# JavaScript Debugging Patterns

Zach Gardner
Keyhole Software
Developer Week 2016

## Who am I?

HTML5/JavaScript consultant

Kansas?

Single Page Applications

Financial and Healthcare clients

JavaScript since 2005

## JavaScript is hard

## Getting your “debugging environment” setup

## “No” Assumptions

## Train your Devs

“No” assumptions

Small trivial changes

Frequent commits of working code

debugger; statement

Build confidence over time

Ask for constant feedback

## Training your QAs

More information is better

Need reproducible steps

Screenshots of the entire screen

Screenshare only if necessary

## HTTP Proxy

Telerik’s Fiddler is #1

Microsoft Network Monitor is #2

Dev Tools is #3

## Dev Tools

Always leave it open

Break on uncaught exceptions

Turn on async stack trace

Rarely break on caught exceptions

## Debugging JavaScript Patterns

## Clean Slate

Isolate what you think is wrong

Go to jsfiddle.net

Prove it

## Binary Tree

Take half the code, comment it out

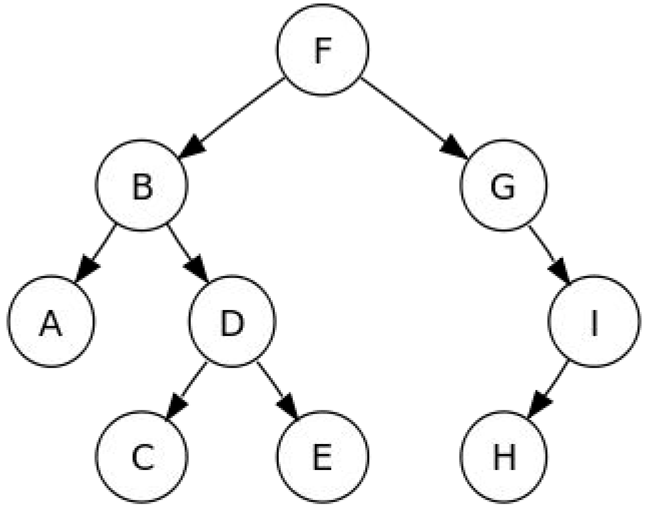
See if the bug still happens

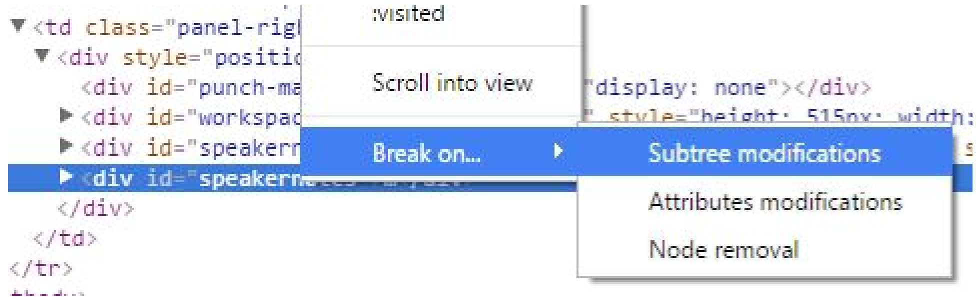
If yes, then bug's in uncommented

If no, then but's in commented

Good for brute force, no better ideas

High time commitment





```
▼<td class="panel-rig
  ▼<div style="positic
     <div id="punch-ma                         "display: none"></div>
    ▶<div id="workspac                         " style="height: 515px; width:
    ▶<div id="speakern
    ▶<div id="speakern
   </div>
 </td>
</tr>
```

