓
});

var *_Array_* = **Contract**.**With**({*OutsideArray:Array*}, *_Array_*);

Contract Constructor

- Building block for dependent, parameterized, abstract, and recursive contracts
- Constructor gets evaluated in a sandbox, like a predicate
- Returns a contract
- No further sandboxing for predicates

```
var __Type__ = Contract.Constructor(function (type) {
  return Contract.Base(function (arg) {
    return (typeof arg) === type;
  });
});

var _Number_ = __Type__('number');
```

TreatJS-Sandbox

Transaction-based Sandboxing of JavaScript

TreatJS-Sandbox

- Language-embedded sandbox for full JavaScript
- Inspired by JSConTest2 and Revocable References
- Adapts SpiderMonkey's compartment concept to run code in isolation to the application state
- Provides features known from transaction processing in database systems and transactional memory

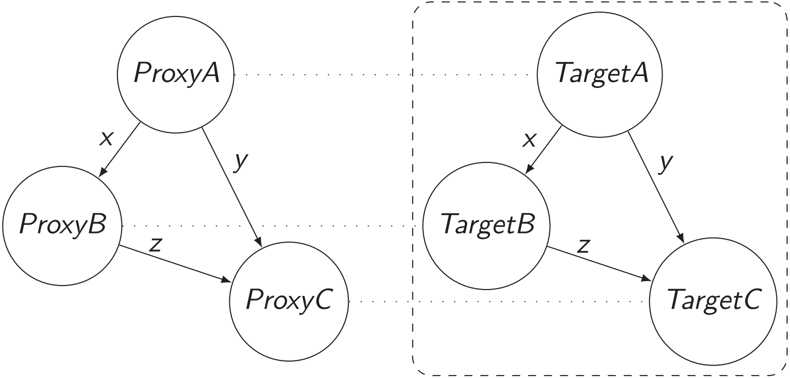
Sandbox Encapsulation

- A reference is the right to access an object
- Requires to control property read and property write

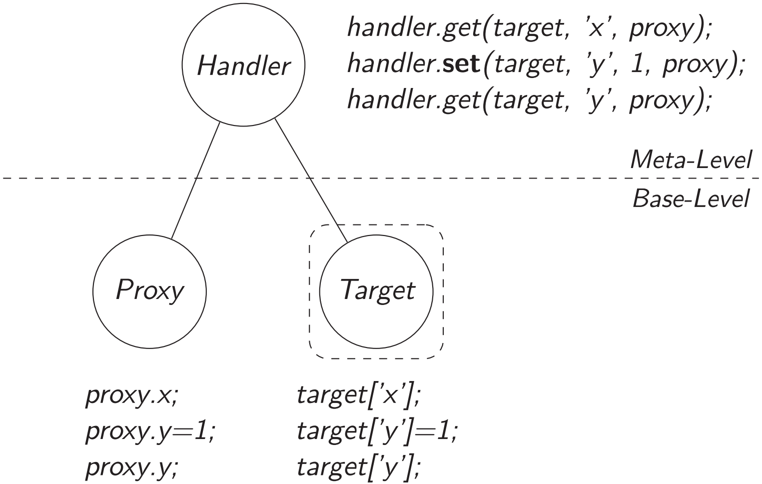
Sandbox Encapsulation

1. Place a write protection on objects
2. Remove external bindings of functions

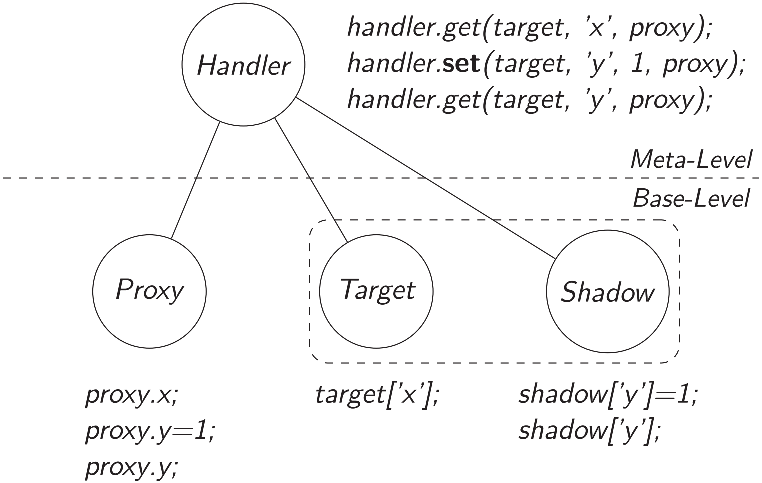
Identity Preserving Membrane



JavaScript Proxies



Shadow Objects



Function Recompilation

- Function decompilation uses the **Function**.*prototype.toString* method to return a string that contains the source code of that function
- Applying *eval* to the string creates a fresh variant
- A **with** statement places a proxy in top of the scope chain
- The *hasOwnProperty* trap always returns true

JavaScript Scope Chain

```
var x = 1;

  function f (y){

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

  }
```

Sandbox Scope Chain

```
var x = 1;

  with(sbxglobal){

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

  }
```

Conclusion

- JSConTest/ JSConTest2: Effect monitoring for JavaScript
- Enables to specify effects using access permission contracts

- TreatJS: Language embedded, dynamic, higher-order contract system for full JavaScript
- Support for intersection and union contracts
- Contract constructors with local scope

- Sandbox: Language embedded sandbox for full JavaScript
- Runs code in a configurable degree of isolation
- Provides a transactional scope

Ongoing Work

- Temporal/ Computation Contracts
- Lemma Contracts
- Invariants
- Different blaming semantics (Lax, Picky, Indy)

Further Challenges

Limitations

- Dynamic contract checking impacts the execution time
- Arbitrary combinations of contracts lead to unprecise error messages

