

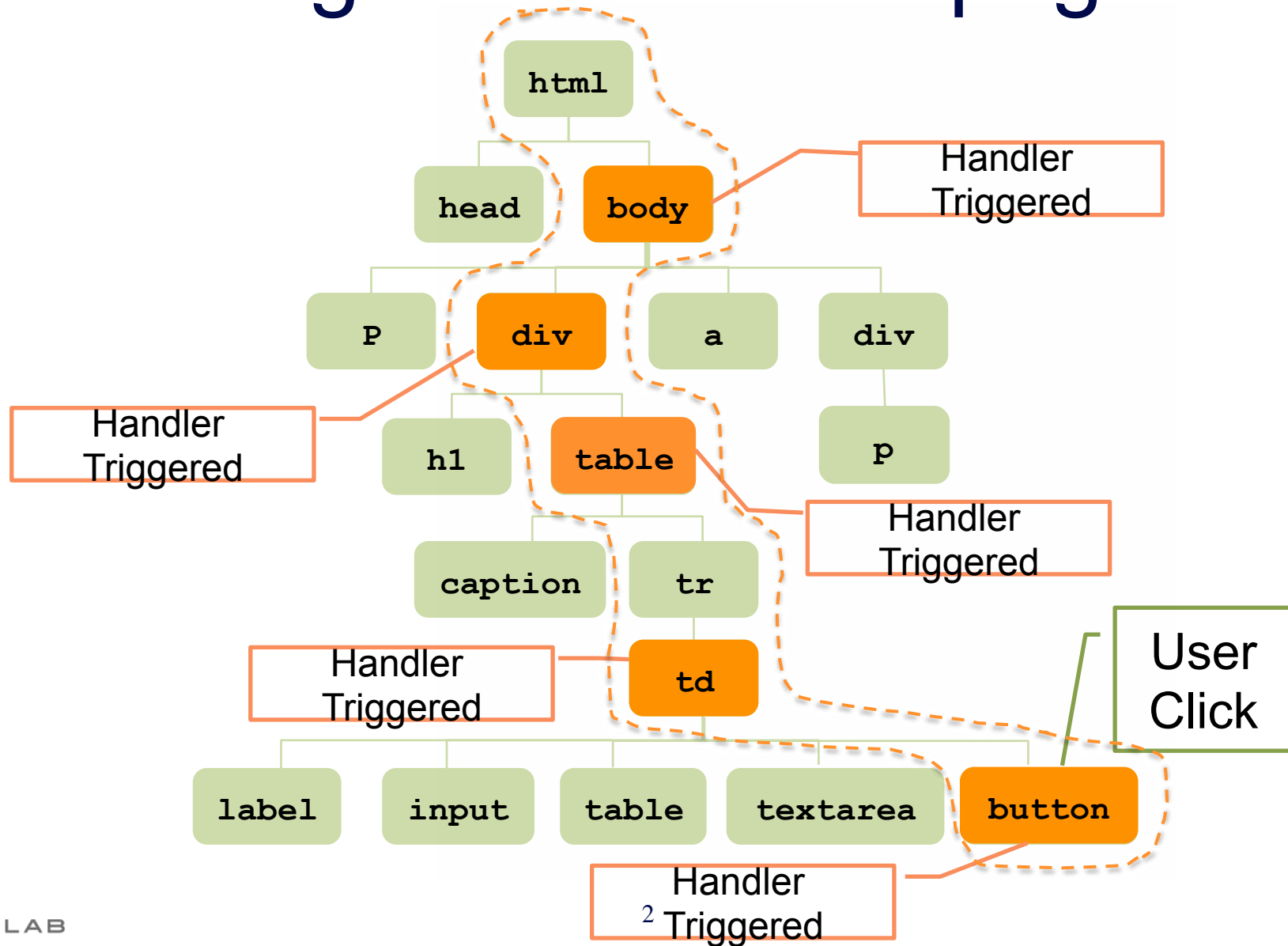
# Understanding JavaScript Event-Based Interactions

Saba Alimadadi  
Sheldon Sequeira  
Ali Mesbah  
Karthik Pattabiraman

# Motivation

- JavaScript
  - Event driven, dynamic, asynchronous
- Difficult to understand the dynamic behavior and the control flow
  - Lower level events
  - Their interactions

# Challenge 1: Event Propagation



# Challenge 2: Asynchronous Events



Timeout for page expiry  
Server request for login  
Server response for login

# Challenge 2: Asynchronous Events



**Timeout** for page expiry  
Server **request** for login  
Server **response** for login  
Server **request**  
Server **request**  
Server **response**  
Server **response**

# Challenge 2: Asynchronous Events



**Timeout** for page expiry  
Server **request** for login  
Server **response** for login  
Server **request**  
Server **request**  
Server **response**  
Server **response**  
**Timeout** for next image

