Building Reliable Software for the Web, IoT and Beyond ….

Karthik Pattabiraman,

Frolin Ocariza, Kartik Bajaj, Saba Alimadadi, Sheldon Sequiria,

Kumseok Jung, Julien-Gascon Samson, and Ali Mesbah

Electrical and Computer Engineering

University of British Columbia (UBC)

http://blogs.ubc.ca/karthik/

My Research

- **Building error resilient and secure software systems**

- **Three areas**
  - Software error resilience [DSN'18A[DSN'18B][DSN'17][SC'17][DSN'16][SC'16][DSN'15]
  - Web applications' reliability [ICSE'18][ASE'17][ICSE'16][ICSE'15][ASE15][ICSE'14A][ICSE'14B]
  - IoT Security and Reliability [ECOOP'18][FSE'17][ACSAC'16][EDCC'16]

- **This talk: Web Applications' and IoT Reliability**

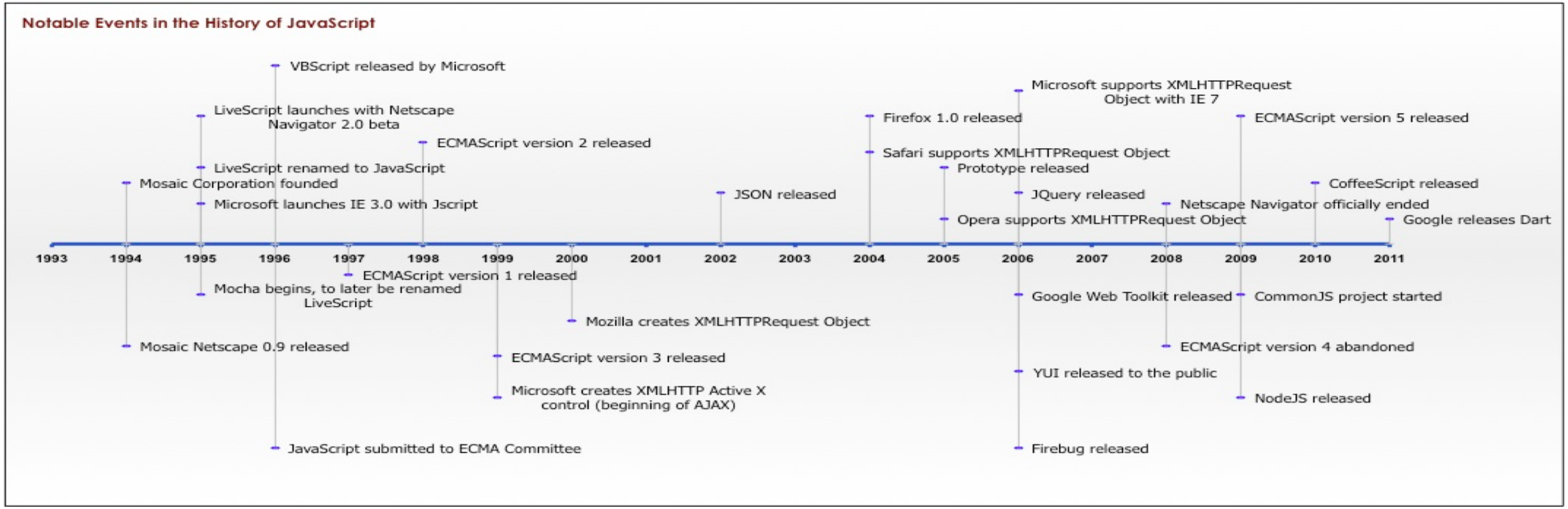
Modern Web Applications: Examples

Modern Web Applications: JavaScript

- JavaScript: Implementation of ECMAScript standard
  - Client-Side JavaScript: used to develop web apps
- Executes in client's browser – send AJAX messages
- Responsible for web application's core functionality
- Not easy to write code in – has many "evil" features

JavaScript: History

Brief History of JavaScript (Source: TomBarker.com)

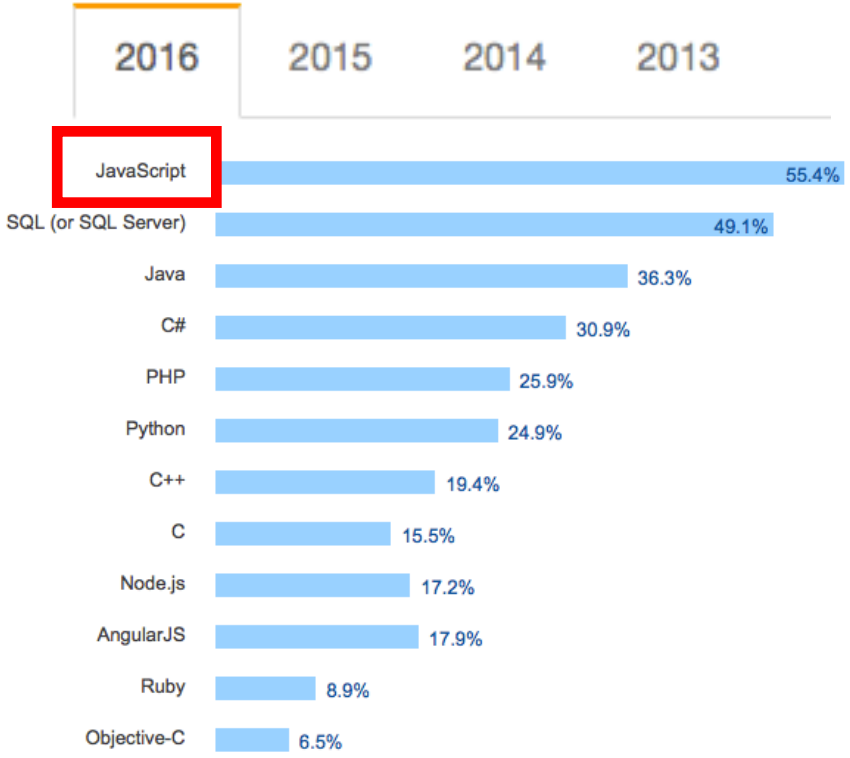


JavaScript (JS) had to "look like Java" only less so, be Java's dumb kid brother or boy-hostage sidekick. Plus, I had to be done **in ten days** or something worse than JS would have happened

– Brendan Eich (Inventor of JavaScript)

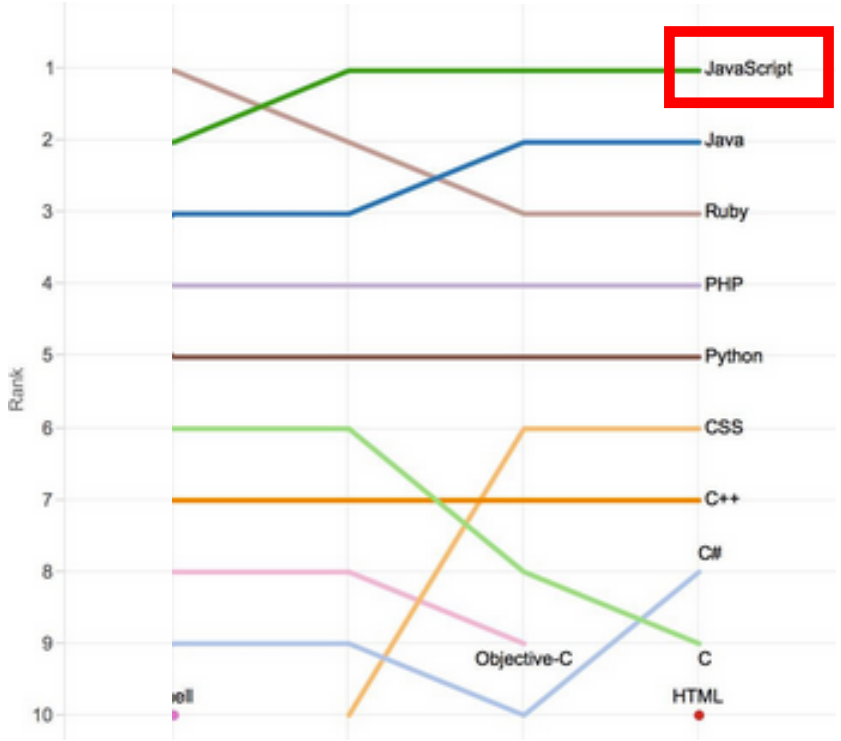
stackoverflow

JavaScript:
Most popular language

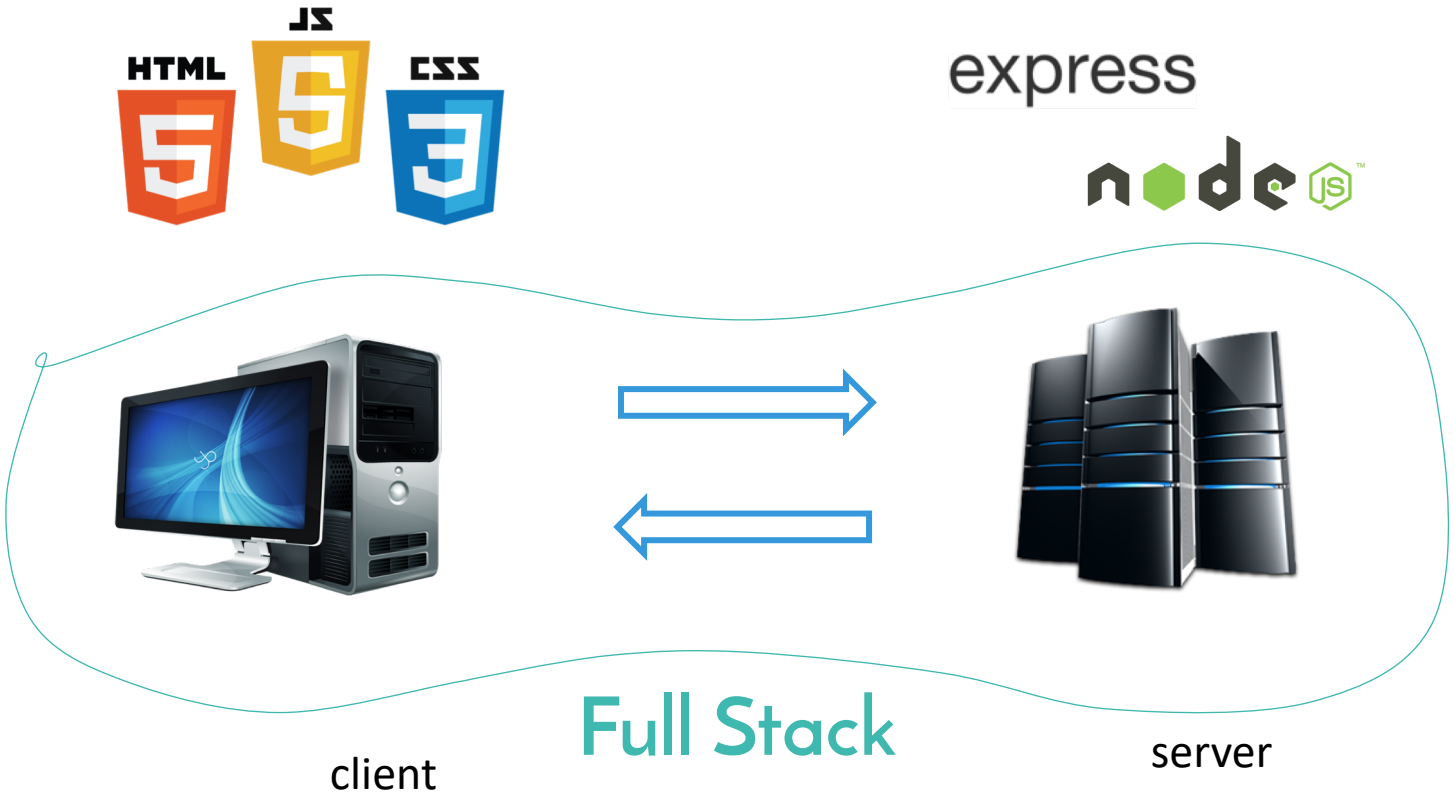


GitHub

JavaScript:
Top languages on GitHub



JavaScript Today



Studies of JavaScript Web Applications

Performance and parallelism:
JSMeter [Ratanaworabhan-2010], [Richards-2009], [Fortuna-2011]

Reliability

?

Security and Privacy:
[Yue-2009], Gatekeeper[Guarnieri-2009], [Jang-2010]

Does Reliability Matter ?