1. Hybrid contract checking
2. Static pre-checking of contracts
3. Optimization, contract rewriting

Appendix

JSConTest

Access Permission Contracts for Scripting Languages

JSConTest2

Efficient Access Analysis Using JavaScript Proxies

JSConTest2

Redesign and reimplementation of JSConTest based on JavaScript proxies

JSConTest2

Redesign and reimplementation of JSConTest based on JavaScript proxies

Advantages

- Full interposition for the full language
  - Including dynamically loaded code and `eval`
- Safe for future language extensions
  - No transformation to maintain
- Runs faster in less memory
  - Efficient representation of access paths
  - Incremental path matching
- Maintenance is simplified
  - No custom syntax for contracts

Contracts on Functions

Contracts on Functions

var *func* = *APC.permitArgs('arguments.0.(a.?+b∗)',*
function(*arg0*) {
  // do something
});

- *permitArg* wraps a function with permissions
  1. contract applied to function arguments
  2. function
- Arguments accessed by position *arguments.0*
  - No reliable way to access parameter names
  - Function may use unlisted parameters
  - Parameter names may not be unique

Interaction of Contracts

Interaction of Contracts

var *obj* = *APC.permit('((a+a.b)+b.b.@)', {a:{b:3}, b:{b:5}});*
obj.a = obj.b; // APC.permit('b.@', {b:5});
y = obj.a; // APC.permit('b & b.@', {b:5});
y.b = 7; ✗ *violation*

- Line 2 reads *obj.b* and writes *obj.a*
- Afterwards, *obj.b* and *obj.a* are aliases
- JSConTest2 enforces *both* contracts reaching *obj.b* and *obj.a*
- *obj.a* carries contract $!'(\epsilon+b)\&b.@'! = !'b.@'!$
- Thus, writing to *obj.a.b* is not permitted

Limitations

1. Cannot directly protect DOM objects because of the browser's sandbox
2. Proxies are not transparent with respect to equality, i.e. for distinct proxies == and === returns false, even if the target object is the same

TreatJS

Higher-Order Contracts for JavaScript

Object Contract

- Defined by a mapping from property names to contracts
- Represented as a pair of a *getter* and a *setter* function
- *Getter* and *Setter* apply the property's contract

var *arraySpec* = **Contract**.**AObject**(*{length:_Number_}*);

Object Contract

- Defined by a mapping from property names to contracts
- Represented as a pair of a *getter* and a *setter* function
- *Getter* and *Setter* apply the property's contract

var *arraySpec* = **Contract**.**AObject**(*{length:_Number_}*);

var *faultyObj* = **assert**(*{length:'1'}, arraySpec*);

Object Contract

- Defined by a mapping from property names to contracts
- Represented as a pair of a *getter* and a *setter* function
- *Getter* and *Setter* apply the property's contract

var *arraySpec* = **Contract.AObject**(*{length:_Number_}*);

var *faultyObj* = **assert**(*{length:'1'}, arraySpec*);

var *faultyObj.length;* ✗ *blame the subject*

Object Contract

- Defined by a mapping from property names to contracts
- Represented as a pair of a *getter* and a *setter* function
- *Getter* and *Setter* apply the property's contract

var *arraySpec* = **Contract.AObject**(*{length:_Number_}*);

var *faultyObj* = **assert**(*{length:'1'}*, *arraySpec*);

var *faultyObj.length* = *'2'*; ✗ *blame the context*

Function Contract

- *Function Contract* is build from one contract for the domain and one contract for the range of a function
- An *Object Contract* serves as the domain portion of a *Function Contract* with zero or more arguments
- **AFunction** defines an *Object Contract* from the array

Contract.**AFunction**(*[_Number_, _Number_], _Number_));*

Function Contract

- *Function Contract* is build from one contract for the domain and one contract for the range of a function
- An *Object Contract* serves as the domain portion of a *Function Contract* with zero or more arguments
- **AFunction** defines an *Object Contract* from the array

Contract.**AFunction**(*[_Number_, _Number_], _Number_));*

Contract.**Function**(
  Contract.**AObject**(*[_Number_, _Number_]), _Number_));*

Recursive Contract

- Similar to *Constructors*
- Unrolls the constructor when asserted
- Contract is given as argument to the constructor

```
var __LinkedList__ = Contract.Recursive(
  Contract.Constructor(function(__LinkedList__) {
    return Contract.AObject({
       value:_Number_,
       next:__LinkedList__});
}));
```

Union Contract

```
// (Number × Number → Number) ∪ (Number × Number → String)
function plusBroken (x, y) {
  return (x>0 && y>0) ? (x + y) : 'Error';
}
```

Union Contract

```
// (Number × Number → Number) ∪ (Number × Number → String)
function plusBroken (x, y) {
  return (x>0 && y>0) ? (x + y) : 'Error';
}

var plusBroken = assert(plusBroken, Contract.Union(
  Contract.Function([_Number_, _Number_], _Number_)
  Contract.Function([_Number_, _Number_], _String_));
```

Union Contract

```
// (Number × Number → Number) ∪ (Number × Number → String)
function plusBroken (x, y) {
  return (x>0 && y>0) ? (x + y) : 'Error';
}

plusBroken('a', 'a'); ✗ blame the context
```

- Context gets *blamed* for $\mathcal{C} \cup \mathcal{C}'$ iff:
  (Context gets *blamed* for $\mathcal{C}$) $\vee$ (Context gets *blamed* for $\mathcal{C}'$)

Union Contract

```
// (Number × Number → Number) ∪ (Number × Number → String)
function plusBroken (x, y) {
  return (x>0 && y>0) ? (x + y) : 'Error';
}

plusBroken(1, 1); ✓
plusBroken(0, 1); ✗ blame the subject
```

- Subject f gets <u>*blamed*</u> for $\mathcal{C} \cup \mathcal{C}'$ iff:
  (f gets *blamed* for $\mathcal{C}$) $\wedge$ (f gets *blamed* for $\mathcal{C}'$)