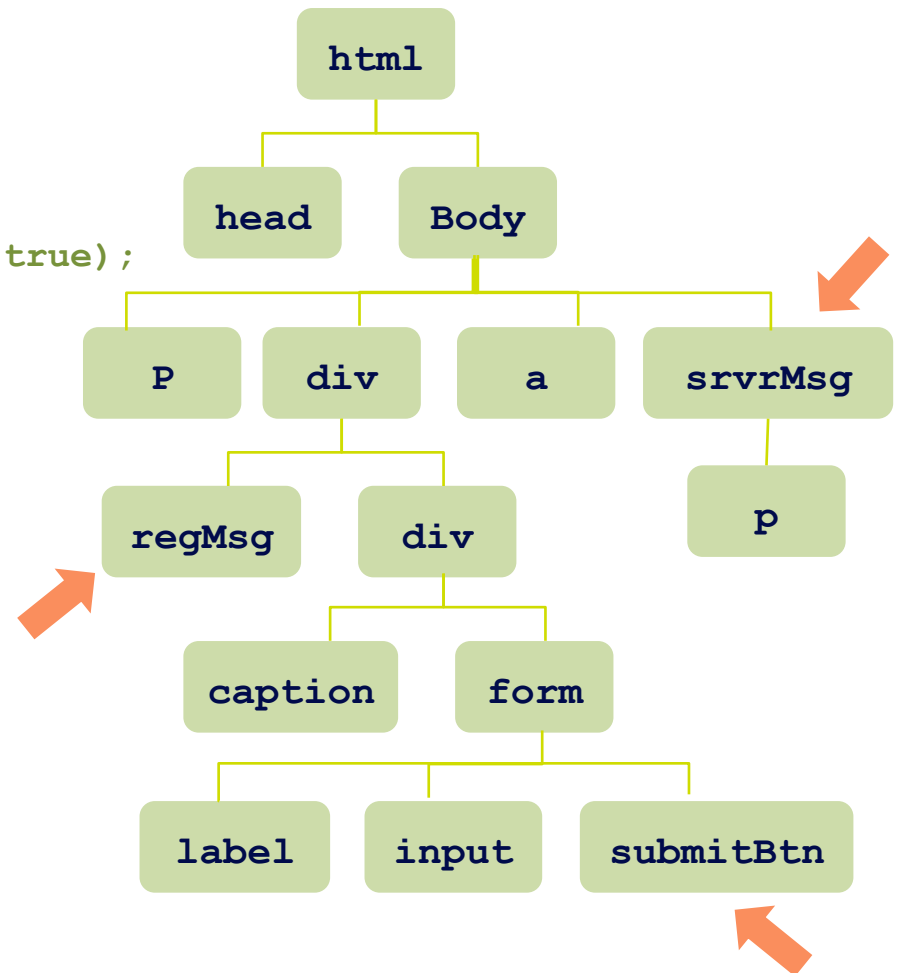
# Challenge 2: Asynchronous Events



- Timeout** for page expiry
- Server **request** for login
- Server **response** for login
- Server **request**
- Server **request**
- Server **response**
- Server **response**
- Timeout** for next image
- Server **request** image
- Server **response**
- Timeout** callback
- Timeout** callback page expiry

# Challenge 3: DOM State

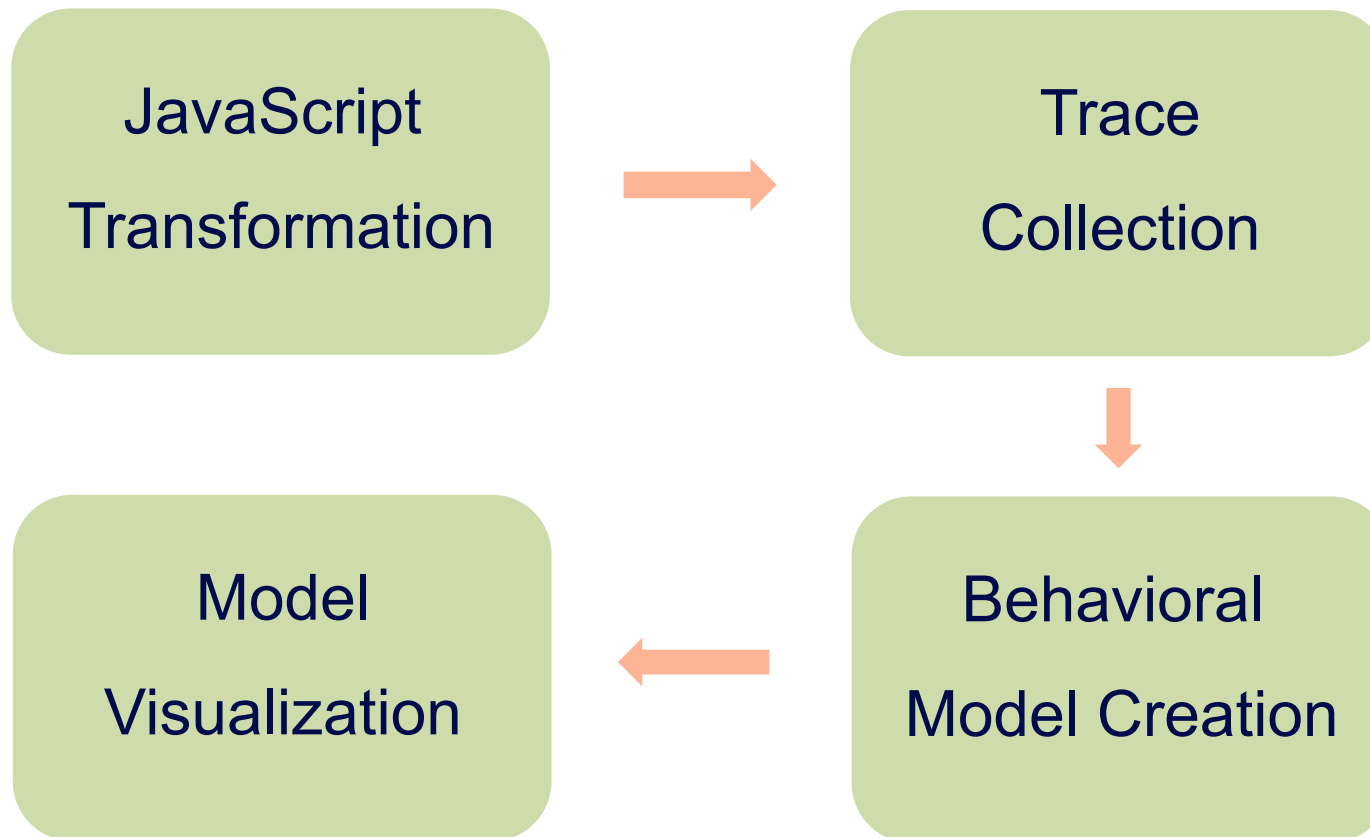
```
function submissionHandler(e) {  
    $('#regMsg').html("Submitted!");  
    var email = $('#email').val();  
    if (isEmailValid(email)) {  
        informServer(email);  
        $('#submitBtn').attr("disabled", true);  
    }  
}  
...  
function informServer(email) {  
    $.get('/register/', { email }  
    , function(data) {  
        $('#srvrMsg').append(data);  
    });  
}
```



