Here Be Dragons
JavaScript Debugging

Rami Sayar - @ramisayar
Technical Evangelist
Microsoft Canada

Here be dragons!

http://www.desertislandsupplyco.com/mapper-delight-here-be-dragons

Agenda

- Types of Errors in JavaScript
- Frequent Locations of Errors in JavaScript
- Debugging and Introspection Tools
- Remote Debugging Node Processes
- Remote Debugging Front-End JavaScript


code. ship. repeat.
JS Open Day MTL - @RAMISAYAR

99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.

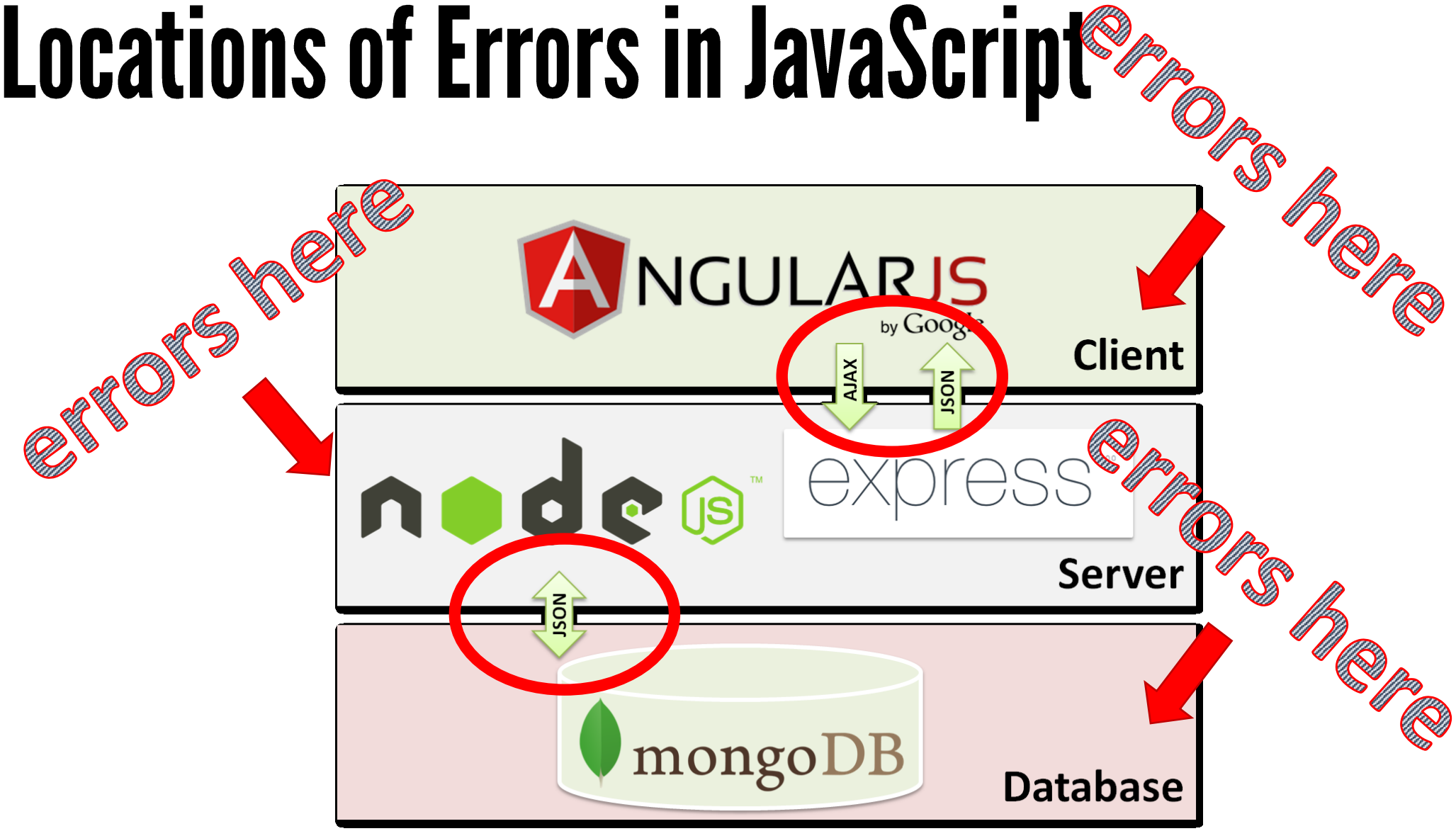
127 little bugs in the code...

