

Expect the unexpected

Dealing with JavaScript errors in modern web apps

Mats Bryntse
Founder, @bryntum

Who is Mats Bryntse?

- From Stockholm 🇸🇪
- Founder of Bryntum
- Gantt & Scheduling JS UI components
- Web dev tools (testing, monitoring)
- @bryntum
- www.bryntum.com



JavaScript error handling 101

What's a JavaScript error?

- JavaScript errors are unhandled exceptions in **your code base**
- Or in the **frameworks** you use
- Doesn't matter where errors happen, poor user impression
- With JS codebases in the size of MBs, cannot ignore error handling + logging
- Good news - it's easy 😊

When a web site error happens, you as a dev see...

Developer Tools - http://localhost:8123/

Elements Console Sources Network Timeline Profiles Application Security Audits

chrome-extension://gp...jpkfnbl Preserve log

[HMR] Waiting for update signal from WDS...

✖ Uncaught Error: Cannot find module "main.js"

✖ Failed to load resource: the server responded with a status of 404 (Not Found)

[WDS] Hot Module Replacement enabled.

[WDS] Errors while compiling.

✖ ▶ multi main

Module not found: Error: Cannot resolve module 'main.js' in C:\Users\firas.koubaa\Desktop\reactapp

resolve module main.js in C:\Users\firas.koubaa\Desktop\reactapp

looking for modules in C:\Users\firas.koubaa\Desktop\reactapp\node_modules

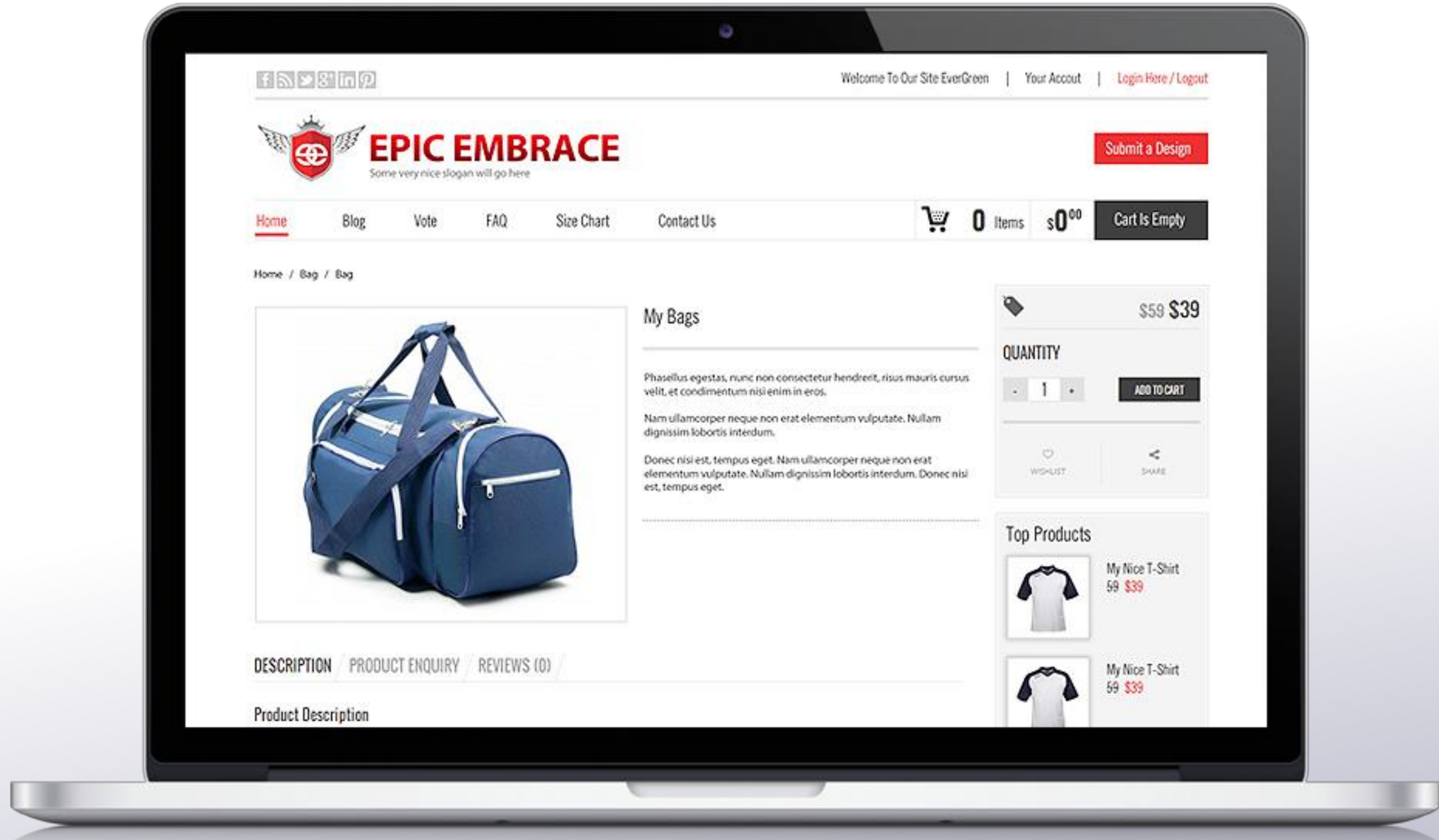
C:\Users\firas.koubaa\Desktop\reactapp\node_modules\main.js doesn't exist (module as directory)

