



Debugging Effectively

Colin O'Dell

@colinodell

Colin O'Dell

- Lead Web Developer at Unleashed Technologies
- PHP developer since 2002
- PHP League Member
 - league/commonmark
 - league/html-to-markdown
- PHP 7 Upgrade Guide e-book
- @colinodell / www.colinodell.com

Overview

- I. Importance of debugging
- II. Debugging process
- III. Tools & Techniques
- IV. Q&A

Debugging is...


A person in winter gear, including a backpack and a hat, is walking away from the camera on a snowy mountain path. In the background, a large, snow-covered mountain peak rises against a dark, cloudy sky. A fence made of wooden posts and wire runs along the right side of the path. The overall scene is dimly lit, suggesting a late afternoon or early morning setting.

(adjective)

Debugging is...

important

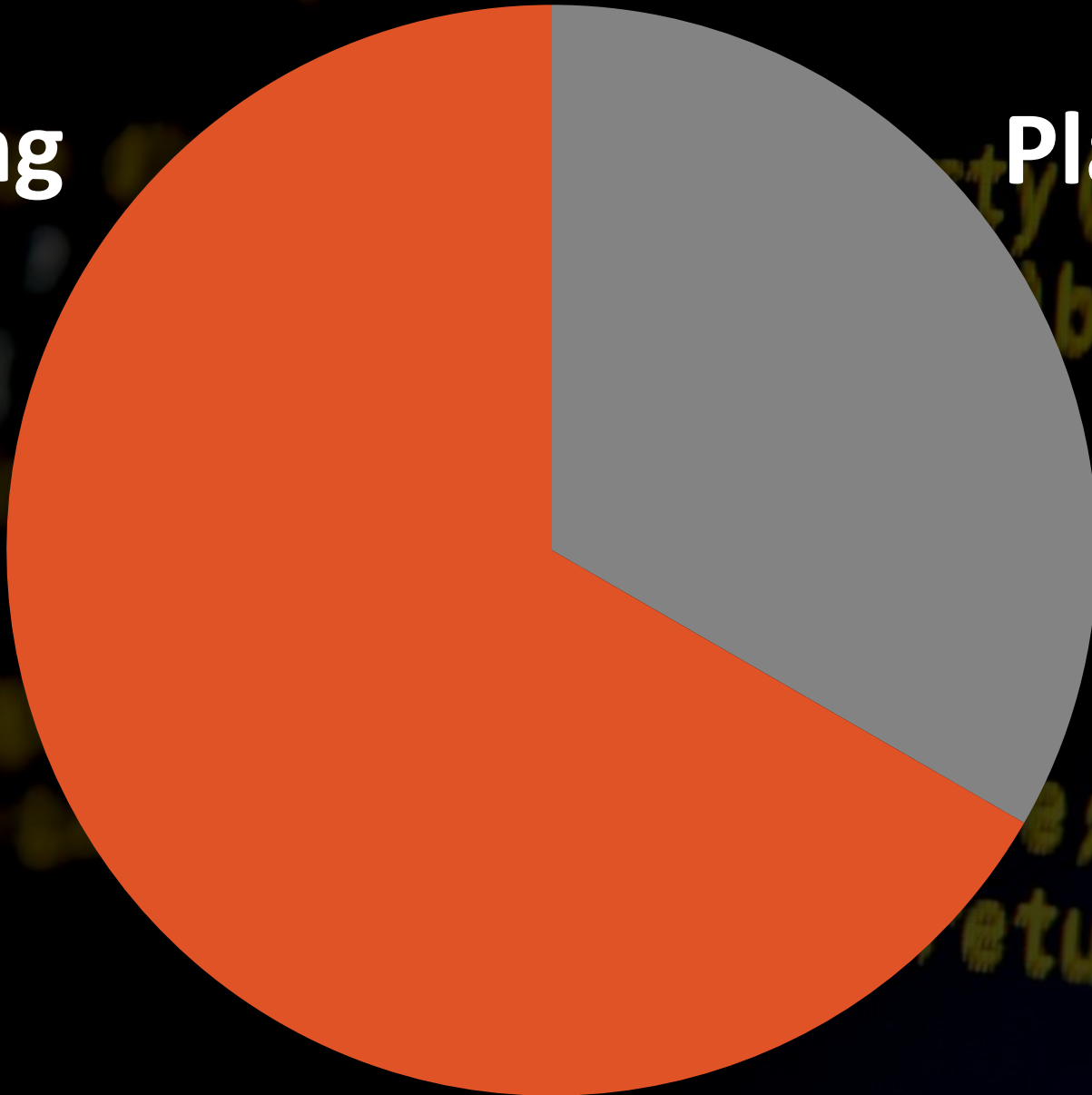


A photograph of two hikers with large backpacks ascending a steep, snow-covered mountain trail. The hiker in the foreground is on the left, and the one further up is in the center. A wooden fence runs along the right side of the path. The background shows a vast, snowy landscape under a cloudy sky.

Debugging is...
the single most
important skill in
programming.

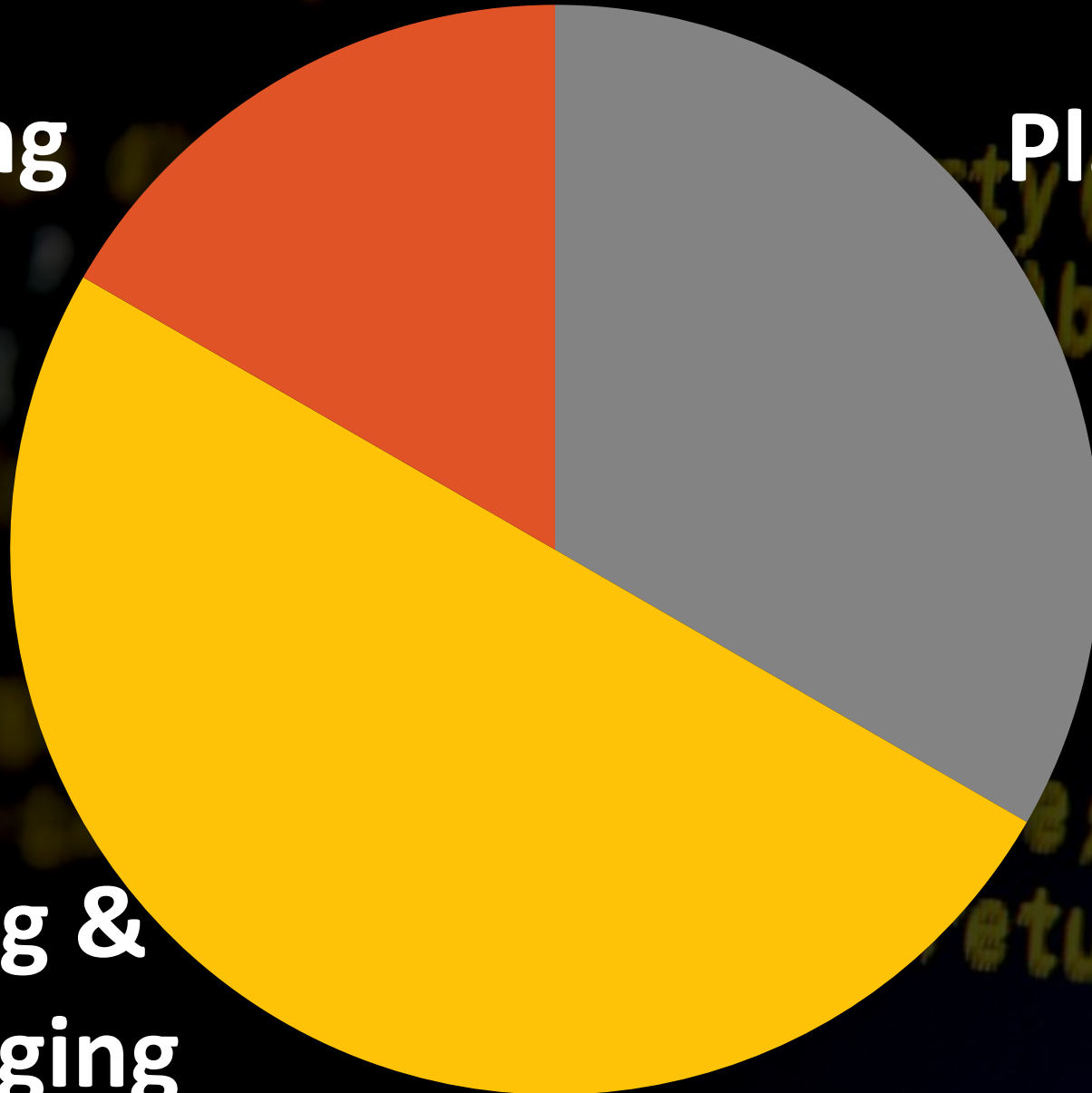
Coding

Planning



Coding

**Testing &
Debugging**



Planning



Debugging is the **process** of finding and resolving bugs or defects that prevent correct operation of computer software or a system.

– Wikipedia

Process is the foundation of effective debugging



Process

Gain **experience** with tools and code



Develop a “**sixth sense**”



“Sixth Sense”

Experience

Process

Junior Developers

- Try the “usual” steps
 - `app/console cache:clear`
 - `composer install`
 - `chmod -R 777 *`
- Google the error
 - Try every solution
- Ask somebody else
 - Co-worker
 - StackOverflow post
- Give up

XY Problem

- I want to solve problem X
- How do I solve X?

XY Problem

- I want to solve problem X
- ~~How do I solve X?~~
- Solution Y might work
- How can I do Y?