# Summary of Challenges

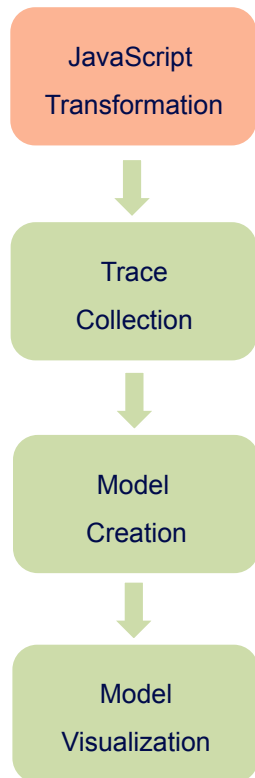
- Event propagation
- Asynchronous events
- Implications of events

# Approach



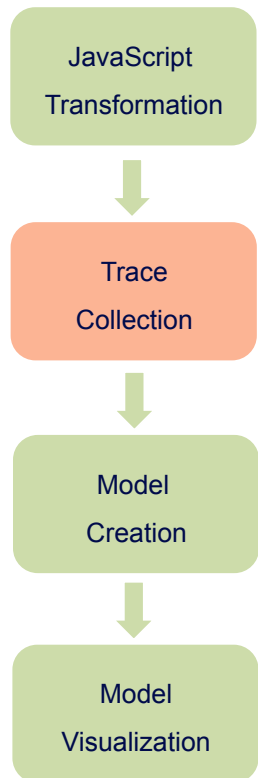
# JavaScript Transformation

- Interposing on DOM events
- Capturing timeouts and XHRs
- Recording function traces
- Extracting DOM mutations



# Trace Collection

- Interposing on DOM events
- Capturing timeouts and XHRs
- Recording function traces
- Extracting DOM mutations

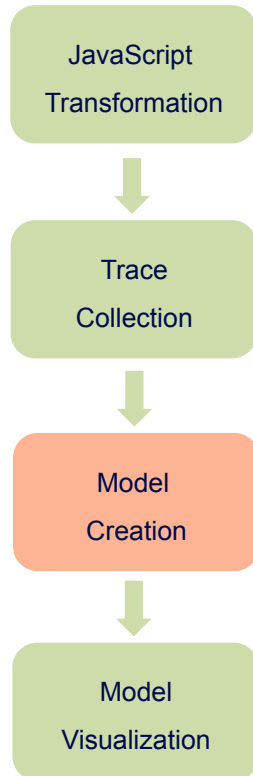


=> Detailed Trace

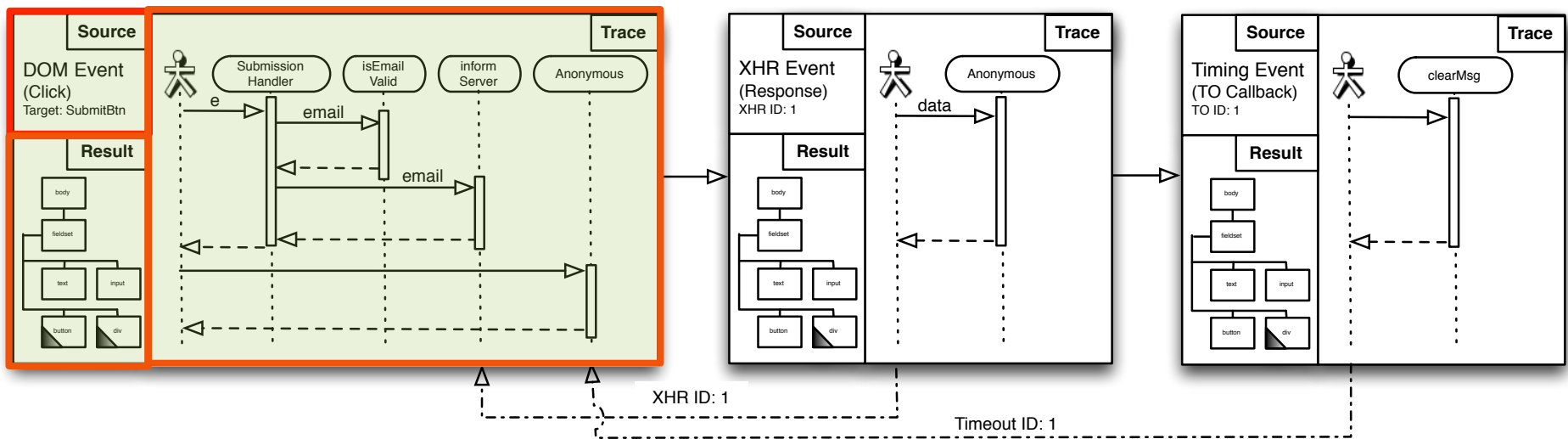
DOM events  
functions  
timeouts  
XHRs  
DOM mutations