Contract Abstraction

```
// T × T → T
function plus (x, y) {
  return (x + y);
}
```

Contract Abstraction

```
// T × T → T
function plus (x, y) {
  return (x + y);
}

var __Plus__ = Contract.Constructor(function (_Type_) {
  return Contract.Function([Type, Type], Type);
});
```

Contract Abstraction

```
// T × T → T
function plus (x, y) {
  return (x + y);
}

var __Plus__ = Contract.Constructor(function (_Type_) {
  return Contract.Function([Type, Type], Type);
});

var Plus = assert(plus, __Plus__);
```

Contract Abstraction

```
// T × T → T
function plus (x, y) {
  return (x + y);
}

var __Plus__ = Contract.Constructor(function (_Type_) {
  return Contract.Function([Type, Type], Type);
});

var Plus = assert(plus, __Plus__);

Plus(_Number_)(1, 2); ✓
```

Dependent Contract

```
// T × T → T
function plus (x, y) {
  return (x + y);
}

var __Type__ = Contract.Constructor(function(x, y) {
  return Contract.Base(function (arg) {
    return ((typeof x) === (typeof y)) &&
      ((typeof x) === (typeof arg));
  });
});
```

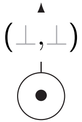
Dependent Contract

```
// T × T → T
function plus (x, y) {
  return (x + y);
}

var __Type__ = Contract.Constructor(function(x, y) {
  return Contract.Base(function (arg) {
    return ((typeof x) === (typeof y)) &&
      ((typeof x) === (typeof arg));
  });
});

var plus = assert(plus, Contract.Dependent(__Type__));

plus(1, 2); ✓
```

Callback Graph

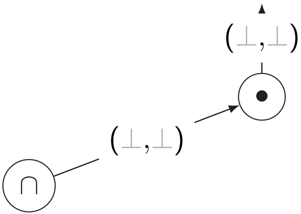
```
// (Number → Number) ∩ (String → String)
function addOneBroken (x) {
  return (x + '1');
}

addOneBroken('a'); ✓
addOneBroken(1); ✗ blame the subject
```

Callback Graph



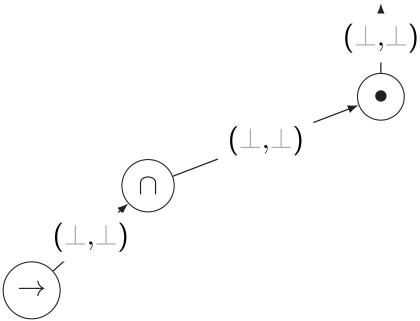
// (Number → Number) ∩ (String → String)
addOneBroken('a');

Callback Graph



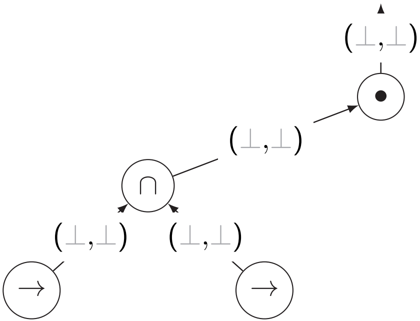
// (Number → Number) ∩ (String → String)
addOneBroken('a');

Callback Graph



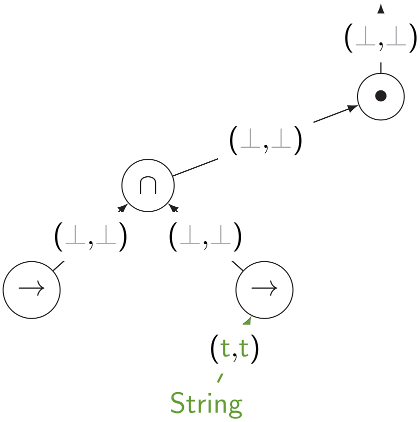
// (Number → Number) ∩ (String → String)
addOneBroken('a');

Callback Graph



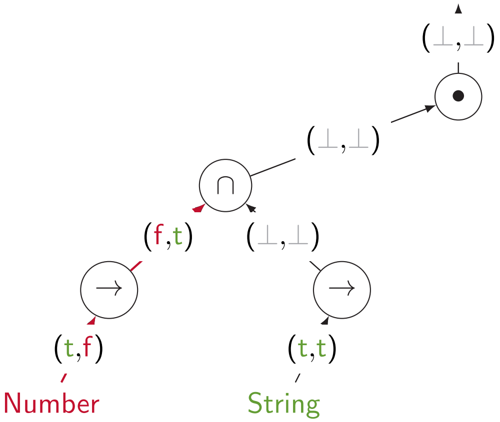
// (Number → Number) ∩ (String → String)
addOneBroken('a');

Callback Graph



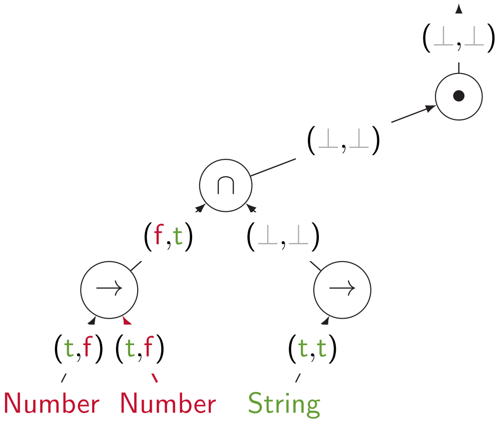
// (Number → Number) ∩ (String → String)
addOneBroken('a');

