Detecting Inconsistencies in JavaScript MVC Applications

Frolin S. Ocariza, Jr., Karthik Pattabiraman and Ali Mesbah

University of British Columbia
Used by 90% of all websites

Most popular language on GitHub (RedMonk survey)

JavaScript

4 JavaScript error messages on average per website [ISSRE’11]

68% of JavaScript faults are DOM-related [ESEM’13]
DOM-Related Faults

DOM-Related Faults are...

Prevalent  Impactful  Non-Trivial to Fix

JS CODE

var elem = getElementById("car")
elem.innerHTML = "Hello World!"

DOM API Method

Erroneous parameter
MVC Frameworks to the Rescue!

MVC Frameworks use **two-way data binding** ➔ No DOM method calls!
MVC Frameworks to the Rescue!

MODEL

```javascript
$scope.msg = "";
```

VIEW

```html
<input type="text" ng-model="msg" />
<div ng-bind="msg"></div>
```

No DOM method calls ➔ No DOM-Related Faults in MVC Applications!
MVC Frameworks to the Rescue?

MODEL

DEFINES

MODEL VARIABLES (MV)

VIEW

USES

USES

CONTROLLER FUNCTIONS (CF)

USES

CONTROLLER

USES

DEFINE

IDENTIFIER INCONSISTENCIES
MVC Frameworks to the Rescue?

MODEL VARIABLES (MV) TYPES

CONTROLLER FUNCTIONS (CF) TYPES

MODEL DEFINES

VIEW USES

CONTROLLER USES

USES DEFINES

IDENTIFIER INCONSISTENCIES

TYPE INCONSISTENCIES
MVC Frameworks to the Rescue?

Identifier and type Inconsistencies happen in real life

Inconsistencies are hard to find

• "What's wrong with AngularJS?"
  By B. Kinoshita

• "Why you should not use AngularJS"
  By E. Koshelko

• e.g., No error messages thrown!
To design a way to automatically detect identifier and type inconsistencies in MVC applications
The **most popular JS MVC framework** in GitHub, StackOverflow, and YouTube

300% increase in AngularJS usage over the past year
Our Approach

STATIC ANALYSIS
Our Approach

- **MODEL**
  - String: a
  - Boolean: b
  - Object: c

- **VIEW**
  - String: d
  - String: fun()
  - Boolean: e
  - String: bar()

- **CONTROLLER**
  - String: foo()
  - Object: c
  - Number: foo()

Inconsistent types!

"e" is not defined in model!
Challenges

**Nested Objects**

```javascript
$scope.obj = {
    prop1: 2,
    prop2: {
        subprop1: "hello",
        subprop2: true
    }
    prop3: "world"
};
```

Obj is a model variable, but so is `obj.prop1`, `obj.prop2.subprop1`, etc.

**Aliasing**

```javascript
$scope.arr = [...]

<li ng-repeat="temp in arr">
    {{temp.val}}
</li>
```

Temp is an alias!

**Groupings**
Running Example

Movie Search

- Type Username
- List User's Favourite Movies

Results Page

Welcome User #1
- Star Wars
- 8 1/2
2 movies

List of movies

Number of movies in list

Which User?
“Search” Grouping

“Search” Model

“SearchCtrl” Controller

“search.html” View

“Results” Grouping

“Results” Model

“ResultsController” Controller

“results.html” View
ng-repeat attribute used to loop through movies in the movie list...

…but userData.count is assigned a String in the model

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

userName model variable not defined in model

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expectsuserData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList

The view expects userData.count to be of type Number...

…but ng-repeat loops through userData instead of userData.movieList
Finding the Models, Views, and Controllers

JavaScript

Model  Controller

View
Inferring Identifiers

Model Variable and Controller Function Definitions
Done via $scope identifier

Model Variable and Controller Function Usages
Embedded in HTML code

“Straw man” approach: Just take whatever identifiers you see in the above parts of the code
“Results” Model

userData

This is a nested object

“SearchCtrl” Controller

userName

searchUser()

“Search” Model

userName

“ResultsCtrl” Controller

userName

alertUserName()

“search.html” View

userName

alertUserName()

