

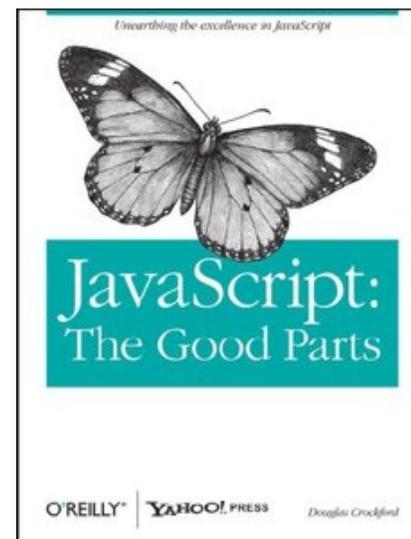
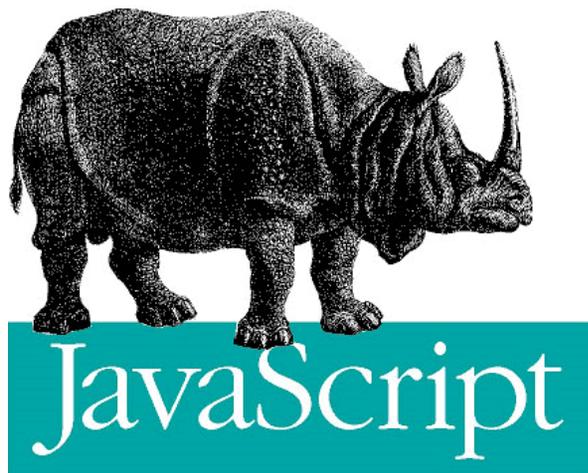
How I Learned to Stop Worrying and Love the DOM



Karthik Pattabiraman,
Frolin Ocariza, Saba Alimadadi,
Kartik Bajaj, Sheldon Sequiera, and
Ali Mesbah
University of British Columbia (UBC)

Web Applications: JavaScript

- JavaScript: Implementation of ECMAScript standard
- Executes in client's browser – thousands of lines of code
- 97 of the Alexa top 100 websites use JavaScript
- Number 1 language on both **Github** and **StackOverflow**
- Not easy to write code in – has many “evil” features



Web Applications: Prior Studies

Performance and parallelism:

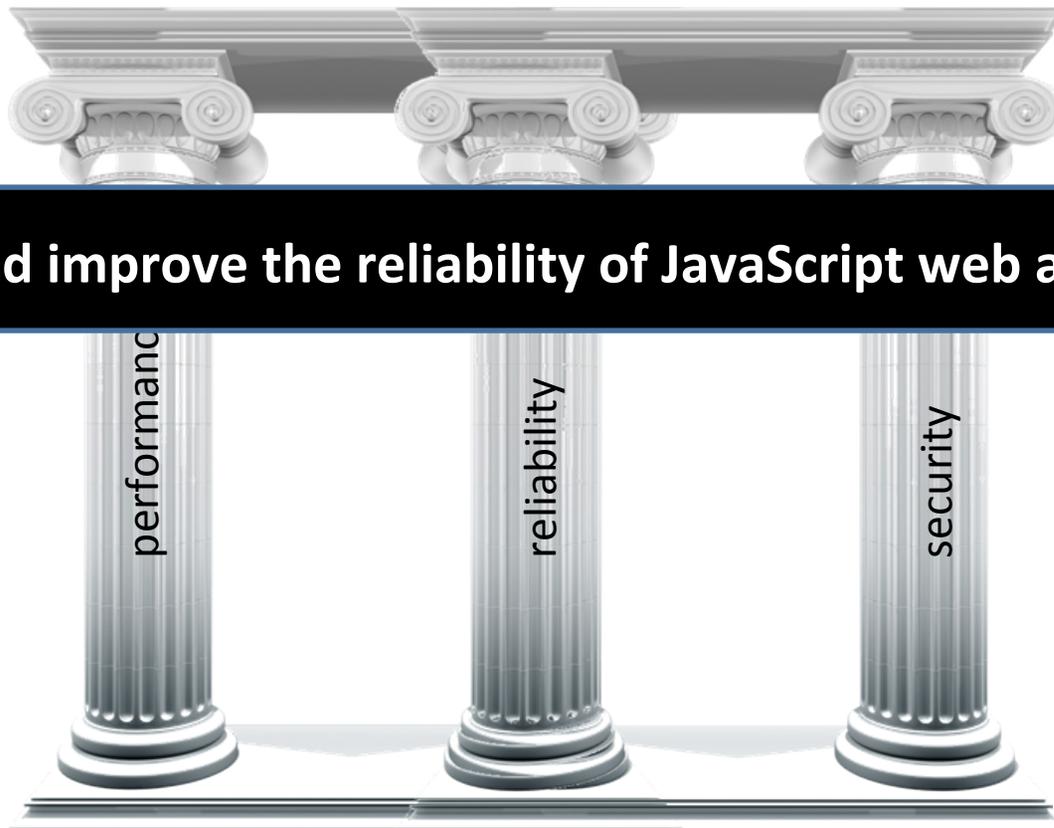
JSMeter [Ratanaworabhan-2010],
[Richards-2009], [Fortuna-2011]

Reliability

?

Security and Privacy:

[Yue-2009],
Gatekeeper[Guarnieri-2009],
[Jang-2010]



Goal: Study and improve the reliability of JavaScript web applications

Web Applications: Study Method

- Collect bug reports from bug repositories
 - Focus on bugs that are marked fixed to avoid spurious bugs
 - Manually verify that the fix involves JavaScript



Web Applications: Research Questions

- What programming errors or bugs *cause* JavaScript faults?
- What *impact* do JavaScript faults have?
- How long does it take to fix these errors?



Bug Report Study of 19 popular and open source JavaScript applications & libraries

- Over a span of 10 years
- Over 500 bug reports



Web Applications: Research Questions

- What programming errors or bugs **cause** JavaScript faults?
- What **impact** do JavaScript faults have?
- How long does it take to fix these errors?