Callback Graph



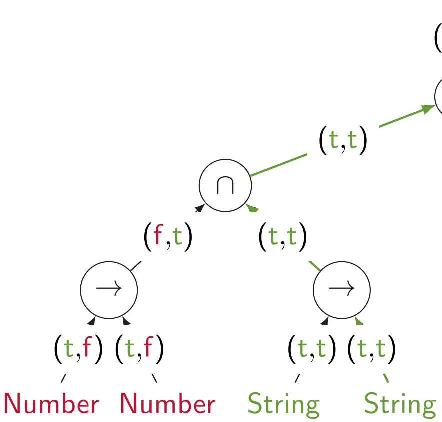
// (Number → Number) ∩ (String → String)
addOneBroken('a');

Callback Graph



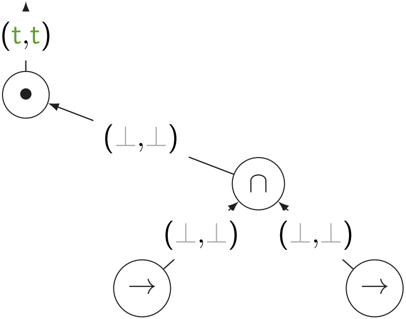
// (Number → Number) ∩ (String → String)
addOneBroken('a');

Callback Graph



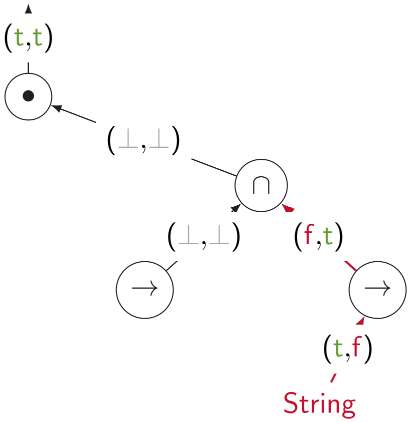
// (Number → Number) ∩ (String → String)
addOneBroken('a'); ✓

Callback Graph



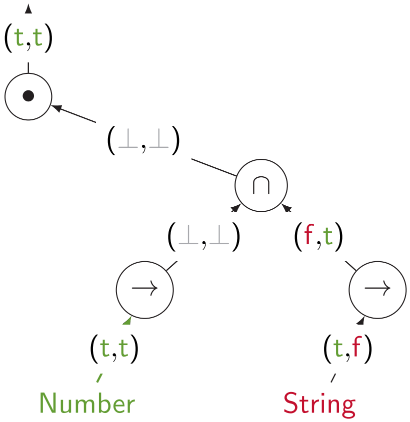
// (Number → Number) ∩ (String → String)
addOneBroken(1);

Callback Graph



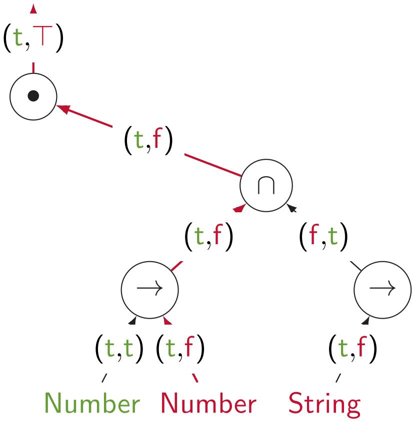
// (Number → Number) ∩ (String → String)
addOneBroken(1);

Callback Graph



// (Number → Number) ∩ (String → String)
addOneBroken(1);

Callback Graph



// (Number → Number) ∩ (String → String)
addOneBroken(1); blame the subject

Scores

Benchmark	Full	System	Baseline
Richards	0.391	0.519	11142
DeltaBlue	0.276	0.360	17462
Crypto	11888	12010	11879
RayTrace	1.09	1.45	23896
EarleyBoyer	5135	5292	5370
RegExp	1208	1181	1207
Splay	20.6	27.8	9555
SplayLatency	73.1	99.7	6289
NavierStokes	6234	7159	12612
pdf.js	9191	9257	9236
Mandreel	12555	12542	12580
MandreelLatency	18741	18883	19398
Gameboy Emulator	6.80	9.07	23801
Code loading	6245	6785	9324
Box2DWeb	3.57	4.67	12528
zlib	29108	28708	29185
TypeScript	187	248	11958

Scores

Benchmark	Full	w/o C	w/o D	w/o M	w/o P
Richards	0.391	0.582	0.782	0.781	0.903
DeltaBlue	0.276	0.409	0.544	0.544	0.625
Crypto	11888	11912	11914	11986	11979
RayTrace	1.09	1.82	2.51	2.51	3.02
EarleyBoyer	5135	5126	5205	5233	5242
RegExp	1208	1205	1199	1212	1178
Splay	20.6	31.2	42.5	42.5	49.7
SplayLatency	73.1	109	151	151	177
NavierStokes	6234	7924	9176	8943	9456
pdf.js	9191	9548	9156	9222	9152
Mandreel	12555	12586	12549	12346	12431
MandreelLatency	18741	18741	18883	19027	18955
Gameboy Emulator	6.80	10.8	14.9	14.9	17.7
Code loading	6245	6937	7372	7335	7533
Box2DWeb	3.57	5.72	7.80	7.82	9.19
zlib	29108	29025	29047	28926	29063
TypeScript	187	290	400	396	463

Statistic

Benchmark	Contract A	Contract I	Contract P
Richards	24	1599377224	935751200
DeltaBlue	54	2319477672	1340451212
Crypto	1	5	3
RayTrace	42	687240082	509234422
EarleyBoyer	3944	89022	68172
RegExp	0	0	0
Splay	10	11620663	7067593
SplayLatency	10	11620663	7067593
NavierStokes	51	48334	39109
pdf.js	3	15	9
Mandreel	7	57	28
MandreelLatency	7	57	28
Gameboy Emulator	3206	141669753	97487985
Code loading	5600	34800	18400
Box2DWeb	20075	172755100	112664947
zlib	0	0	0
TypeScript	4	12673644	8449090

