

DOMPLETION

DOM-Aware JavaScript Code Completion

An approach to assist the Web Developers in
writing DOM manipulating JavaScript code

Kartik Bajaj, Karthik Pattabiraman, Ali Mesbah

University of British Columbia

{kbajaj, karthikp, amesbah}@ece.ubc.ca

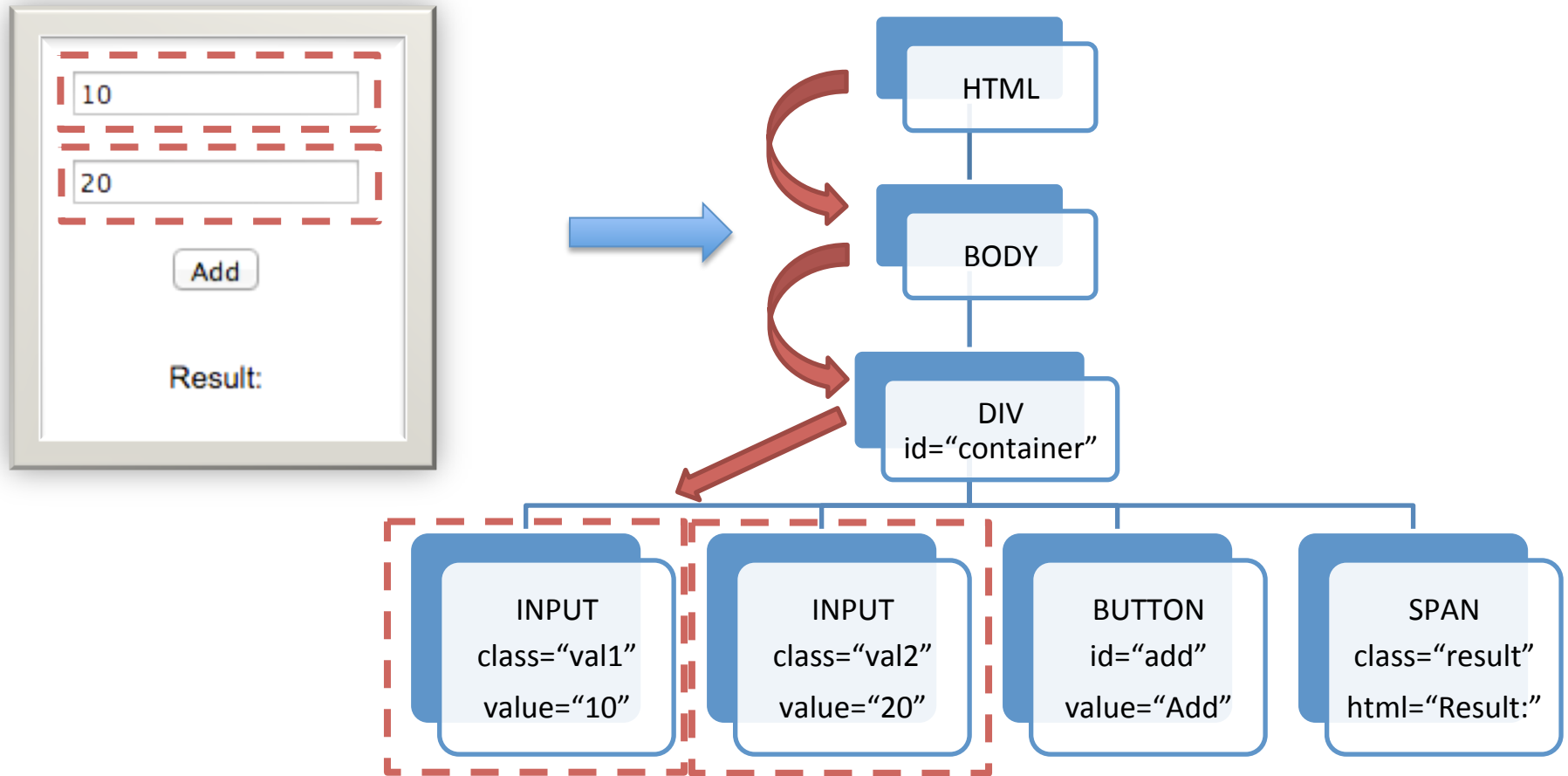
Running Example (JavaScript)

```
1. function add(a,b) {  
2.     return a + b;  
3. }  
4.  
5. var input1 = 10;  
6. var input2 = 20;  
7. if(input1 != 0) {  
8.     var sum = add(input1,input2);  
9.     alert(sum);  
10. } else {  
11.     var sum = input2;  
12.     alert(sum);  
13. }
```



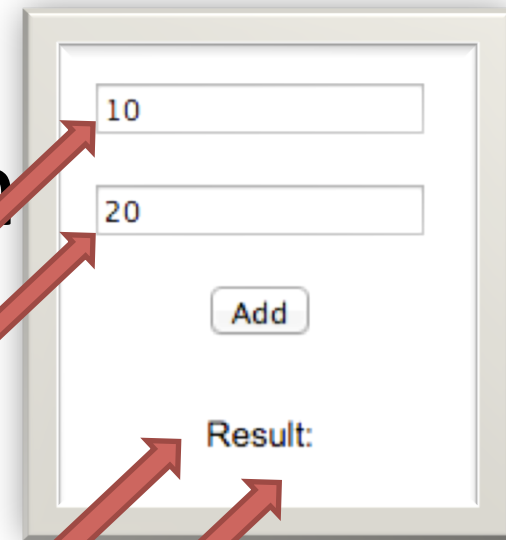
A web form with a light gray border and a white background. It contains two text input fields, one above the other. The first input field contains the number '10' and the second contains '20'. Below the input fields is a rounded rectangular button with the text 'Add'. At the bottom of the form is the text 'Result:'.