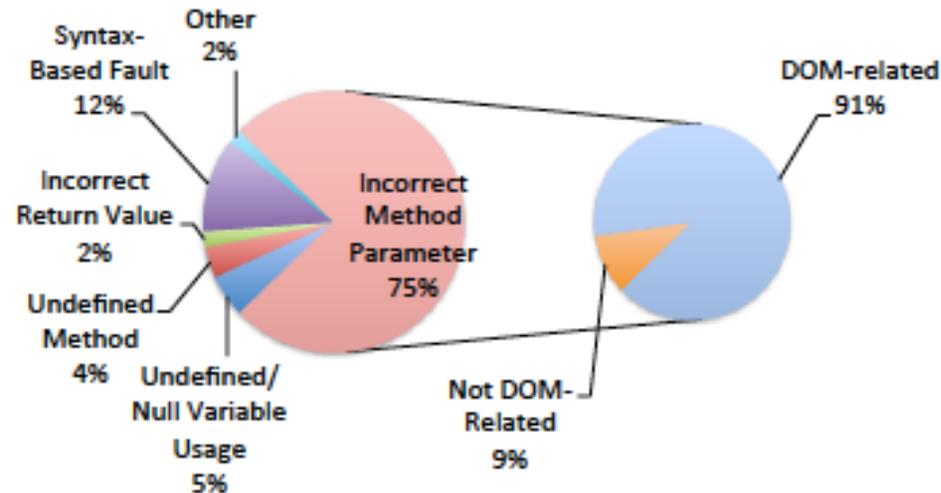


Bug Report Study of 19 popular and open source JavaScript applications & libraries

- Over a span of 10 years
- Over 500 bug reports



Web Applications: Bug Categories



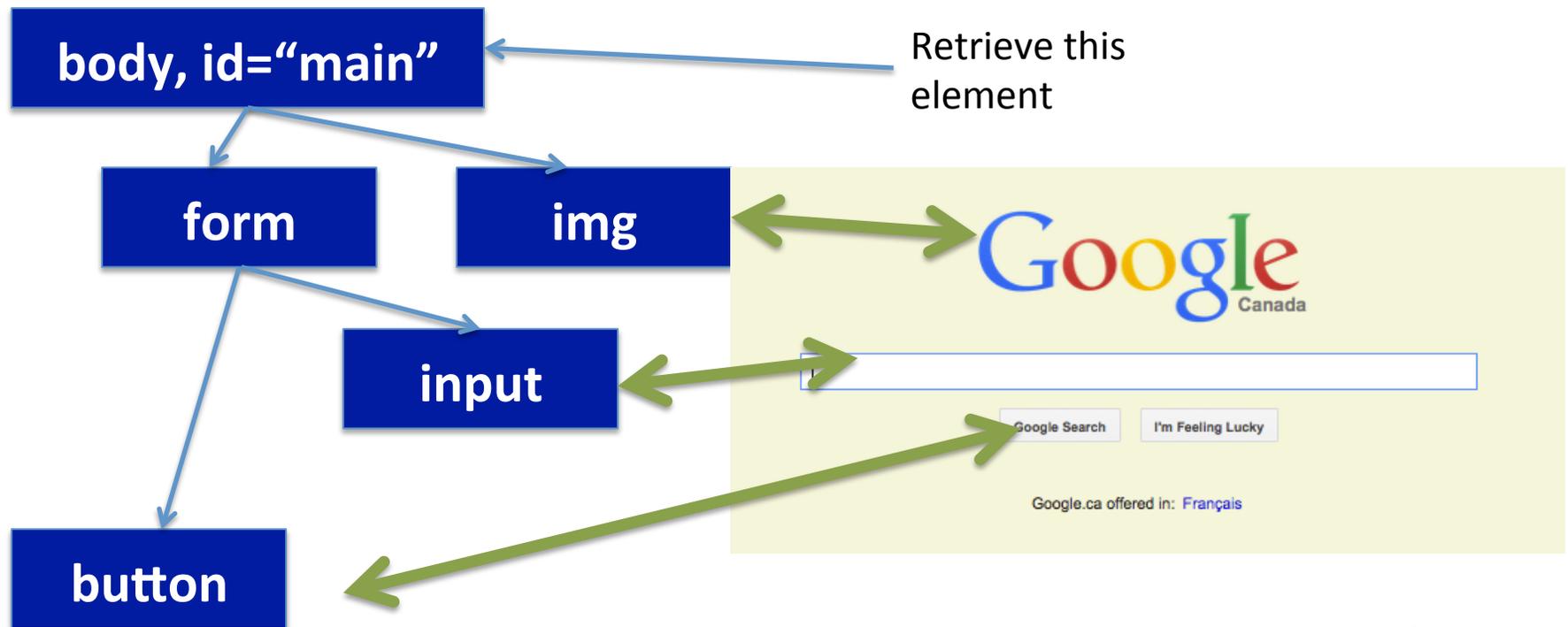
Incorrect Method Parameter Fault: Wrong or invalid value passed to some JavaScript method

DOM-Related Fault: The method is a DOM API method
- Account for nearly two-thirds of JavaScript Faults (68%)

Web Applications: DOM-Related Faults

DOM (Document Object Model)