Statistic

Benchmark	Sandbox M	Sandbox D
Richards	4678756000	4
DeltaBlue	6702256060	5
Crypto	15	3
RayTrace	2546172110	4
EarleyBoyer	340860	6
RegExp	0	0
Splay	35337965	5
SplayLatency	35337965	5
NavierStokes	195545	5
pdf.js	45	4
Mandreel	140	4
MandreelLatency	140	4
Gameboy Emulator	487439925	5
Code loading	92000	4
Box2DWeb	563324735	5
zlib	0	0
TypeScript	42245450	2

Statistic

Benchmark	Callback
Richards	3351504000
DeltaBlue	4744203248
Crypto	13
RayTrace	2190186074
EarleyBoyer	309120
RegExp	0
Splay	26231845
SplayLatency	26231845
NavierStokes	177197
pdf.js	39
Mandreel	128
MandreelLatency	128
Gameboy Emulator	399084085
Code loading	70400
Box2DWeb	469141435
zlib	0
TypeScript	33796350

Timings

Benchmark	Full	Baseline
Richards	1 day, 18 hours, 21 min, 20 sec	8 sec
DeltaBlue	2 days, 10 hours, 36 min, 49 sec	4 sec
Crypto	9 sec	8 sec
RayTrace	23 hours, 12 min, 37 sec	4 sec
EarleyBoyer	1 min, 9 sec	53 sec
RegExp	9 sec	8 sec
Splay	19 min, 19 sec	3 sec
SplayLatency	19 min, 19 sec	3 sec
NavierStokes	11 sec	4 sec
pdf.js	6 sec	6 sec
Mandreel	5 sec	5 sec
MandreelLatency	5 sec	5 sec
Gameboy Emulator	4 hours, 28 min, 28 sec	5 sec
Code loading	12 sec	9 sec
Box2DWeb	5 hours, 19 min, 49 sec	6 sec
zlib	11 sec	11 sec
TypeScript	22 min, 46 sec	24 sec

Timings

Benchmark	Predicate	Baseline
Richards	1 day, 6 hours, 59 min, 42 sec	8 sec
DeltaBlue	1 day, 20 hours, 58 min, 17 sec	4 sec
Crypto	8 sec	8 sec
RayTrace	17 hours, 1 min, 25 sec	4 sec
EarleyBoyer	1 min, 3 sec	53 sec
RegExp	8 sec	8 sec
Splay	13 min, 55 sec	3 sec
SplayLatency	13 min, 55 sec	3 sec
NavierStokes	9 sec	4 sec
pdf.js	6 sec	6 sec
Mandreel	5 sec	5 sec
MandreelLatency	5 sec	5 sec
Gameboy Emulator	3 hours, 13 min, 13 sec	5 sec
Code loading	12 sec	9 sec
Box2DWeb	3 hours, 52 min, 34 sec	6 sec
zlib	12 sec	11 sec
TypeScript	17 min, 8 sec	24 sec

TreatJS-Sandbox

Transaction-based Sandboxing of JavaScript

Motivation


CNN
News Regions Video TV Features Opinions More...
International Edition
Search CNN
U.S. China Asia Middle East Africa Europe Americas
NASA finds 'Earth's bigger, older cousin'
By Michael Pearson, CNN
Updated 1435 GMT (2135 HKT) July 24, 2015 | Video Source: CNN/NASA
NASA
KEPLER-452B IS 1,400
LIGHT YEARS FROM EARTH
Kepler-452 System
Kepler-186 System
Solar System
Kepler-186f
Mercury Venus Earth
Mars
Kepler-452b
Now Playing
NASA finds 'Earth's bigger, older cousin'
03:25
Mysterious 'heartbeat' caused by sunspot cycle
01:17
Whole new look at Pluto
01:06
New Horizons passes Pluto after 3 billion-mile journey
01:01
See NASA New Horiz arrives at P
DIE WELT DIGITAL
Smartphone + Tablet + Web
0,00€
Wie standfest ist die EU?
DIE WELT
Advertisement
Story highlights
The planet is the most Earth-like yet found in the habitable zone of a star like ours
It's about 60% larger than our own planet and "almost certainly has an atmosphere"
(CNN)—NASA said Thursday that its Kepler spacecraft has spotted "Earth's bigger, older cousin": the first nearly Earth-size planet to be found in the habitable zone of a star similar to our own.
Though NASA can't say for sure whether the planet is rocky like ours or has water and air, it's the closest
Space news

Non-interfering Contract Execution

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

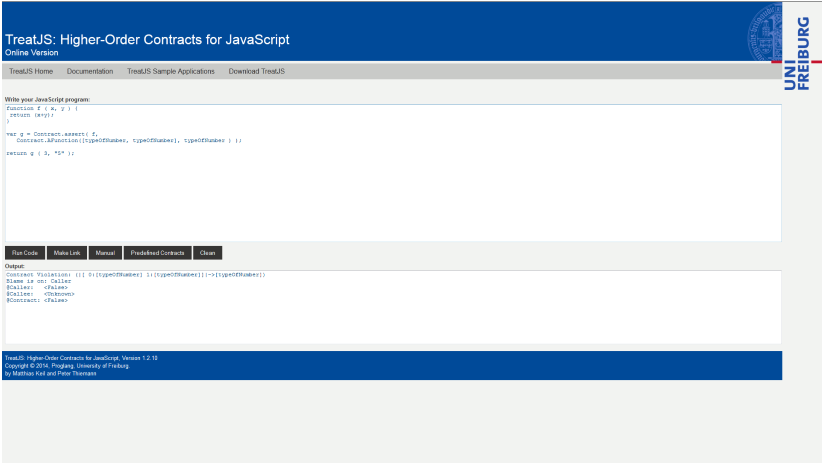
Non-interfering Contract Execution

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

Read-only access

```
var _Array_ = Contract.Base(function (arg) {
  return (arg instanceof OutsideArray);
});
var _Array_ = Contract.With({OutsideArray:Array}, _Array_);
```