Document Object Model (DOM)



```
document.getElementById("val1").value="10";
document.getElementById("val2").value="20";
document.getElementById("add").value="Add";
document.getElementById("result").html="Result:";
```

DOM JavaScript Interaction



10

20

Add

Result:

```
1. function add(a,b) {
2.   return a + b;
3. }
4.
5. var input1 = 10;
5'. var input1 = document.getElementsByClassName("value")[0].value;
6. var input2 = 20;
6'. var input2 = document.getElementsByClassName("value")[0].value;
7. if(input1 != 0) {
8.   var sum= add(input1,input2);
9.   alert(sum);
9'.   var html = "<p id='true'> + add(input1,input2) + "</p>";
9''.   document.getElementById("result").innerHTML += html;
10. } else {
11.   var sum = input2;
12.   alert(input2);
12'.   var html += "<p id='false'> + sum+ "</p>";
12''.   document.getElementById("result").innerHTML += html;
13. }
```

Updated DOM State(s)

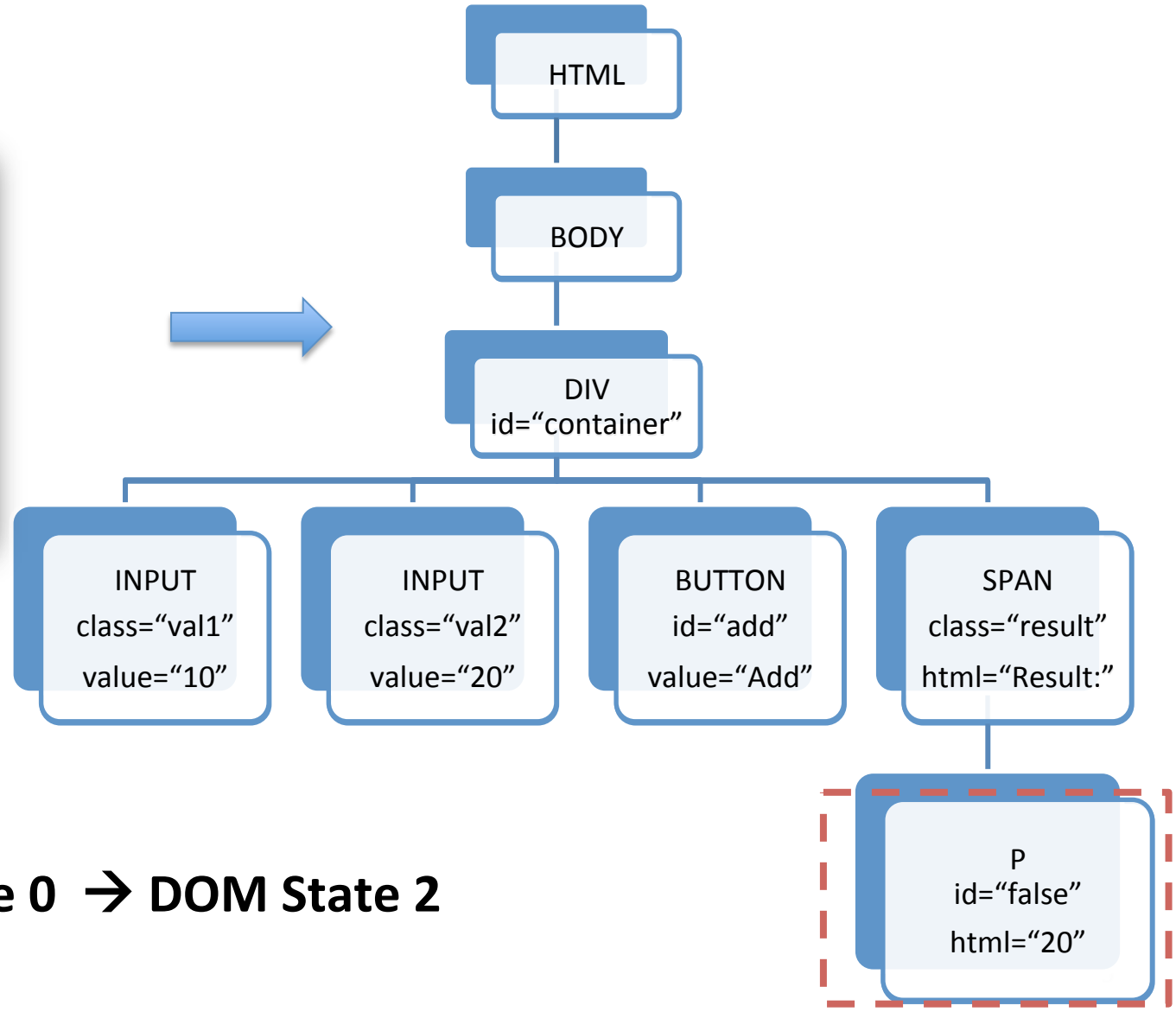
Input 2

0

20

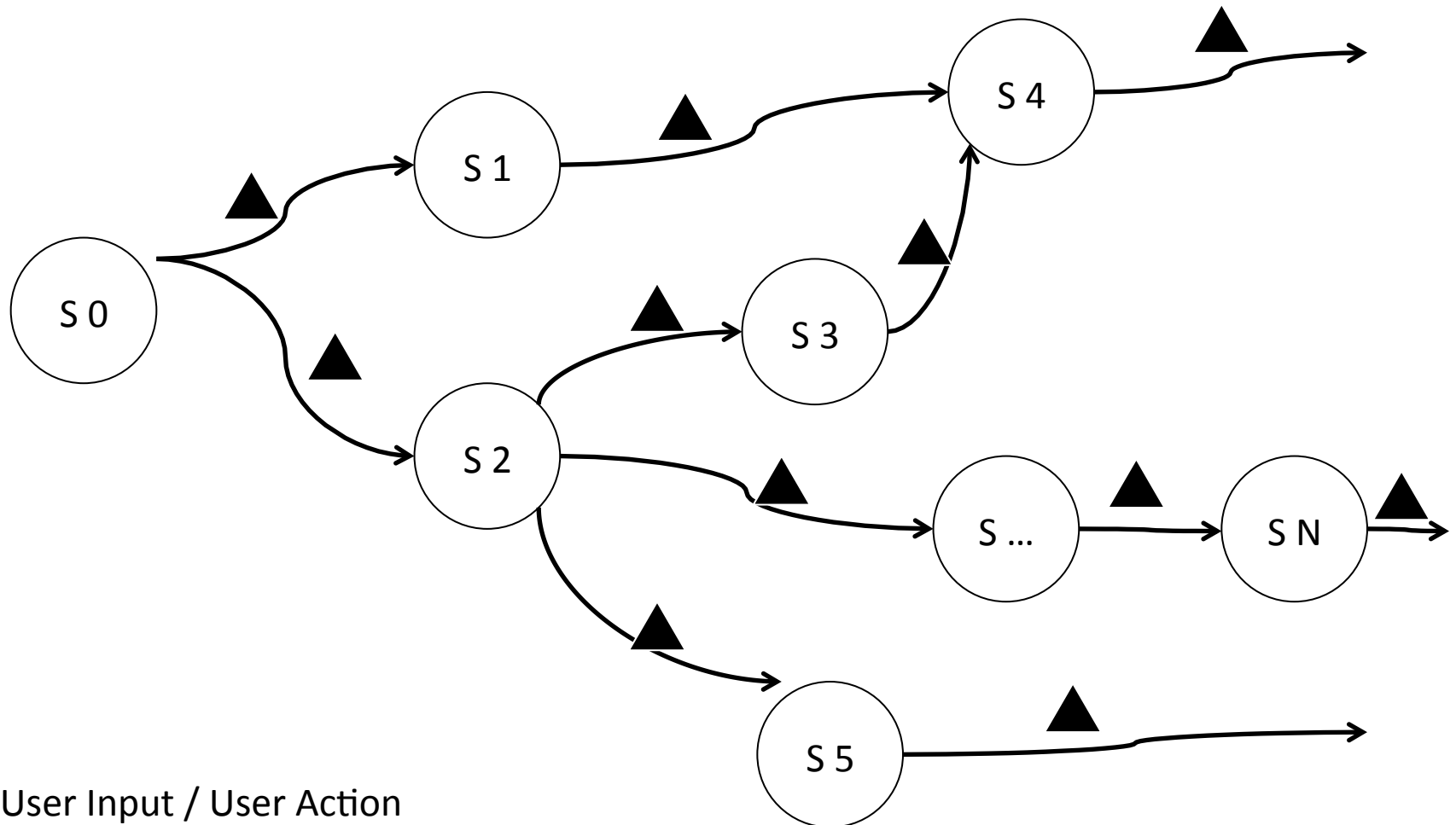
Add

Result 20



DOM State 0 → DOM State 2

DOM State Transition



Challenges

Is this class name correct?
Does the element exist in DOM?

Does it have the property "value"?

```
3. }  
4.  
5. var input1 = document.getElementsByClassName("val1")[0].value;  