resolve 'file' main.js in C:\Users\firas.koubaa\Desktop\reactapp\node_modules

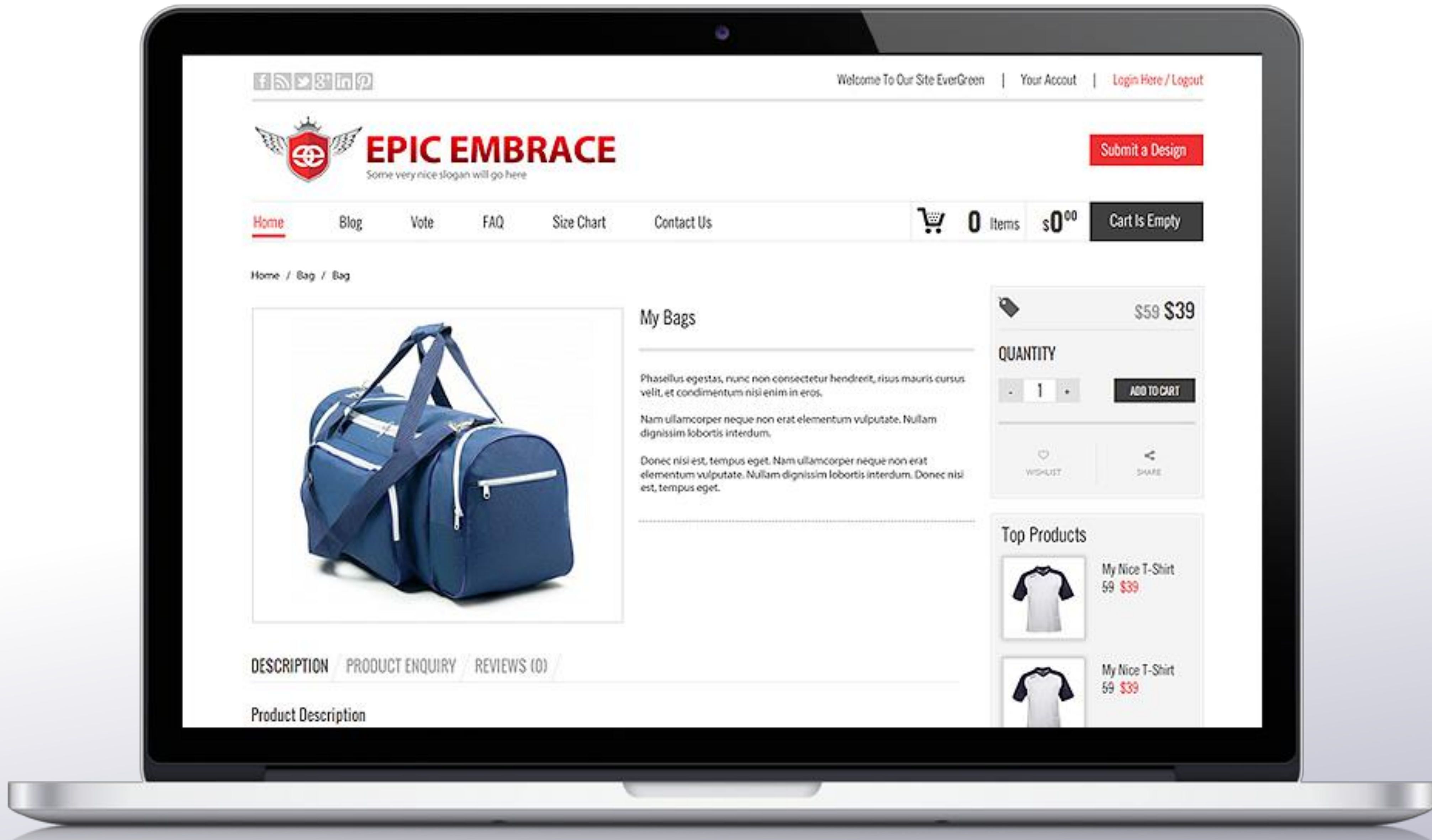
resolve file

C:\Users\firas.koubaa\Desktop\reactapp\node_modules\main.js doesn't exist

What does the user see when there is a JS error?





Nothing





Two scenarios for the end user:

- Error is captured by you. **User notified**
- **Or.....**
- Nothing happens for user, will probably try same action again.
- If you're really lucky, user will contact you  

Live demo of an error



Beware of 'Script error.' messages

- Happens for scripts on external domains
- No message or callstack
- Fix: Add **crossorigin="anonymous"** to the script tag

```
<script crossorigin="anonymous" src="https://unpkg.com/react.production.min.js"></script>
```



Making an error logger in < 10 minutes

1. Create single table db

- date, message, file, line, callstack etc

```
CREATE TABLE `error` (  
  `msg` char(60),  
  `callstack` char(1000),  
  ...  
) ENGINE=InnoDB DEFAULT  
CHARSET=utf8
```

2. PHP script to receive error data and store it in DB

```
<?php  
  
// LOG TO DB  
$link = getLink();  
  