Application Scenarios
TreatJS Online



Running Example

```
function Node (value, left, right) {
  this.value = value;
  this.left = left;
  this.right = right;
}
function heightOf (node) {
  return node ? Math.max(heightOf(node.left)+1), heightOf(
      node.right)+1) : -1;
}
function setValue (node) {
  node.value=heightOf(node);
  if(node.left) setValue(node.left);
  if(node.right) setValue(node.right);
}
```

Function Invocation

var *root = new Node(0, new Node(0), new Node(0));*

Function Invocation

```
var root = new Node(0, new Node(0), new Node(0));
```

```
var sbx = new Sandbox(this, {/* some parameters */});
```

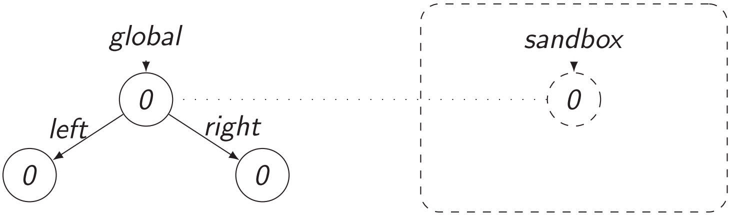
Function Invocation

```
var root = new Node(0, new Node(0), new Node(0));
```

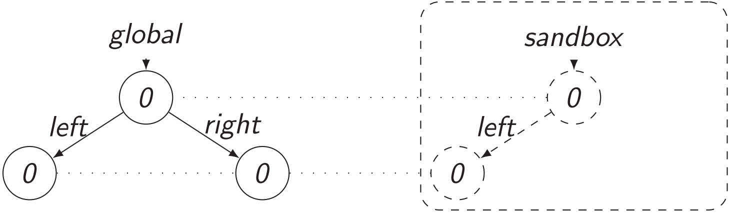
```
var sbx = new Sandbox(this, {/* some parameters */});
```

```
sbx.call(heightOf, this, root);
```

Cross-Sandbox Access
Read Access

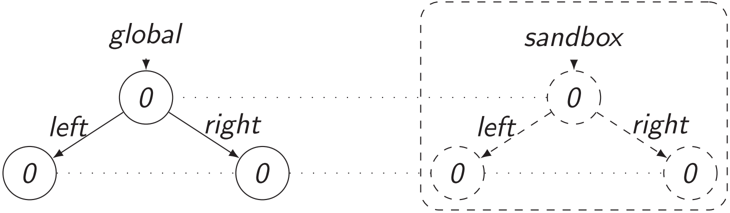


function *heightOf (node)* {
return *node ? Math.max(heightOf(node.left)+1), heightOf(node.right)+1) : −1;*
}




```
function heightOf (node) {
  return node ? Math.max(heightOf(node.left)+1), heightOf(
      node.right)+1) : −1;
}
```

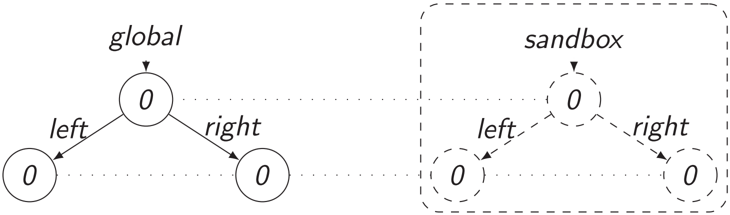
UNI FREIBURG



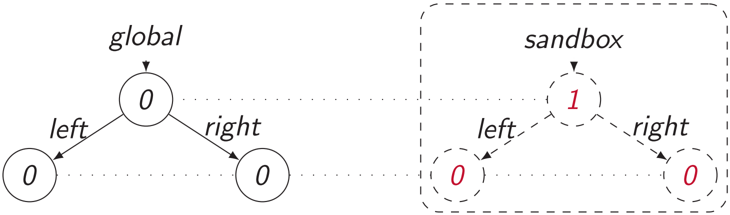

```
function heightOf (node) {
  return node ? Math.max(heightOf(node.left)+1), heightOf(
       node.right)+1) : −1;
}
```

Cross-Sandbox Access

Write Access

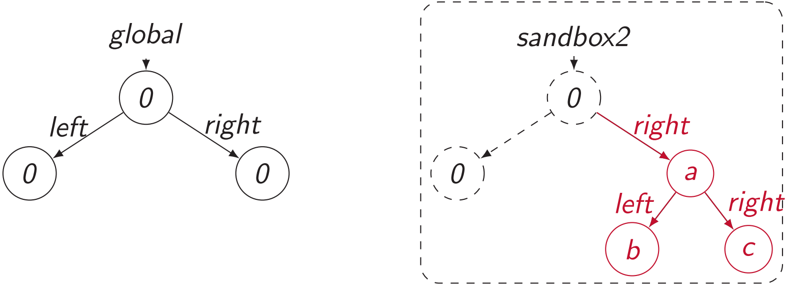



```
function setValue (node) {
  node.value=heightOf(node);
  if(node.left) setValue(node.left);
  if(node.right) setValue(node.right);
}
```

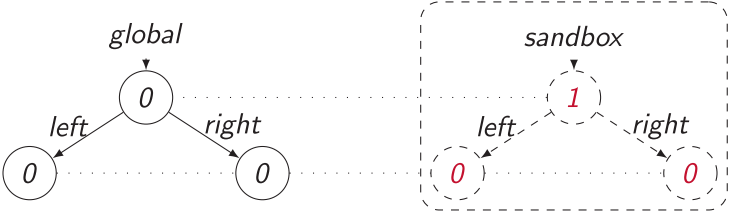



```
function setValue (node) {
  node.value=heightOf(node);
  if(node.left) setValue(node.left);
  if(node.right) setValue(node.right);
}
```