## Retrospective

Figure out a bug's side effect

Try to break on the side effect

Look at the call stack to find bug

Good for DOM-related bugs

Difficult for JS-only bugs

## Infinite Loop

Dev Tools logs most of the call stack

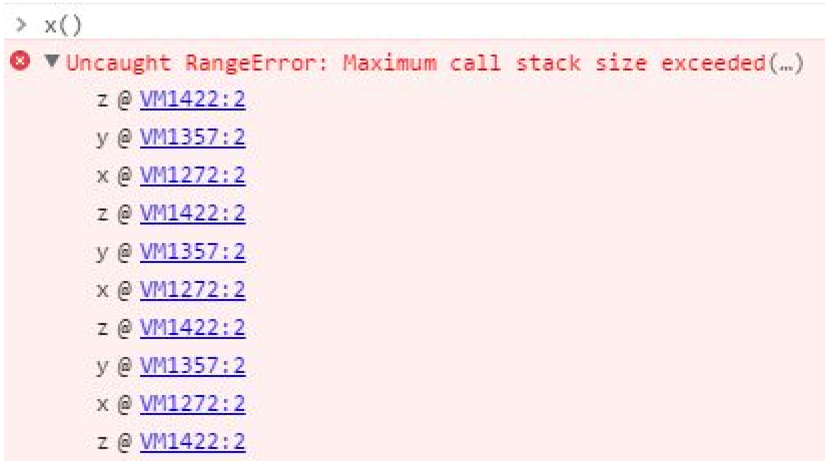
Find a pattern in the call stack

Step through the calls in the pattern

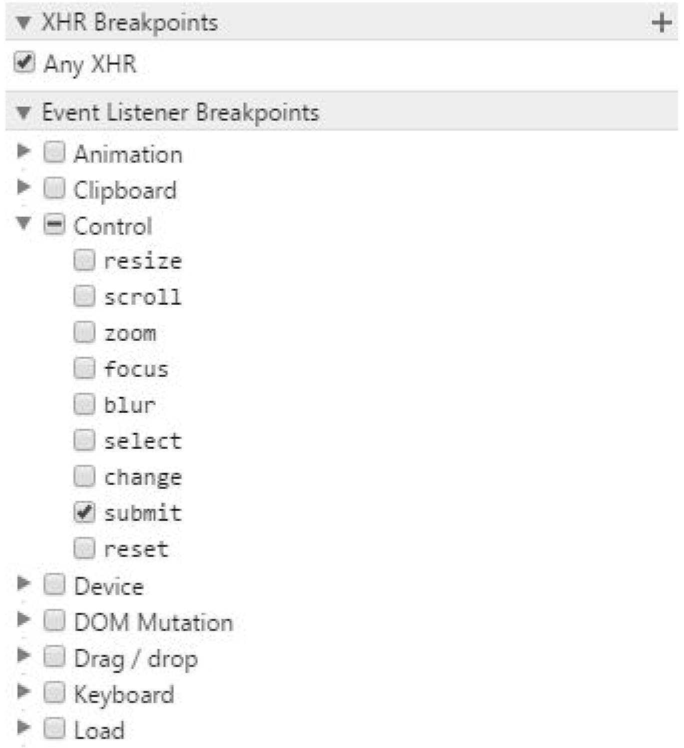
Find what breaks the assumptions

Only works for infinite loops

Can be hard to find the pattern



```
> x()
✖ ▼Uncaught RangeError: Maximum call stack size exceeded(…)
      z @ VM1422:2
      y @ VM1357:2
      x @ VM1272:2
      z @ VM1422:2
      y @ VM1357:2
      x @ VM1272:2
      z @ VM1422:2
      y @ VM1357:2
      x @ VM1272:2
      z @ VM1422:2
```



▼ XHR Breakpoints +

[✓] Any XHR

▼ Event Listener Breakpoints

- ▶ ☐ Animation
- ▶ ☐ Clipboard
- ▼ ☐ Control
  - ☐ resize
  - ☐ scroll
  - ☐ zoom
  - ☐ focus
  - ☐ blur
  - ☐ select
  - ☐ change
  - [✓] submit
  - ☐ reset
- ▶ ☐ Device
- ▶ ☐ DOM Mutation
- ▶ ☐ Drag / drop
- ▶ ☐ Keyboard
- ▶ ☐ Load