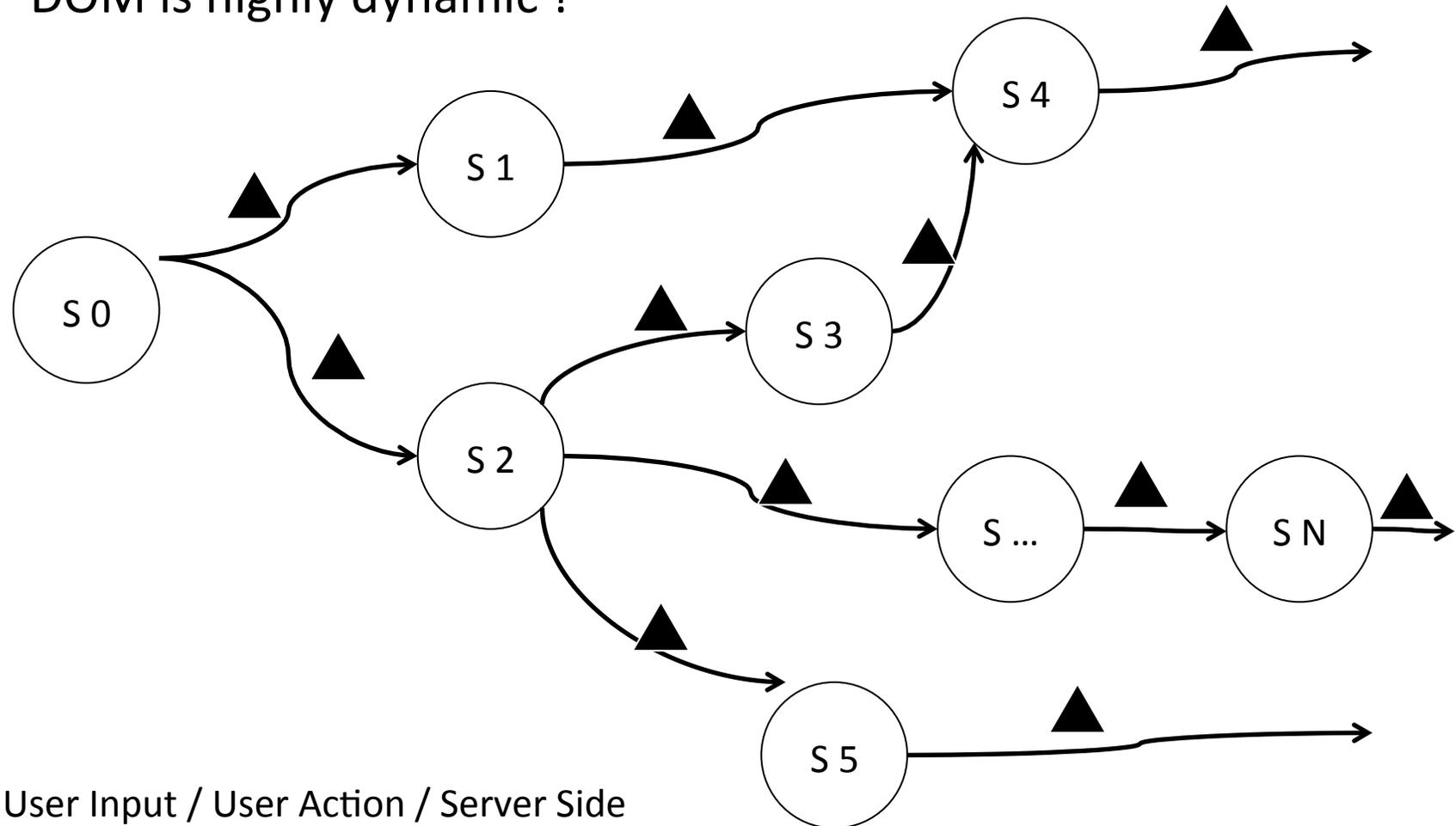
Webpage



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

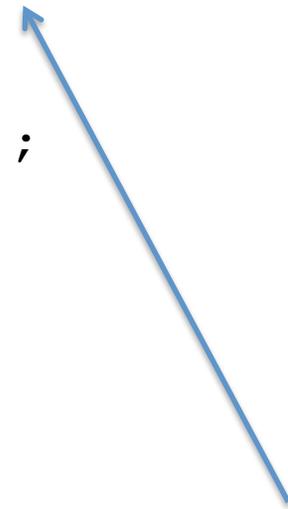
Web Applications: Challenge

DOM is highly dynamic !