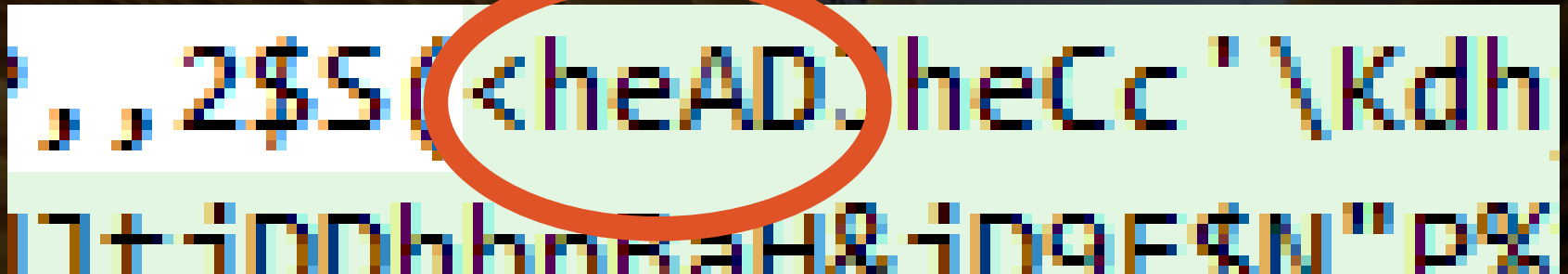
```

+::^`0QXq<]5=2KC03q7d/[]_IP,,2$S(<heADJheCc'\KdhjTq/8!kje*?lmkB&TX:,?!;i
QGtsZ(/Zn=CUoigLcr&+i%SJUJtjDDhbpBaH&jD9F$N"P%1pQgp+[RRKnrmcjA?[i1b3@F
?><meta http-equiv="X-UA-Compatible" content="IE=8" />kFj+#DePW"=6`i-#
$tag = '<meta http-equiv="X-UA-Compatible" content="IE=8" />';

$regex = '{(<head[>]*?>)}i';

$response->setBody(
    preg_replace($regex, '$1'.$tag, $response->getBody())
);

```



,,2\$S(<heADJheCc'\Kdh

1. Don't parse HTML with regex
2. Solve problems the right way

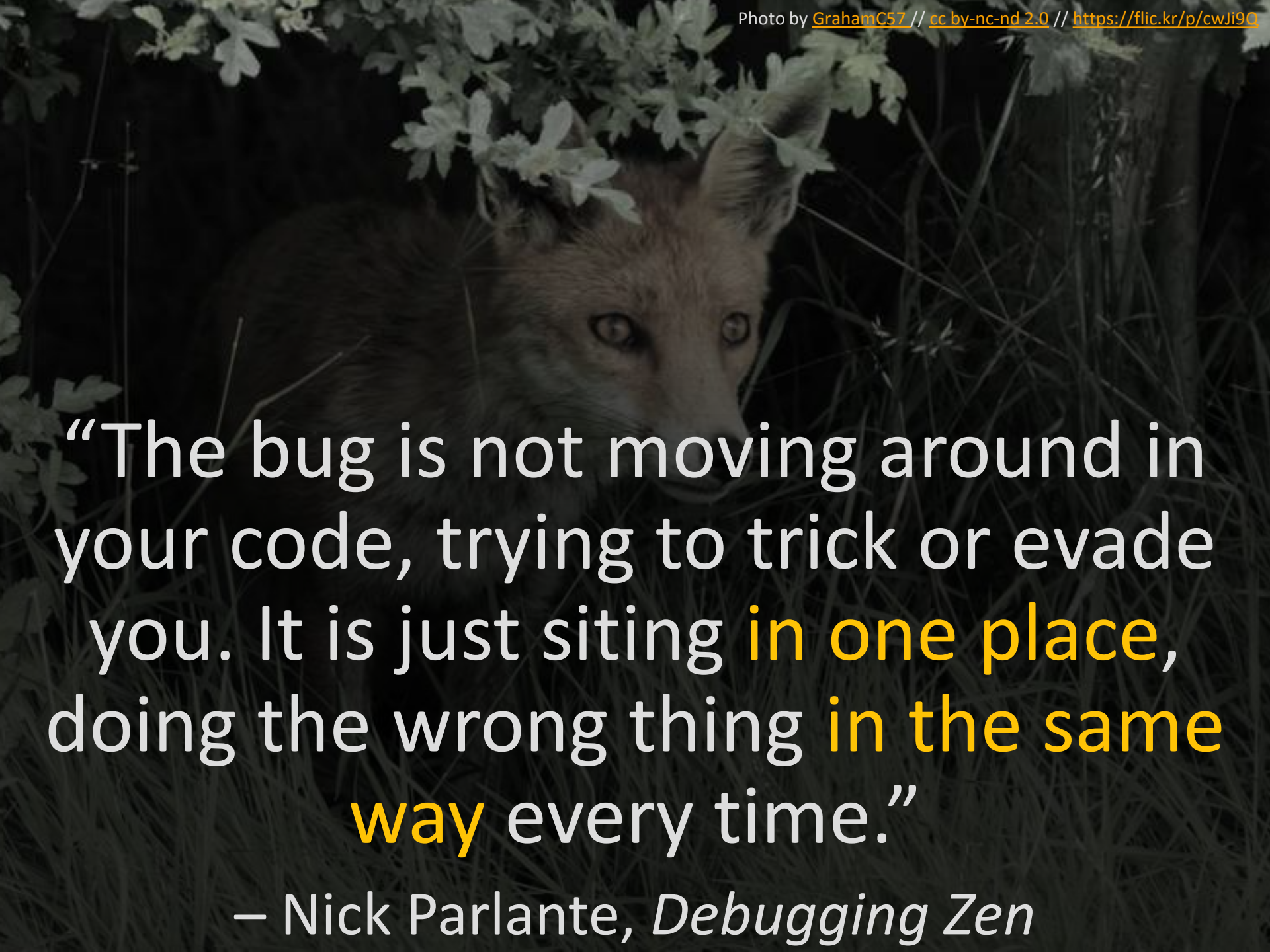
“I don’t know why”

“For some reason”

“Doesn’t make sense”

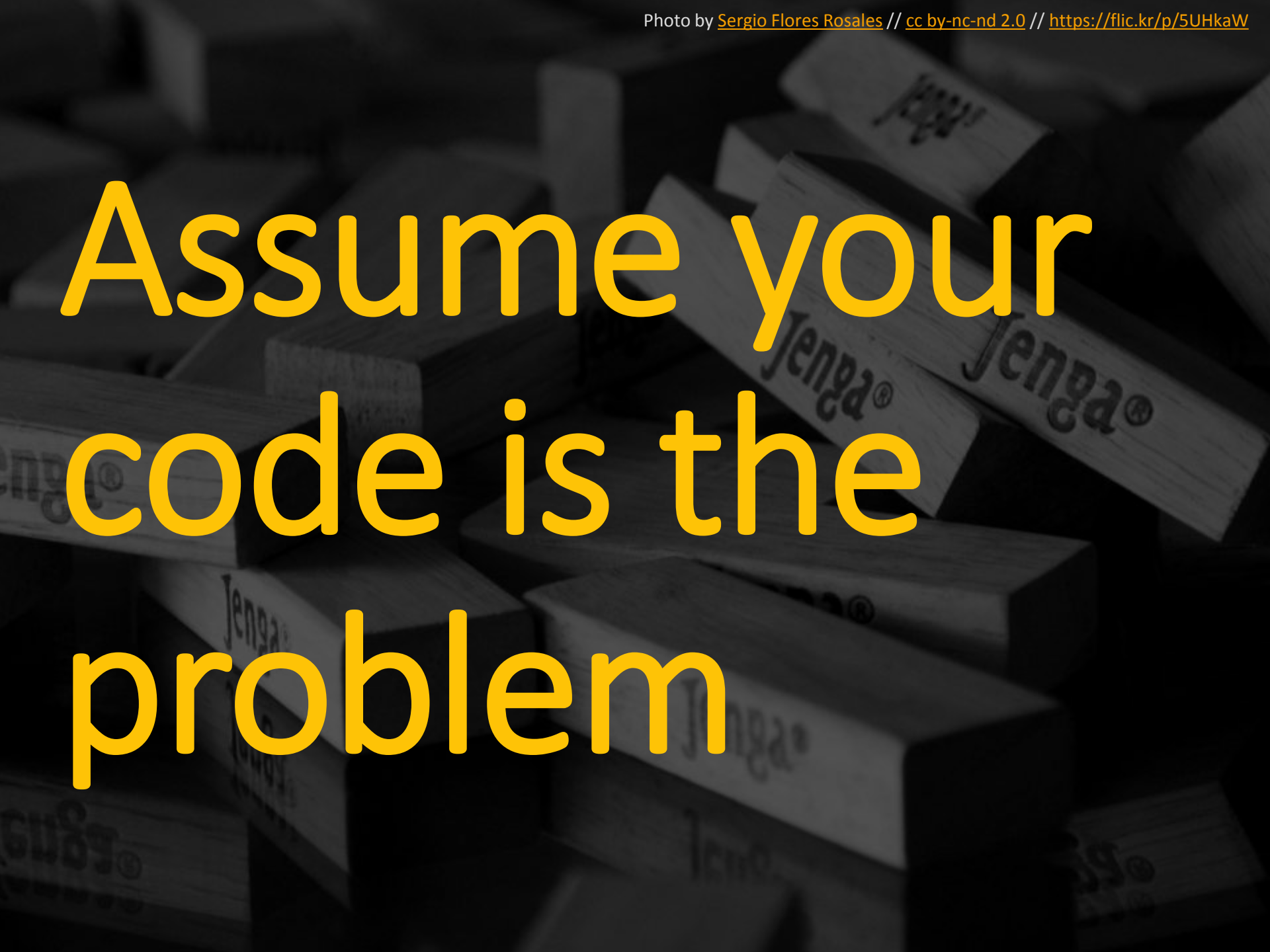
A close-up photograph of Spock, a Vulcan character from the Star Trek franchise, sitting in the command chair of a starship's bridge. He has his characteristic dark hair with a high forehead and a serious, logical expression. He is wearing a dark blue Starfleet uniform. The background is dark with several glowing rectangular panels, likely control consoles or viewscreens.