$command = "call insert_error('$msg', '$url', '$stack', ...)";  
  
$result = mysqli_query($link, $command);
```

3. Setup client side logging

- Log message, file, line, stack etc..
- Add any extra meta relevant for your debugging (userId/name/...)

```
// Poor mans JS error logger
```

```
window.onerror = (msg) => {  
    new Image().src = `log.php?msg={msg}`;  
}
```

```
throw new Error("Ooops");
```

Live demo: logging an error

Manual error logging, things to consider

- Store error logs in a database on a **non-production** server
- Throttle logging on client side + server side
- Probably we only care about the first error on a page



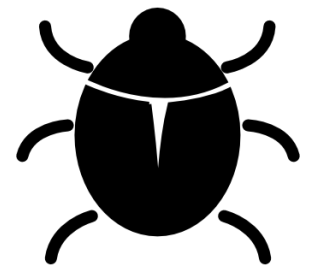
How do **JavaScript bugs** relate to testing?

Let your users help you

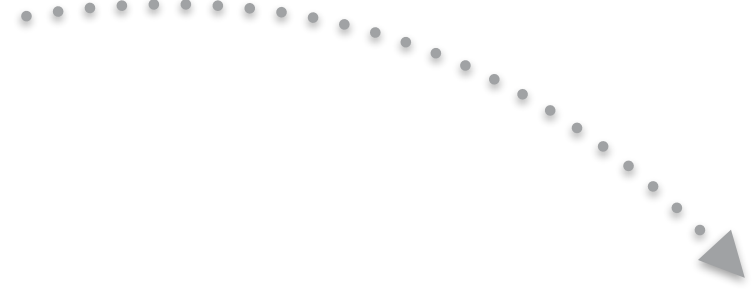
- Not every SaaS company can afford a full time QA department
- Your users will eventually encounter bugs in production
- Users === “*Smart monkey testers*”
- *How* users trigger bugs is **valuable data** which we can harvest
- Ideally, a session crash would generate a **runnable test case**