- Snapshot of iFeng.com: Leading media website in China

an error occurred when processing this directive

[an error occurred while processing this directive]

李克强宣布广州亚残运会开幕

火炬手攀登点燃主火炬|数开幕式十宗"最"
亚残运开幕解密|广州亚残运会开幕式特写

广州亚运会圆满闭幕 高清大图

[组图]仁川十分钟：Rain连唱三曲|暖场演出
童谣《月光光》拉开序幕|大郅出任中国旗手

女排上演绝地逆转战胜韩国夺冠

周苏红发威女排逆转|韩国输球再斥裁判丑陋
女排逆转令洪钢哽咽|俞觉敏：我为队员骄傲

[高清]冠军球员搭讪礼仪小姐

裁判引导韩朝摔跤手赛场握手|摔跤精彩瞬间
男篮绝杀伊朗进决赛|朝鲜女足失冠背向升旗

- "铁血女将"黄蕴瑶暂列亚运英雄榜之首
- 中华台北选手罹癌参赛 携奖牌返家无遗憾
- 日本男女足亚运齐称霸 统治亚洲足坛获证
- 霍启刚温文尔雅态度和蔼 与郭晶晶差别大
- 快讯：广州亚运会发生第二起兴奋剂事件
- 阿联酋绝杀韩国队 将与日本争男足金牌
- 韩朝射箭选手只关注比赛 不知两国冲突

王治郅闭幕式上挥舞国旗入场

2 of 3

Talk Outline

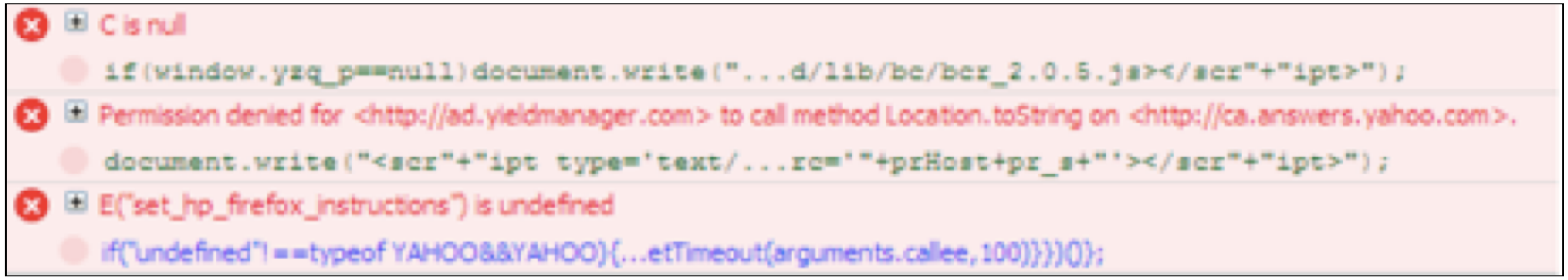
- Motivation and Goals
- Web Applications' Reliability
- IoT Reliability
- Conclusions and Future Directions

Our Prior Work [ISSRE'11]

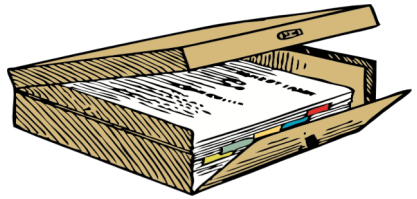
- Empirical study based on Console Error Messages: Alexa top 100

- **Popular web applications experience four distinct JavaScript errors (i.e., exceptions) on average (up to 20 errors), in normal operation**

- Many errors were non-deterministic or dependent on event order - hard to determine the root cause and impact of the error



Bug Report Study: Methodology



Collected 502 **fixed** bug reports from 19 web applications over 10 years (2004-2014)

Qualitatively analyzed and classified bug reports manually and reading the commits

Aggregated data for further analysis

Bug Report Study: 19 Objects (15 applns, 4 libraries)

Bug Report Study: Research Questions

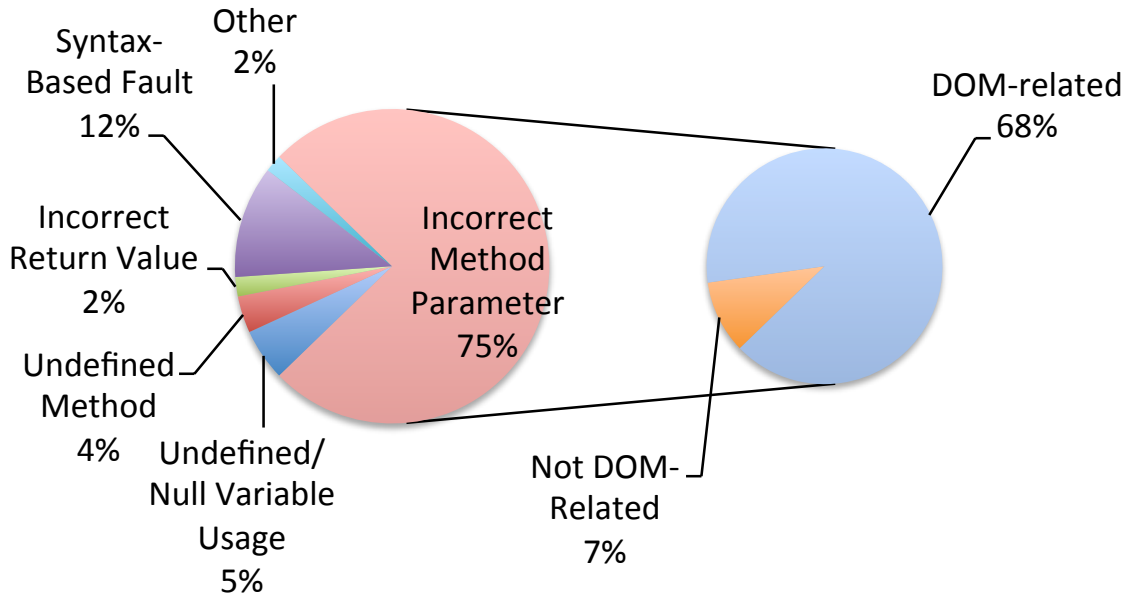
- What mistakes ***cause*** JavaScript faults?

- What ***impact*** do JavaScript faults have?

- How long does it take to fix these errors?

- How many of these faults are browser-specific ?
- How many faults can be caught by strong typing ?

Bug Report Study: Categories

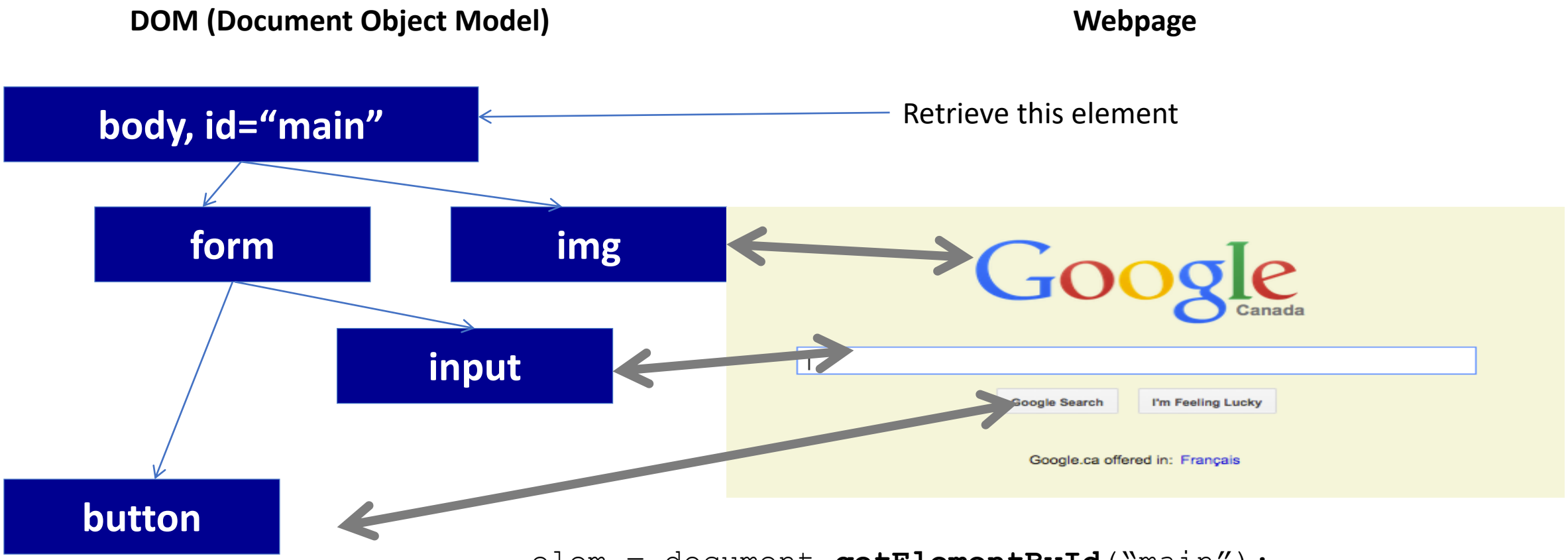


Incorrect Method Parameter Fault: Unexpected or invalid value passed to JS method or assigned to JS property

DOM-Related Fault: The method is a DOM API method

- Account for more than two-thirds of JavaScript Faults

Bug Report Study: DOM-Related Faults




```
elem = document.getElementById("main");
```

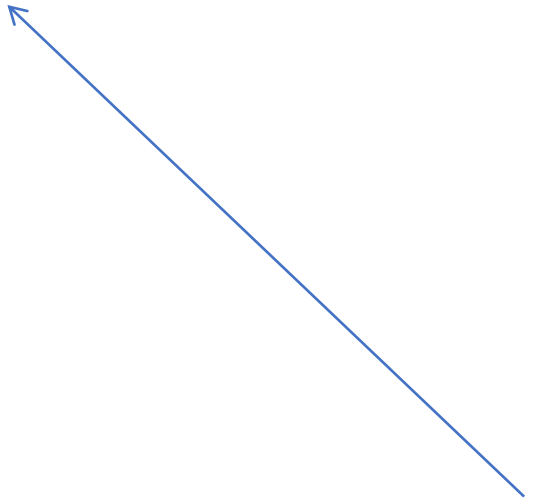
Programming Error that propagates to a DOM method parameter

DOM-Related Fault: Example

```
var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
      var prefix = "id_";
      elem = $("#" + prefix + retrievedStr);
}
else {
      elem = $(retrievedStr);
}
elem[0].focus()
```

DOM-Related Fault: Example

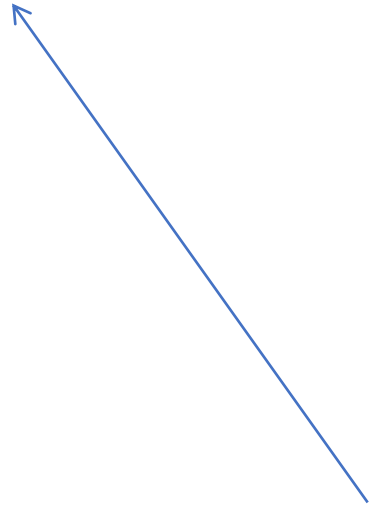
```
var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
      var prefix = "id_";
      elem = $("#" + prefix + retrievedStr);
}
else {
      elem = $(retrievedStr);
}
elem[0].focus()
```



Retrieved string via XHR

DOM-Related Fault: Example

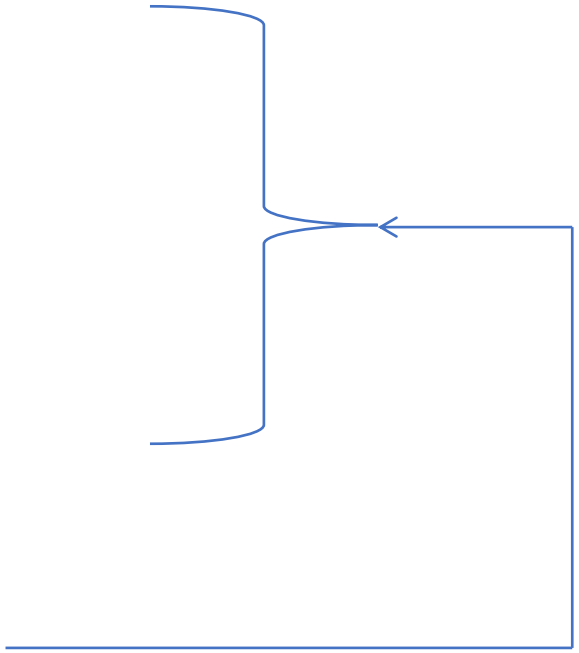
```
var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
      var prefix = "id_";
      elem = $("#" + prefix + retrievedStr);
}
else {
      elem = $(retrievedStr);
}
elem[0].focus()
```



Find the number of dots in the string

DOM-Related Fault: Example

```
var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
      var prefix = "id_";
      elem = $("#" + prefix + retrievedStr);
}
else {
      elem = $(retrievedStr);
}
elem[0].focus()
```

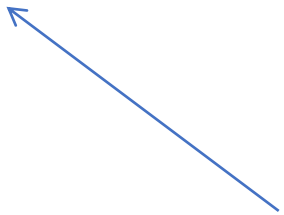


If there are no dots, prepend “id_” to the string and access it via $(). Otherwise, leave it as is, and access it via $().