6. var input2 = document.getElementsByClassName("val2")[0].value;  
7. if(input1 != 0) {  
8.   var sum= add(input1,input2);  
9.   var html = "<p id='true'> + add(input1,input2) + "</p>";  
10.  document.getElementById("result").innerHTML += html;  
11. } else {  
12.   var sum = input2;  
13.   var html += "<p id='false'>" + sum + "</p>";  
14.   document.getElementById("result").innerHTML += html;  
15. }
```

What is the updated DOM structure?

Straw man Approach

Manually inspect the DOM

Develop, Execute, Repeat

Tedious and time consuming

Problem Statement

- JavaScript code is challenging to develop and analyze
 - Handling interactions between JS and DOM
[MSR'14][ESEM'13]
- Lack of tool support [ESEM'13]
 - Instant feedback required

Related Work

???

JavaScript DOM Analysis

ICSE'14
DOM Js Fixes

ESEC'11
DOM Model
Static Analysis

FSE'13
DOM Call Graph

SA'09
Type Analysis

ASE'09
Abbreviations

ICSE'12
Graphical Input

ESEC'09
Suggestion Filtering

SIGCHI'10
Search Engine

ASE'07
Keywords

ICSE'11
Hierarchical Information

Code Completion

Proposed Solution

Analyze each DOM state and assist the developer while writing JavaScript Code

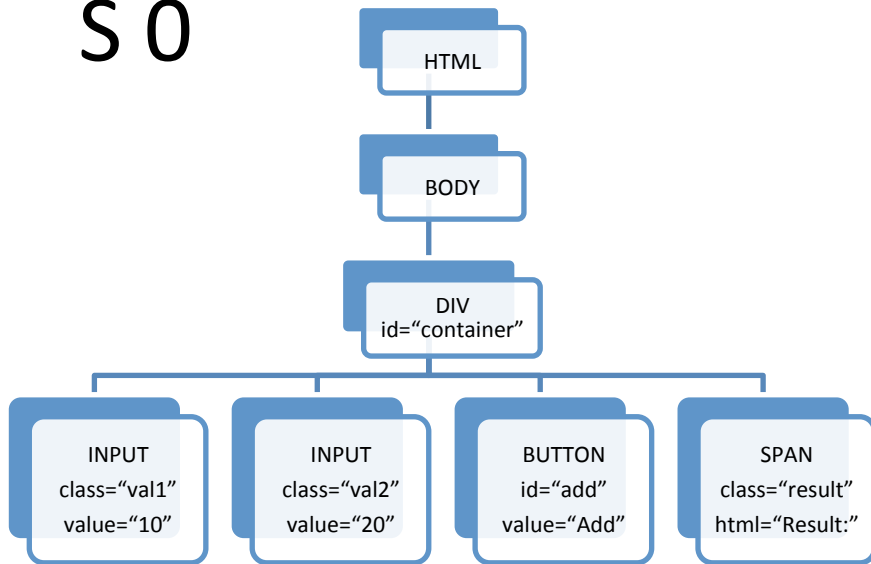
BUT

Number of DOM states can be infinite!!!