**Bugs are
logical**

A photograph of a fox's head and shoulders, partially obscured by green leaves and branches. The fox is looking directly at the camera with a calm expression. The background is dark and out of focus, suggesting a natural, wooded environment.

“The bug is not moving around in your code, trying to trick or evade you. It is just sitting **in one place**, doing the wrong thing **in the same way** every time.”

— Nick Parlante, *Debugging Zen*

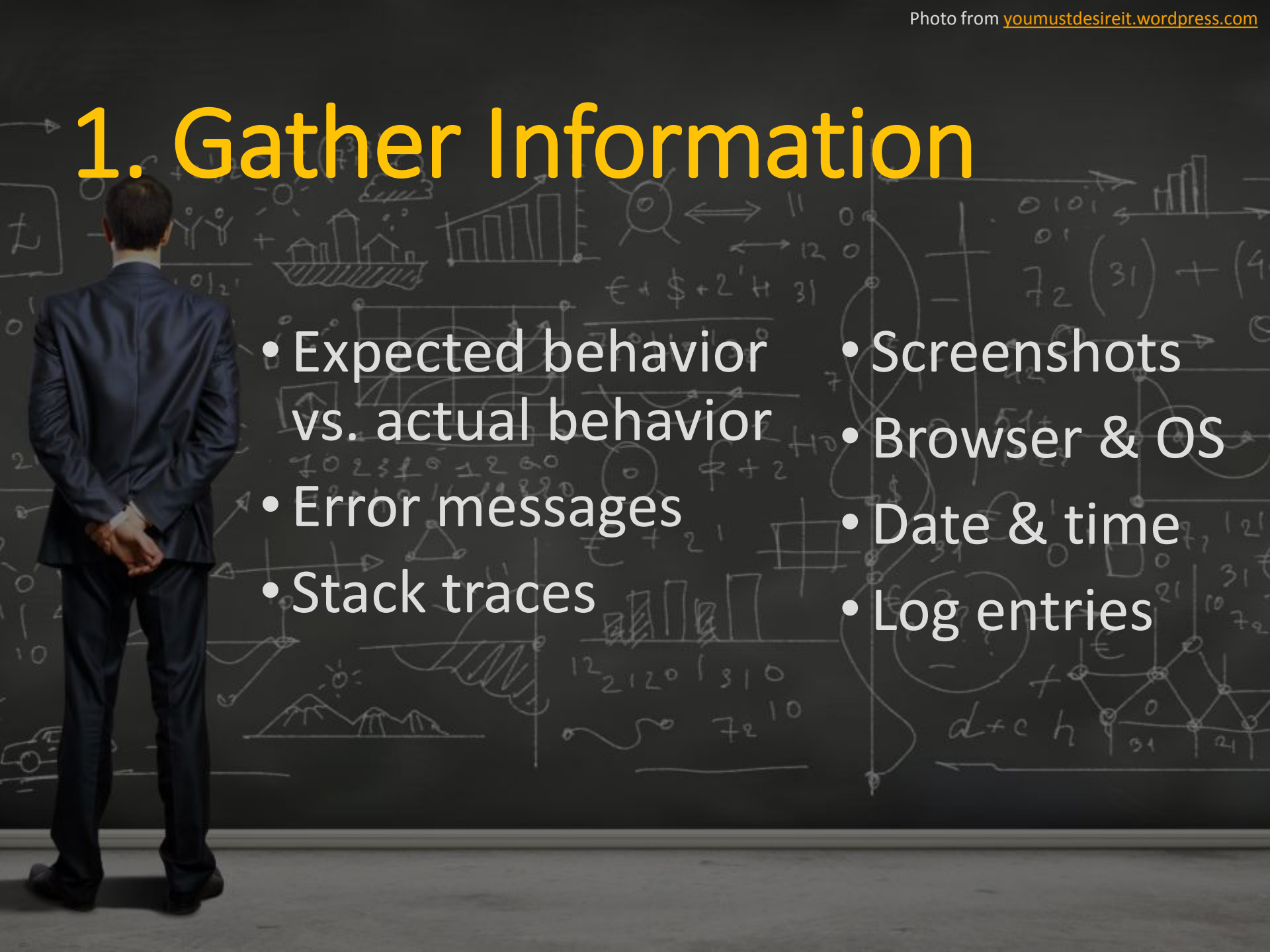
A background image showing a pile of wooden Jenga blocks. The blocks are scattered and some have the 'Jenga' logo visible. Overlaid on this background is the text 'Assume your code is the problem' in a large, bold, yellow font.

Assume your code is the problem

Systematic Approach

1. Gather information
2. Replicate the issue
3. Identify the culprit
4. Fix it & re-test
5. Mitigate future occurrences

1. Gather Information

- 
- Expected behavior vs. actual behavior
 - Error messages
 - Stack traces
 - Screenshots
 - Browser & OS
 - Date & time
 - Log entries

2. Replicate the Issue

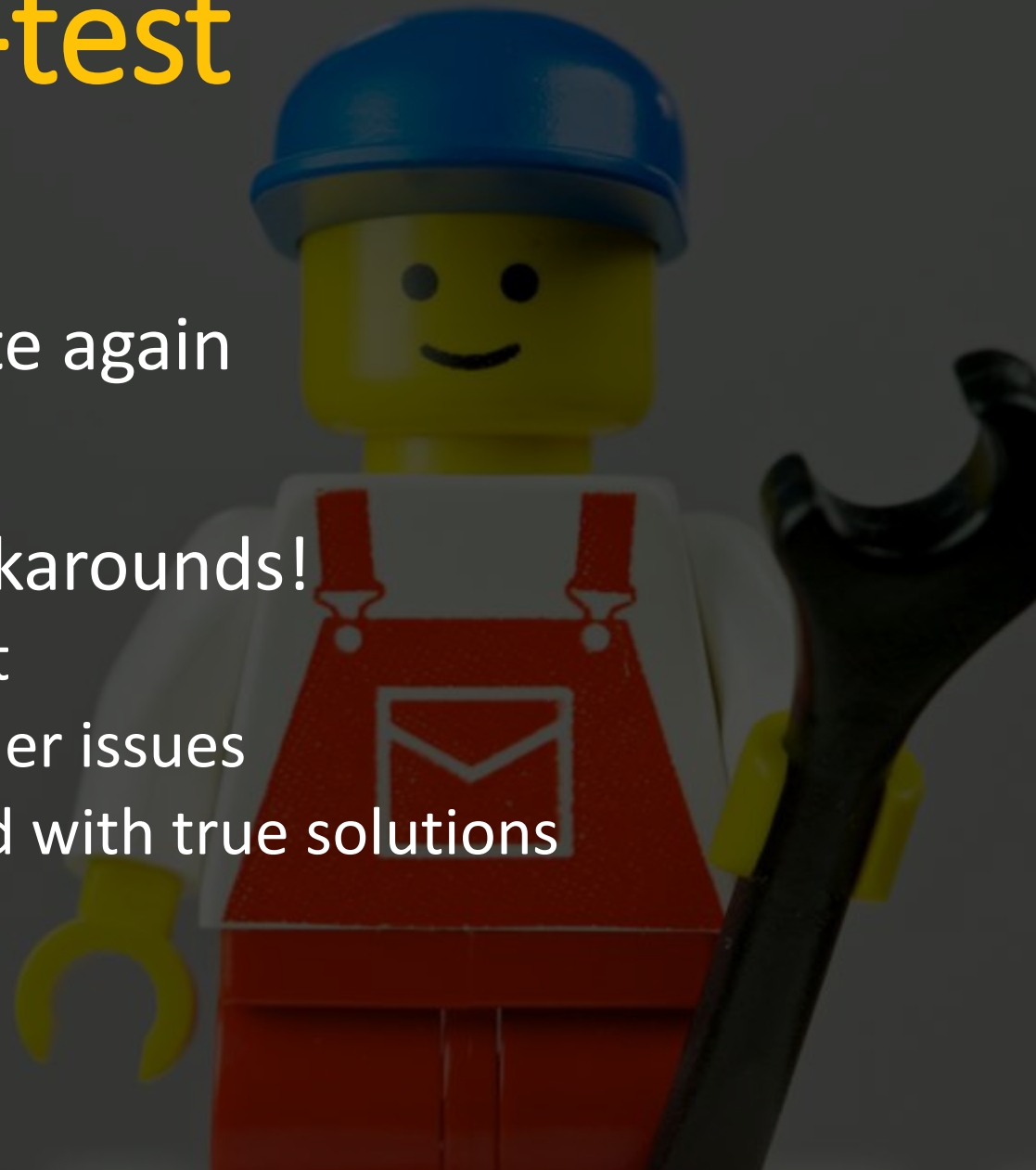
Be able to replicate with **100% certainty**

3. Identify the Culprit

- Be methodical
- Make no assumptions
- Understand the bug

4. Fix & Re-test

- Attempt to replicate again
- Avoid XY problem
- No temporary workarounds!
 - Add technical debt
 - May introduce other issues
 - Never get replaced with true solutions



5. Mitigate Future Occurrences

- Add an automated test
- Share your new knowledge
 - Project documentation
 - Blog post
 - StackOverflow
- Submit patch upstream