Cross-Sandbox Access
Write Access (cont'd)

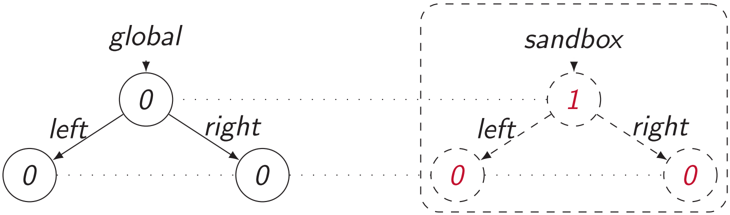



```
var sbx2 = new Sandbox(this, {/* some parameters */});
function appendRight (node) {
  node.right = Node('a', Node('b'), Node('c'));
}
sbx2.call(appendRight, this, root);
```



*sbx.effectsOf(this).foreach(***function***(i, e) {print(e)});*

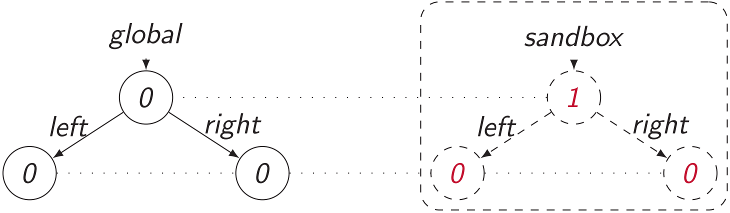
;;; Effects **of** *this*
(1425301383541) has [name=heightOf]
(1425301383541) get [name=heightOf]
(1425301383543) has [name=Math]
(1425301383543) get [name=Math]



*sbx.writeeffectsOf(root).foreach(**function**(i, e) {print(e)});*

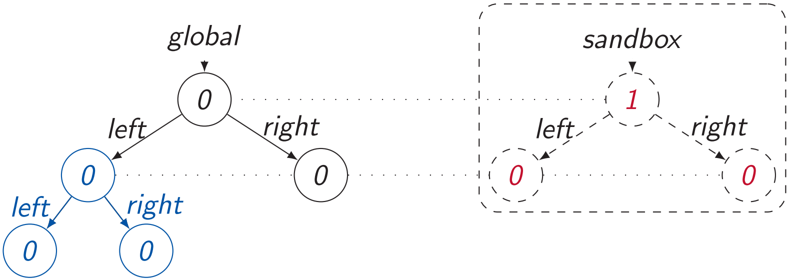
;;; Write Effects **of** *root*
(1425301634992) **set** *[name=value]*

Inspecting a Sandbox
Changes



sbx.hasChanges; // returns true
*sbx.changes.foreach(***function***(i, e) {print(e)});*

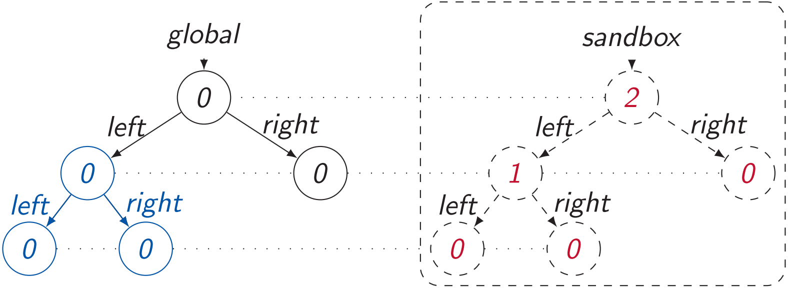
;;; All Changes
Change: (1425300876577) **set** *[name=value]@SBX001*



sbx.hasDifferences; // *returns true*
*sbx.differences.foreach(***function***(i, e) {print(e)});*

;;; Differences **of** *root*
Difference: (1425300932388) **set** *[name=value]@SBX001*

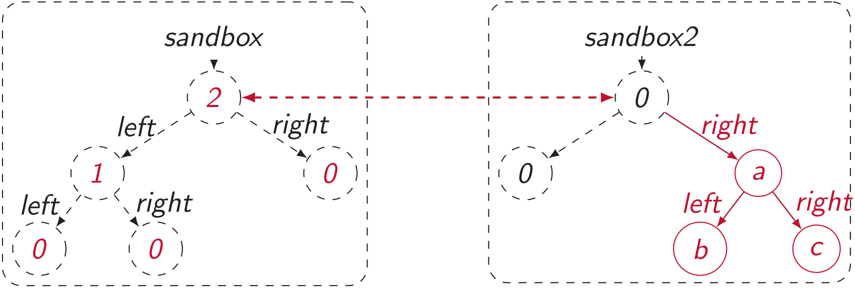
Inspecting a Sandbox
Differences




```
sbx.hasDifferences; // returns true
sbx.differences.foreach(function(i, e) {print(e)});
```

;;; *Differences* **of** *root*

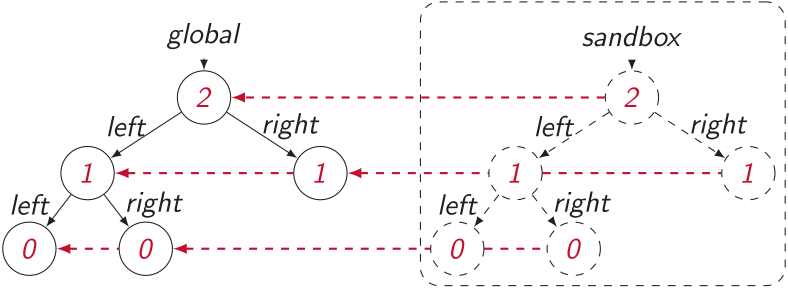
Difference: (1425300932388) **set** *[name=value]@SBX001*