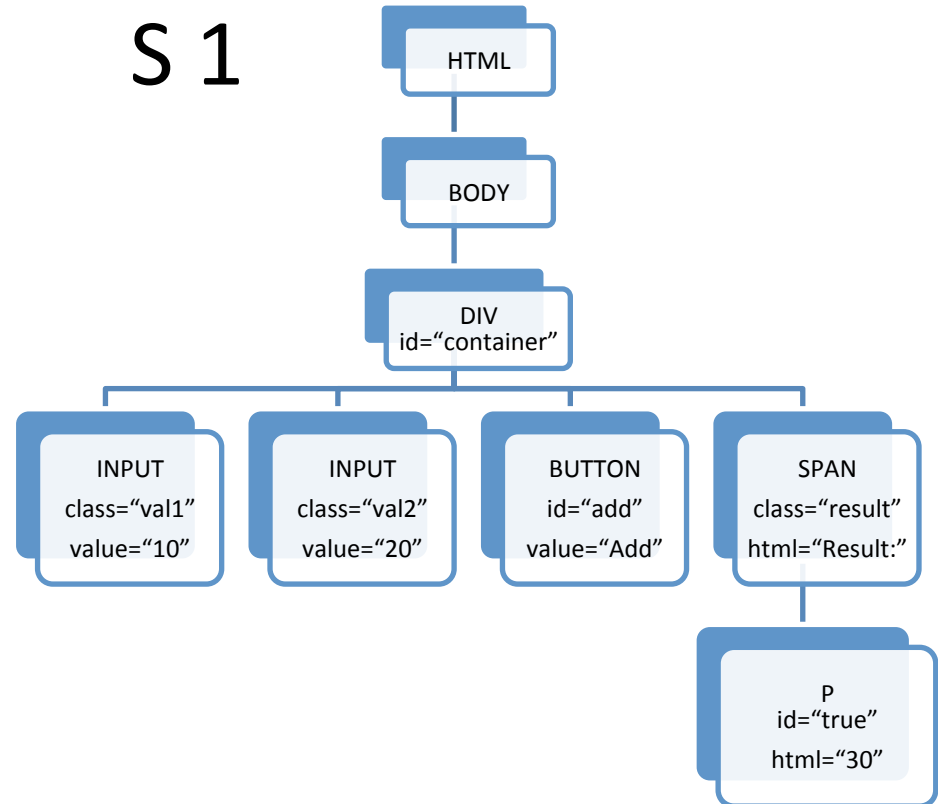
Intuition

DOM states exhibit patterns

S 0

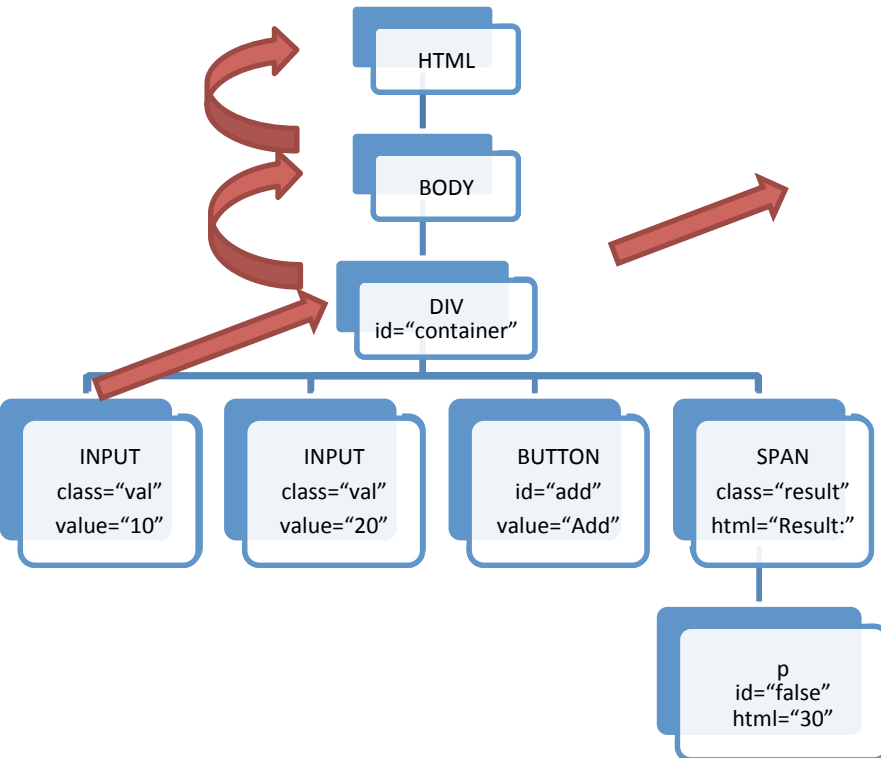


S 1



DOM Element Locators

S 2



```
html body div#container input.val1
```

```
html body div#container input.val2
```

```
html body div#container button#add
```

```
html body div#container span.result
```

```
html body div#container input.val1
```

```
html body div#container input.val2
```

```
html body div#container button#add
```

```
html body div#container span.result p#true
```

```
html body div#container input.val1
```

```
html body div#container input.val2
```

```
html body div#container button#add
```

```
html body div#container span.result p#false
```

Compression Techniques

```
html body div#container input.val1  
html body div#container input.val1.val2  
html body div#container input.val2  
html body div#container input.val1  
html body div#container button#add
```

```
html body div#container span.result
```

```
html body div#container input.val1
```

```
html body div#container input.val2
```

```
html body div#container button #add
```

```
html body div#container span.result p#true
```

```
html body div#container span.result p#true#flase
```

```
html body div#container input.val1
```

```
html body div#container input.val2
```

```
html body div#container button #add
```

```
html body div#container span.result p#false
```

Duplicates Removed

Similar IDs combined

Similar classes combined

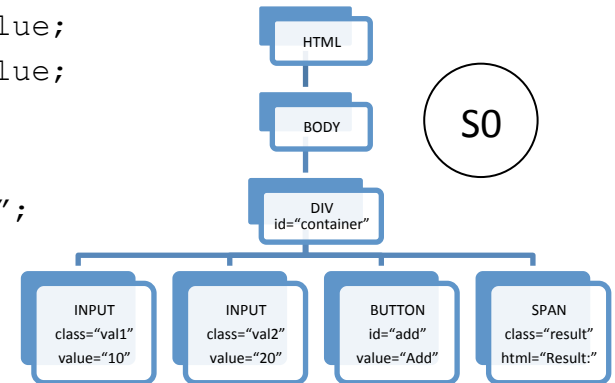
DOM Analysis

- Manual
- Automatic

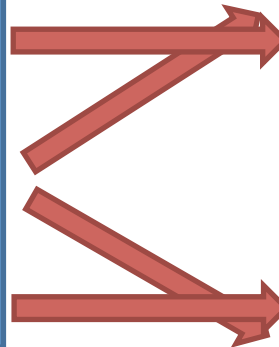
- Crawl available DOM states.
- Convert each DOM state to a list of DOM Element Locators.
- Detect patterns in DOM Element Locators.

