Monitoring Javascript-Applications with Kieker

Daniel Schmidt

23.11.2016
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation
   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation
   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
Goals

Motivation

- Evaluation of Existing Aspect Oriented Programming Frameworks and Libraries
Goals

- Evaluation of Existing Aspect Oriented Programming Frameworks and Libraries
- Implementation of a Method to Monitor Javascript Applications
Goals

- Evaluation of Existing Aspect Oriented Programming Frameworks and Libraries
- Implementation of a Method to Monitor Javascript Applications
- Implementation of an Efficient Way to Adapt Kieker for Javascript Applications
Goals

Evaluation of Existing Aspect Oriented Programming Frameworks and Libraries
Implementation of a Method to Monitor Javascript Applications
Implementation of an Efficient Way to Adapt Kieker for Javascript Applications
Evaluation of Developed Way to Monitor Javascript Applications with Kieker
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation
   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
- Kieker
- Aspect Oriented Programming
- Javascript
- Web Worker
Javascript Foundations

- Scripting Language
- Dynamically Typed
- Prototypal Inheritance
- Environments
  - Browser
  - Node.js®
```javascript
function myFunction() {...} // Function Declaration
var myFunction = function() {...}; // Function Expression
```
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation
   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
Approach

Developed Approach

Original Application

Javascript Files

package.json

Instrumented Application

Javascript Files

Original Source Code

Kieker-javascript

package.json

Figure: Structure of kieker-javascript
Developed Approach

- core & worker
- writer
  - console
  - http
- advices
  - environment
  - function
  - method
- records
- cli (Command Line Interface)
Figure: Function Advice usage in kieker-javascript
kieker.sendEnvironment();

var qs = kieker.instrumentFunction(function (selector) {
    return document.querySelector(selector);
});

var items = kieker.instrumentMethod({
    add: function () {
        // ...
    }
});
Command Line Interface

Figure: Architecture of kieker-javascript-cli
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation
   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation

   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
- Existing solution don’t provide the necessary functionality
- Own implementation needed to be developed
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation
   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
Demo time
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation
   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
Execution Overhead
RAM Consumption
Throughput
Figure: Execution Overhead of TODO MVC
RAM Consumption

Figure: Flexboxfroggy Uninstrumented
Figure: Flexboxfroggy Instrumented
Figure: Throughput of Records received by Kieker Data Bridge
- Miliseconds
- More fine-granular solutions exist
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation
   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
Achievements

Conclusion

- Evaluation of Aspect Oriented Programming Approaches
- Instrumentation of Javascript Applications
  - Manual
  - Automatic
Problems

Conclusion

- Automatic Instrumentation Fails in Certain Cases
- Execution Overhead is significant
- Only Browser Environment is supported
1. Motivation

2. Foundations

3. Developed Approach

4. Evaluation
   Aspect Oriented Programming Frameworks
   Feasibility
   Performance

5. Conclusion

6. Future Work
Future Work

- Performance
  - asm.js or WebAssembly
  - alternative AOP approaches
- Unsuccessful automated instrumentation in certain scenarios
var a = 42;
printA();

function printA() {
    console.log(a);
}
var a = 42;
printA();

var printA = function printA() {
    console.log(a);
};
```javascript
var a = 42;
var printA = function printA() {
    console.log(a);
};
printA();
```
Future Work

- Precision
  - Use better clock, e.g. `Performance.now`

- Environment
  - Add support for Node.js and similar environments
Thank you for your attention