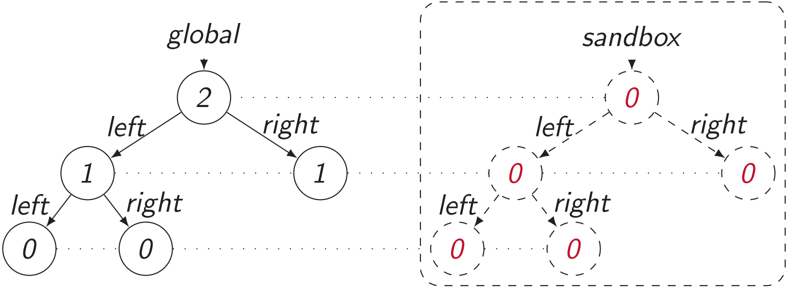

```
sbx.inConflictWith(sbx2); // returns true
sbx.conflictsWith(sbx2).foreach(function(i, e) {print(e)});
```

Confict: (1425303937853) get [name=right]@SBX001 – (1425303937855) **set** *[name=right]@SBX002*

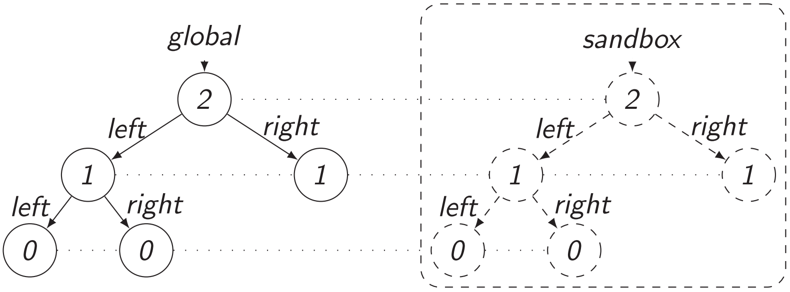
Transaction Processing
Commit



sbx.commit();



sbx.rollback();


global
2
left
right
1
1
left
right
0
0
sandbox
2
left
right
1
1
left
right
0
0

sbx.revert();

Further Features

- Fine-graned handling of commit, rollback and revert
- Transparent sandboxing
- Snapshot Mode
- Wrapping of values

Sandbox Scope Chain

```
var node = new Node(/*some sub−nodes */);
var sbx = new Sandbox(this, /*some parameters */);

with(sbxglobal){

    function setValue (node) {
      node.value=heightOf(node);
      setValue(node.left);
      setValue(node.right);
    }

}
```

Policies

- A new sandbox starts with a blank environment
- Read access by binding values when instantiating a new sandbox
- Fine-grained write access by committing effects
- Combination with *Access Permission Contracts* and *Revocable References* possible

Features

- Enables scripts to run in a configurable degree of isolation
- Enables fine-grained access control
- Provides a transactional scope that performs effect logging
- Effects can be committed to the application state or rolled back

Limitations

- Function recompilation needs a string that contains the source code of that function
- **Function**.*prototype.toString* did not work for all functions
  - A native function did not have a string representation
  - The **Function**.*prototype.bind* method creates a new bound function without a string representation

Experimental Evaluation
Scores

Benchmark	Sandbox F	Sandbox w/o E	Transparent F	Transparent w/o E	Baseline
Richards	3919	4270	4070	4078	6618
DeltaBlue	177	333	225	333	4778
Crypto	1508	1917	1666	1959	11012
RayTrace	158	211	157	220	3941
EarleyBoyer	624	677	618	663	2648
RegExp	883	900	887	903	1192
Splay	697	774	706	1394	1439
SplayLatency	1986	2567	1983	2276	1234
NavierStokes	915	865	938	934	11835
pdf.js	84	121	85	127	9229
Mandreel	95	122	96	124	8521
MandreelLatency	455	621	437	632	16562
Gameboy Emulator	4413	5081	4577	5027	22307
Code loading	5123	5264	5148	5202	6721
Box2DWeb	84	121	82	120	13853
zlib	-	-	-	-	28970
TypeScript	3006	3339	2940	3242	11813

Experimental Evaluation
Timings

Benchmark	Sandbox	Transparent	Baseline
Richards	14 sec	16 sec	9 sec
DeltaBlue	263 sec	265 sec	13 sec
Crypto	59 sec	58 sec	9 sec
RayTrace	865 sec	854 sec	23 sec
EarleyBoyer	265 sec	249 sec	75 sec
RegExp	14 sec	16 sec	8 sec
Splay	32 sec	43 sec	17 sec
SplayLatency	32 sec	43 sec	17 sec
NavierStokes	59 sec	61 sec	5 sec
pdf.js	944 sec	872 sec	9 sec
Mandreel	683 sec	683 sec	9 sec
MandreelLatency	683 sec	683 sec	9 sec
Gameboy Emulator	30 sec	27 sec	7 sec
Code loading	14 sec	14 sec	10 sec
Box2DWeb	1156 sec	1140 sec	6 sec
zlib	-	-	11 sec
TypeScript	86 sec	91 sec	24 sec

Experimental Evaluation
Statistic

Benchmark	Wrap	Cache Miss	Cache Hit	Effect
Richards	164021	3	7	328060
DeltaBlue	7348024	3	7	14696066
Crypto	2329634	3	1046550	3629234
RayTrace	25519224	3	12757207	38281266
EarleyBoyer	5465	3	24	4194942
RegExp	115474	3	65923	230966
Splay	466840	3	232019	1397701
SplayLatency	466840	3	232019	1397701
NavierStokes	1839	3	727	2978
pdf.js	18222921	4	37302	37477198
Mandreel	19954116	3	9974959	29934936
MandreelLatency	19954116	3	9974959	29934936
Gameboy Emulator	467585	4	121250	813987
Code loading	23829	5	4208	45064
Box2DWeb	24876419	3	7800731	115579412
zlib	-	-	-	-
TypeScript	1114199	3	17	26404297