JavaScript Code Analysis

```
1. function add(a,b) {
2.     return a + b;
3. }
4.
5. var input1 = document.getElementsByClassName("val1")[0].value;
6. var input2 = document.getElementsByClassName("val2")[0].value;
7. if(input1 != 0) {
8.     var sum= add(input1,input2);
9.     var html = "<p id='true'> + add(input1,input2) + "</p>";
10.    document.getElementById("result").innerHTML += html;
11. } else {
12.     var sum = input2;
13.     var html += "<p id='false'>" + sum+ "</p>";
14.     document.getElementById("result").innerHTML += html;
15. }
```



```
.....
if(condition) {
    then-statements;
} else {
    else-statements;
.....
```



```
.....
condition;
log true;
then-statements;
.....
```

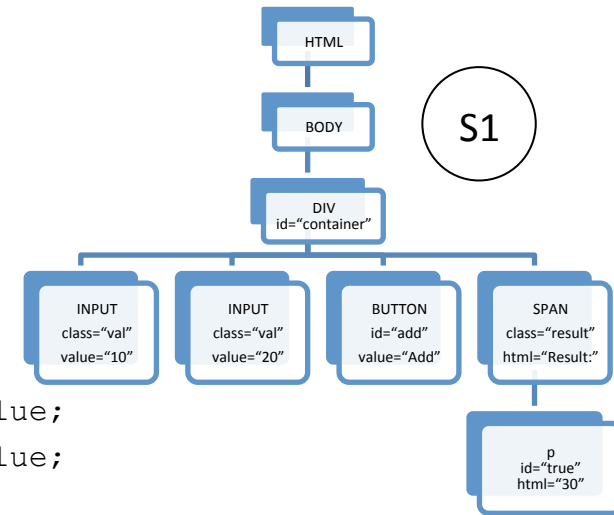
```
.....
condition;
log false
else-statements;
.....
```

Example

```

1. function add(a,b) {
2.     return a + b;
3. }
4.
5. var input1 = document.getElementsByClassName("val1")[0].value;
6. var input2 = document.getElementsByClassName("val2")[0].value;
7. input1 != 0;
8. dompleteLog("main",true);
9. var sum= add(input1,input2);
10. var html = "<p id='true'> + add(input1,input2) + "</p>";
11. document.getElementById("result").innerHTML += html;

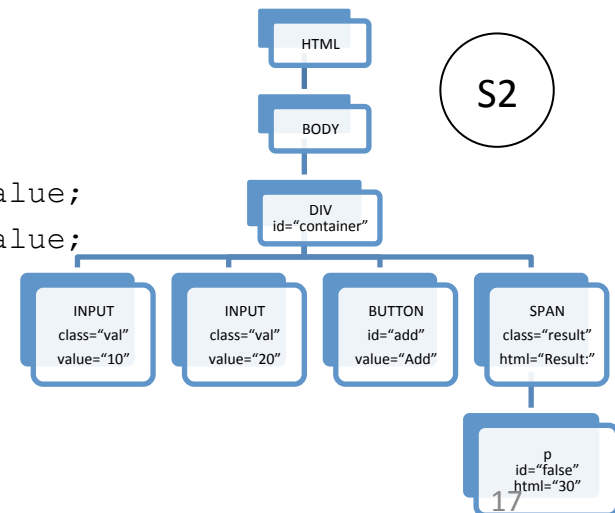
```



```

1. function add(a,b) {
2.     return a + b;
3. }
4.
5. var input1 = document.getElementsByClassName("val1")[0].value;
6. var input2 = document.getElementsByClassName("val2")[0].value;
7. input1 != 0;
8. dompleteLog("main",false);
9. var sum = input2;
10. var html += "<p id='false'>" + sum+ "</p>";
11. document.getElementById("result").innerHTML += html;

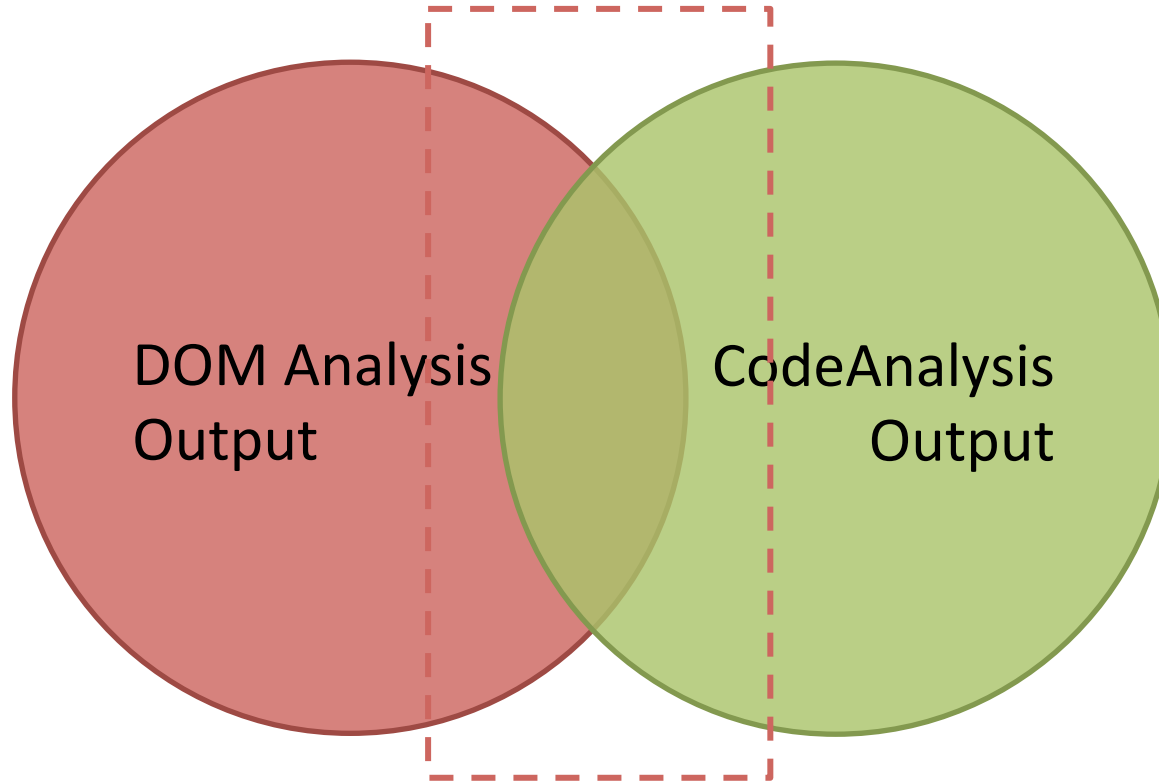
```