## Event Breakpoint

Use to find JS executed on events

Listen for the event in Dev Tools

Step through third party code

Look for application code

Good for an unknown code base

Hard to do with obfuscation

## Logging

Wraps console.log(), .error(), etc

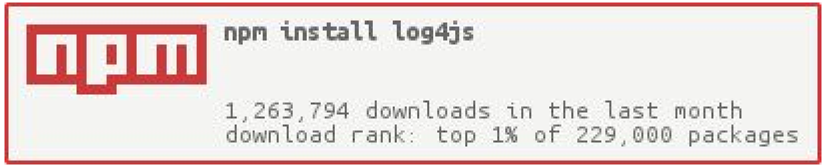
Send logs to a server, writes to DB

Helpful to read author's intention

Can also be used for performance

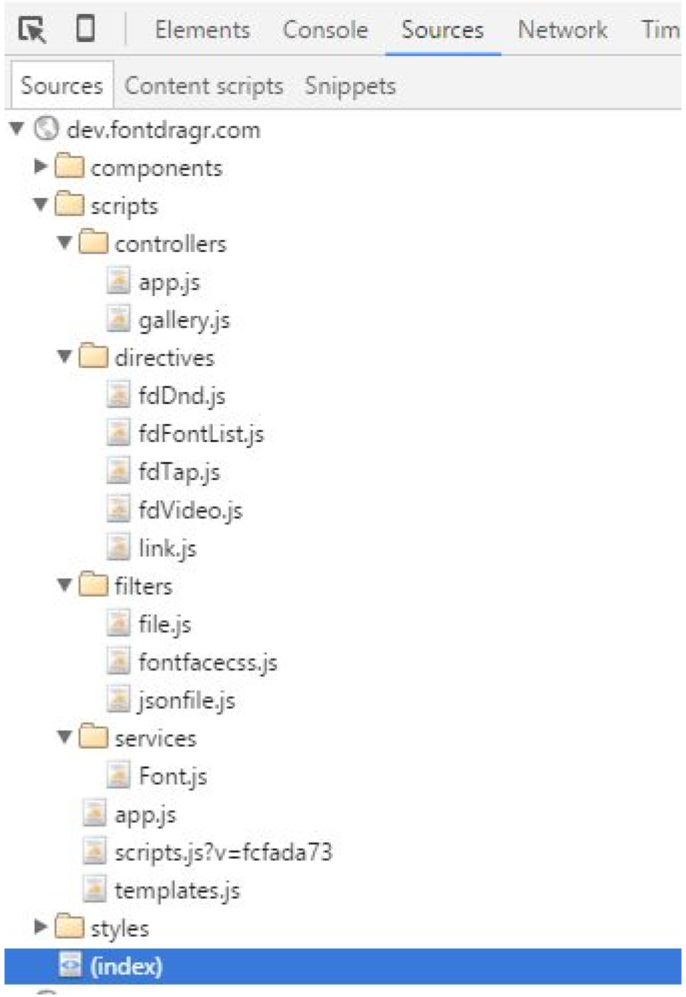
Good if it already exists

Can be painful to add after the fact



```
npm install log4js

1,263,794 downloads in the last month
download rank: top 1% of 229,000 packages
```



```
dev.fontdragr.com
  components
  scripts
    controllers
      app.js
      gallery.js
    directives
      fdDnd.js
      fdFontList.js
      fdTap.js
      fdVideo.js
      link.js
    filters
      file.js
      fontfacecss.js
      jsonfile.js
    services
      Font.js
    app.js
    scripts.js?v=fcfada73
    templates.js
  styles
  (index)
```

## Read the source

Don't be afraid of source code

Source Maps!

Use unminified third party files

Do minification yourself

Good when you have control

Bad for some frameworks (Angular)

## JSDoc

Docs on classes/methods/members

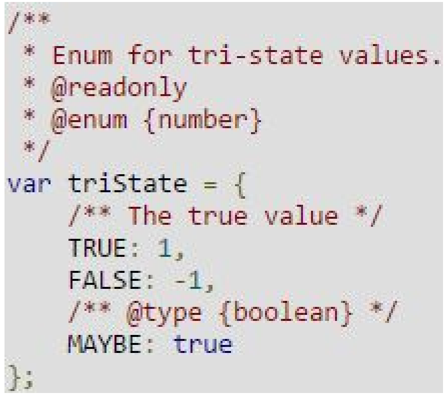
Use @param and @type

Get this working ASAP on new code

TypeScript doesn't have run time

Helps maintain code with intent

Difficult to add in later



```
/**
 * Enum for tri-state values.
 * @readonly
 * @enum {number}
 */
var triState = {
    /** The true value */
    TRUE: 1,
    FALSE: -1,
    /** @type {boolean} */
    MAYBE: true
};
```

# Debugging JavaScript Patterns

“When I was a JS hacker, I used to speak like a hacker, think like a hacker, reason like a hacker; when I became a JS developer, I did away with hacker things.”

Saint JavaScript
46 AE

## Contact Info

Zach Gardner

https://zgardner.github.io/

@zachagardner

HTML5/JavaScript Consultant

Keyhole Software

https://www.keyholesoftware.com

@inthekeyhole