

DOM Transactions for Testing JavaScript

Phillip Heidegger,

Annette Bieniusa, Peter Thiemann

University of Freiburg

2010/09/05

Setting: Whitebox Unit Testing of JavaScript Code

```
1 /** object → bool */  
2 function h(x) {  
3   if ((x) && (x.longpropertyname))  
4     return "false";  
5   return true;  
6 };
```

Setting: Whitebox Unit Testing of JavaScript Code

```
1 /** object → bool */  
2 function h(x) {  
3   if ((x) && (x.longpropertyname))  
4     return "false";  
5   return true;  
6 };
```

- Find counter example for contracts
- Finding bugs in JavaScript programs
- Understanding JavaScript programs

JavaScript – Features

Some important features:

- Weak, dynamic typing
- Object-based language (no classes, but prototypes)
- Functions are first class values
- Side effects

Checking Contracts of pure Functions

- Monitoring of contracts (assertions)
- Unit Testing with random generated data
 - Generate random data for parameters
 - check return contract
- Isolated test cases

Checking Contracts of pure Functions

- Monitoring of contracts (assertions)
- Unit Testing with random generated data
 - Generate random data for parameters
 - check return contract
- Isolated test cases

Side Effects ?



Side effects?

Why not include side effects in the contracts?

- Programmers typically introduces contracts incremental
- Contracts do not specify the complete behavior of the system

Default is:

- Everything that is not forbidden, is allowed

Handling Side Effects

1. Contracts with a `setup` and `teardown` function
 - Means a lot of work
2. Restart browser after each test run
 - Takes seconds
3. ?

Handling Side Effects

Transactions

on HTML pages

Transactions

Body of our test loop:

- Begin a transaction
- Generate test data
- Run unit under test
- Check contract
- Revert changes

Implementation

- Integrated into **JSConTest**
 - Contracts for Specification
 - Random Testing
 - Guided Random Testing
- Log based transactions
- Transformation of code under test
- Browser independent

Implementation

- Integrated into **JSConTest**
 - Contracts for Specification
 - Random Testing
 - Guided Random Testing
- Log based transactions
- Transformation of code under test
- Browser independent

We can not revert every side effect, but

→ setup/teardown or browser restart is rarely necessary

Performance?

Slowdown: ≈ 4.3

Performance?

Slowdown: ≈ 4.3

But we get:

- Contract monitoring
- Transaction system
- Random data generation

→ much better than restart the browser

Conclusion

JSConTest

- Contract testing framework for JS
- Transactions for functions with side effects
- Browser independent
- `http://proglang.informatik.uni-freiburg.de/JSConTest/`