DOM-Related Fault: Example

```
var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
      var prefix = "id_";
      elem = $("#" + prefix + retrievedStr);
}
else {
      elem = $(retrievedStr);
}
elem[0].focus()
```

UNDEFINED EXCEPTION!



Retrieved string of “editor” would go here even though it has no dots, which would erroneously cause $() to use selector “editor”, which doesn’t match any elements.

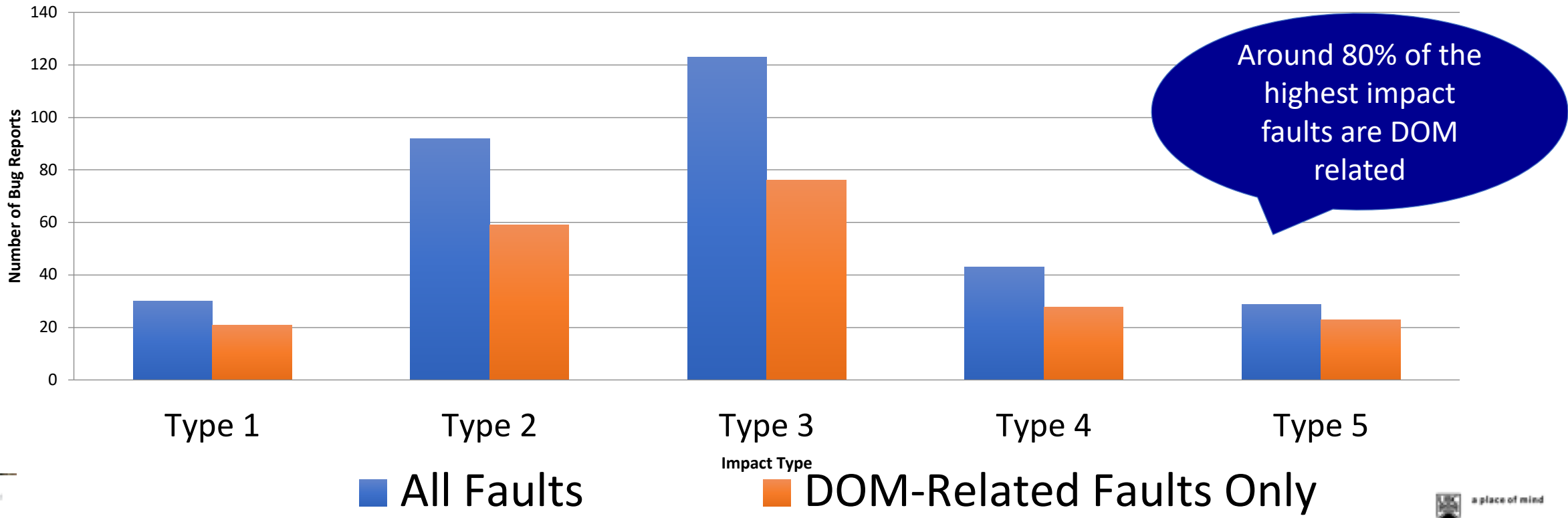
DOM-Related Fault: Example

```
var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
      var prefix = "id_";
      elem = $("#" + prefix + retrievedStr);
}
else {
      elem = $(retrievedStr);
}
elem[0].focus()
```

BUG: The assigned value should be retrievedStr.split(“.”).length – 1, as length() always returns at least 1.

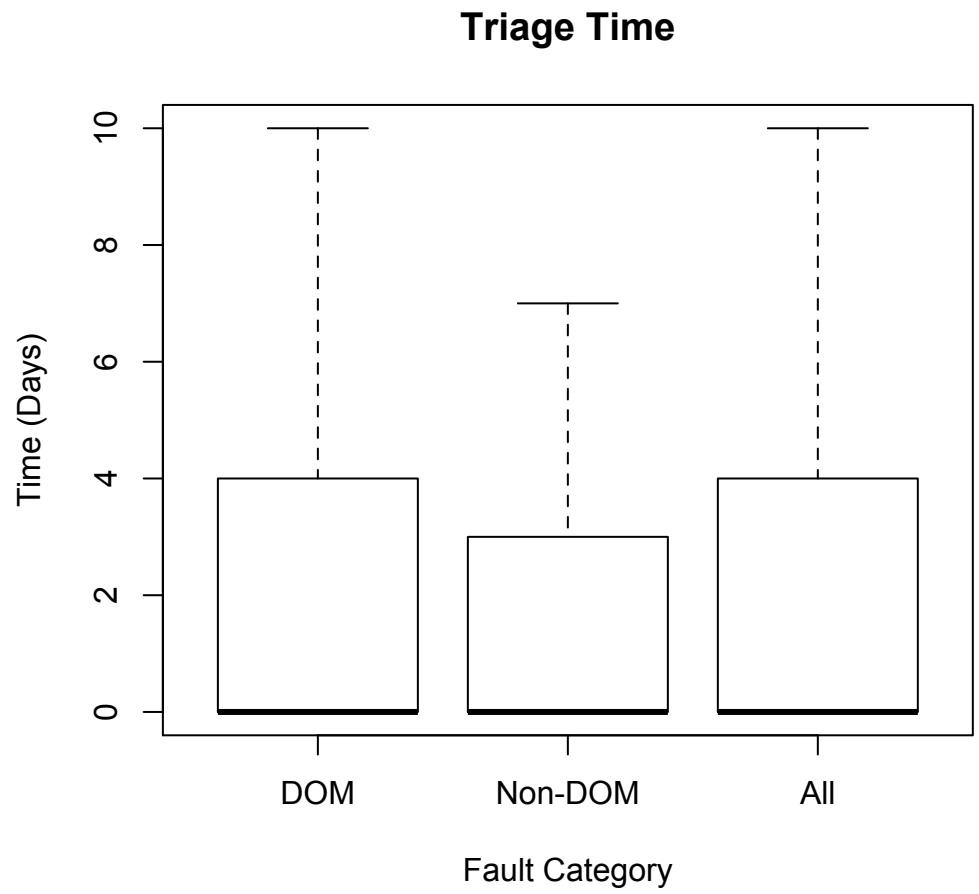
Bug Report Study: Fault Impact

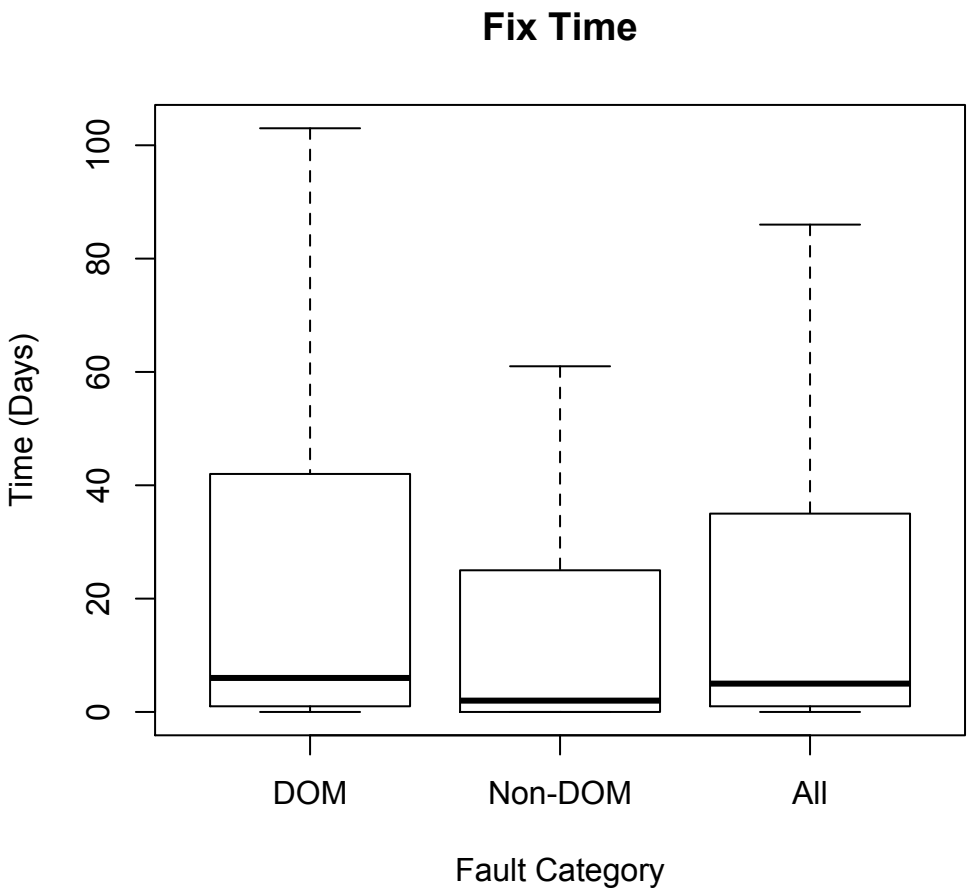
- Impact Types – Based on Bugzilla [ICSE'11]
  - Type 1 (lowest impact – e.g., cosmetic changes)
  - Type 5 (highest impact – e.g., data loss bugs)



Bug Report Study: Fix Times

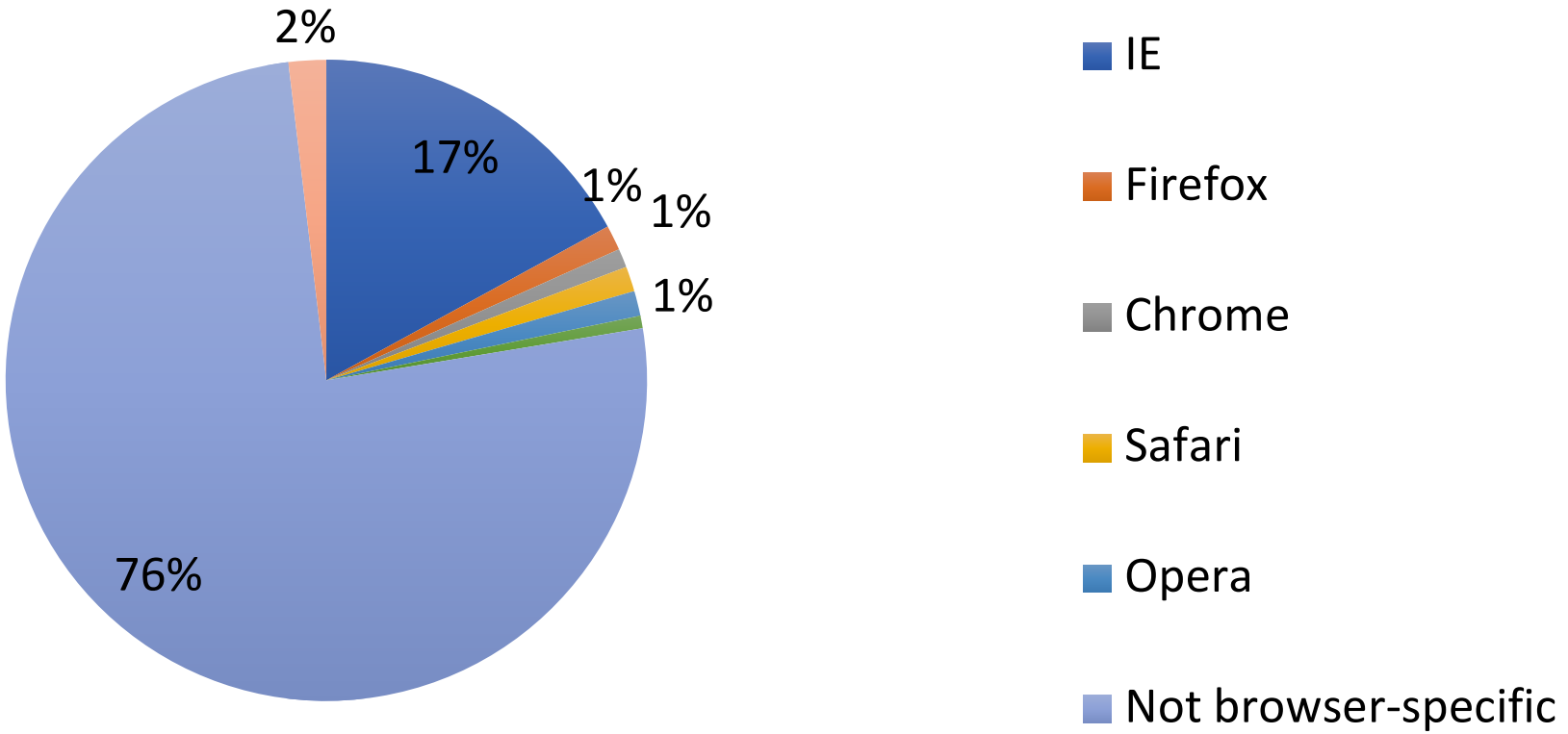
DOM-related faults take much longer to fix than non-DOM faults