Code Analysis

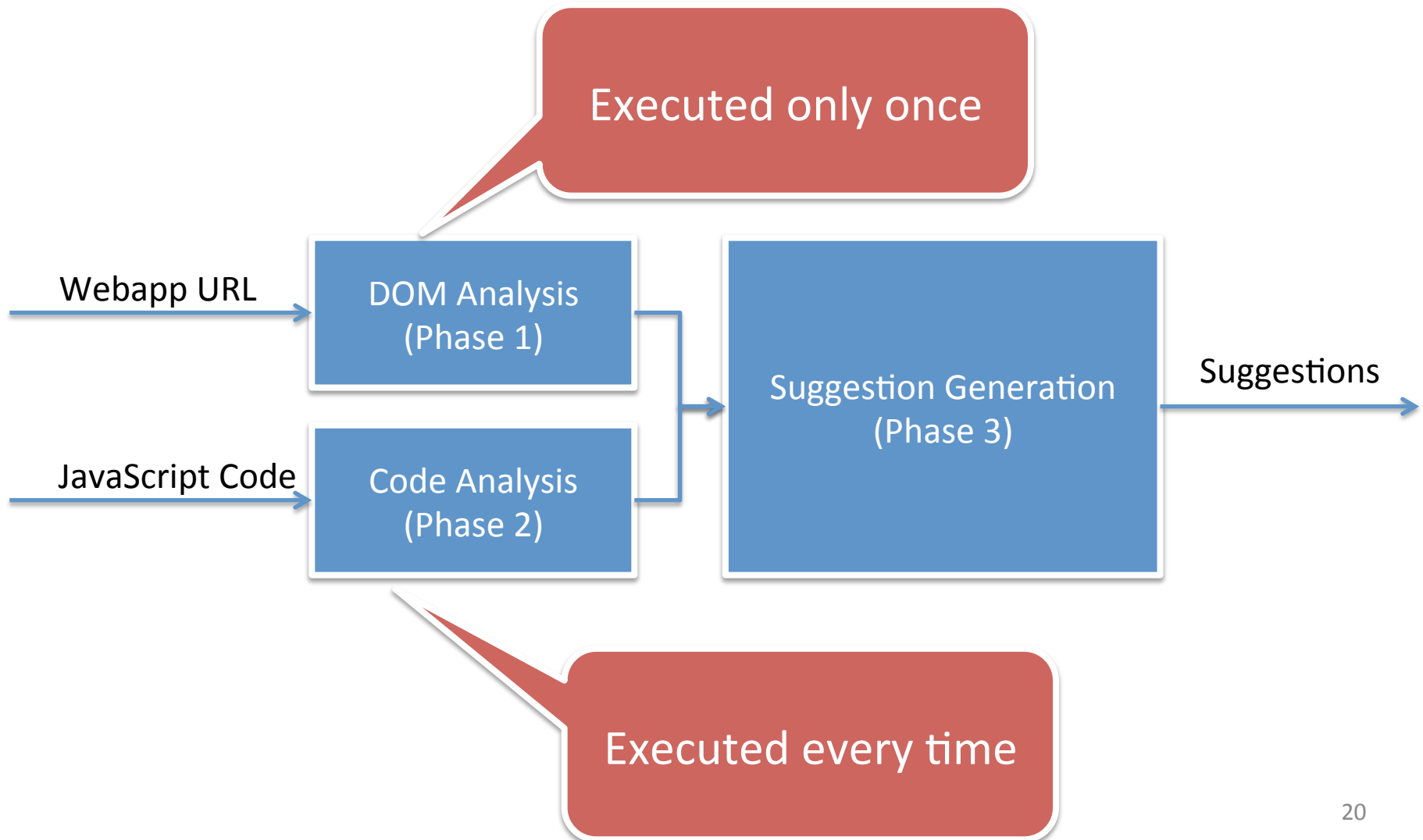
- Generate possible code paths
- Execute JavaScript code
- Intercept calls to DOM API
- Analyze Logs

Suggestion Generation



Suggestions

Approach Summary



DOMpletion (Brackets IDE Plugin)