# Behavioral Model Creation

- Customized graph
- Nodes: episodes
- Links: temporal and causal

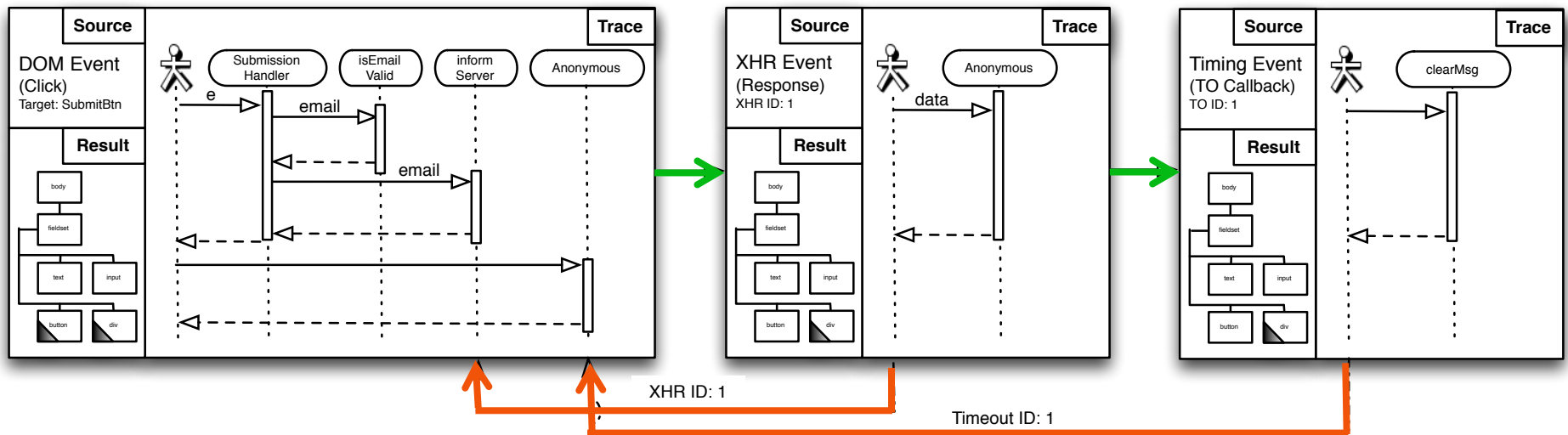


# Model: Episodes



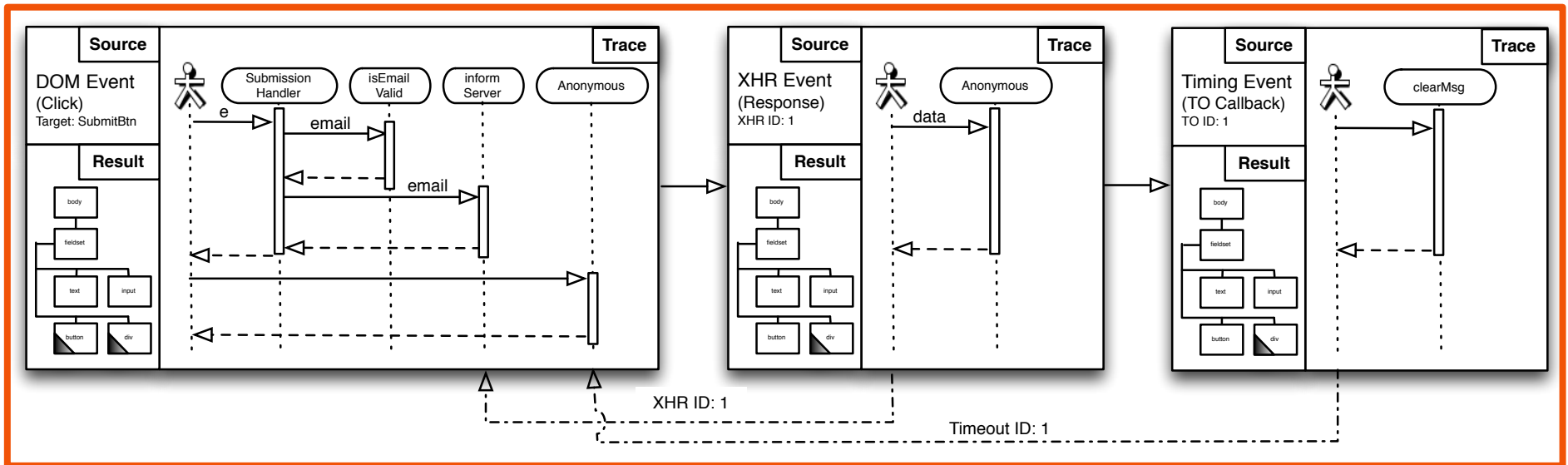
- A period of JavaScript execution
- Start and end points