DOM-Related Faults: 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();
```

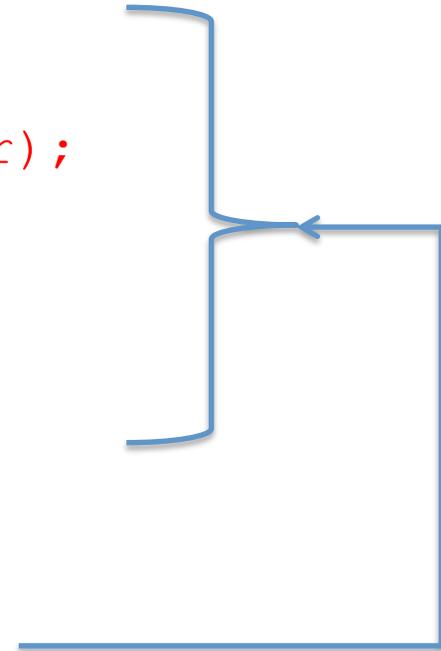


Find the number
of dots in the
string

DOM-Related Faults: 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, prefix "id_" to the string. Otherwise, leave it as is. Access the DOM element via a '\$'



DOM-Related Faults: 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 select "editor", which returns Undefined

DOM-Related Faults: 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.

Web Applications: Research Questions

- What programming errors or bugs *cause* JavaScript faults?
- What *impact* do JavaScript faults have?
- How long does it take to fix these errors?



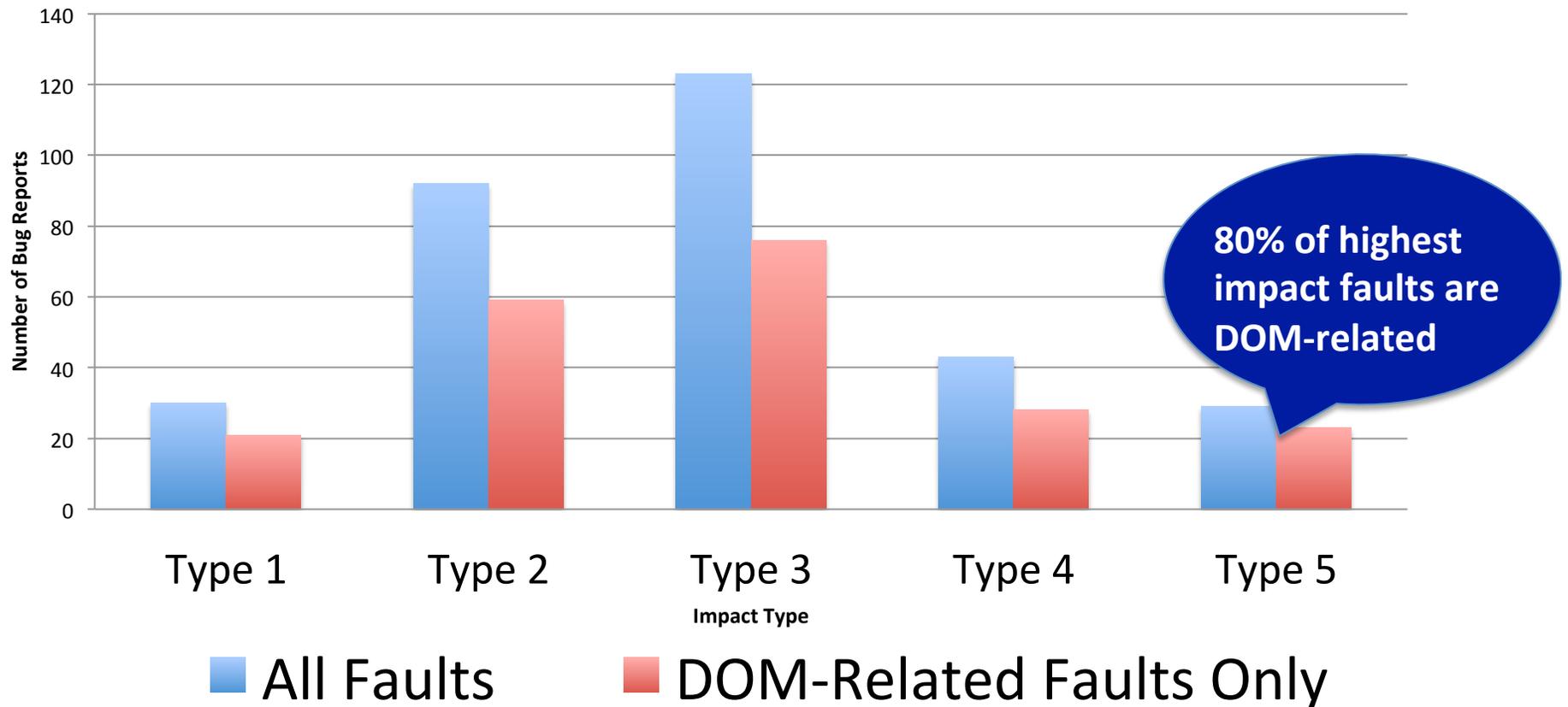
Bug Report Study of 19 popular and open source JavaScript applications & libraries

- Over a span of 10 years
- Over 500 bug reports



Web Applications: Bug Impact

- Impact Types – Based on Bugzilla’s classification
 - Type 1 (lowest impact), Type 5 (highest impact)



Web Applications: Research Questions

- What programming errors or bugs *cause* JavaScript faults?
- What *impact* do JavaScript faults have?
- How long does it take to fix these errors?



Bug Report Study of 19 popular and open source JavaScript applications & libraries