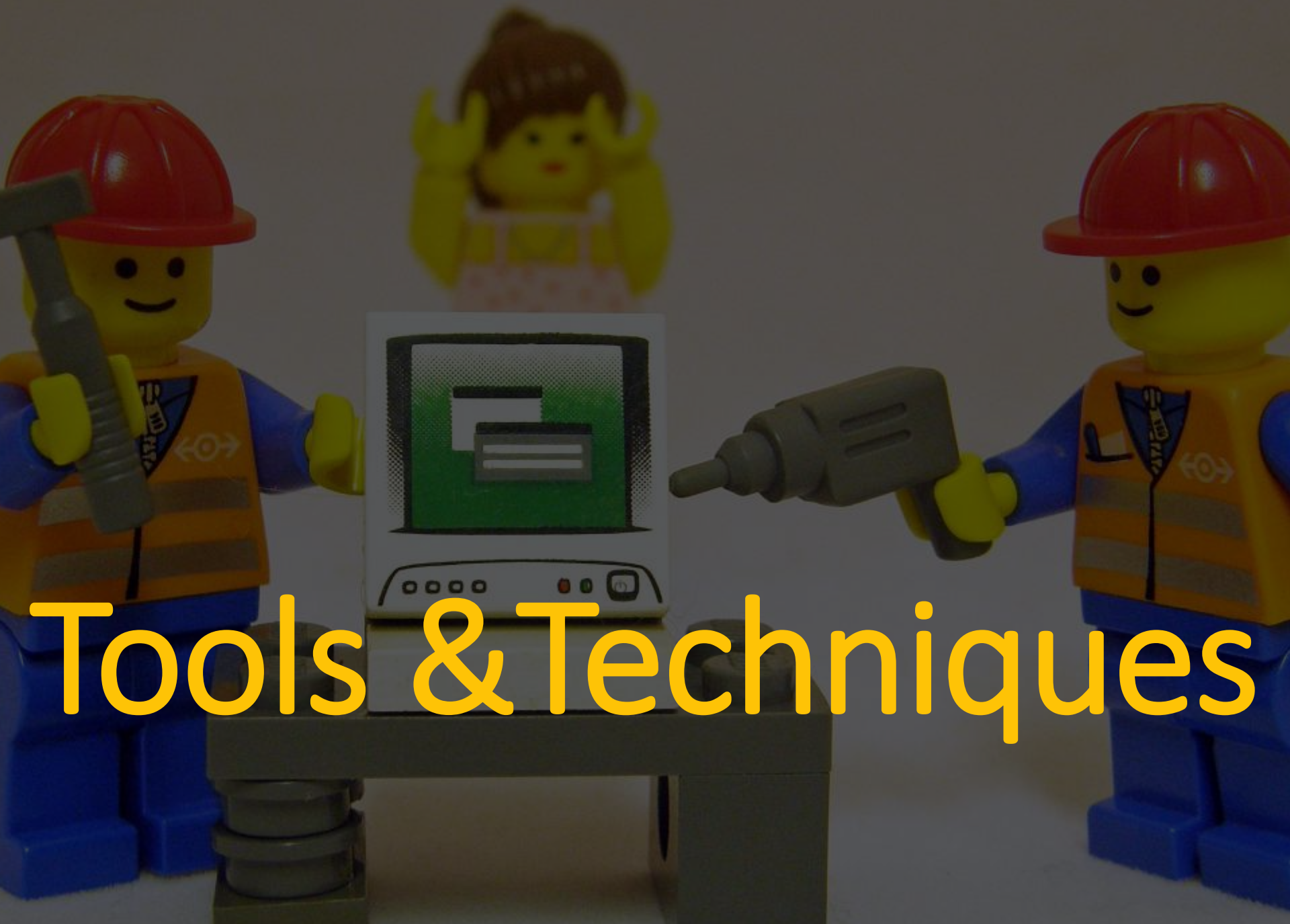


Recap

1. Gather information
2. Replicate the issue
3. Identify the culprit
4. Fix it & re-test
5. Mitigate future occurrences

Long-Term Results

- Gain experience
- Learn how the system works
- Build heuristics
- Boost confidence



Tools & Techniques

Two essential tools

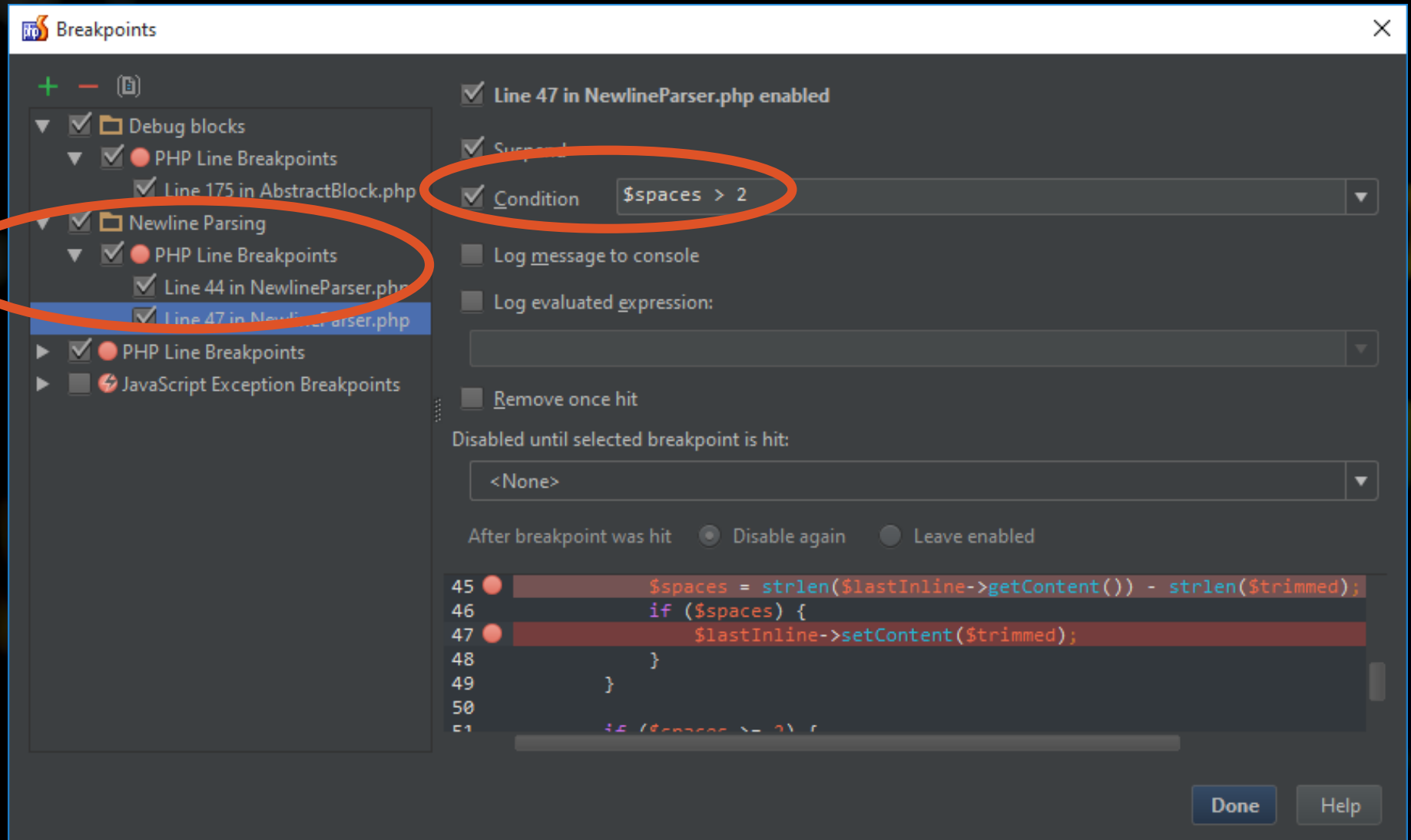
- Integrated development environment (IDE)
- Interactive debugger

Integrated Development Environment

- Minimum features:
 - Syntax highlighting
 - Auto-completion
 - Fast code navigation
 - Debugger

Interactive Debugger

- Pause code execution
 - Breakpoints
 - Conditional breakpoints
- Step through execution
- Examine variables
- Explore call stack