“results.html” View

userData

movie.name

userData.count

This is an alias
Inferring Identifiers: Nested Objects

Object Literal

```javascript
$scope.userData = {
  movieList: {
    movie1: ..., 
    movie2: ...
  },
  count: "two",
  intro: ..., 
  display: true
};
```

Tree, where each node is an object property identifier
Inferring Types

For assigned/returned types: Look at AST node!

```javascript
$scope.bar = "hello"
```

**Insight**: Parts of MVC applications typically written in “declarative” way.
Inferring Types

For expected types: Examine HTML attribute

```html
<ng-pluralize count="userData.count" />
```

`userData.count` is expected to be a Number

```html
<h3 ng-if="userData.display">...
```

`userData.display` is expected to be a Boolean
“Search” Model

Object: userData
String: userData.count
Object: userData.movieList

“SearchCtrl” Controller

userName
Void: searchUser()

“Results” Model

Object: userData
String: userName

“ResultsController” Controller

userName
Void: alertUserName()

“search.html” View

userName
alertUserName()

“results.html” View

Array/Object: userData
userData.count.name
userData.movieList.name
Number: userData.count
Discovering MVC Groupings

Groupings can be inferred via

\texttt{ng-controller} attribute

\texttt{routers}
.when('/', {
    controller: 'SearchCtrl',
    templateUrl: 'search.html'
})

.when('/', {
    controller: 'ResultsController',
    templateUrl: 'results.html'
})
Grouping 1

“Search” Model
“SearchCtrl” Controller
“search.html” View

Grouping 2

“Results” Model
“ResultsCtrl” Controller
“results.html” View
Detecting Inconsistencies

Controller Functions:

String comparison of identifiers and types

Model Variables:

 userData.

userData.count

Traverse

userData

count movieList intro display

movie1 movie2
Message 1:
userName is not defined in “Results” model

Message 2:
userData.count.name not defined in “Results” model

Message 3:
userData.count expected to be of type Number, but is of type String instead in “Results” model
Aurebesh

http://www.ece.ubc.ca/~frolino/projects/aurebesh

APP.JS

```
29 searchApp.controller('ResultsController', function($scope, $routeParams)
30   $scope.userData = {
31     movielist: getList($routeParams.userId),
32     intro: "Welcome User #" + $routeParams.userId,
33     display: true,
34     count: "two"
35   }
36 $scope.movieForms = {
37     one: '[] movie',
38     ...
```

RESULTS.HTML

```
7 <div id="movieCount">
8   <ng-pluralize count="userData.count" when="movieForms">
9   </div>
10   <br />
11 <button ng-click="alertUserName()">Which User?
12   </button>
13 </button>
```

“count” is expected to be a Number!
Research Questions

**RQ1**: Can Aurebesh help developers find real bugs in real-world web applications?

**RQ2**: How accurate is Aurebesh in finding identifier and type inconsistencies?

**RQ3**: How quickly can Aurebesh perform the inconsistency detection analysis?
20 open-source AngularJS applications

100+ to 1000+ lines of JS code

eval() ✗
dynamic type conversions ✗
RQ1: Real Bugs

Aurebesh found **15 bugs** previously undetected (5 were acknowledged by developers)

**Todo Application**
Can’t add todo items…

**Slide Maker**
Can’t change themes and transitions in slides…
RQ2: Accuracy

Fault injection experiment

10 types of mutations

200 injections per application

Recall

\[
\text{Recall} = \frac{\text{# of successful detections}}{\text{# of total injections}}
\]

96% overall

Precision

\[
\text{Precision} = \frac{\text{# of successful detections}}{\text{# of error messages displayed}}
\]

Close to 100% (only one false positive)
RQ3: Performance

Average Execution Time

121 ms

Worst-case Execution Time

849 ms (eTuneBook)
MVC Frameworks to the Rescue!

MVC Frameworks use **two-way data binding** ➔ No DOM method calls!

To design a way to automatically detect identifier and type inconsistencies in MVC applications

Aurebesh

http://www.ece.ubc.ca/~froino/projects/aurebesh

RQ2: Accuracy

Fault injection experiment 10 types of mutations 200 injections per application

Recall

\[ \frac{\text{# of successful detections}}{\text{# of total injections}} \]

96% overall

Precision

\[ \frac{\text{# of successful detections}}{\text{# of error messages displayed}} \]

Close to 100% (only one false positive)