- Over a span of 10 years
- Over 500 bug reports

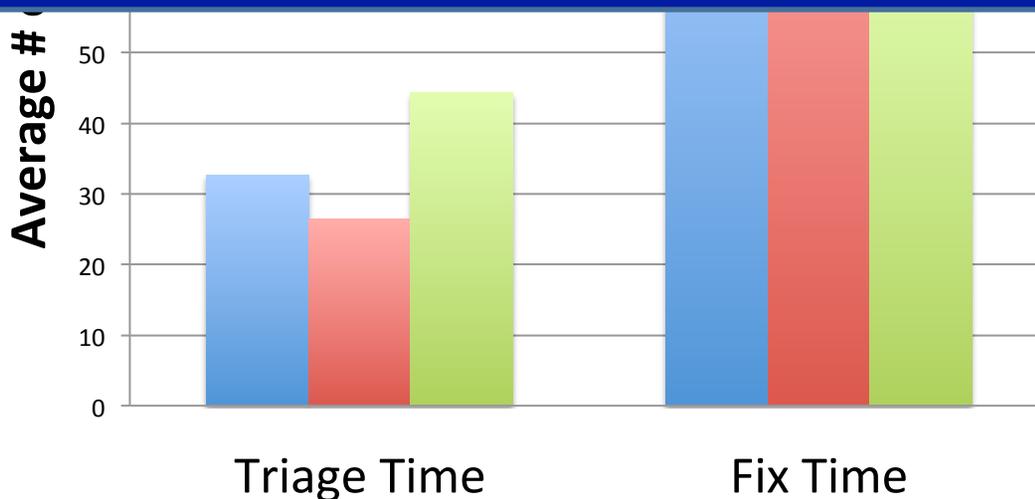


Web Applications: Bug Fix Times

- **Triage Time:** Time it took to assign or comment on the bug
- **Fix Time:** Time it took to fix the bug since it was triaged



DOM-related faults take much longer to fix, through they are triaged quicker



Web Applications: Findings Summary

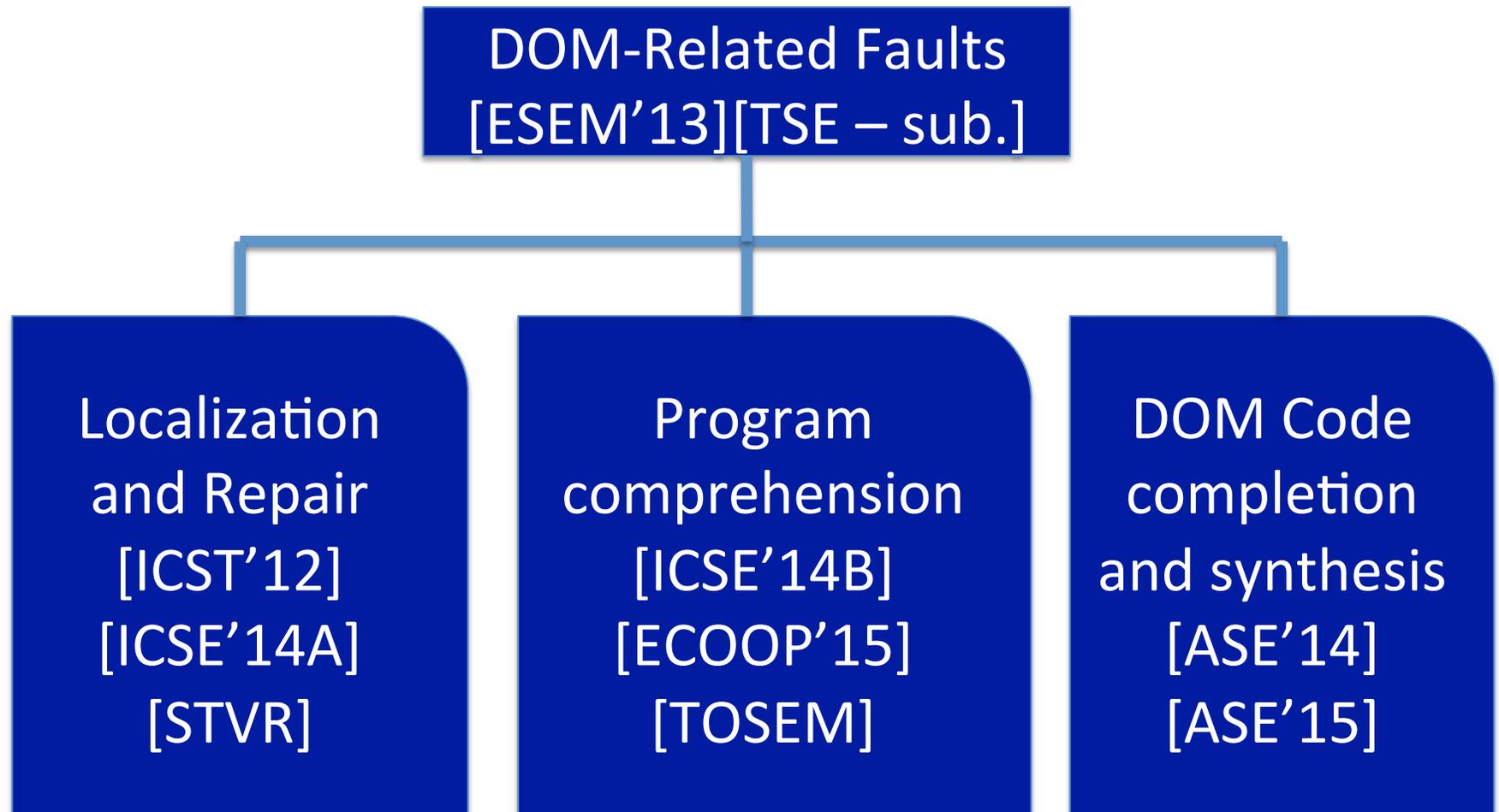
DOM-related faults dominate JavaScript faults

- Responsible for **nearly two-thirds (68%)** of all the JavaScript faults in web applications
- Responsible for **80%** of the **highest impact** faults, including security vulnerabilities
- Take **40%** longer time to fix though they're triaged quicker than non-DOM related faults

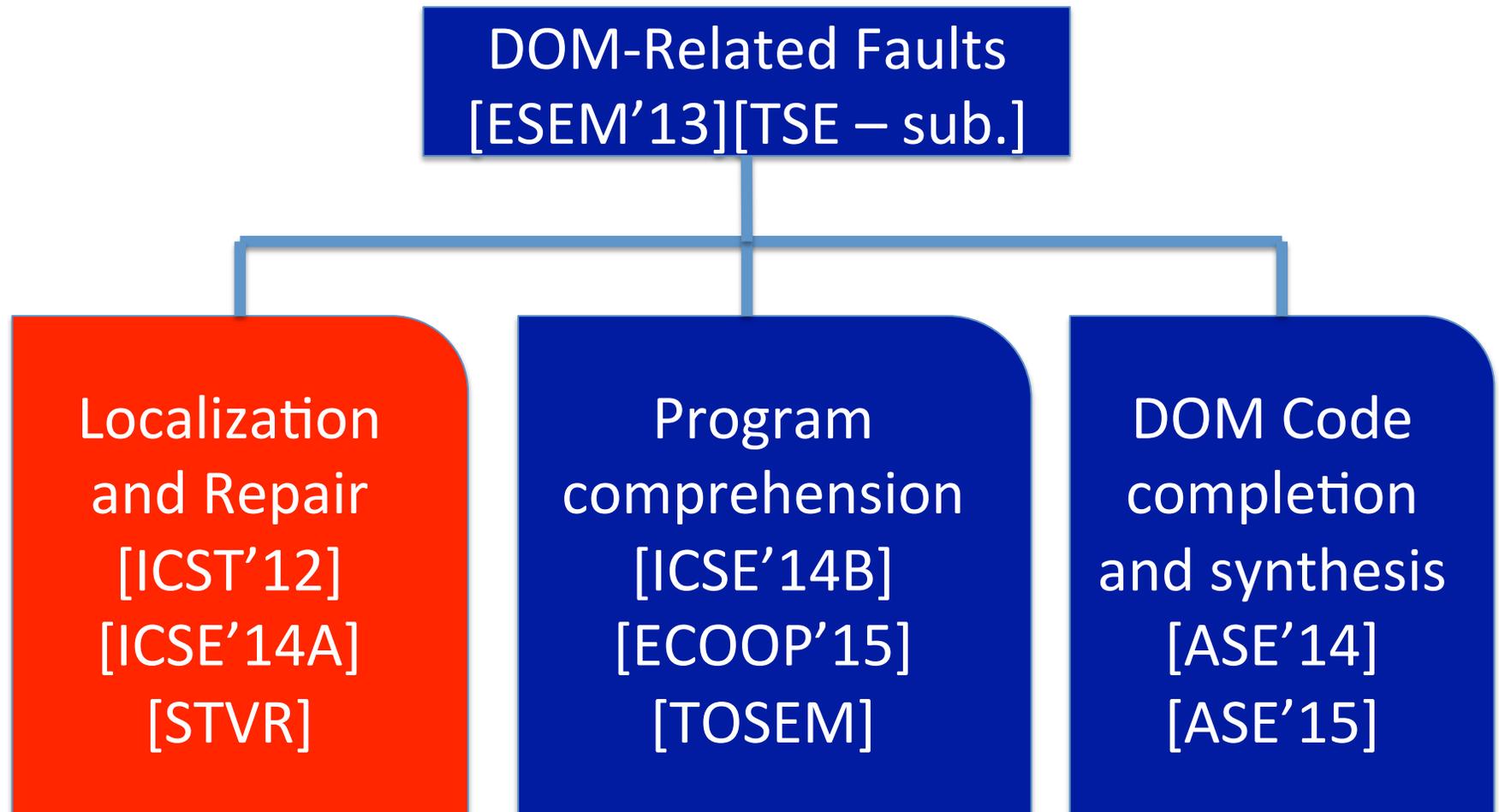
Web Applications: Existing Techniques

- **Add gradual typing to JavaScript (e.g., TypeScript from MS, DART from Google, Flow from Facebook ...)**
 - 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**
 - Spelling Errors and Type Errors [Moeller – FSE'11]
 - Race conditions [Vechev - OOPSLA'13][Livshits - FSE'15]
 - Type Coercion Errors [Pradel – ICSE'15][Pradel – ECOOP'15]

Web Applications: Our Research



Web Applications: Our Research



AutoFlox [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 }
```