The bug fix cycle

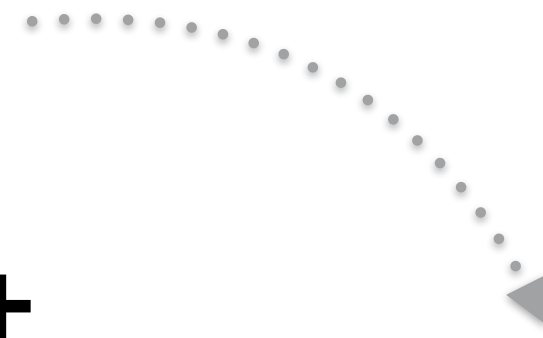
Bug life cycle



Occurrence



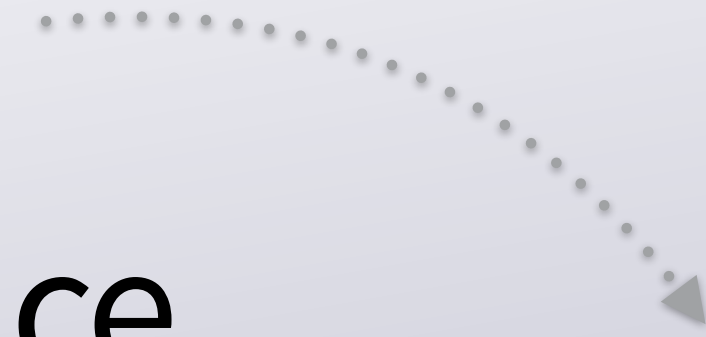
Report



Investigate



Reproduce



Fix



Developers need context to fix a bug

Debug context wish list

1. Error message
2. File / line number
3. Call stack
4. Screenshot
5. Step by step description
6. Log of user / browser session activity
7. Seeing the user reproduce the error
8. Live breakpoint in production environment
9. Live breakpoint on my localhost, in my fav browser



Once you have breakpoint, it's all downhill!

“A live breakpoint is worth a 1000 callstacks”

Error

```
at Object.module.exports.request (/home/vagrant/src/kumascript/lib/kumascript/caching.js:366:17)
at attempt (/home/vagrant/src/kumascript/lib/kumascript/loaders.js:180:24)
at ks_utils.Class.get (/home/vagrant/src/kumascript/lib/kumascript/loaders.js:194:9)
at /home/vagrant/src/kumascript/lib/kumascript/macros.js:282:24
at /home/vagrant/src/kumascript/node_modules/async/lib/async.js:118:13
at Array.forEach (native)
at _each (/home/vagrant/src/kumascript/node_modules/async/lib/async.js:39:24)
at Object.async.each (/home/vagrant/src/kumascript/node_modules/async/lib/async.js:117:9)
at ks_utils.Class.reloadTemplates (/home/vagrant/src/kumascript/lib/kumascript/macros.js:281:19)
at ks_utils.Class.process (/home/vagrant/src/kumascript/lib/kumascript/macros.js:217:15)
```