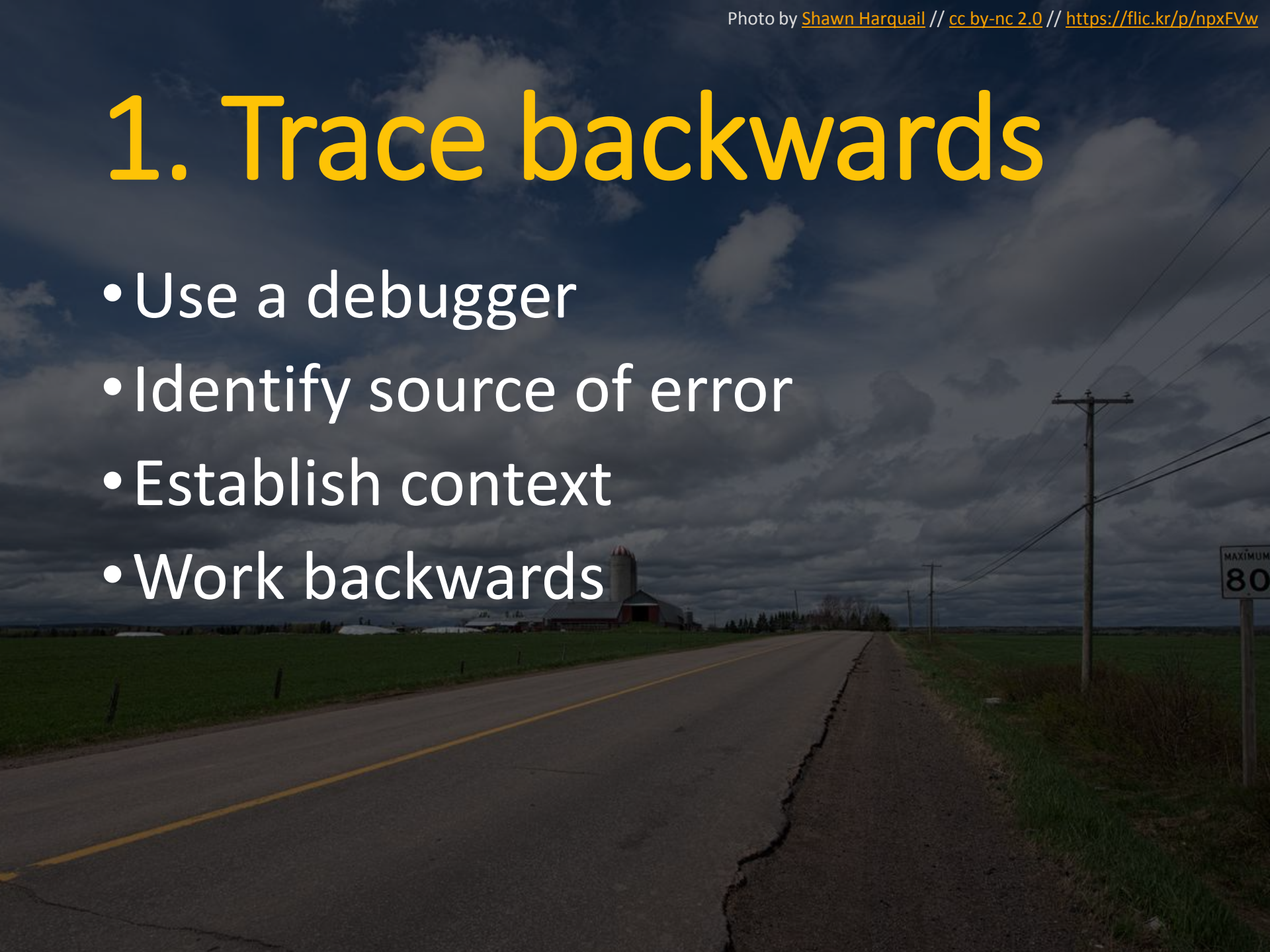
Techniques

1. Trace backwards from known issue
2. Divide & conquer
3. Use tools
4. Get help
5. Take a break

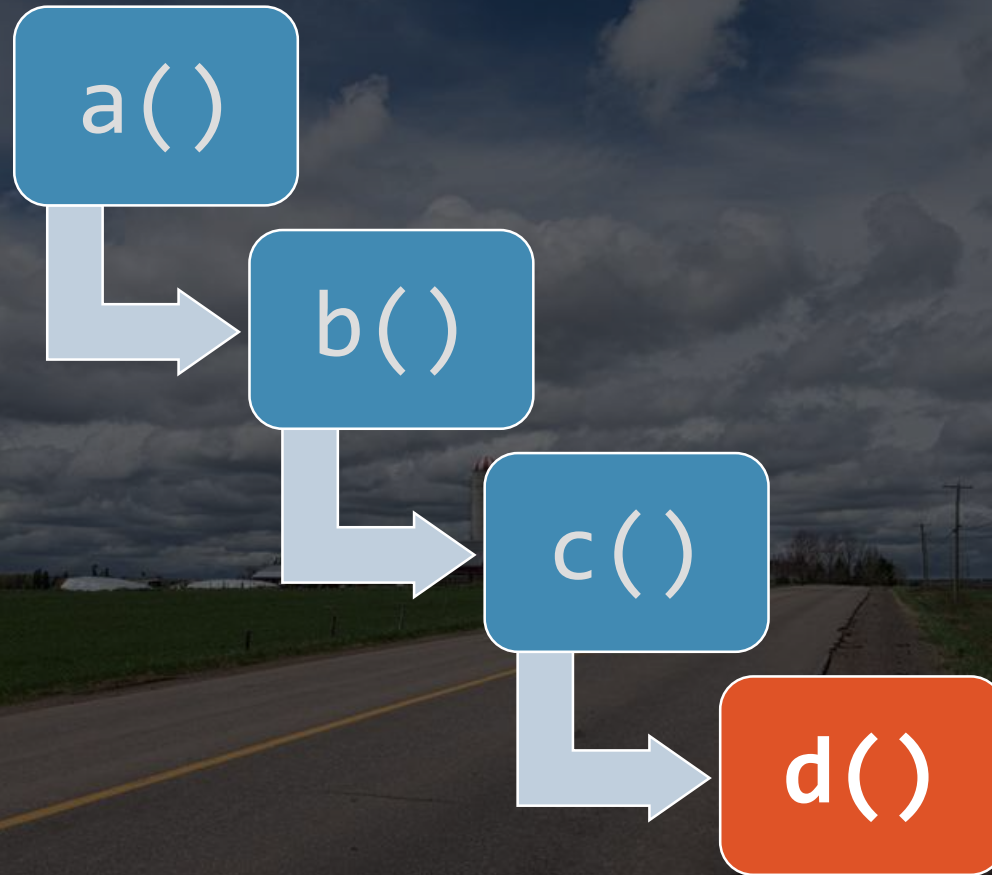


1. Trace backwards

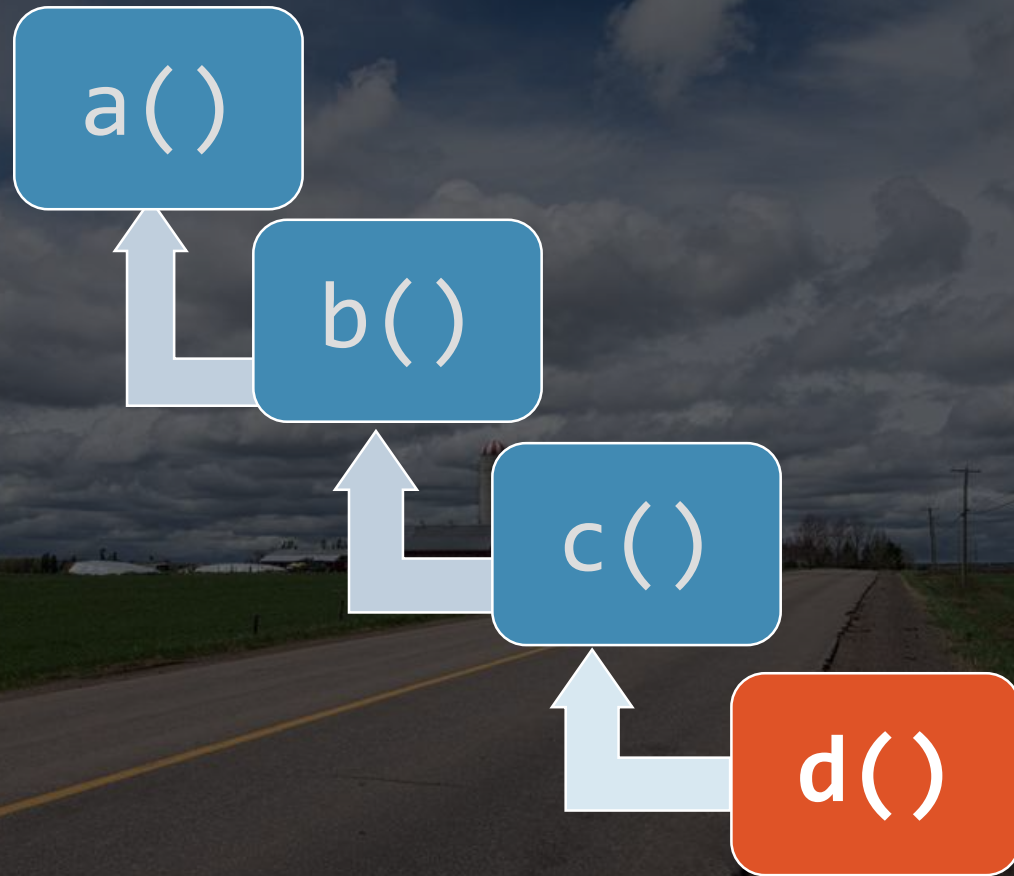
- Use a debugger
- Identify source of error
- Establish context
- Work backwards