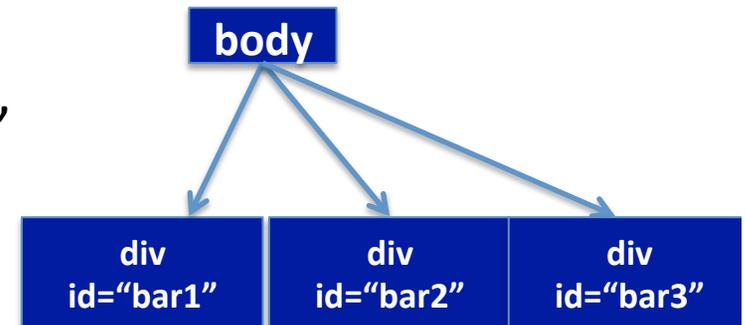
ERROR POINT TO REPAIR

DOM METHOD CALL

FAILURE POINT

Vejovis [ICSE'14A]

```
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 }
```

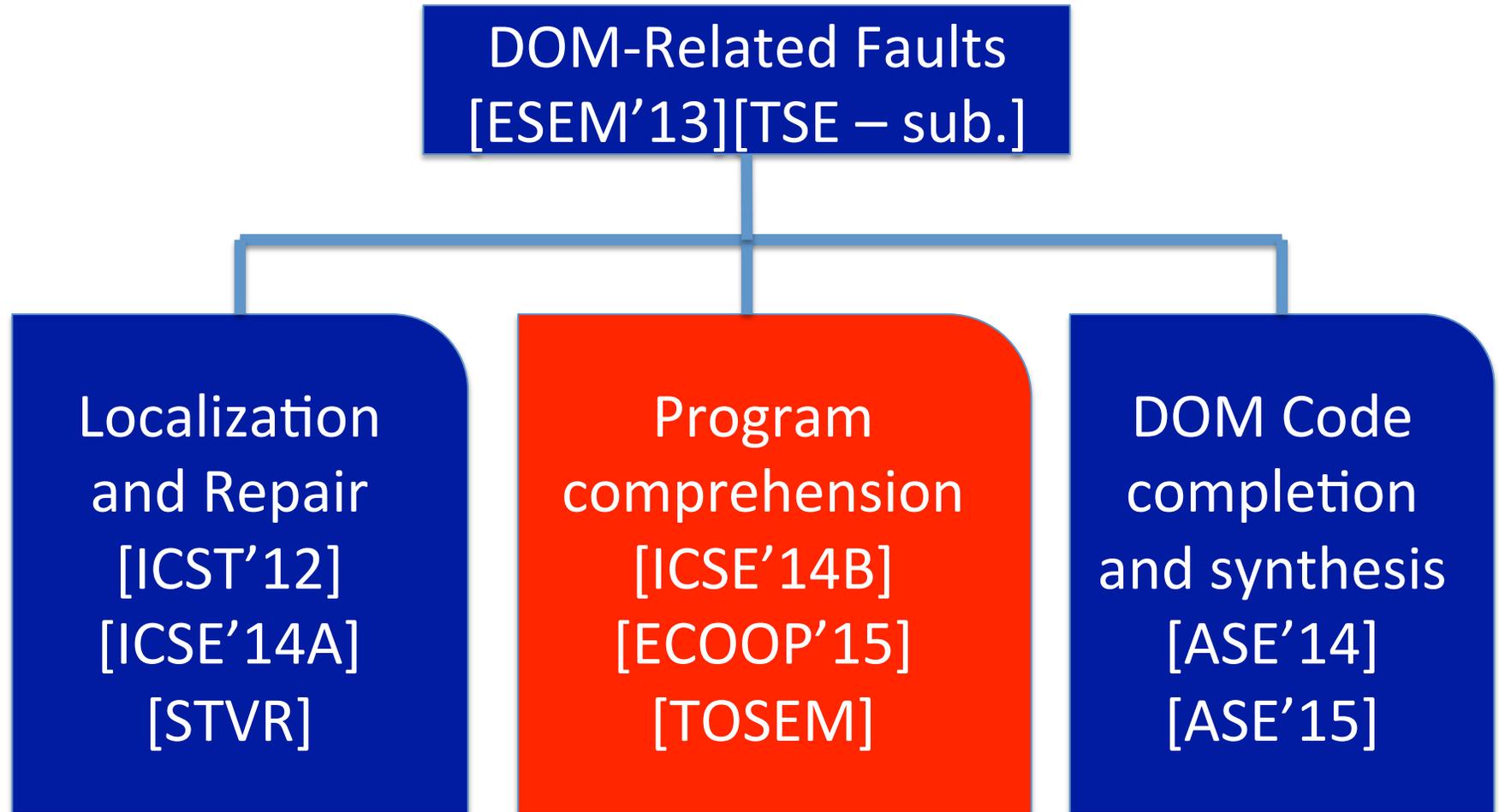


Evaluates to "bar4"

"4"

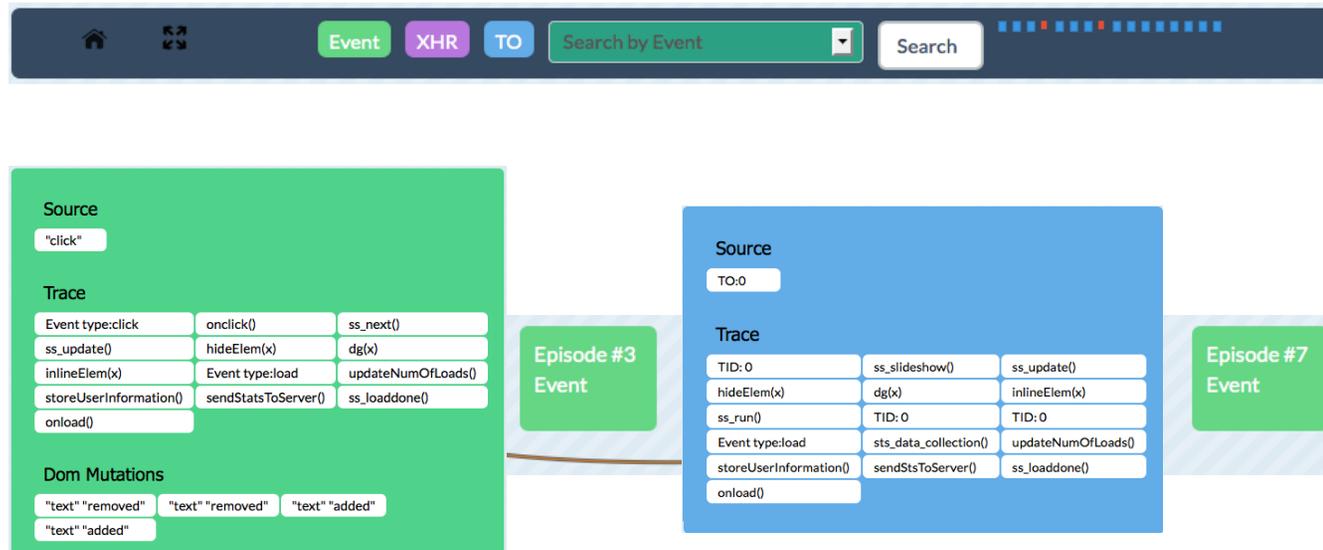
Suggestion: REMOVE last iteration of for loop in line 13