Image from developermeme.com

Types of Errors in JavaScript

- Loading Errors
  - E.g. Incorrect syntax, minification errors, network-related errors, missing files, etc.
- Runtime Errors
  - E.g. Syntax errors, misspelled variables, illegal assignment, variables/classes don't exist, etc.
- Logic Errors
  - E.g. BUGS! But also errors due to format and parameters (JSON vs XML), etc...

Locations of Errors in JavaScript



JAVASCRIPT ERRORS

Debugging & Introspection Tools

DEMOS!

Typical Logic Errors in JavaScript

- Using this wrong an scoping errors
- Accidentally creating memory leaks
  - Dangling references to unused objects
  - Circular references
- Incorrect coercion, comparisons and equality
- Incorrect references to instance objects & prototypical inheritance errors
- So much more...

Debugging & Introspection Tools

- Allow you to control execution and walk through line-by-line
- Debug with breakpoints to conditionally stop execution
- Examine the call stack
- Pause on exceptions
- Stack trace exceptions

Demo: Introspection Tools

https://gist.github.com/sayar/3ffa68c1765b8e6d78e0

Debugging Node Apps – node-inspector

- Node Inspector supports almost all of the debugging features of the Blink DevTools, including:
  - Set breakpoints (and specify trigger conditions)
  - Step over, step in, step out, resume (continue)
  - Inspect scopes, variables, object properties
  - Edit variables and object properties
  - Break on exceptions
  - CPU and HEAP profiling
  - Network client requests inspection
  - Console output inspection

Debugging Node Apps – node-inspector

- Other cool stuff:
  - Node Inspector uses WebSockets, so no polling for breaks.
  - Remote debugging
  - Live edit of running code, optionally persisting changes back to the file-system.
  - Set breakpoints in files that are not loaded into V8 yet - useful for debugging module loading/initialization.
  - Embeddable in other applications

Demo: Node-Inspector

```
npm install -g node-inspector
node-debug app.js
```

IDE Demo: Code

It's free and runs on Linux/OSX/Windows.

There's also WebStorm and Eclipse.

Not All Debugging Happens During Development...

Occasionally...