<https://github.com/saltlab/dompletion>



Brackets



Live Development



Esprima

Screenshot

```
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

Evaluation

RQ1: Do DOM element locators converge?

RQ2: How accurate are the code completion suggestions?

- Precision
- Recall

RQ3: What is the performance overhead incurred?

Experimental Objects



RQ1: Convergence

Crawled each application randomly, until the number DOM element locators tend to stabilize.

facebook

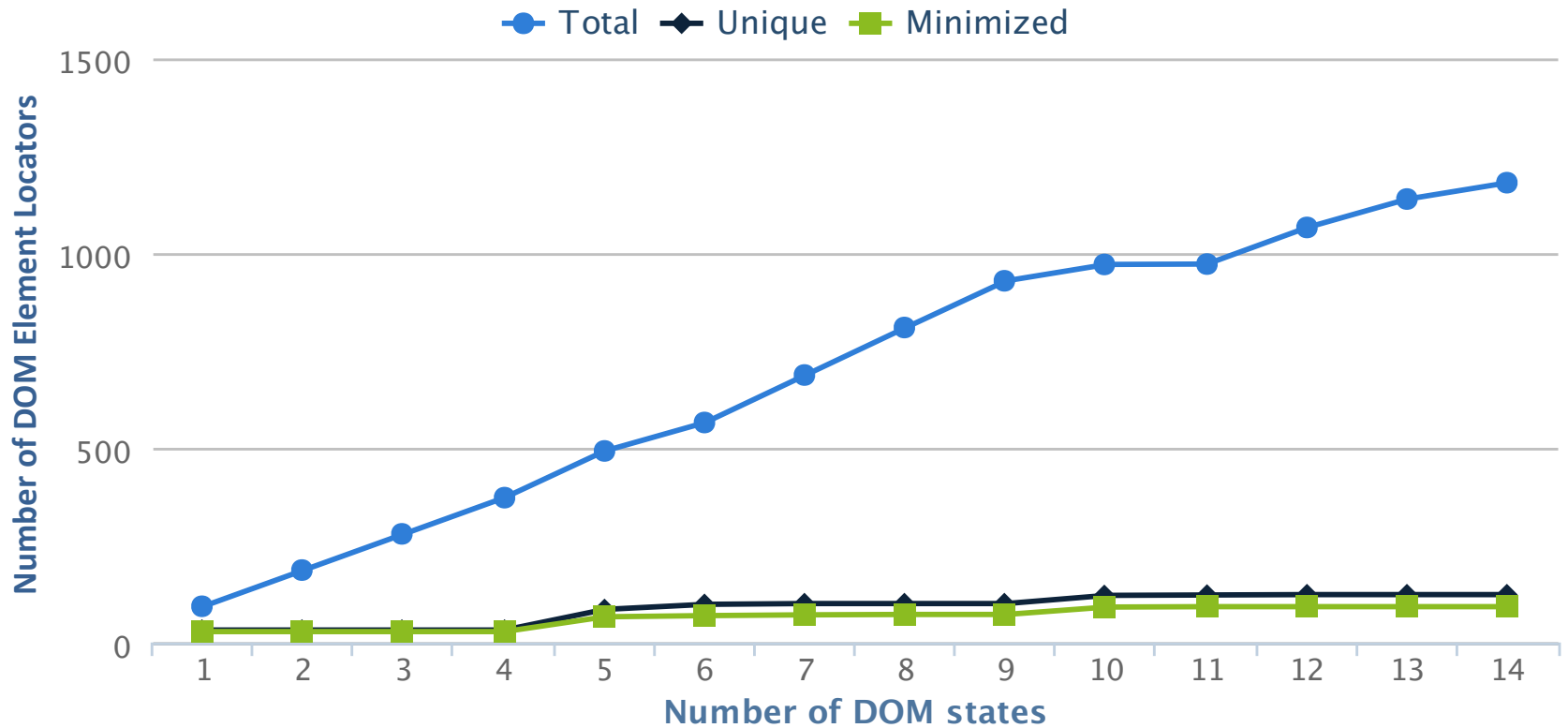
bing



WIKIPEDIA
The Free Encyclopedia



RQ1: Results



Phormer

RQ2: Performance

Performed code-completion in the existing
JavaScript Code.

Compared results with the existing values
used in the code.

RQ2: Execution

Actual Suggestion

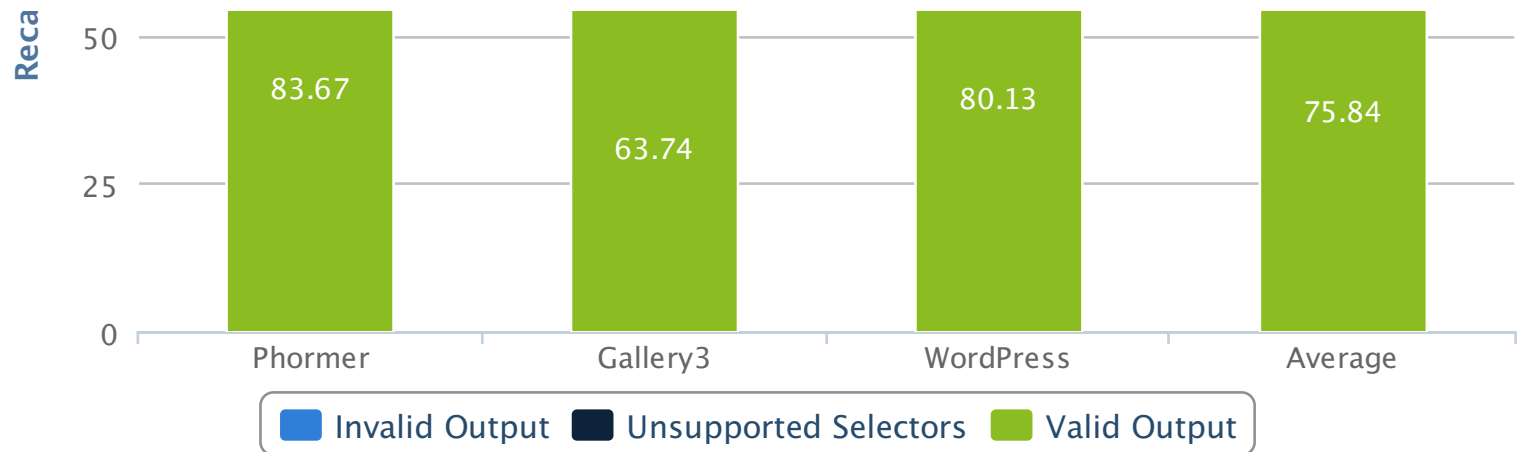
"val2"