Web Applications: Our Research



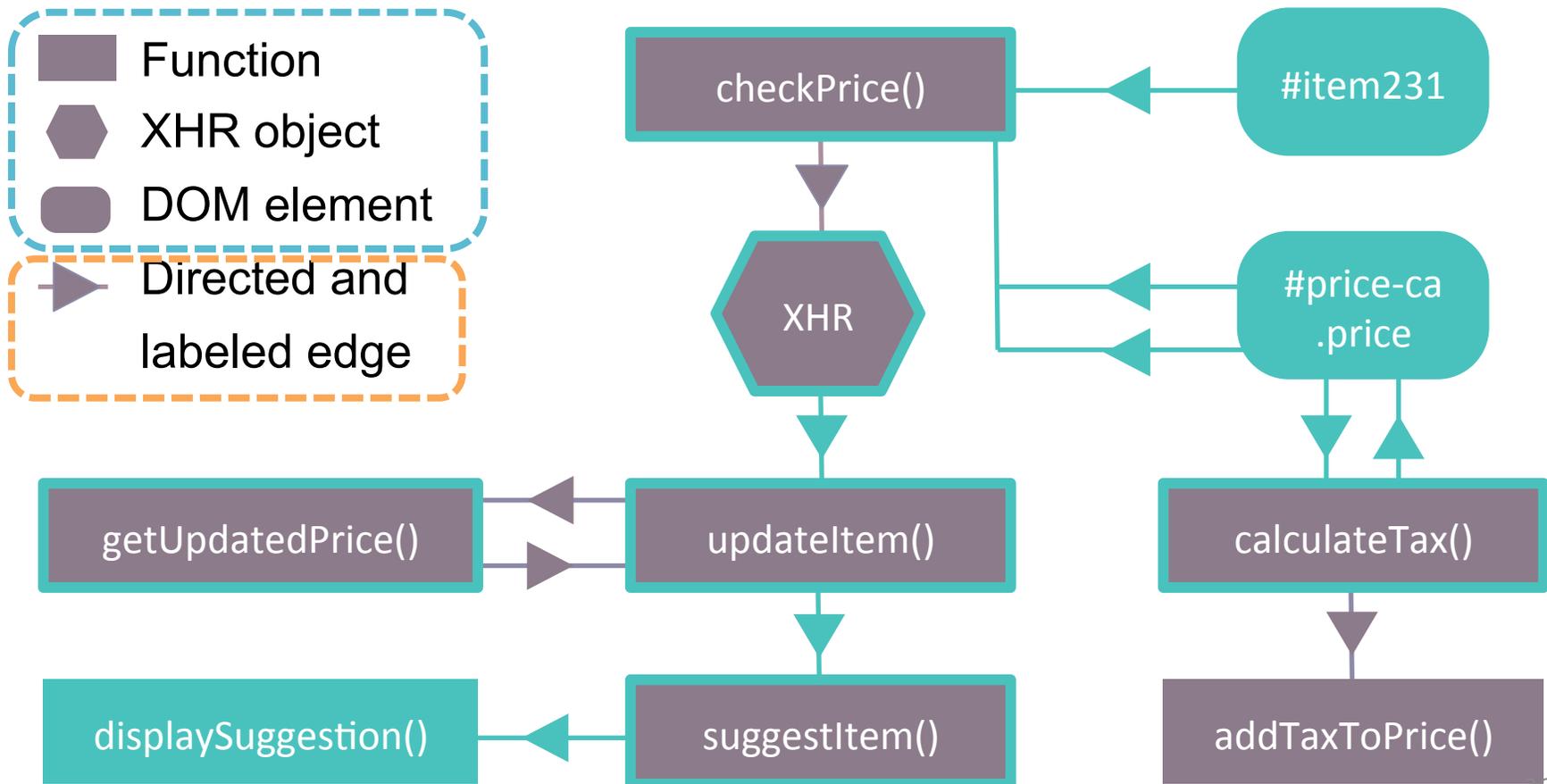
Clematis [ICSE'14B – distinguished paper award][TOSEM]

- **Challenge:** Web applications are complex, and consist of DOM interactions, AJAX messages and timeouts
- Clematis allows users to visualize causal dependencies between events and code, and between async. events

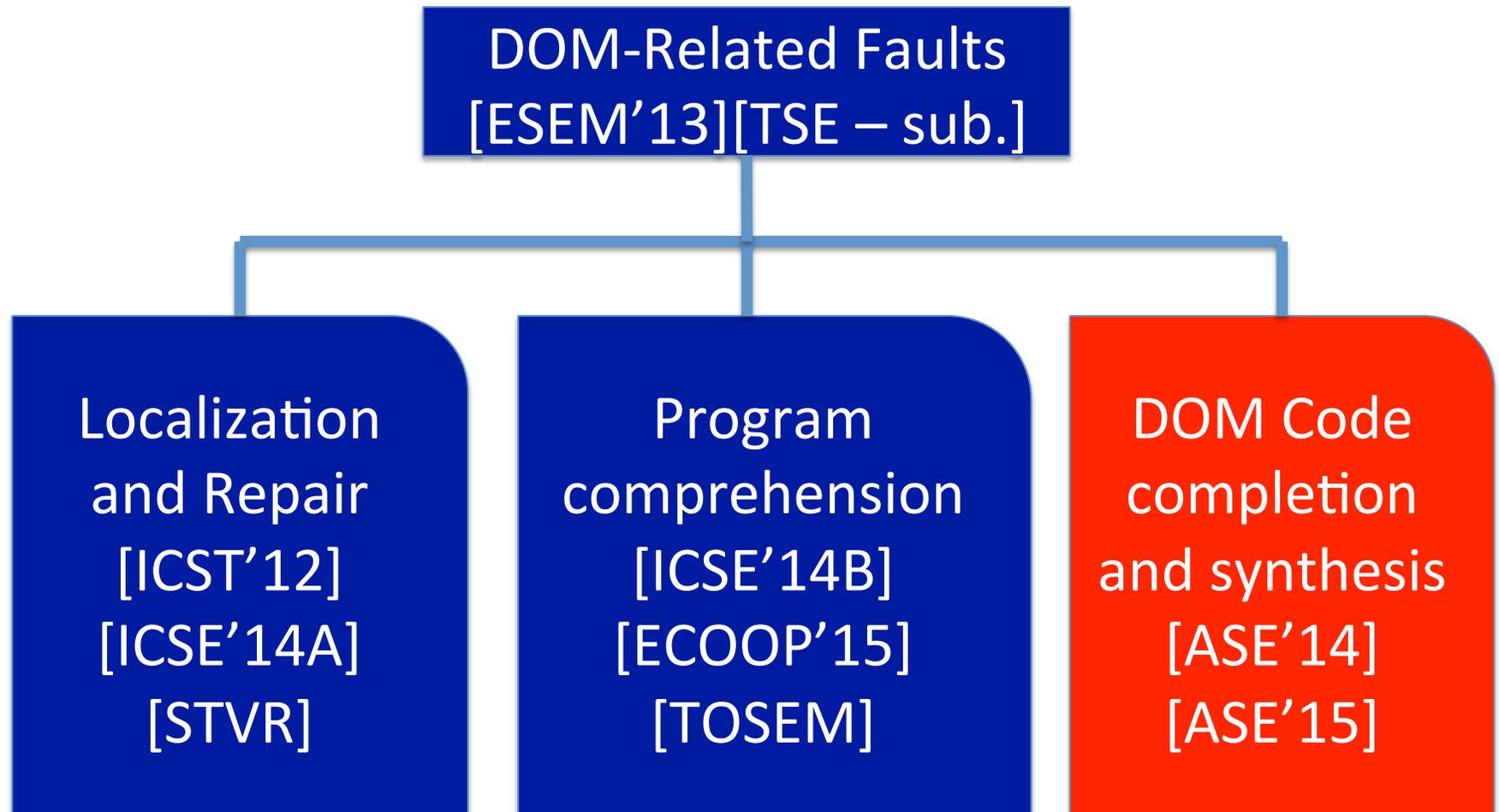


ToChal [ECOOP'15]