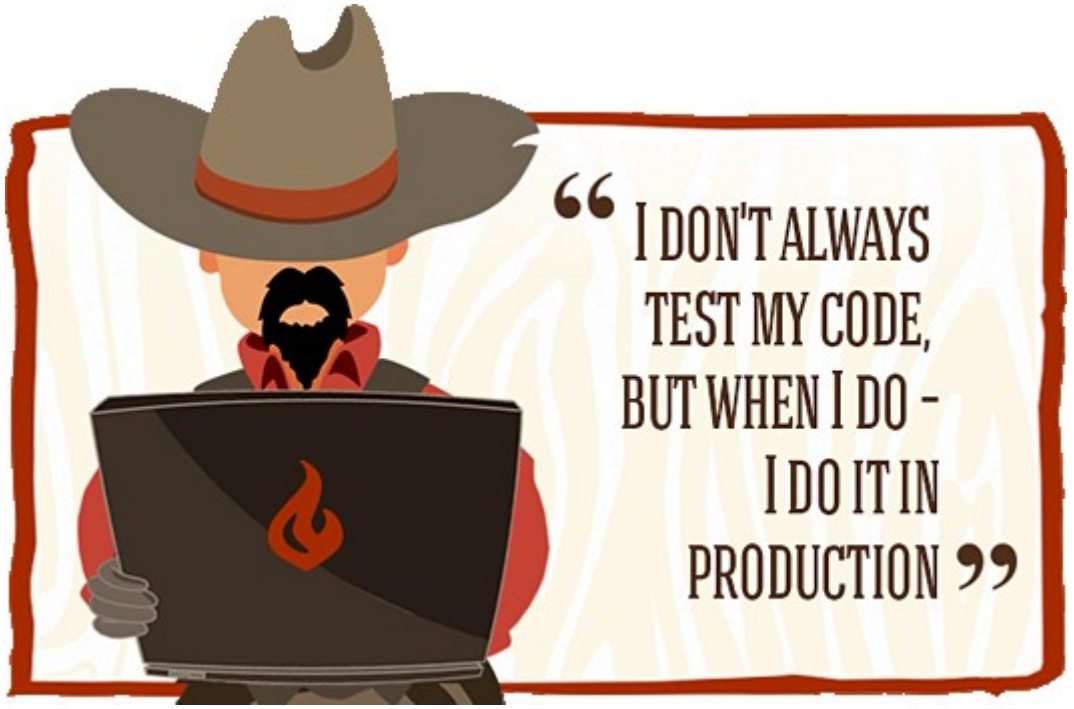


Image from https://www.siteground.com/blog/siteground-staging/

You Can Debug Production Node Applications!

Debugging node applications is easy if you know you want to debug... It gets harder when you want to connect to already-running production applications...

Enabling Debugging on a Node Process

On OSX/Linux...

```
$ pgrep -l node
    2345 node your/node/server.js
$ kill -s USR1 2345
```

On Windows...

```
tasklist /FI "IMAGENAME eq node.exe"
node -e "process._debugProcess()"
```

Connecting to a Remote Node Process

- With built-in node debugger:

```
$ node debug localhost:5858
```

- With node-inspector:

```
$ node-inspector
```

- With Visual Studio Code
  - Use Attach launch setting, debug... ☺

Demo: Debugging Remote Node Process

“your website doesn’t work on my pc”

Who hasn’t heard that one or a variant before?

Debugging JavaScript in Remote Browsers

Wait... what! You can do that?


VORLON.JS

Vorlon.JS

- An open source, extensible, platform-agnostic tool for remotely debugging and testing your JavaScript.
- Plugins:
  - Console: View logs and errors for your application.
  - Modernizr: View a list of supported and unsupported features.
  - DOM Explorer: Inspect the DOM tree and its corresponding styles.
  - Object Explorer: Display the living JavaScript variables tree.
  - XHR Panel: View XHR calls information sent by your devices.
  - ngInspector: Inspect your Angular.js scopes
  - Network Monitor: View network activities (XHR & resources loading).
  - Resources Explorer: Inspect local resources such as localStorage or cookies.

Getting Started with Vorlon.JS

```
$ npm install -g vorlon
$ vorlon
```

Add to your code.

```
<script
src="http://localhost:1337/vorlon.js"></script>
```

Open http://localhost:1337

Using Vorlon.JS to Remote Debug

- Deploy Vorlon.JS to a public server/PaaS/wtv.
  - As simple as a git push

- Add this to your public beta website:

```
<script
src="http://mywebsite.com/vorlon.js"></script>
```

Demo: Public Vorlon.JS - bit.ly/jsopendemo

Note: Don’t do this in production with SSL and authentication setup. Better yet, only do this in staging/beta.

Other Tools: Vantage

Vantage = CLI + SSH + REPL for your live node app

Demo: Vantage

```
$ npm install -g vantage
$ vantage tour
```


FIXED BUG IN CODE
WITHOUT CREATING NEW BUGS

What did we learn?

- Types of Errors & Locations in JavaScript
- Debugging and Introspection Tools
  - node-inspector
  - VS Code
- Remote Debugging Node Processes
  - node-inspector, VS Code
  - Vantage
- Remote Debugging Front-End JavaScript
  - Vorlon.JS

Mendhak

CC Attribution-ShareAlike 2.0 Generic

Thank You! Questions?

tw: @ramisayar | gh: @sayar

slideshare.net/ramisayar

Resources, References, Links

- http://www.toptal.com/javascript/10-most-common-javascript-mistakes
- https://developers.google.com/web/tools/javascript/breakpoints/
- http://www.w3schools.com/js/js_debugging.asp
- http://www.webreference.com/programming/javascript/rg31/index.html
- http://eloquentjavascript.net/08_error.html
- http://www.pluralsight.com/courses/fixing-common-javascript-bugs
- https://trackjs.com/
- http://www.standardista.com/javascript/15-common-javascript-gotchas/