Common approaches to error handling

Do nothing

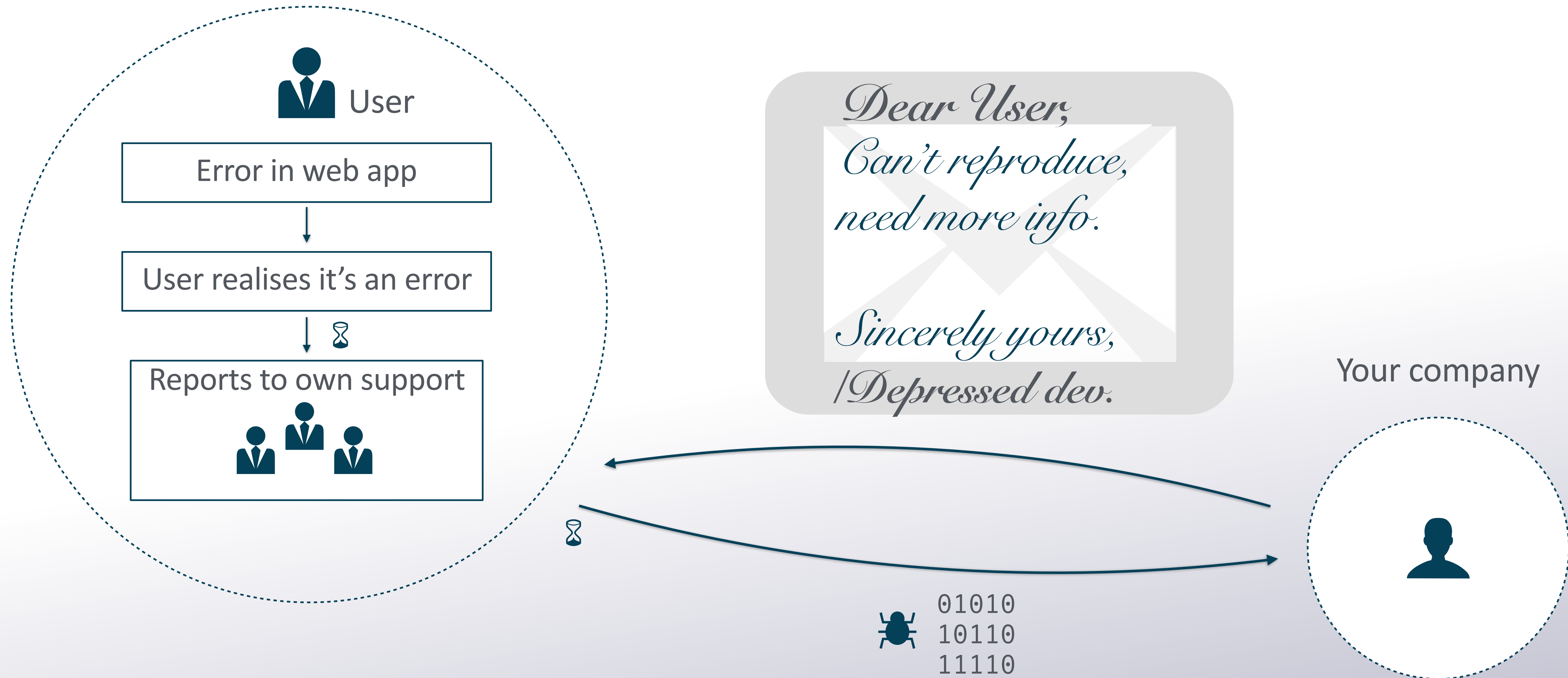
- Pros: Cheap
- Cons: Bugs live forever

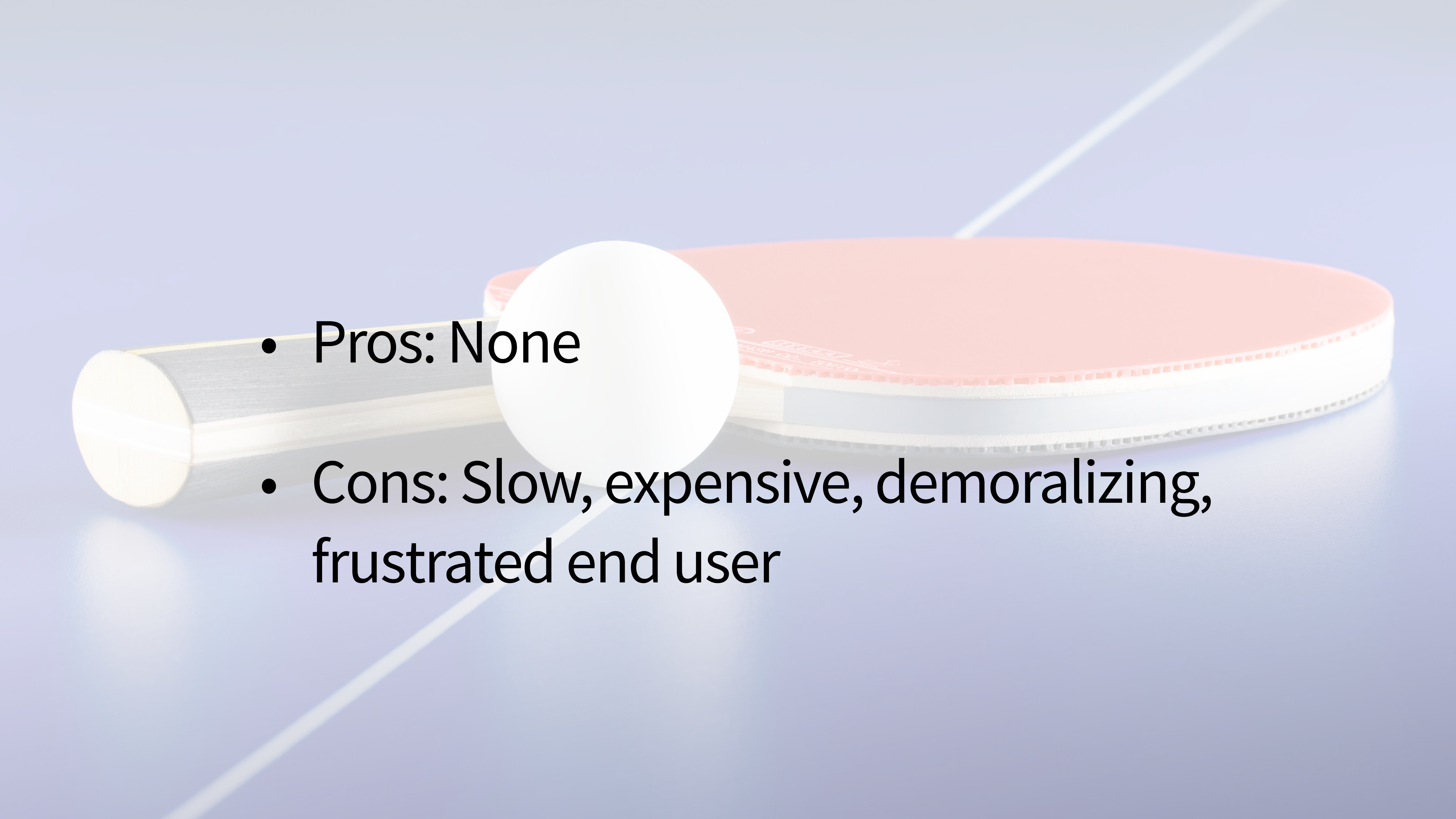


Email ping pong



Email ping pong - Enterprise version



A 3D rendering of a wooden matchstick and a red pill blister pack on a blue background. The matchstick is positioned horizontally on the left, and the blister pack is on the right. A white circle is overlaid on the matchstick, partially obscuring the text.

- Pros: None

- Cons: Slow, expensive, demoralizing, frustrated end user

Roll your own logger

```
1 //notify from web app
2 window.onerror = log;
3
4 function log(msg) {
5     //load 1px GIF, send stack trace
6     new Image().src = 'log.php?msg=...'
7 }
```

Roll your own logger

```
1 //notify from web app
2 window.onerror = log;
```

- Pros: Simple, get basic error info. **Awareness**
- Cons: Lots of code to scan through

Using a 3rd party logger

Airbrake.io

OverOps

- Pros: Tons of data, call stack, console logs, ajax, user activity.
Enables you to search for the error