- Change Impact analysis for JavaScript



Web Applications: Our Research



Dompletion [ASE'14]

- Automatically perform code completion for DOM-JS interactions within the IDE
- Symbolic execution for finding DOM nodes

```
1 a = document.getElementById('maincol').innerHTML;
2 -
3 - if(a == "header") {
4   ... elem = document.getElementById('headerBar');
5 - } else {
6   ... elem = document.getElementById('photoBoxes');
7   }
8 elem.getElementsByClassName('
```

Path: 0	VeryTitle	(span)	DOM Level: 1
Path: 1	photoBox	(div)	DOM Level: 1
Path: 0	topHeadAround	(a)	DOM Level: 2
Path: 1	titlePhotoBox	(span)	DOM Level: 2
Path: 1	darkdot	(span)	DOM Level: 2
Path: 1	spc	(span)	DOM Level: 2
Path: 1	rate	(select)	DOM Level: 2
Path: 1	dot	(span)	DOM Level: 3

LED [ASE'15]

- Synthesize DOM element selectors automatically through programmer-supplied examples

The screenshot displays the LED tool interface, which is used for synthesizing DOM element selectors. At the top, there are navigation tabs: Questions, Tags, Users, Badges, Unanswered, and Ask Question. The main interface is divided into two steps:

Step 1: Drag and Drop DOM elements

This step shows three examples of DOM elements with their corresponding selectors:

- `A#nav-questions ...`
- `A#nav-users ...`
- `A#nav-unanswered ...`

Each example includes a visual representation of the element and a set of controls (arrows, minus, plus, and X) to adjust the selector. The plus sign is highlighted in blue in the third example.

Step 2: Configure Options

This step allows for configuring the options for the selector synthesis:

- id (down)
- classes (up, down)
- tag (up, down)
- mix (up)

Additional options include Depth (4), Max time (10), and a checkbox for "Select only selected elements".

The "Must use (one per line):" section lists:

- action Results

The "Ignore (one per line):" section lists:

- html
- body

A "Generate Selector" button is located at the bottom of the configuration panel. Below it, the "Total CSS Selectors: 1" section shows the generated selectors:

```
#nav-users
#nav-unanswered
```

Our Recent Work

MVC Frameworks for JS

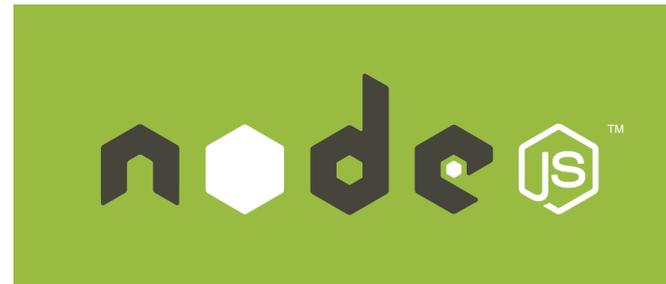
- Static analysis tools for JavaScript MVC Frameworks such as AngularJS [ICSE'15]



ANGULARJS

Server-side JavaScript

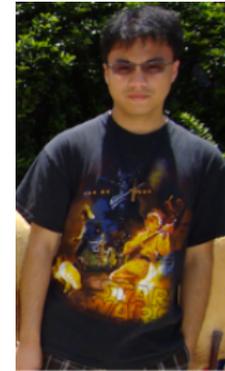
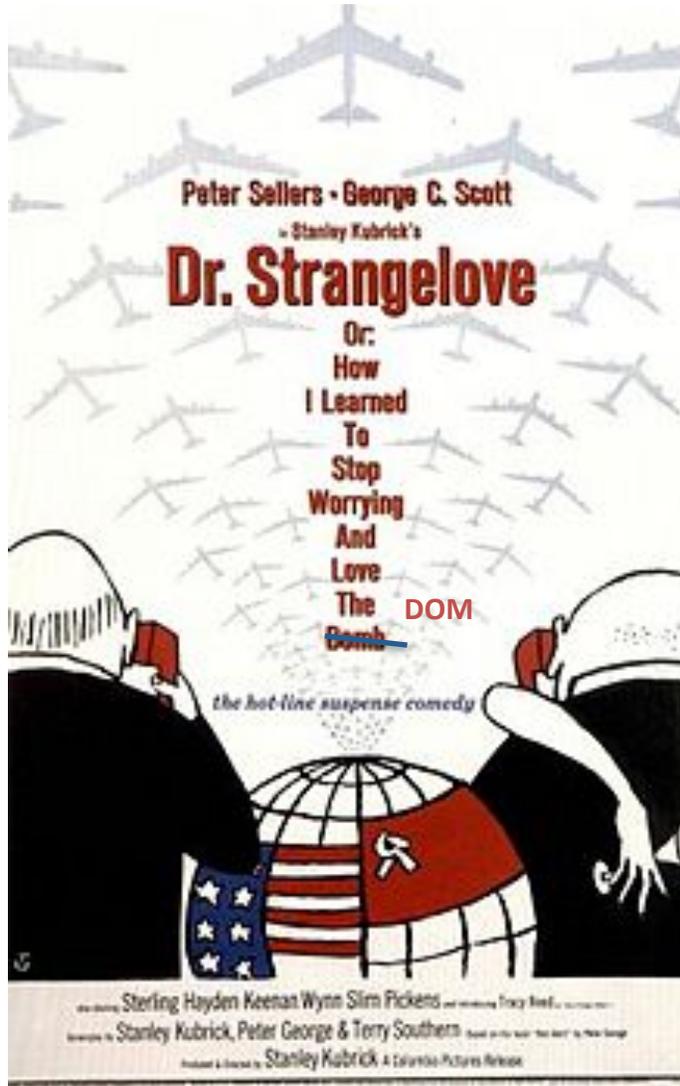
- Program Comprehension for server side JavaScript such as Node.js [ICSE'16]



Conclusion and Future Work

- **DOM-related faults prevalent in JavaScript**
 - Responsible for 2/3rds of all real-world bugs, and 80% of the most critical ones (highest impact)
 - Need efficient techniques to fix the bugs, understand root causes, and write error-free code
- **Future Work**
 - Extensions for JavaScript in the IoT context
 - Considering security in addition to reliability
 - Extension to web languages beyond JavaScript

Coming Soon to a Theater Near You !



Frolin Ocariza



Saba Alimadadi



Kartik Bajaj



Sheldon Sequira

Software and Datasets:

<http://blogs.ubc.ca/karthik/software>