1. Trace backwards

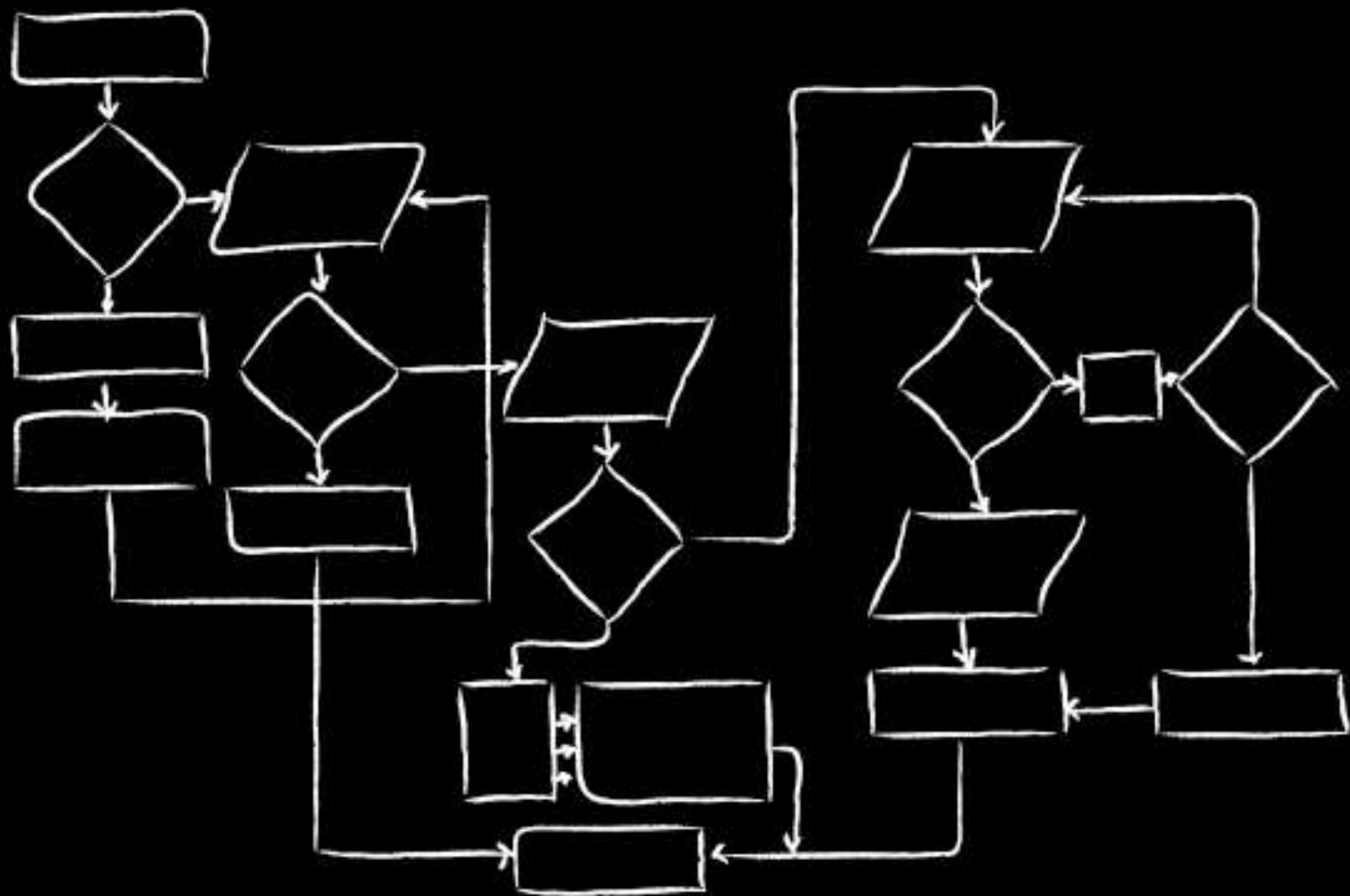


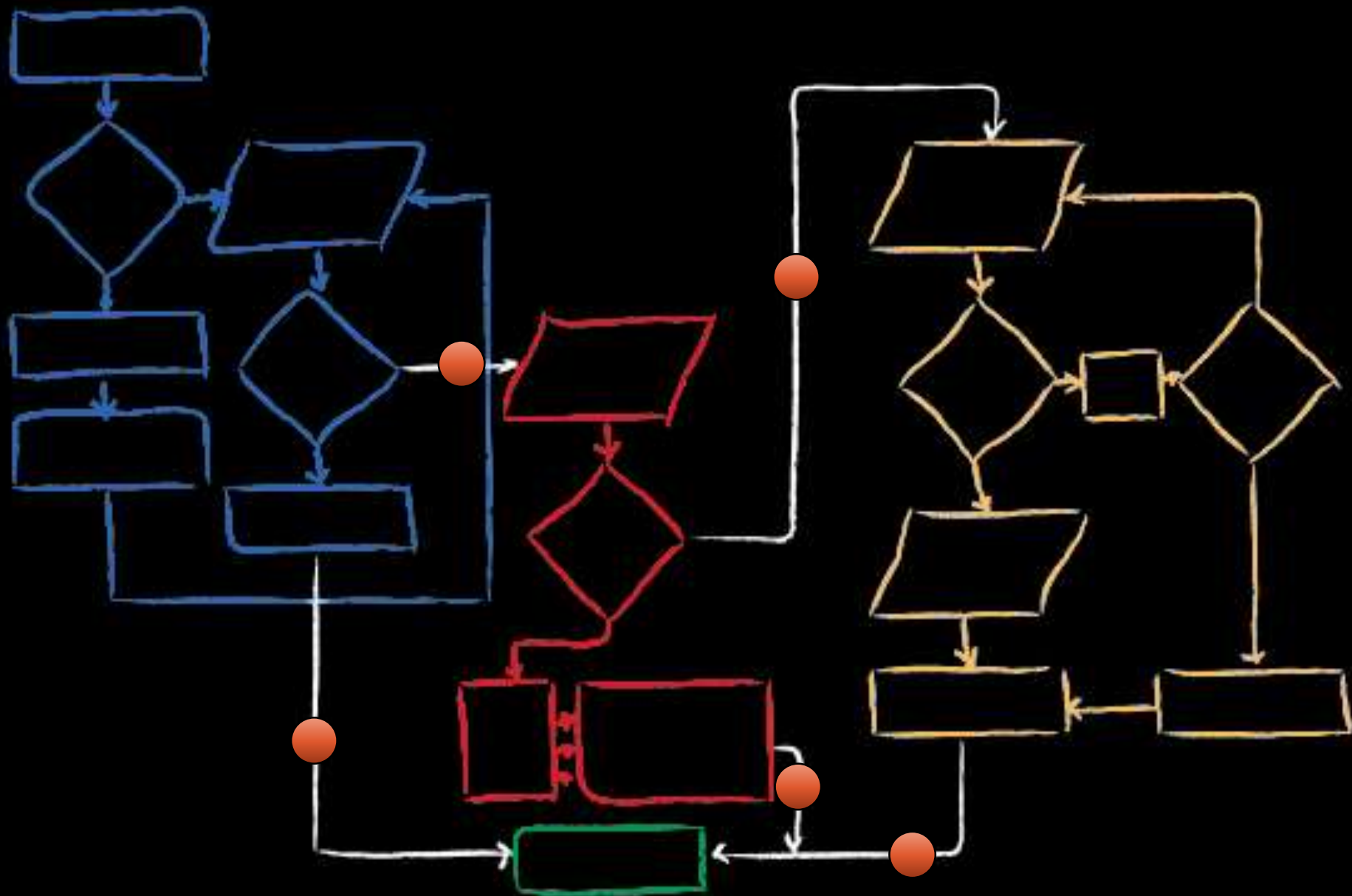
1. Trace backwards

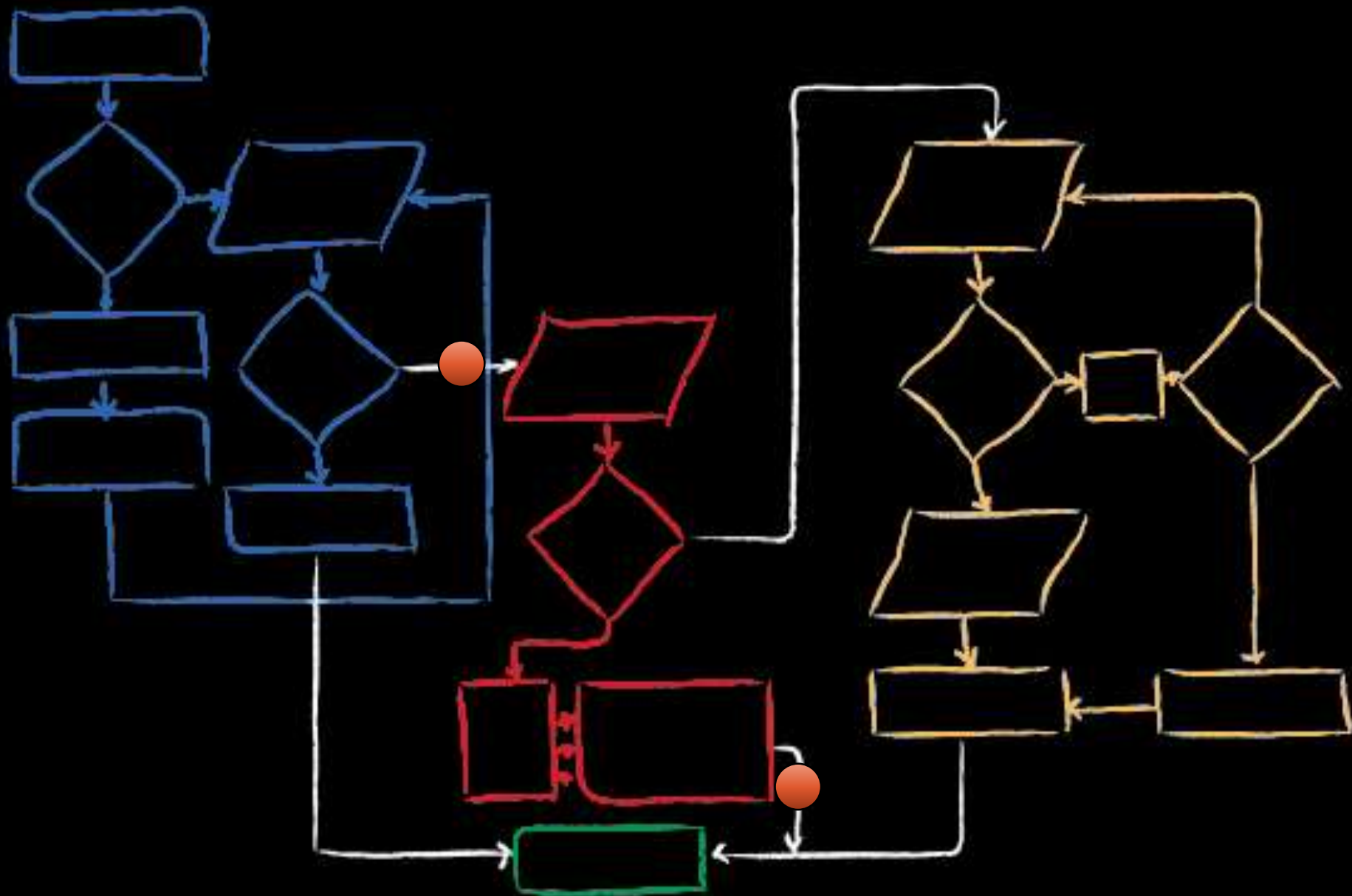


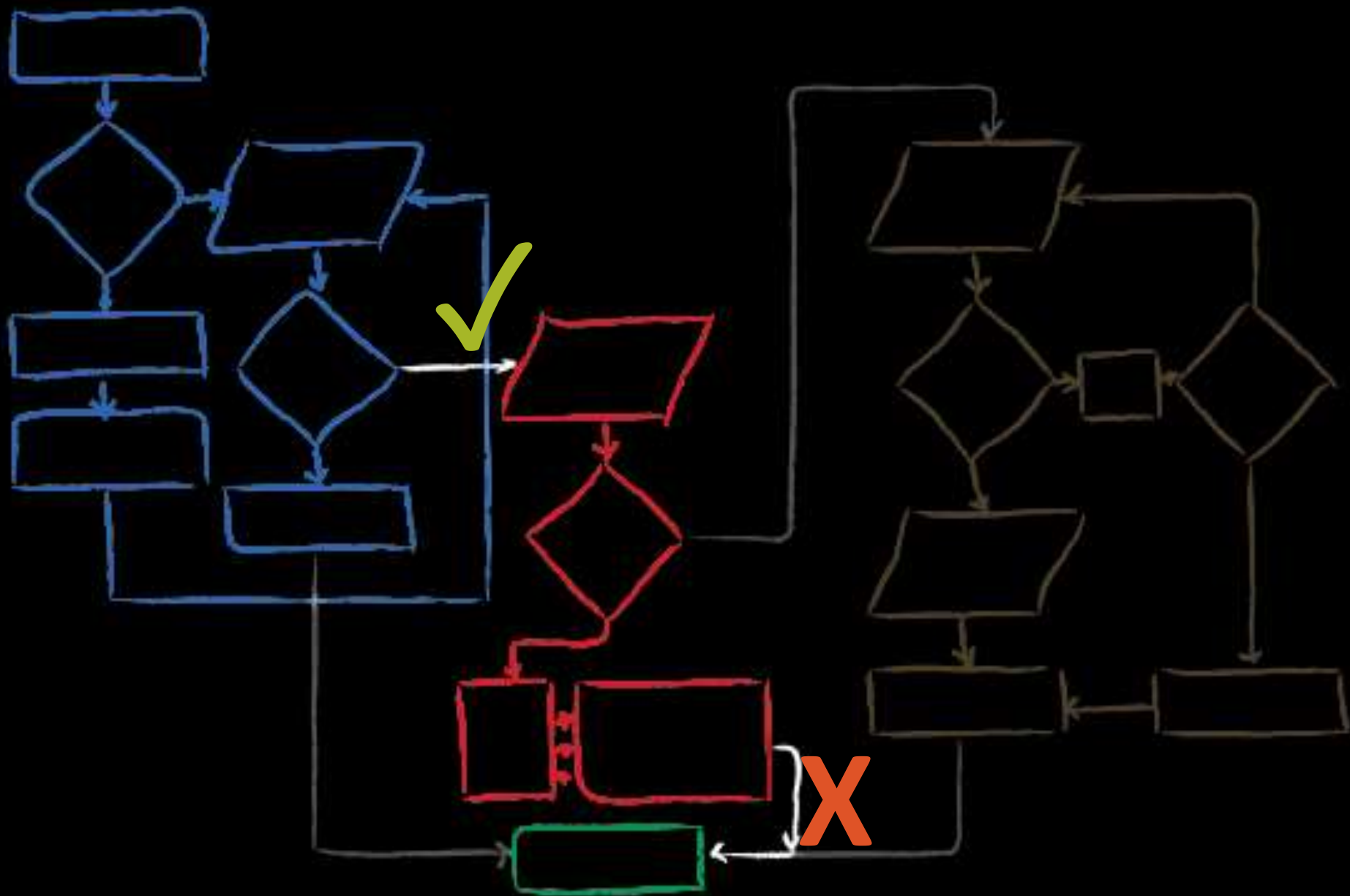
2. Divide & Conquer

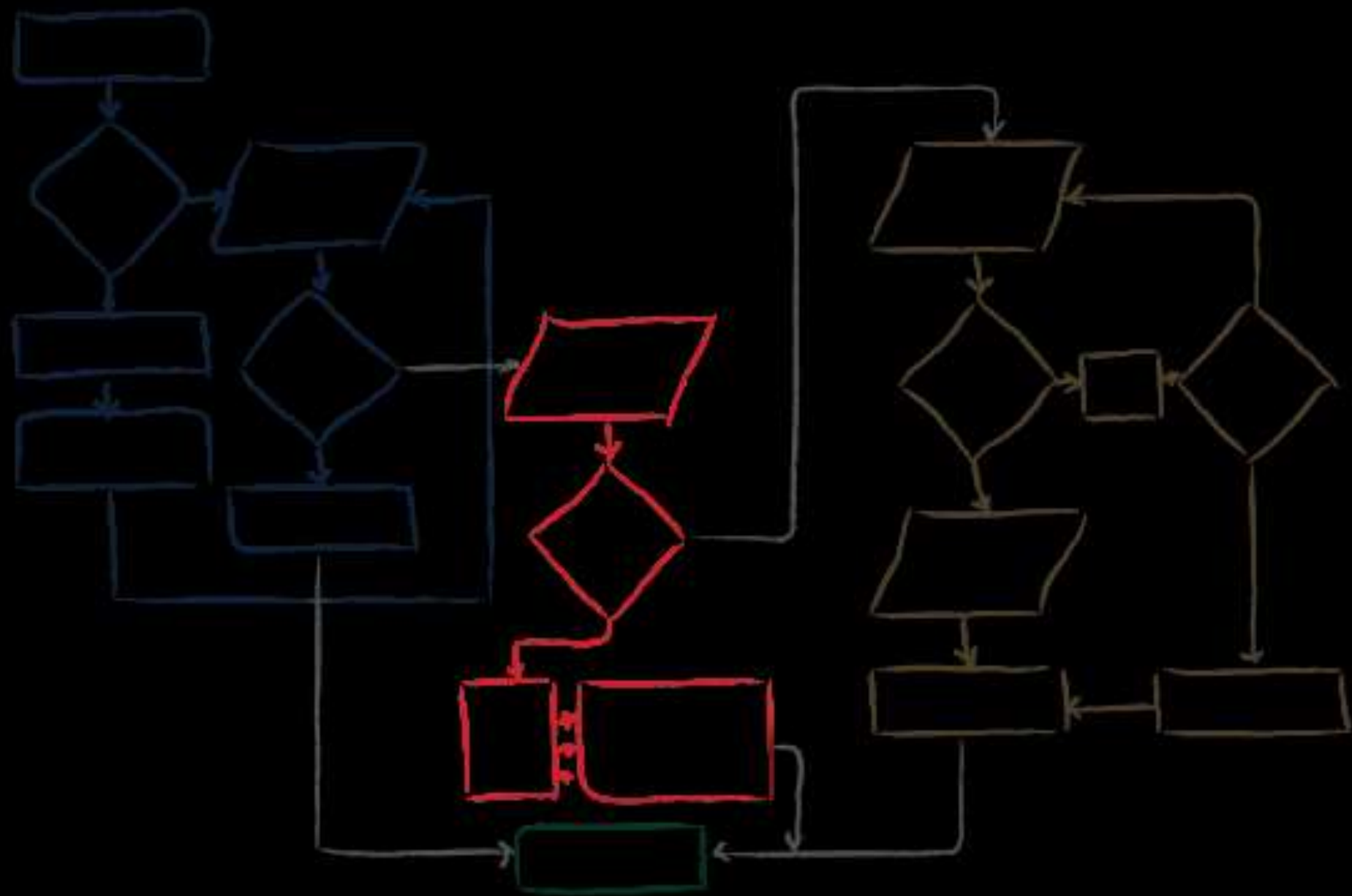
- Identify different code sections
- Set breakpoints at the boundaries
- Isolate issue to one particular area
- Focus efforts on that area











3. Use tools

- VarDumper
- Debug toolbars
- Console utility
- Profilers
- git bisect
- netcat
- curl
- strace
- etc.

VarDumper

Twig:

```
{{ dump(foo) }}  
{% dump foo %}
```

PHP:

```
dump($somevar);
```

dump()

in DefaultController.php line 15:

```
appDevDebugProjectContainer {#212 ▾  
  -parameters: array:523 [▶]  
  #parameterBag: null  
  #services: array:67 [ ...67]  
  #methodMap: array:226 [ ...226]  
  #aliases: array:16 [ ...16]  
  #scopes: array:1 [ ...1]  
  #scopeChildren: array:1 [ ...1]  
  #scopedServices: array:1 [ ...1]  
  #scopeStacks: []  
  #loading: []  
}
```

in DefaultController.php line 15:

```
Request {#7 ▾  
  +attributes: ParameterBag {#10  
    ▶}  
  +request: ParameterBag {#8 ▶}
```

Debug toolbars

The image shows a screenshot of a Drupal Themer Information debug toolbar. The toolbar is a dark grey overlay with white text. It displays the following information:

- Drupal Themer Information** (Title)
- Parents:** `phptemplate_node_submitted` < `node.tpl.php` < `page.tpl.php`
- Function called:** `theme_username()`
- Candidate function names:**
 - `minnelli_username` < `phptemplate_username` < `garland_username` < `theme_username`
- Duration:** 0.6 ms
- Function Arguments** (Section Header)
 - `... (Array, 1 element)`
 - `0 (Object) stdClass`
 - `nid (String, 2 characters) 25`

Console Utility

Symfony:	<code>bin/console</code>
Drupal:	<code>drush</code>
	<code>Drupal Console</code>
Magento:	<code>n98-magerun</code>
Laravel:	<code>artisan</code>