- Cons: Slow, tons of data to parse, manual work, code to review

bugsnap

STACKHUNTER

Quick poll



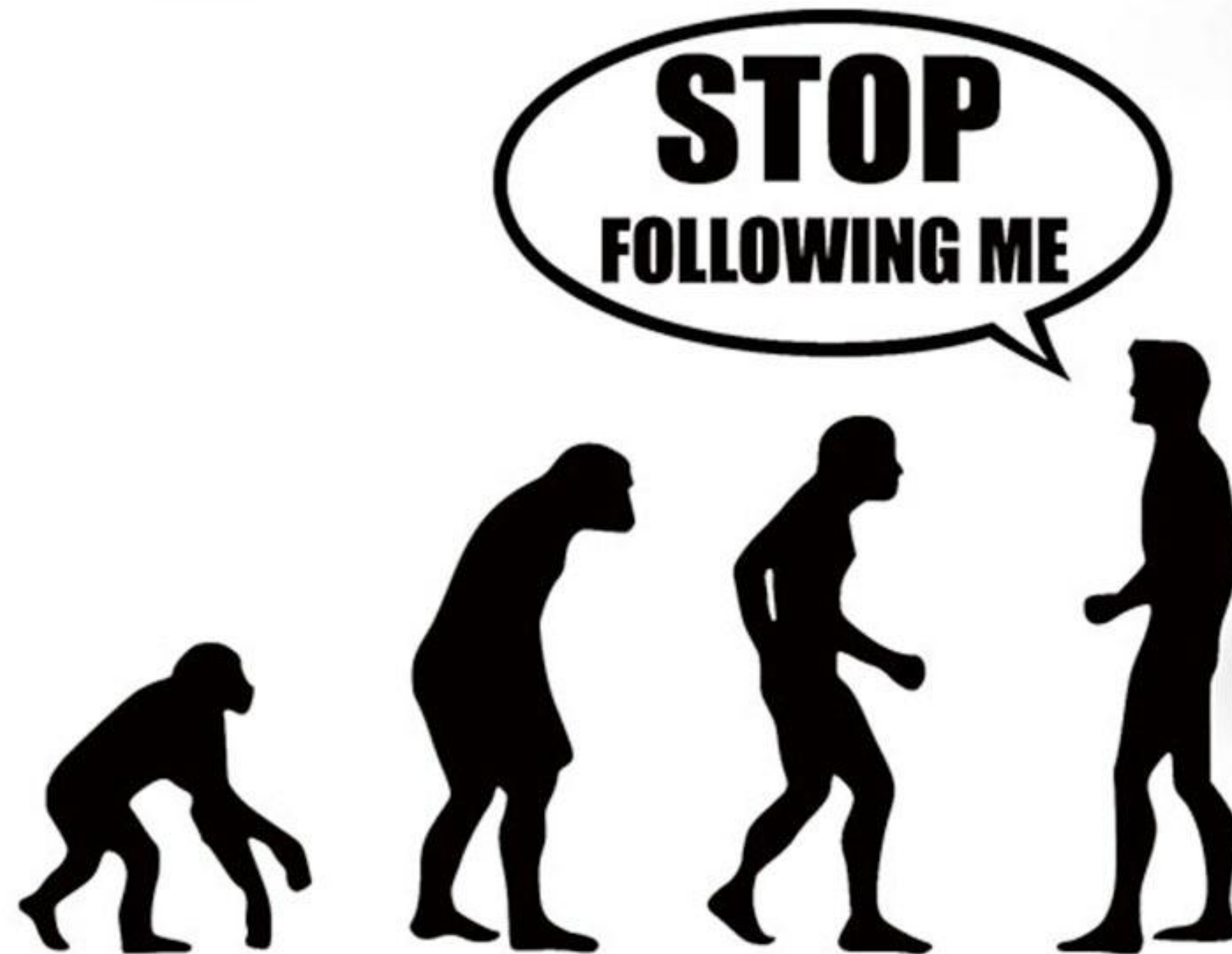
```
1 //notify from web app
2 window.onerror = log;
3
4 function log(msg) {
5     //load lpx GIF, send stack trace
6     new Image().src = 'log.php?msg=...'
7 }
```

[Airbrake.io](#) [OverOps](#)

[Rollbar](#) [RAYGUN](#) [SENTRY](#)

[bugsnag](#) [STACKHUNTER](#)

Evolution of monitoring tools



First generation tools

- Sentry, Rollbar, TrackJS, RayGun, NewRelic, StackHunter...
- Basic error logging
- Call stack + context
- Timeline
- Dashboard
- Statistics
- Focus on raising **awareness**, data gathering

Second generation tools

- LogRocket, SessionStack, FullStory...
- Generates video of the user session
- Replay & **view** the error
- Focus on **showing you the bug**, not just data gathering
- Bonus: videos help reveal bad UX design

But, to reproduce the bug we would like a test case.



RootCause - debugging JavaScript errors in 2018

RootCause Person 2

Secure https://app.therootcause.io/#organizations/Bryntum/dashboard

Select app... mats ? ☰

Bryntum > dashboard Dashboard Errors Feedback Applications

Daily Activity

14D 7D

Date	Activity Count
Jan 07	14
Jan 08	11
Jan 09	31
Jan 10	9
Jan 11	7
Jan 12	8
Jan 13	4
Jan 14	0
Jan 15	2
Jan 16	7
Jan 17	6
Jan 18	10
Yesterday	8
Today	3

Latest Activity

- maxim** marked an issue as Fixed 5 hours ago
- maxim** marked an issue as Fixed 6 hours ago
- mats** marked an issue as Reproduced 19 hours ago
- mats** marked an issue as Reproduced a day ago

Latest Errors

- Cannot read property 'scrollTo' of null 4 hours ago in Ext Scheduler
- Cannot read property 'isEqual' of undefined 13 hours ago in Ext Gantt
- null is not an object (evaluating '_0x5e77x1[_0x255f[466]]') 18 hours ago in TaskBoard
- Unresolved is not defined a day ago in Root cause app

FEEDBACK

DEMO TIME!