# Model: Links

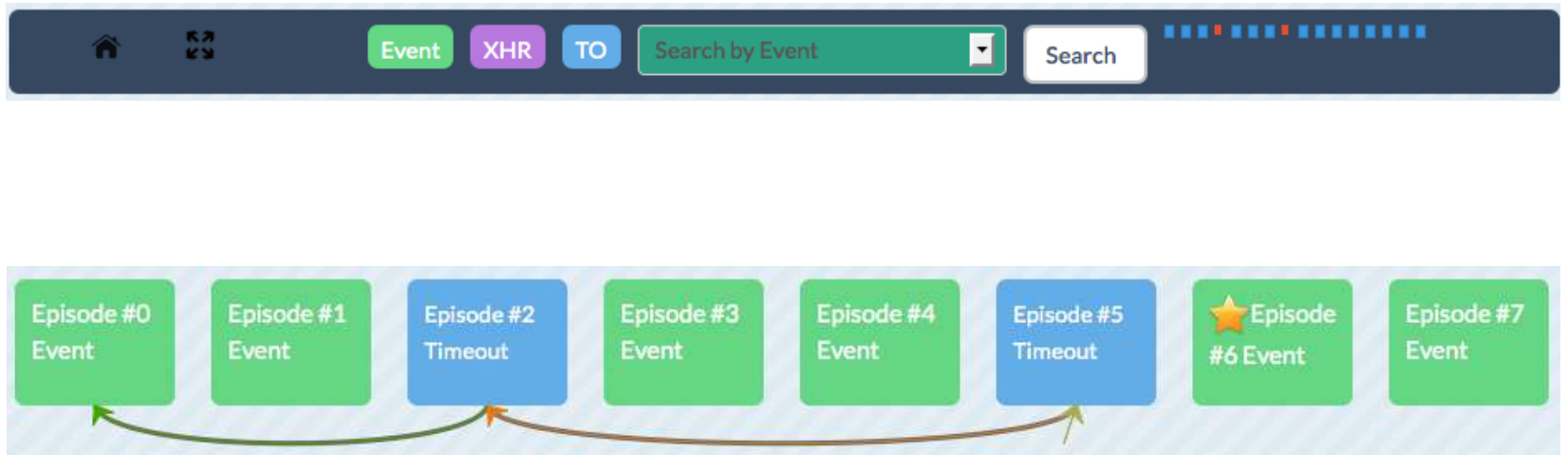
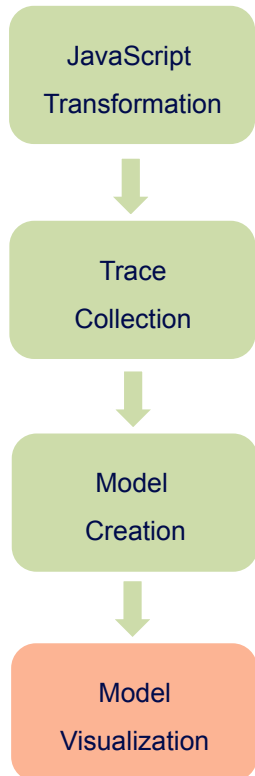


Temporal  
Causal

# Model: Story



# Visualization: Overview



# Visualization: Zoom Level 1

Navigation bar with icons for home, refresh, and zoom. Filter buttons: Event (green), XHR (purple), TO (blue). Search by Event dropdown menu, Search button, and a progress indicator.

**Source**  
"click"

**Trace**

Event type:click	onclick()	ss_next()
ss_update()	hideElem(x)	dg(x)
inlineElem(x)	Event type:load	updateNumOfLoads()
storeUserInformation()	sendStatsToServer()	ss_loaddone()
onload()		

**Dom Mutations**

"text" "removed" "text" "removed" "text" "added"  
"text" "added"

Episode #3  
Event

**Source**  
TO:0

**Trace**

TID: 0	ss_slideshow()	ss_update()
hideElem(x)	dg(x)	inlineElem(x)
ss_run()	TID: 0	TID: 0
Event type:load	sts_data_collection()	updateNumOfLoads()
storeUserInformation()	sendStatsToServer()	ss_loaddone()
onload()		