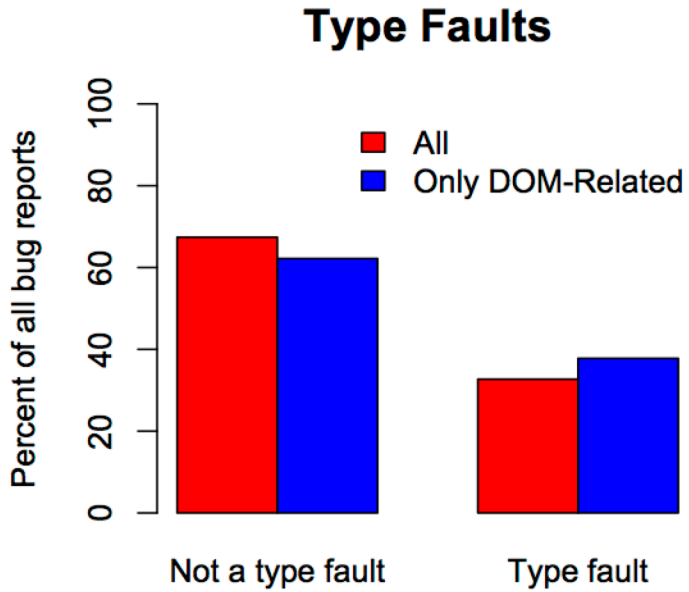
Bug Report Study: Browser Specificity

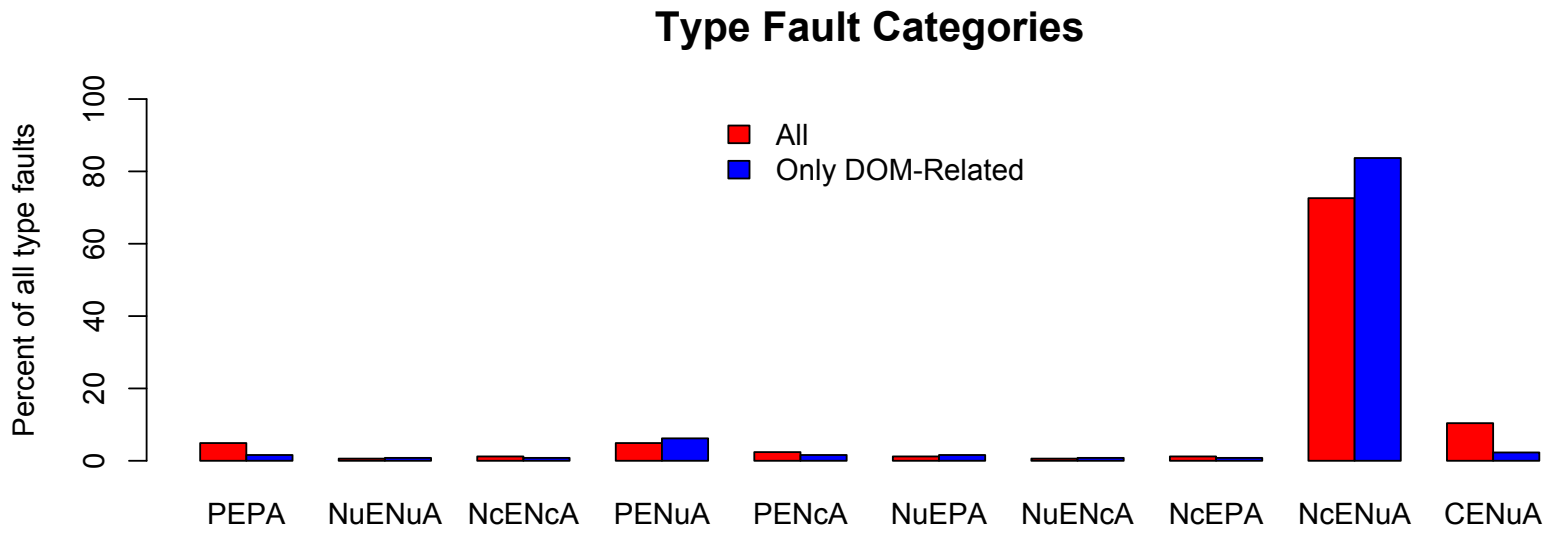
Most JavaScript faults are not browser-specific



Bug Report Study: Type Faults

- Most DOM-related faults are NOT type faults
  - Among the type faults, a single category dominates





Bug Report Study: Summary

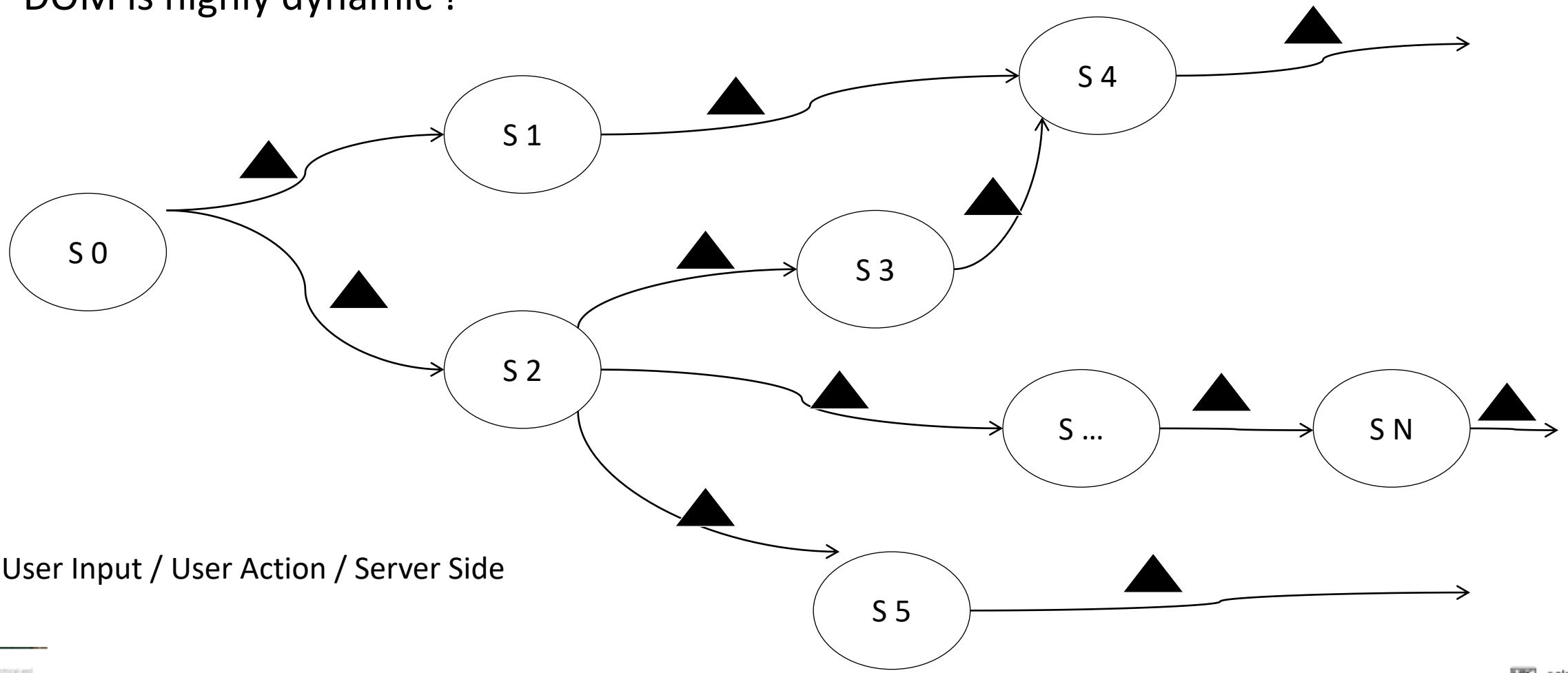
- **Bug report study of 19 applications: JS faults**
  - Over 500 bug reports analyzed; only fixed bugs considered

- **DOM-related faults dominate JavaScript faults**
  - Responsible for nearly two-thirds of all JavaScript faults
  - Responsible for 80% of highest impact faults
  - Take 50% longer time to fix for developers
  - Majority are not specific to web browser platform
  - Most DOM-related faults are NOT type errors

Need techniques to detect and prevent DOM-related faults

Web Applications: Challenge

DOM is highly dynamic !



User Input / User Action / Server Side

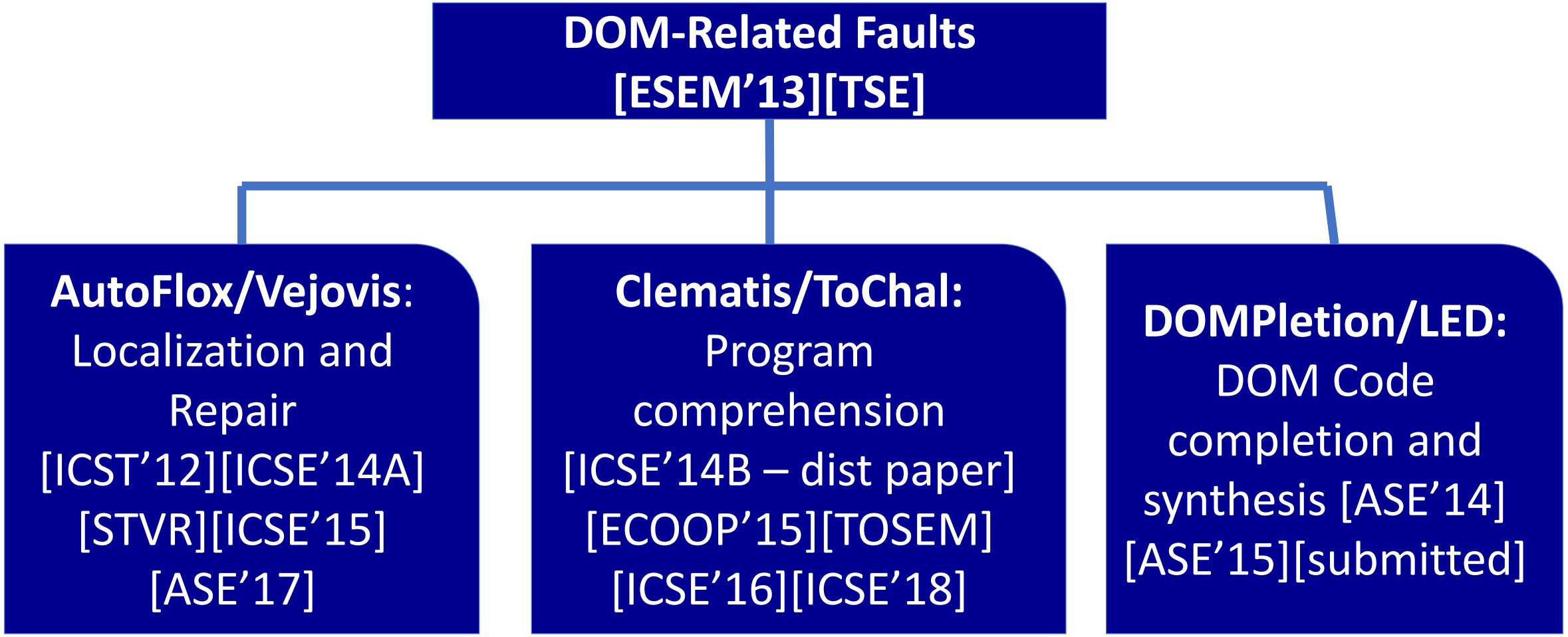
Web Applications: Existing Techniques

- **Add gradual typing to JavaScript (e.g., TypeScript, DART, Flow)**
  - Typically ignore the DOM or provide only limited support