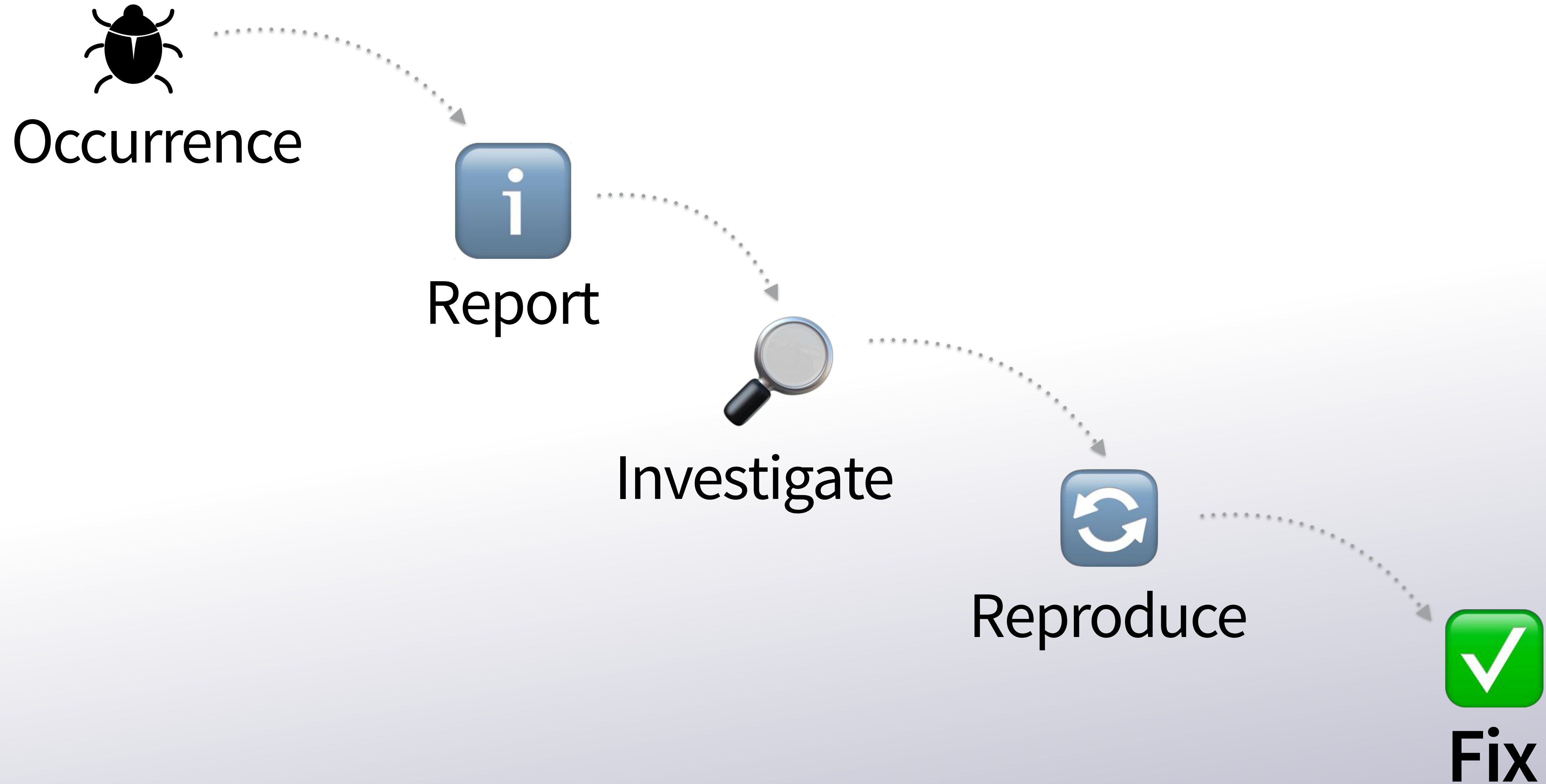
Cuts 99% of communication out

- **No** need for QA / end users to email devs with crash reports, step by step
- **No** need for devs to notify QA that bug is fixed

Technical details

- Recorder: 100% vanilla JS
- Screenshots: HTML2Canvas
- Dashboard: Ext JS
- Replay studio powered by Siesta

>> Fast forward into context



Privacy concerns / GDPR...?



Summing up:

- Fix your external script tags. Never see “Script error”. Ever.
- Your users can help you find bugs with the right tool
- Don’t rely on users reporting bugs manually
- Automate your error reporting

Sign up for free here: <https://app.therootcause.io>

Questions?



@the_rootcause
@bryntum

Join discussion at [slido.com](https://www.slido.com) with #test2018