Episode #7  
Event



# Implementation

- **Clematis**

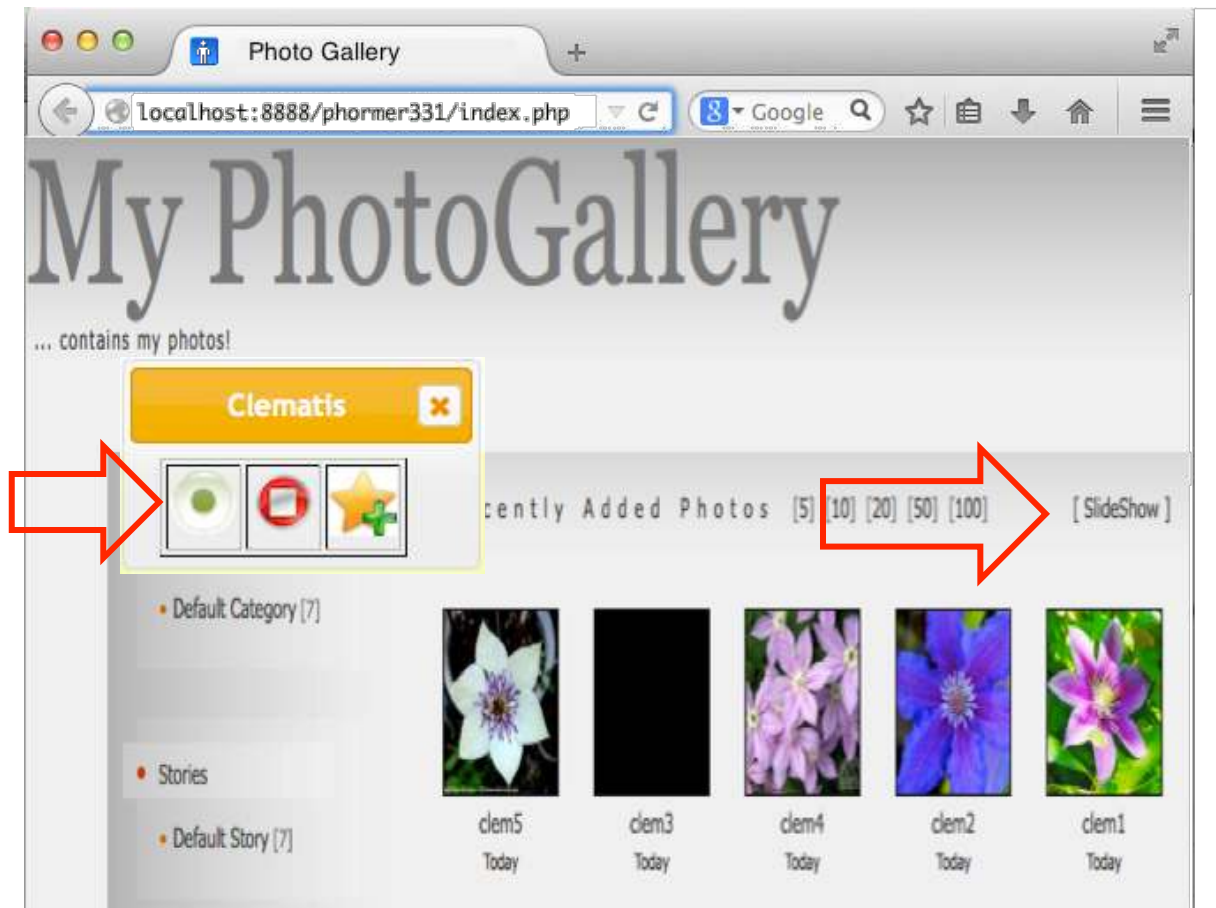
<https://github.com/saltlab/clematis>

- Languages: Java, JavaScript
- Transform JavaScript & inject toolbar via proxy
- Provide a RESTful API for retrieving data
- Render a web-based visualization