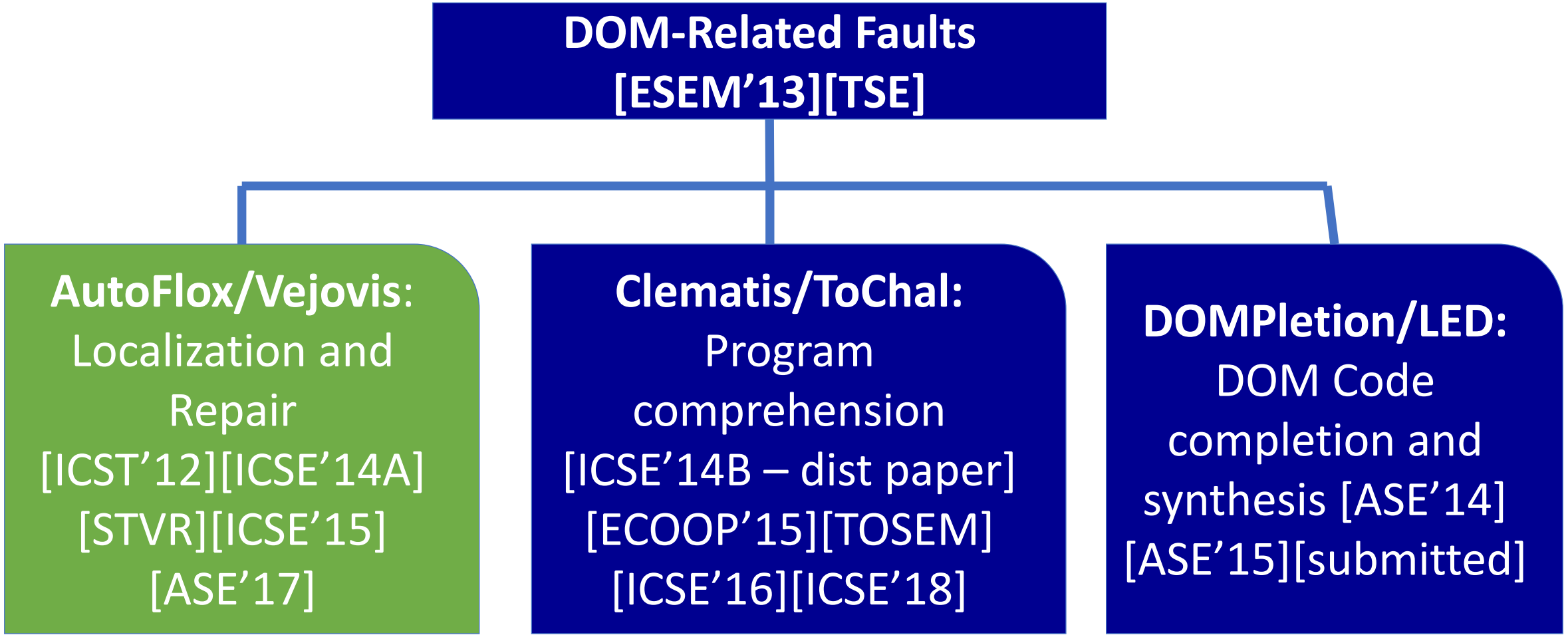
- **Use higher-level programming idioms in JavaScript**
  - MVC Frameworks (e.g., AngularJS)
  - Functional Reactive Programming (e.g., RxJS)

- **Detecting errors in web applications: Ignore DOM**
  - Race conditions [Vechev-OOPSLA'13][Livshits-FSE'15]
  - Type Coercion Errors [Pradel – ICSE'15][Moeller – OOPSLA'14]

Web Applications: Reliability Improvement



Web Applications: Reliability Improvement



Web Applications: Program Repair (Vejovis)

WRONG

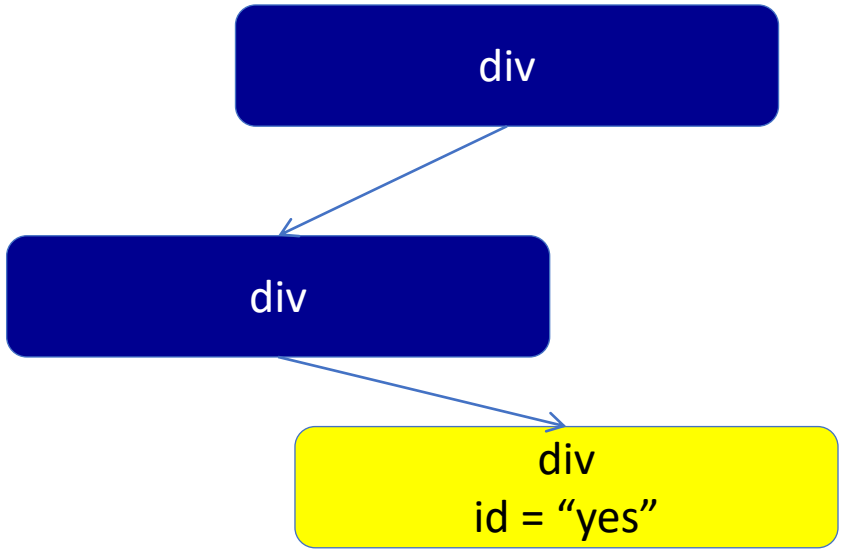
```
getElementById(“no”)
```

RIGHT

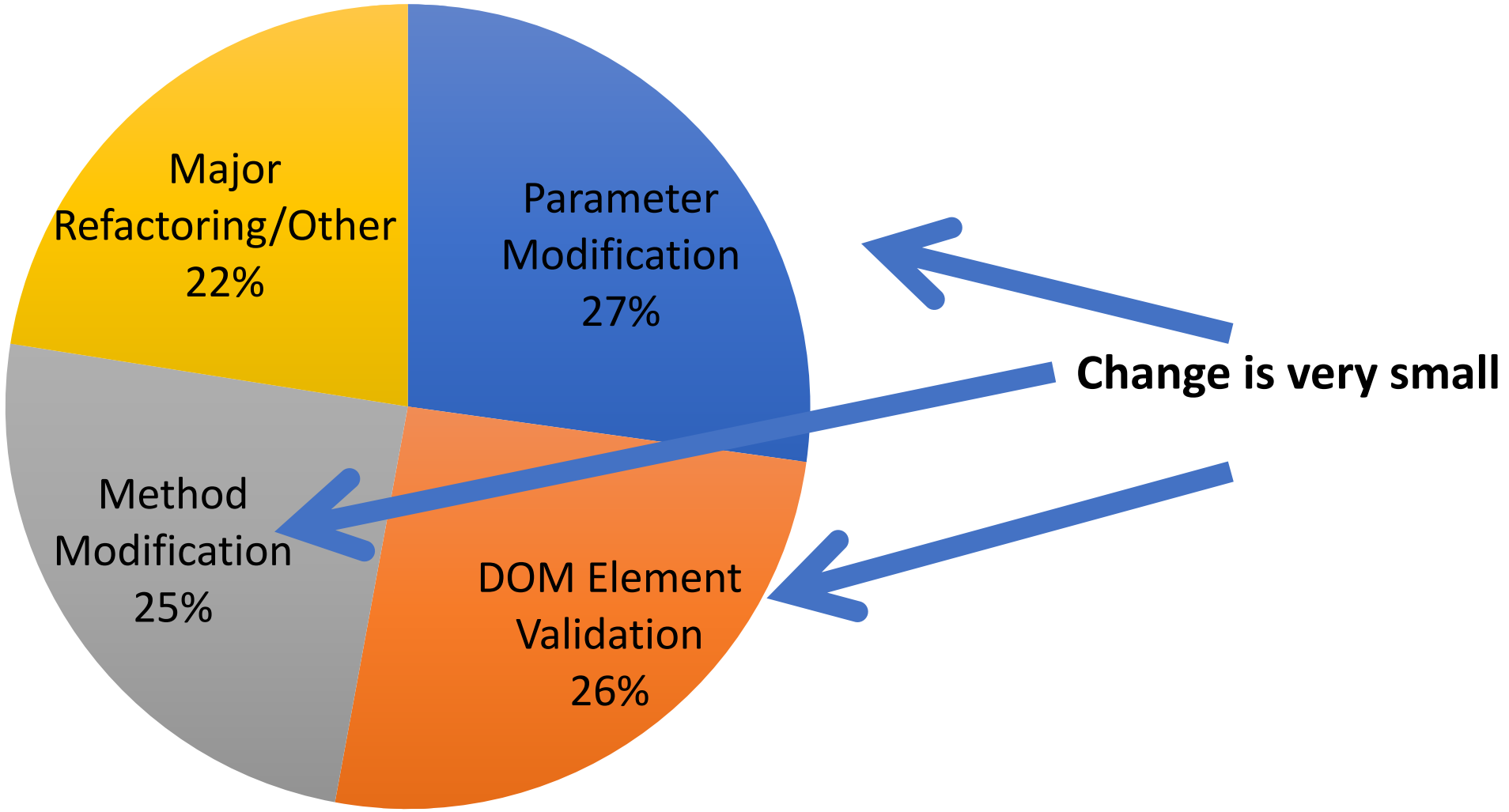
```
getElementById(“yes”)
```

Question: How do we know that we should replace “no” with “yes”

Answer: We use the DOM structure, and the JavaScript code structure

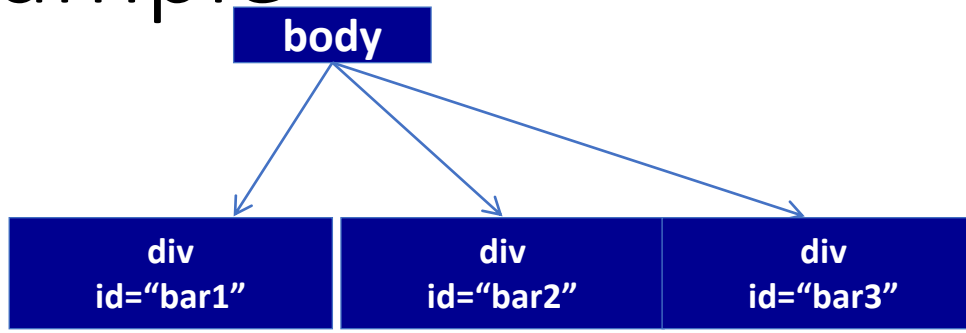


Web Applications: Bug Fix Patterns



Web Applications: Running Example

```
1  function generateId(index) {
2            var prefix = "bar";
3            var id = prefix + index;
4            return id;
5  }
6
7  function retrieveElement(index) {
8            var id = generateId(index);
9            var e = document.getElementById(id);
10           return e;
11 }
12
13 for (var i = 1; i <= 4; i++) {
14           var elem = retrieveElement(i);
15           elem.innerHTML = "Item #" + i;
16 }
```



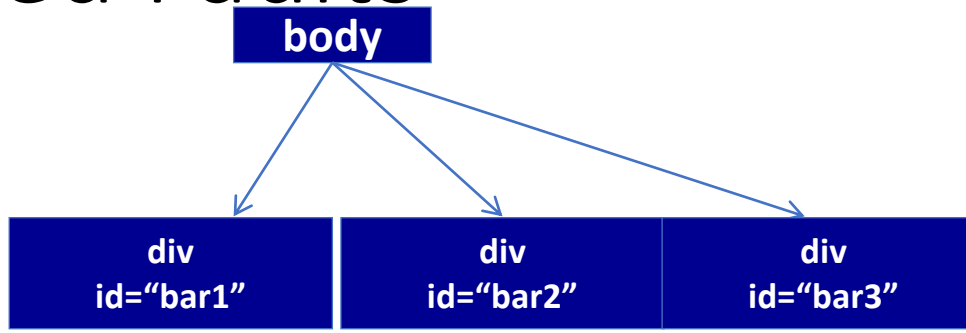
Add the "bar" prefix to the ID

Retrieve the element with index i

Update retrieved element

Web Applications: DOM-Related Faults

```
1   function generateId(index) {
2               var prefix = "bar";
3               var id = prefix + index;
4               return id;
5   }
6
7   function retrieveElement(index) {
8               var id = generateId(index);
9               var e = document.getElementById(id);
10              return e;
11  }
12
13  for (var i = 1; i <= 4; i++) {
14              var elem = retrieveElement(i);
15              elem.innerHTML = "Item #" + i;
16  }
```



This should be "<", not "<="

Evaluates to "bar4" in 4th iteration

NULL EXCEPTION!