```
1. function add(a,b) {
2.   return a + b;
3. }
4. var input1 = document.getElementsByClassName("val1")[0].value;
5. var input2 = document.getElementsByClassName("val2")[0].value;
6. var input2 = document.getElementsByClassName("val2")[0].value;
7. if(input1 != 0) {
8.   var sum= add(input1,input2);
9.   var html = "<p id='true'> + add(input1,input2) + "</p>";
10.  document.getElementById("result").innerHTML += html;
11. } else {
12.   var sum = input2;
13.   var html += "<p id='false'>" + sum+ "</p>";
14.   document.getElementById("result").innerHTML += html;
15. }
```

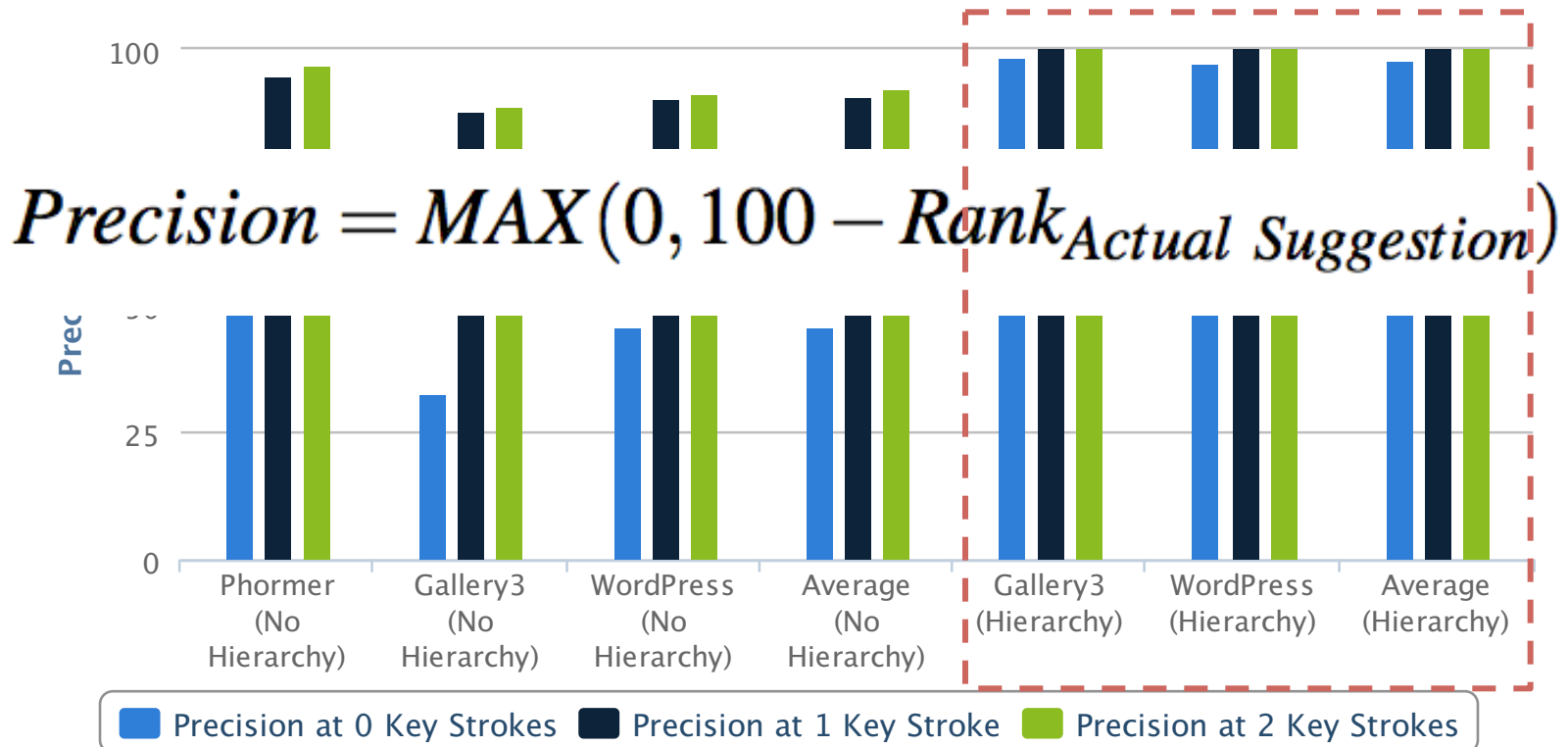
RQ2: Results (Recall)



$$\text{Recall} = \frac{\text{Valid Output}}{\text{Valid Output} \cup \text{Invalid Output} \cup \text{Unsupported Selectors}}$$



RQ2: Results (Precision)



RQ3: Performance

DOM Analysis Phase

- Crawled each web application until DOM element locators merge.
- Recorded time elapsed by the end of each state

Code Analysis Phase

- Recorded time taken to list the suggestions

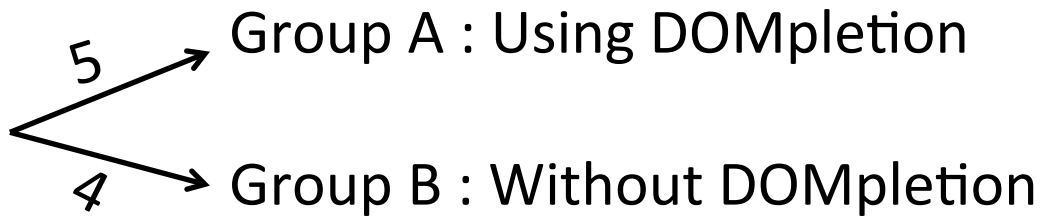
RQ3: Results

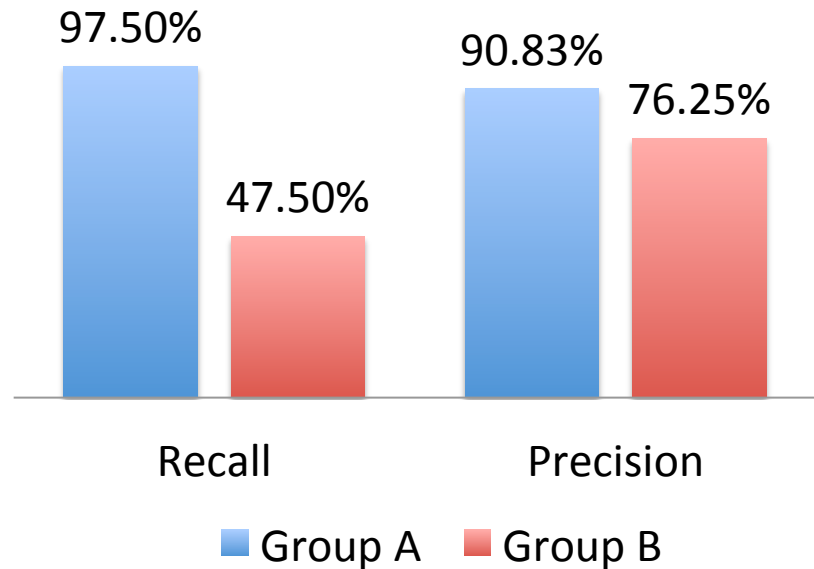
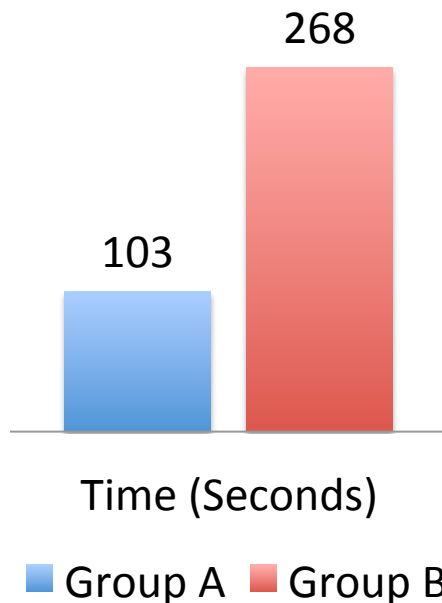
- DOM Analysis
 - 173 seconds (3 minutes approx.)
- Code Analysis
 - Min time: 1 second
 - Max time: 6 seconds
 - Avg. time: 2.8 seconds



Incurred only once

User Study

- 9 Participants 
 - Group A : Using DOMpletion
 - Group B : Without DOMpletion
- 4 Tasks to analyze the DOM JavaScript interactions



Contributions

1. Discussed challenges behind DOM based code completion.
2. Fully automated code completion technique using static and dynamic analysis of JavaScript code and DOM.
3. Implementation in an open source tool called DOMpletion.
4. Empirical evaluation to assess DOMpletion.

<https://github.com/saltlab/dompletion>

Up to 83 % Recall and 90% Precision

2.8 seconds average time

Precision

$$precision = \left(\frac{totalSuggestions - actualSuggestion}{totalSuggestions} \right) * 100$$

$$precision = \left(\frac{1000 - 300}{1000} \right) * 100 = 70\%$$

$$precision = \left(\frac{\max(0, 100 - actualSuggestion)}{100} \right) * 100$$

$$precision = \max(0, 100 - actualSuggestion)$$