MODELING AND REASONING ABOUT DOM EVENTS

Benjamin Lerner, Matthew Carroll, Dan Kimmel, Hannah Quay-de la Vallee, Shriram Krishnamurthi

Brown University

WebApps 2012
Those pesky ads…

Click here and type “expo!” to win!!
Those pesky ads…

Click here and type “expo!” to win!!
Those pesky ads...

Click here and type "expo!" to win!!
What's really going on here?
Event dispatch, informally

Click here and type “exp!” to win!!
Event dispatch, informally

Handle ‘x’, ‘!’ keypresses

Click here and type “expo!” to win!!
Event dispatch, informally

Handle ‘x’, ‘!’ keypresses

Cancel all but ‘x’, ‘!’

Click here and type “eXpo!” to win!!
To understand the execution of any web page, we have to understand the model for event dispatch.
Event dispatch, informal summary

Dispatch path
- div #page
  - div #conv
    - div #msg1
      - x
    - div #msg2
    - div #reply
  - div #AD

Bubbling
- Event target
- Event listener
- Events
  - function(evt) {...}
  - Cancel all but ‘x’, ‘!’
Event dispatch – the subtleties

Interactions between mutations and order of operations

- Multiple listeners per event per element
- Tree mutation
- Adding/removing listeners during dispatch
- Legacy “handlers”
- Default actions
Core dispatch algorithm

Build dispatch path

Start

Collect list of listeners

Next node?

Done

Run

Dispatch

Dispatch-next

Dispatch-default

Pre-dispatch

Dispatch-collect

Dispatch-next

Dispatch-default
Surely this is all specified?

• Yes, but 😊
• Specification is 113 pages long
  • (Mostly definitions of specific event types)
• Core dispatch algorithm is 16 pages,
  • With side references to other specifications!
• Specification is not self-consistent
Example: addEventListener

addEventListener

Registers an event listener, depending on the useCapture parameter, on the capture phase of the DOM event flow or its target and bubbling phases.

Parameters

type : DOMString
  Specifies the Event.type associated with the event for which the user is registering.

listener : EventListener
  The listener parameter must be an object that implements the EventListener interface or a function. If listener is a function then it must be used as the callback for the event; otherwise, if listener implements EventListener, then its handleEvent method must be used as the callback.

useCapture : boolean
  If true, useCapture indicates that the user wishes to add the event listener for the capture and target phases only, i.e., this event listener will not be triggered during the bubbling phase. If false, the event listener must only be triggered during the target and bubbling phases.

This parameter must be optional. If not provided, the EventTarget.addEventListener method must behave as if useCapture were specified to be false.
Modeling the event dispatch

We built a model in Redex of the event dispatch algorithm

- 1000 lines of commented code
- Analyzable
- Testable
- Executable
- Composable
Redex: what and why

- Redex is a framework designed for language engineers
- Makes it easy to:
  - Specify operational semantics
  - Simulate running of programs
  - Examine syntax and state of programs as they run
- *Particularly* convenient when trying to match web specs:
  - Mostly written in an idiomatic, step-by-step manner
addEventListener, revisited

(define-metafunction DOM
  [(addListener LS string_type bool_useCapture loc_listener)
   (addListenerHelper
    (addListenerHelper
     (addListenerHelper
      LS string_type target bool_useCapture loc_listener)
     string_type
     ,(if (term bool_useCapture) (term capture) (term bubble))
     bool_useCapture
     loc_listener))))
Using the model: formal analysis

• Common knowledge about event dispatch:
  • “Modifying the tree shouldn’t impact the current dispatch.”
  • “Every node gets visited twice (capture and bubble) except the target.”
  • “Event dispatch is deterministic.”
  • “Event dispatch terminates.”

• All of these are theorems that hold of our model
  • Good for user understanding.
  • Good for analyses that rely upon them.
Example: dispatch path is fixed

Build dispatch path

Pre-dispatch

Start

Dispatch-collect

Collect list of handlers

Next node?

Dispatch

Done

Dispatch-next

Run

Dispatch-default

Only state to modify dispatch path – QED.
That’s nice, so? Model relevance

What assurance do we have that the model reflects reality?

Annotate the correspondence explicitly
Spec text ←annotations→ model rules

An informed reader could read both together and confirm they match.
(Compare the spec for addEventListener with our model)
Using the model: automatic testing

Can **automatically** construct test cases

- All small trees, random larger ones…
- All pairs of 1-line listeners; random longer ones…
Using the model: debug execution

Two real-world Thunderbird extensions:
  • Nostalgia
Using the model: debug execution

Two real-world Thunderbird extensions:
• Thunderbird Conversations
Using the model: debug execution

• Nostalgia: hot-keys for saving messages
  • Type ‘S’, then a folder name ➔ save message to folder

• Conversations: “Gmail-like” quick-reply box

• What should happen when you quick-reply with a word containing ‘s’?

• More importantly, when the “wrong thing” happens, why? And how should we fix it?
Future Uses

- A full account of dynamic web behavior:
  - Events (this work)
  - JavaScript
  - DOM
  - Network
  - Storage
- Testing and verification of larger web applications
- …
Recap: Contributions

• A **tractable, formal** model of web event dispatch
• **Analyzable**
  • Amenable to traditional PL techniques
• **Testable**
  • Has found actual bugs in current browsers
• **Executable**
  • Can help explain odd app behaviors or debug broken extensions
• **Composable**
  • Can be combined with other models for increased precision