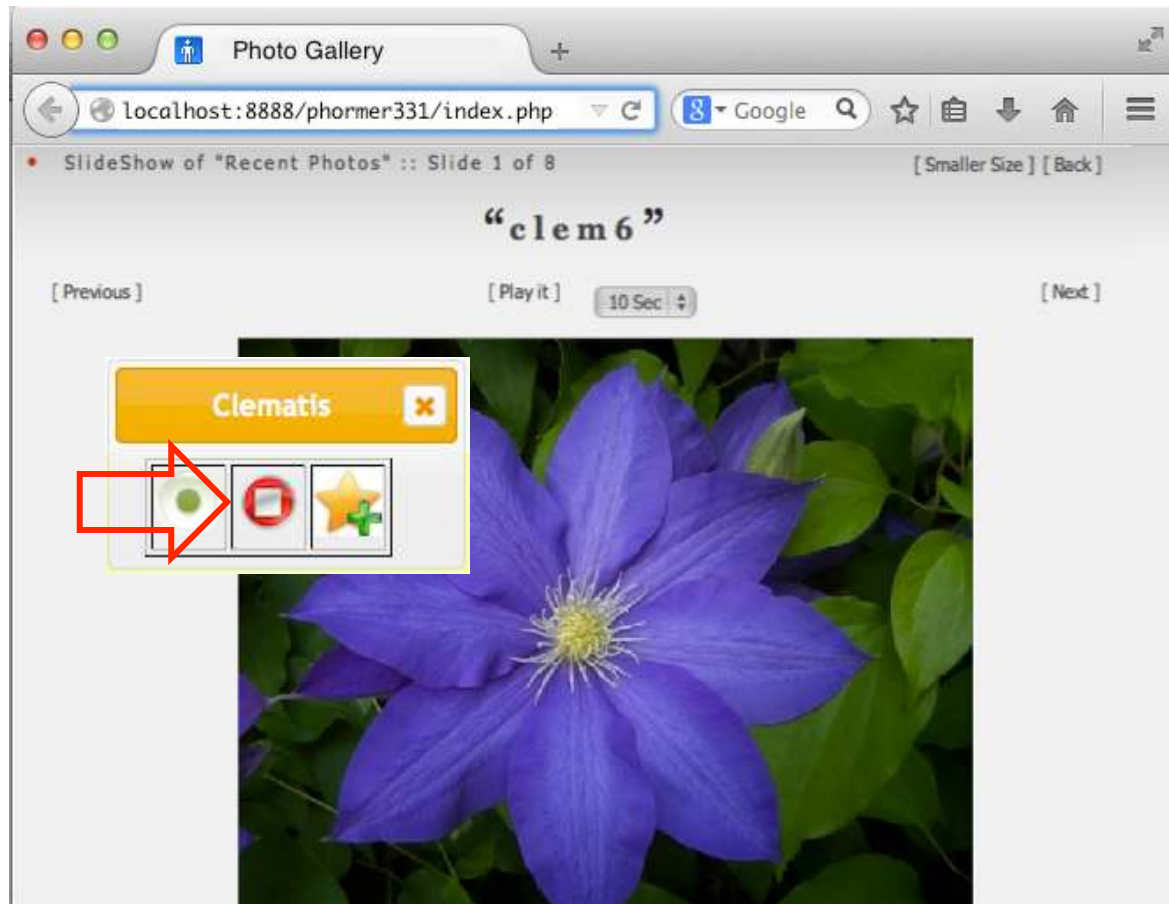
# Usage Scenario



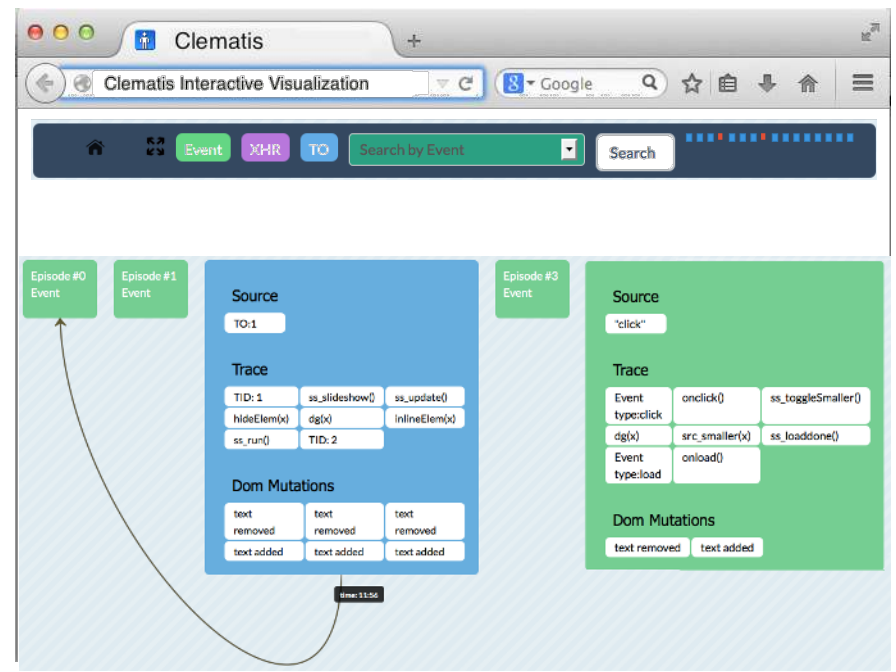
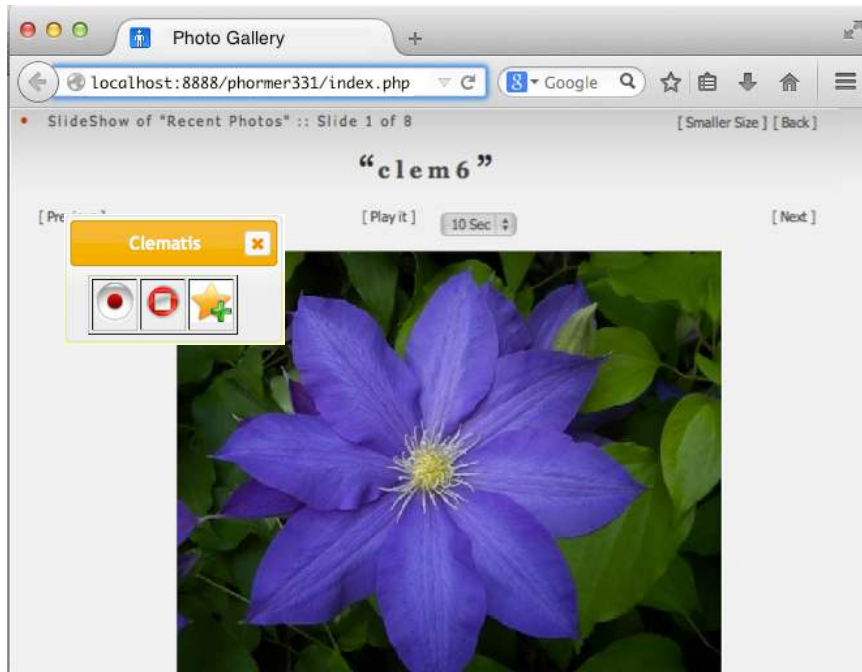
# Usage Scenario



# Usage Scenario



# Usage Scenario



# Evaluation

- RQ1) Does using Clematis decrease the task completion *duration* for web application comprehension?
- RQ2) Does using Clematis increase the task completion *accuracy* for web application comprehension?
- RQ3) Are there any certain categories of tasks for which Clematis improves the performance most?
- RQ4) Is the overall performance of Clematis acceptable?

# Industrial Controlled Experiments

- Participants
  - 20 software developers (from a large SW company)
  - Experimental group: Clematis
  - Control group: Chrome, Firefox & Firebug
- Procedure
  - 5 minute tutorial on Clematis
  - Tasks: control flow, feature location, DOM mutations, ...
- Data collection
  - Task completion duration & accuracy

# Results: Duration



## Average Time (mm:ss) Per Task

Task	Clematis		Other	
T1	7:00	<<	11:27	(39%↑)
T2	3:51	<<	7:27	(48%↑)
T3	2:02	<<	6:18	(68%↑)
T4	2:44	<	4:00	(32%↑)

## Average Time (mm:ss) in Total

Task	Clematis		Other	
All	15:37	<<	29:12	(47%↑)

# Results: Accuracy



## Average Accuracy (%) Per Task

Task	Clematis		Other	
T1	84	>>	28	(67%↑)
T2	97	>>	57	(41%↑)
T3	100	>	80	(20%↑)
T4	95	>>	30	(68%↑)

## Average Accuracy (%) in Total

Task	Clematis		Other	
All	90	>>	35	(61%↑)

# Results



## Duration

Task	Improvement
T1	(39%↑)
T2	(48%↑)
T3	(68%↑)
T4	(32%↑)