AutoFlox: Fault Localization [ICST'12][STVR]

```
1   function generateId(index) {
2               var prefix = "bar";
3               var id = prefix + index;
4               return id;
5   }
6
7   function retrieveElement(index) {
8               var id = generateId(index);
9               var e = document.getElementById(id)
10              return e;
11  }
12
13  for (var i = 1; i <= 4; i++) {
14              var elem = retrieveElement(i);
15              elem.innerHTML = "Item #" + i;
16  }
```

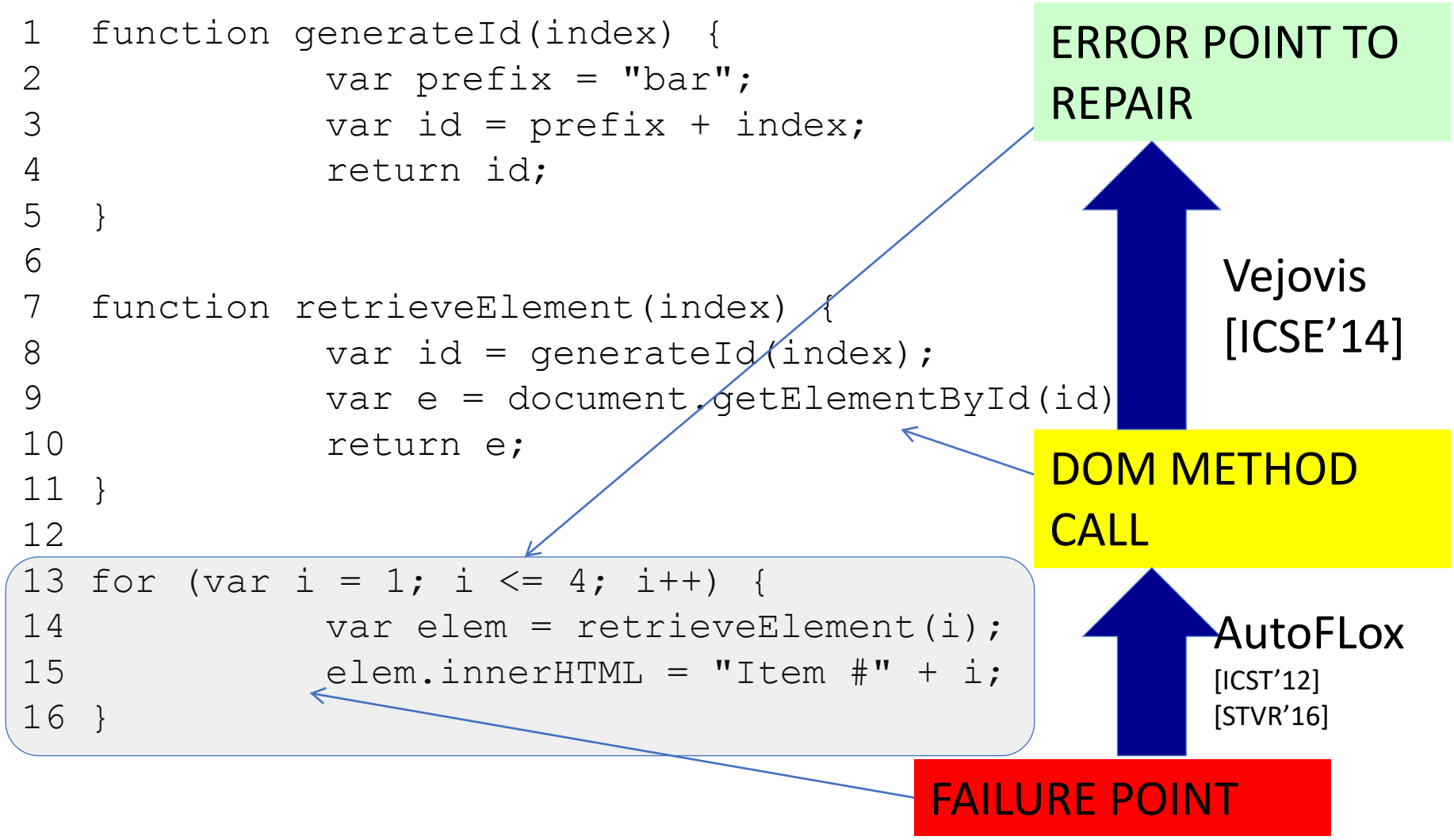
ERROR POINT TO REPAIR

Our Goal

FAILURE POINT

AutoFlox: Fault Localization [ICST'12][STVR]

```
1   function generateId(index) {
2               var prefix = "bar";
3               var id = prefix + index;
4               return id;
5   }
6
7   function retrieveElement(index) {
8               var id = generateId(index);
9               var e = document.getElementById(id)
10              return e;
11  }
12
13  for (var i = 1; i <= 4; i++) {
14              var elem = retrieveElement(i);
15              elem.innerHTML = "Item #" + i;
16  }
```



Vejovis: Fault Repair [ICSE'14]

```
1  function generateId(index) {
2              var prefix = "bar";
3
4
5  }
6
7  f
8
9
10
11 }
12
13 for (var i = 1; i <= 4; i++) {
14             var elem = retrieveElement(i);
15             elem.innerHTML = "Item #" + i;
16 }
```

"OFF-BY-ONE, SO REMOVE LAST ITERATION OF FOR LOOP"

"bar"

and
ady
in
ions

Evaluates to "bar4"

"4"

"4" comes from iterator i

Web Applications: Vejovis Actions

- Determine **action** for programmer to match valid DOM element selectors (based on fix patterns)
  - Use String Solver (HAMPI) [Kiezun09] to find the action
  - Rank fixes based on their edit distances from original

MESSAGE TYPES
REPLACE STRING
REPLACE STRING AT ITERATION
OFF BY ONE AT BEGINNING
OFF BY ONE AT END
MODIFY UPPER BOUND
EXCLUDE ITERATION
ENSURE THAT

Web Applications: Vejovis Recall

Subject	Bug Report #1	Bug Report #2
Drupal		
Ember.js		
Joomla		
jQuery		X
Moodle		
MooTools		
Prototype		
Roundcube		X
TYPO3		
WikiMedia		
WordPress		

We consider a fix to be correct if and only if it matches the programmers' fix for the bug.

Overall Recall: (20/22) = 91%

Web Applications: Vejovis Precision

Subject	Bug Report #1	Bug Report #2
Drupal	31 / 40	1 / 4
Ember.js	1 / 2	1 / 3
Joomla	1 / 88	1 / 88
jQuery	2 / 108	-
Moodle	2 / 37	1 / 37
MooTools	2 / 2	1 / 2
Prototype	1 / 6	1 / 2
Roundcube	4 / 79	-
TYPO3	1 / 187	1 / 1
WikiMedia	6 / 24	1 / 71
WordPress	13 / 30	1 / 170

Many suggestions provided for some cases

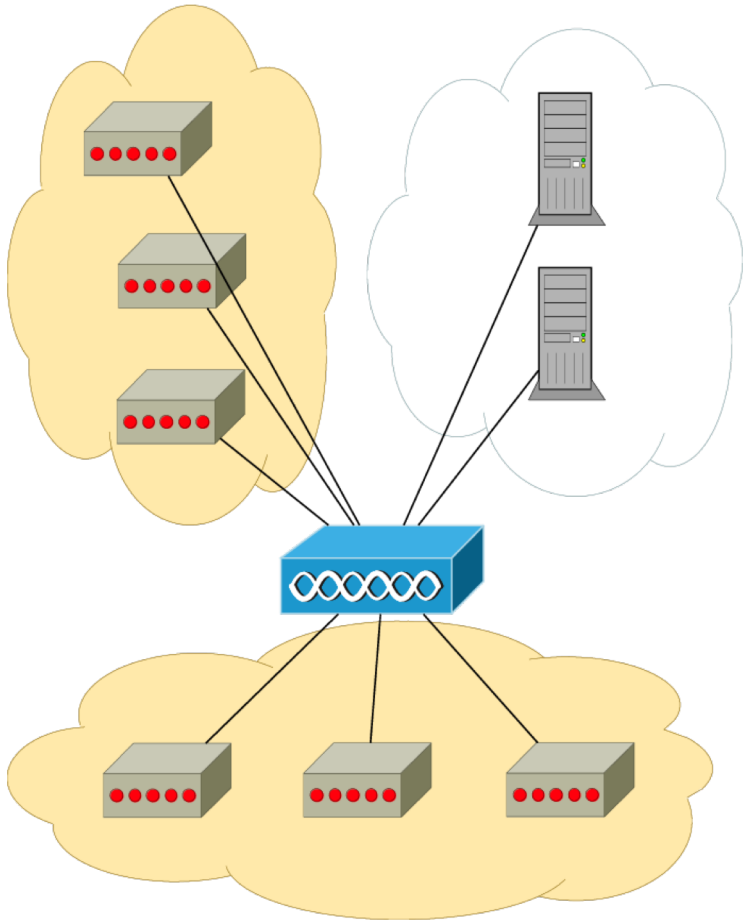
Ranking function for fix

#1 Ranking in 13 out of 20 bugs (#2 in 3)

Talk Outline

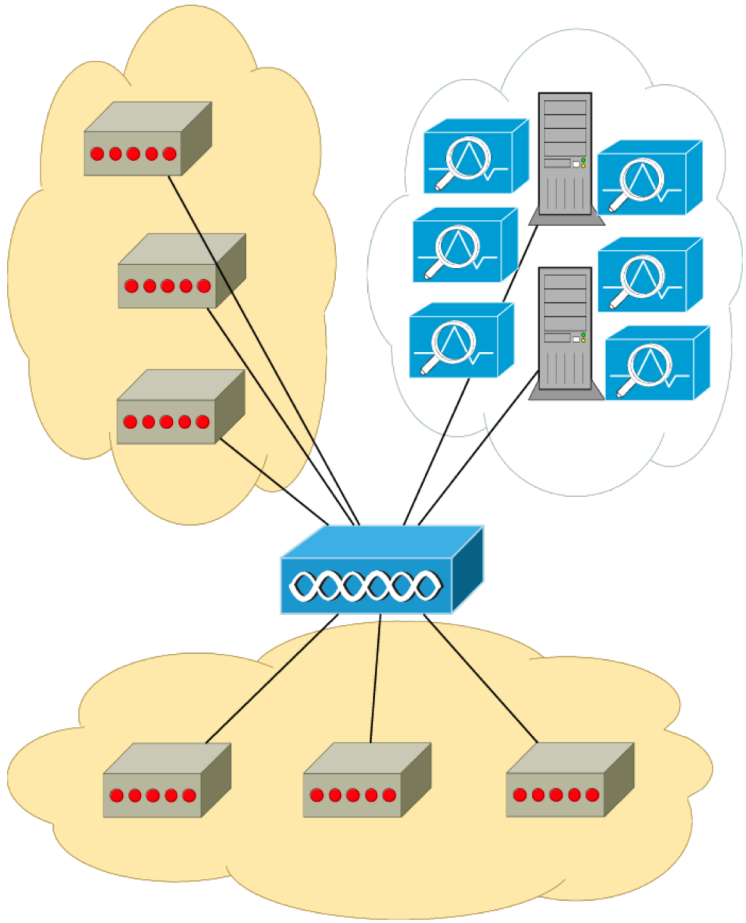
- Motivation and Goals
- Web Applications' Reliability
- IoT Reliability
- Conclusions and Future Directions

Motivation



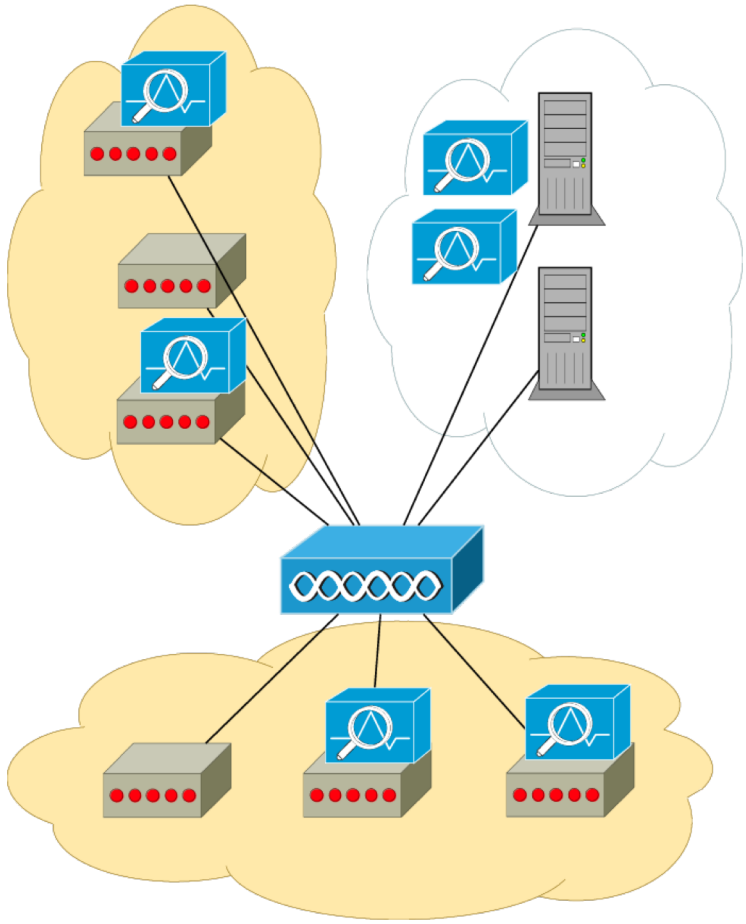
- World of IoT growing very fast

Motivation



- World of IoT growing very fast
- Traditionally, processing was done in the cloud

Motivation

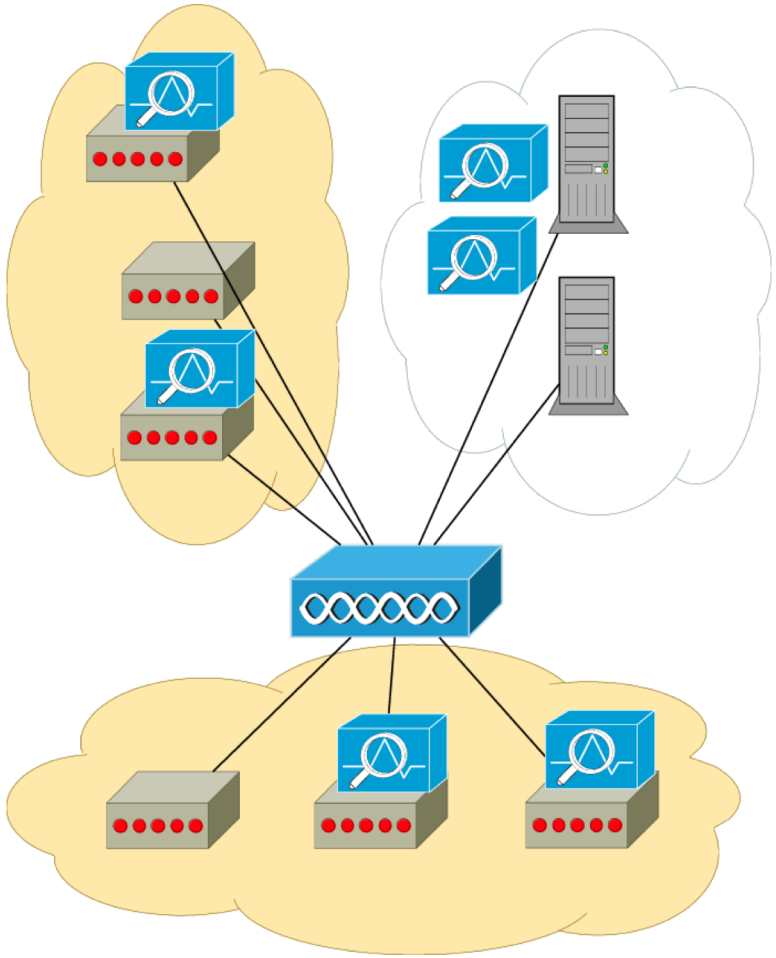


- World of IoT growing very fast
- Traditionally, processing was done in the cloud
- Emerging trend: running applications on the IoT devices themselves (edge)
  - Performance, costs, reliability

Goals

- **ThingsJS**: a framework for developing and deploying *high-level* applications on IoT devices

- **JavaScript !**

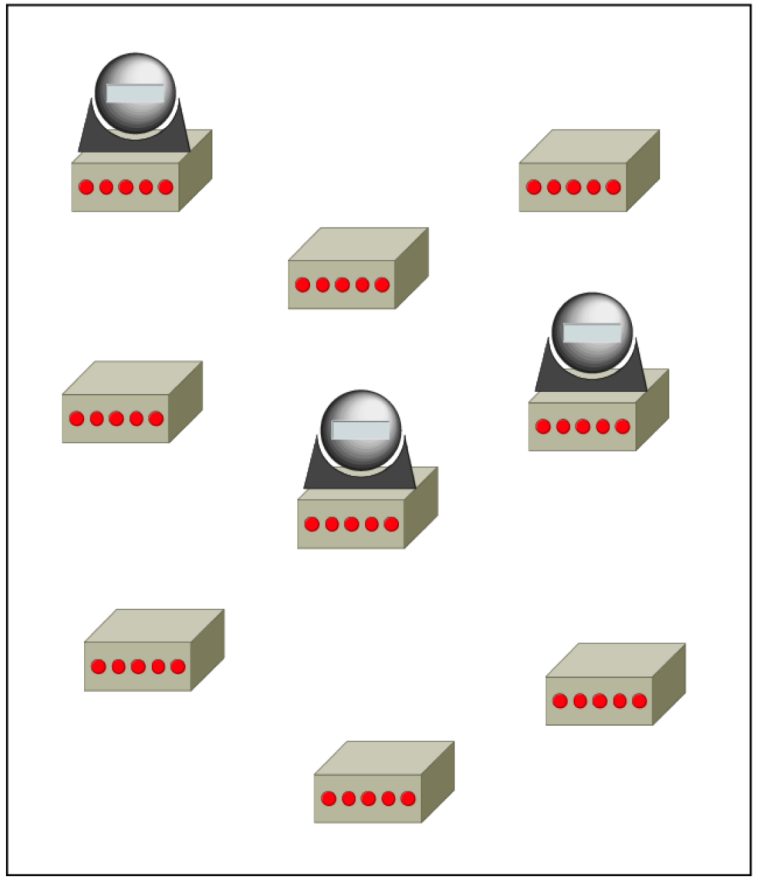


Goals

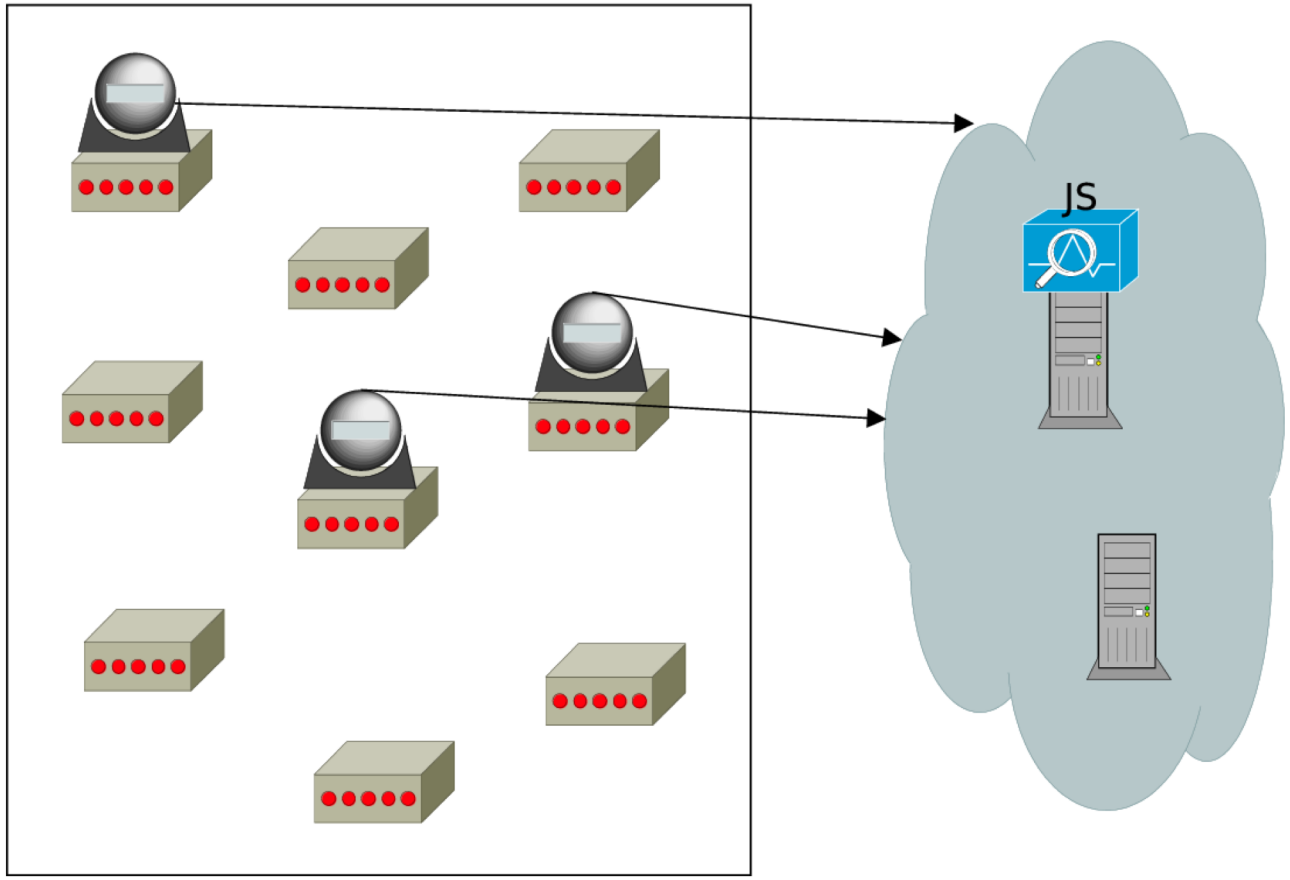
- News Flash: JavaScript VMs can run on IoT !
  - Samsung IoT.js
  - Intel XDK
  - DukServer
  - Smart.js
  - **Node.js / V8 (Google Chrome)**
  - **ChakraCore (MS Edge)**
  - **SpiderMonkey (Mozilla Firefox)**

But they are hard to program and unreliable ….

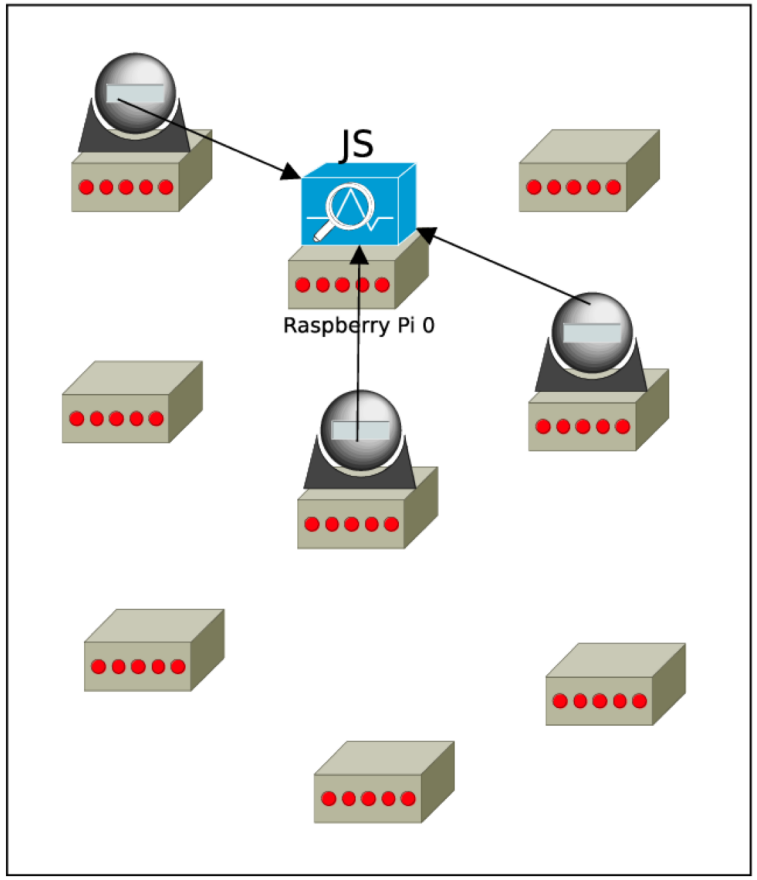
Scenario: Video Surveillance



Scenario: Video Surveillance



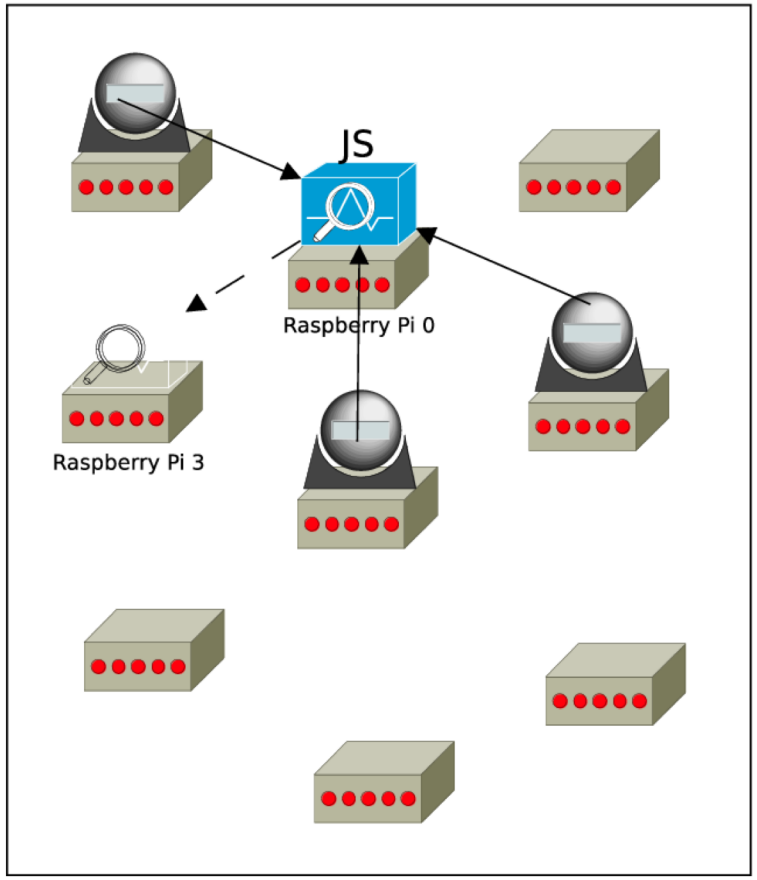
Scenario: Video Surveillance



ThingsJS:

Executing High-Level Applications on IoT/Edge devices.