## Accuracy

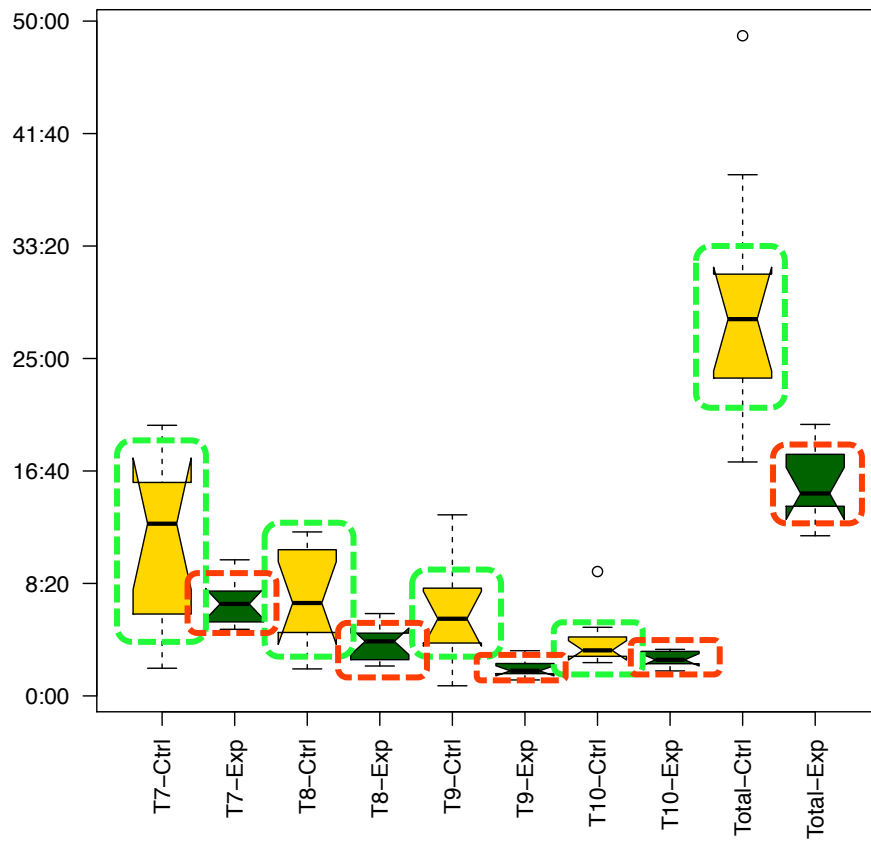
Task	Improvement
T1	(67%↑)
T2	(41%↑)
T3	(20%↑)
T4	(68%↑)

Task	Description
T1	Following control flow in presence of asynchronous events
T2	Finding DOM mutations caused by a DOM event
T3	Locating the implementation of a malfunctioning feature
T4	Detecting control flow in presence of event propagation

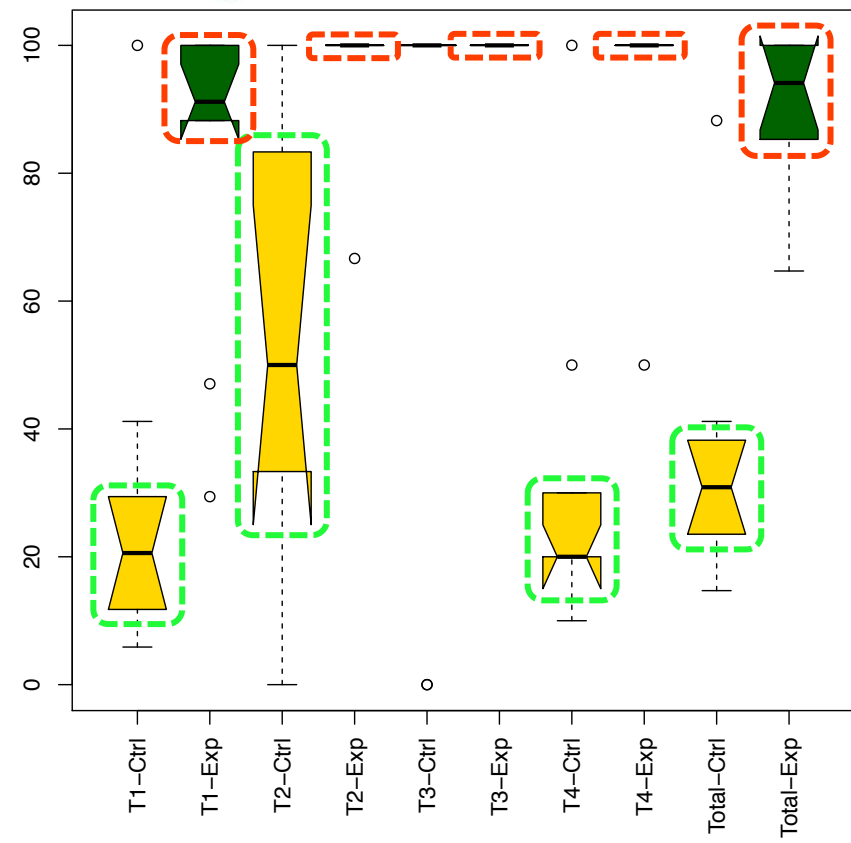
# Consistent Performance



Duration (mm:ss)



Accuracy (%)



# Understanding JavaScript Event-Based Interactions

Saba Alimadadi

Sheldon Sequeira

Ali Mesbah

Karthik Pattabiraman

Electrical and Computer Engineering  
University of British Columbia  
Vancouver, BC, Canada

{saba, sheldon, amesbah, karthikp}@ece.ubc.ca