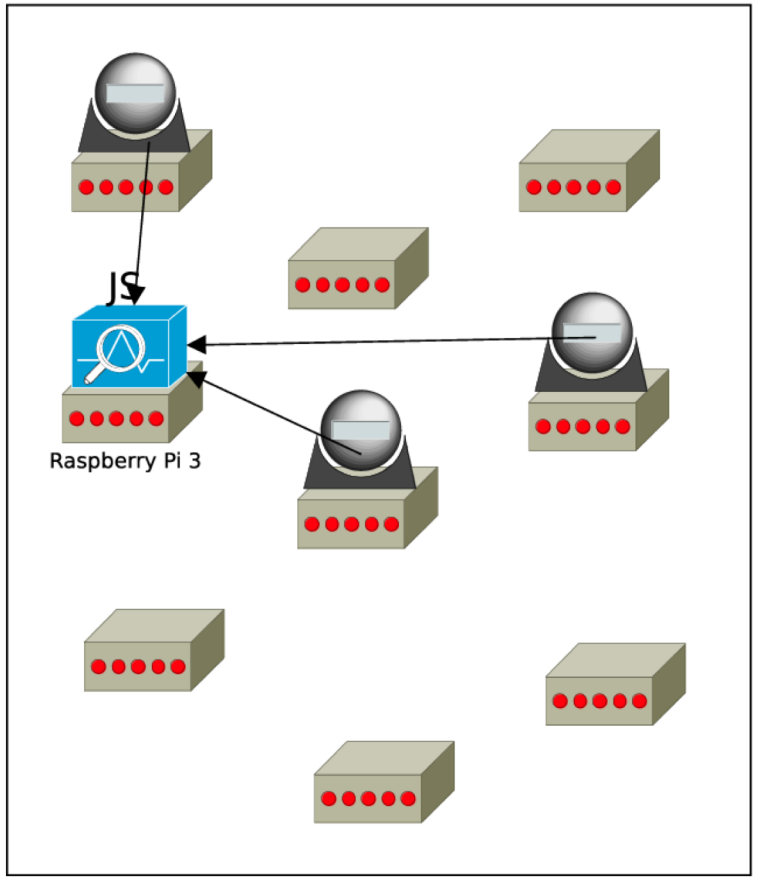
Scenario: Video Surveillance



ThingsJS:

Executing High-Level Applications on IoT/Edge devices.

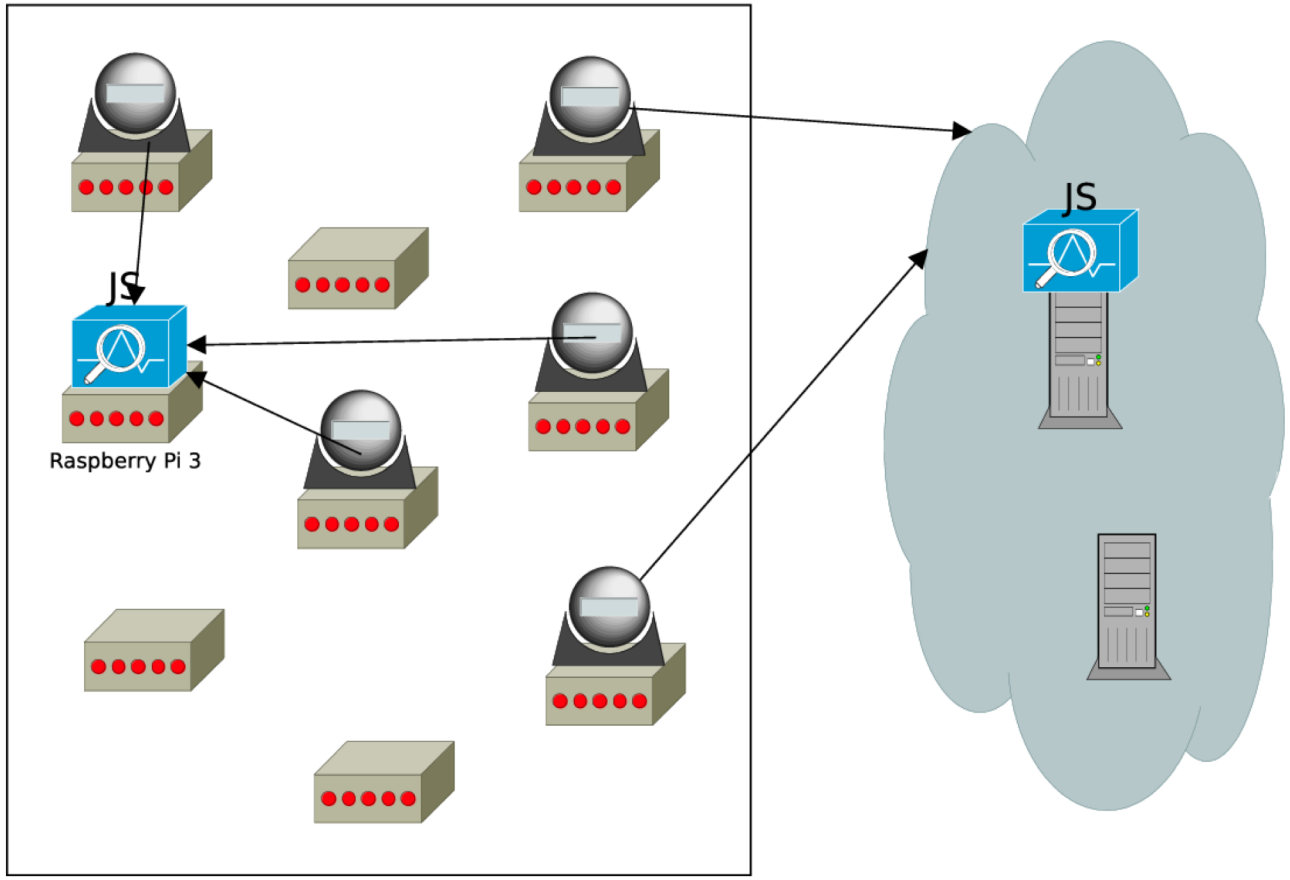
Scenario: Video Surveillance



ThingsMigrate:

Transparently migrating JS applications between IoT/edge devices.

Scenario: Video Surveillance



Challenge: Many Devices, OSes & VMs!

- **Goal: capturing the state of the JavaScript VM**
  1. Closures (data encapsulation in functions)
  2. Timers
  3. Classes & prototypes
  4. Asynchronous Model (Event-Based)

```
1  function Counter() {
2     var value = 0;
3
4     return function() {
5        value += 1;
6        return value;
7     }
8  };
9
10 var c = Counter();  // value in c is 0
11 console.log( c() ); // prints 1
12 console.log( c() ); // prints 2
13 setInterval(function() { c(); },1000);}
```

Approach: Code Instrumentation

```
1  function Counter() {
2     var value = 0;
3
4     return function() {
5        value += 1;
6        return value;
7     }
8  };
9
10 var c = Counter();  // value in c is 0
11 console.log( c() ); // prints 1
12 console.log( c() ); // prints 2
13 setInterval(function() { c(); },1000);}
```

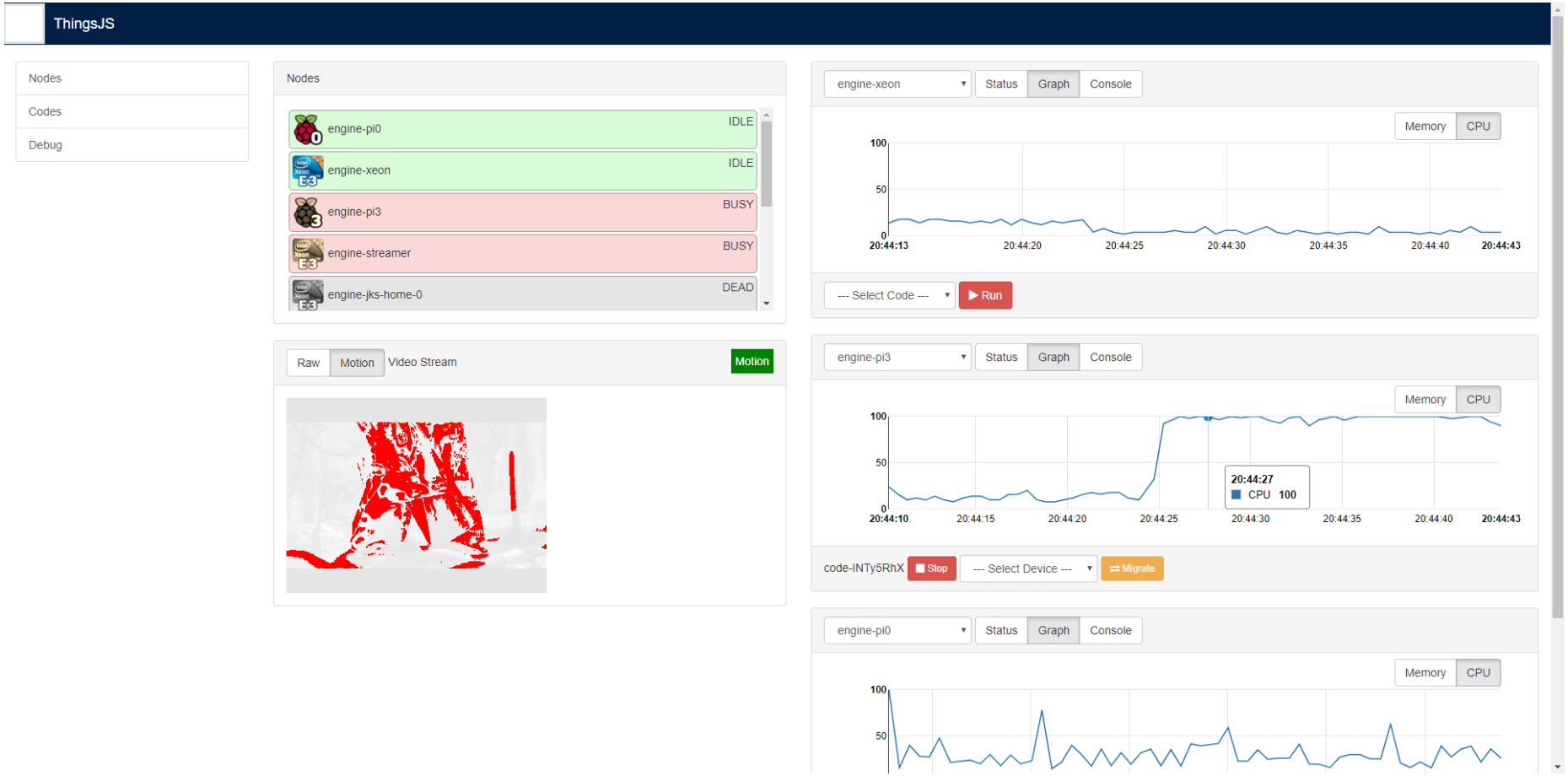
```
1  var global = new Scope("global");
2  function Counter() {
3     counter. = new Scope(global, "Counter");
4     var value = 0;
5     counter.addVar("value", value);
6
7     var anon1 = function() {
8        anon1 = new Scope(createcounters, "anon1");
9        value += 1;
10       anon1.setVar("value", value);
11
12       return value;
13    }
14
15    counter.addFunction("anon1", anon1);
16    return anon1;
17 };
```

ThingsMigrate: Results

- **Test Bed**
  - Devices: Raspberry Pi 3, Raspberry Pi 0, Cloud Server
  - Chrome benchmarks and real-world IoT applications

- **Overhead:**
  - Execution time (CPU): ~30%
  - Memory: About 2X overhead
  - Support for multiple migrations without additional overhead

GUI – Web Dashboard



Ongoing Efforts: ThingsJS

- Scheduling for Edge Devices
- Failure Prediction and Live Snapshotting
- Global, distributed file system
- Security constraints and isolation

Talk Outline

- Motivation and Goals
- Web Applications' reliability
- IoT Reliability
- Conclusions and Future Directions

Conclusion and Future Directions

- **Web and IoT applications represent the future of Software !**

- **Reliability is not clearly defined in these applications**
  - Need to understand what constitutes a fault or a failure
  - Need to provide transparent reliability to applications

- **Future Directions**
  - Understanding failures in edge-computing devices and applications
  - Program synthesis for generating reliable code from specifications

More info at: http://blogs.ubc.ca/karthik