Performance Profiling

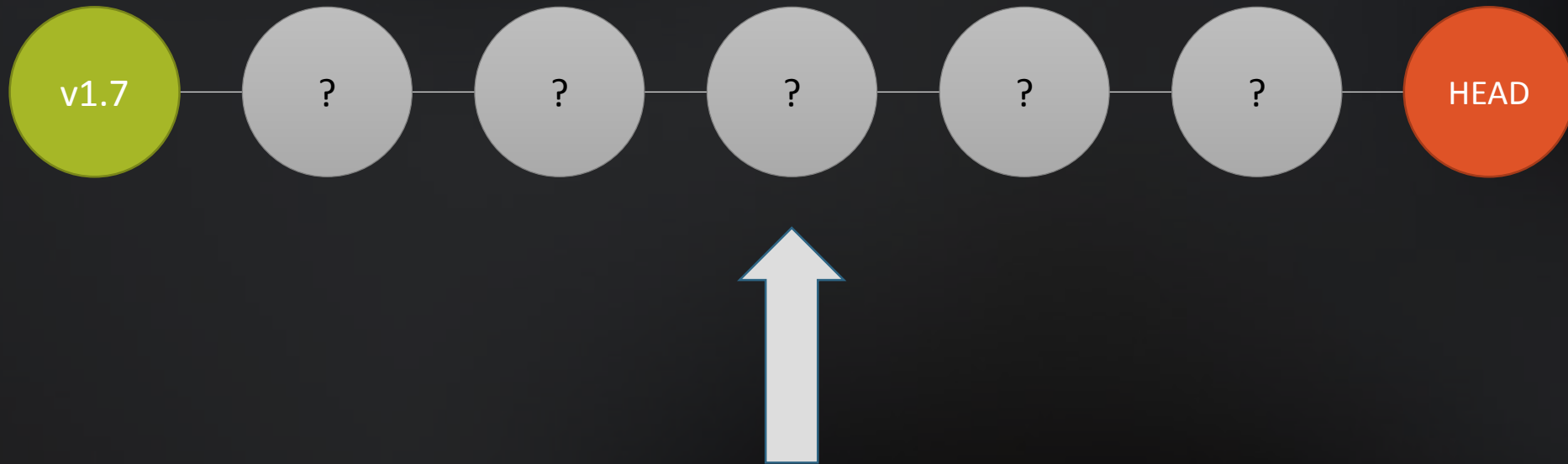
Identify slowness:

- Bottlenecks
- Resource hogs
- Inefficient code

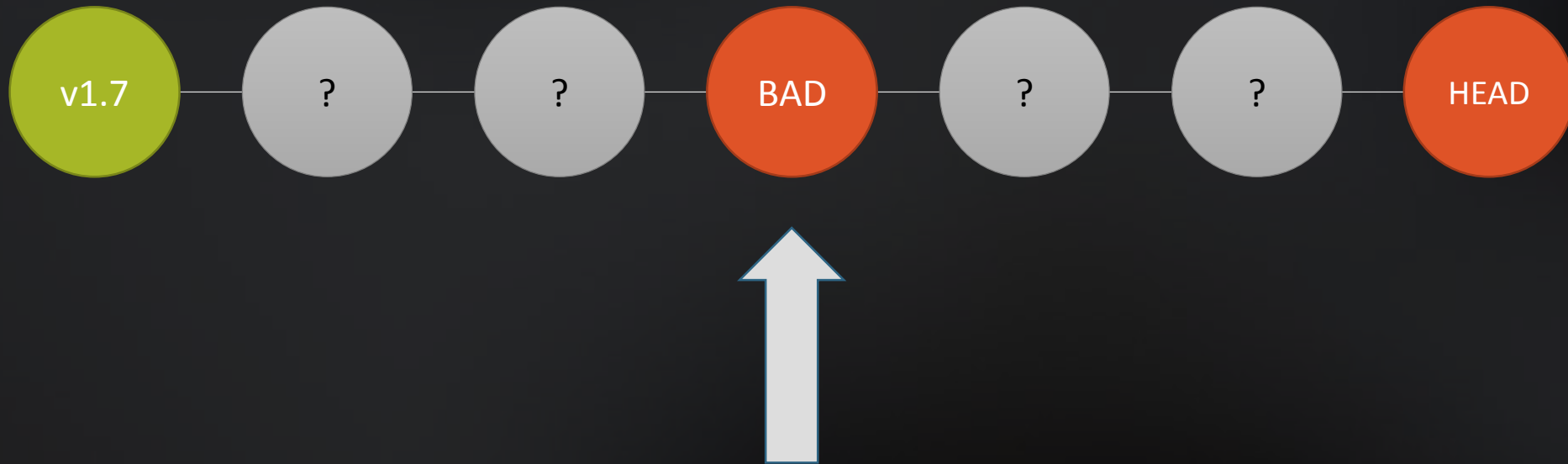
Tools:

- Blackfire (freemium)
- New Relic (freemium)
- xhprof (open-source)

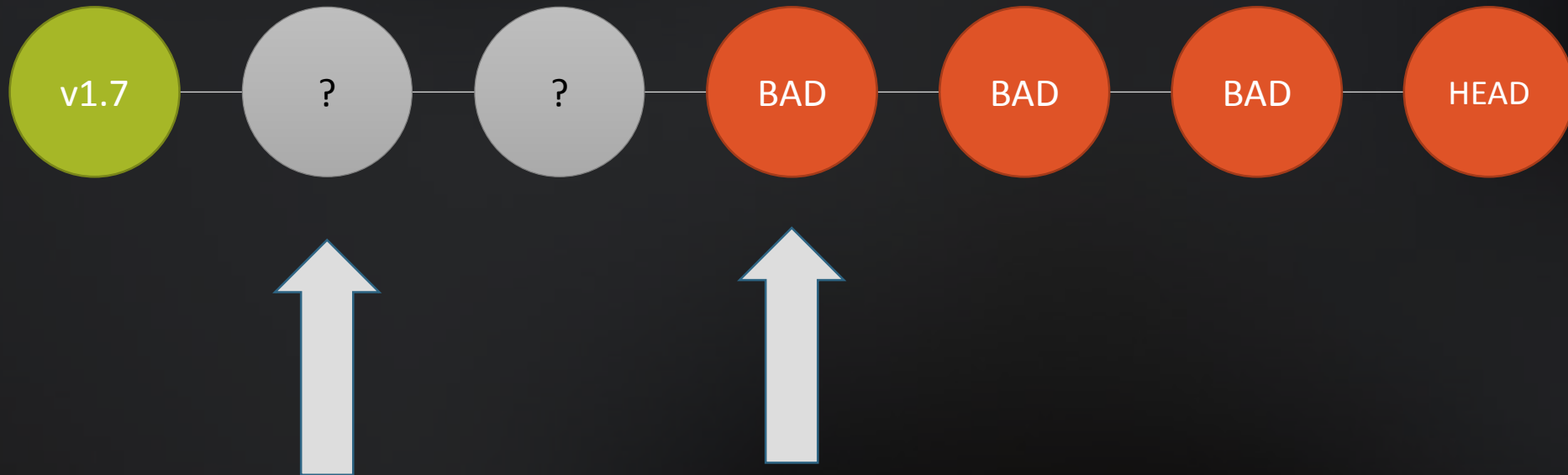
git bisect



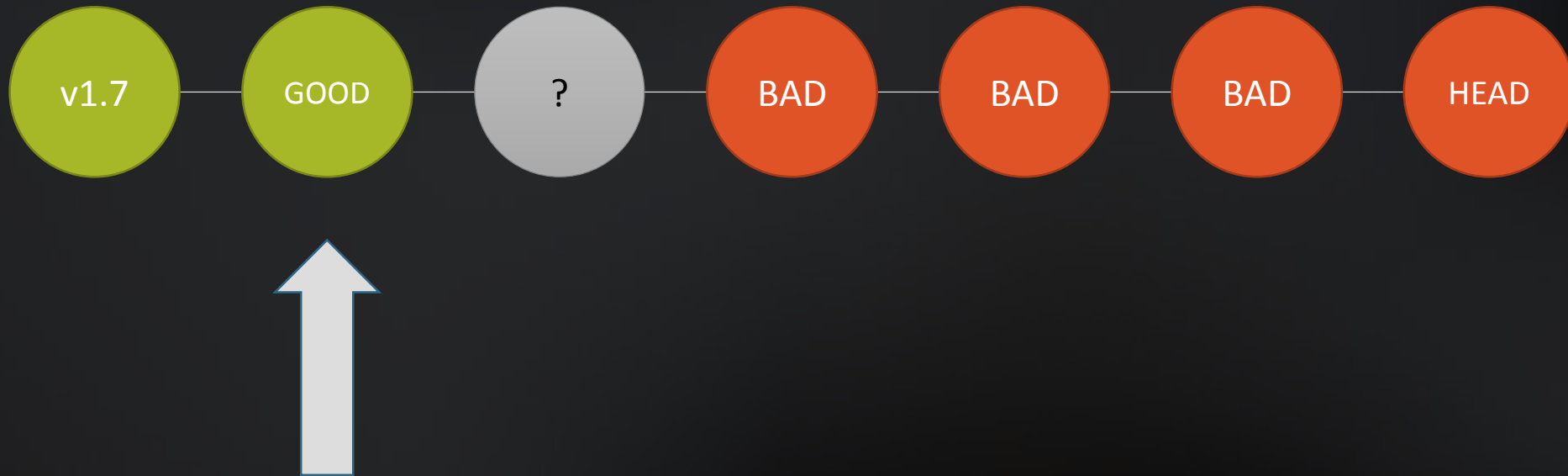
git bisect



git bisect



git bisect



git bisect



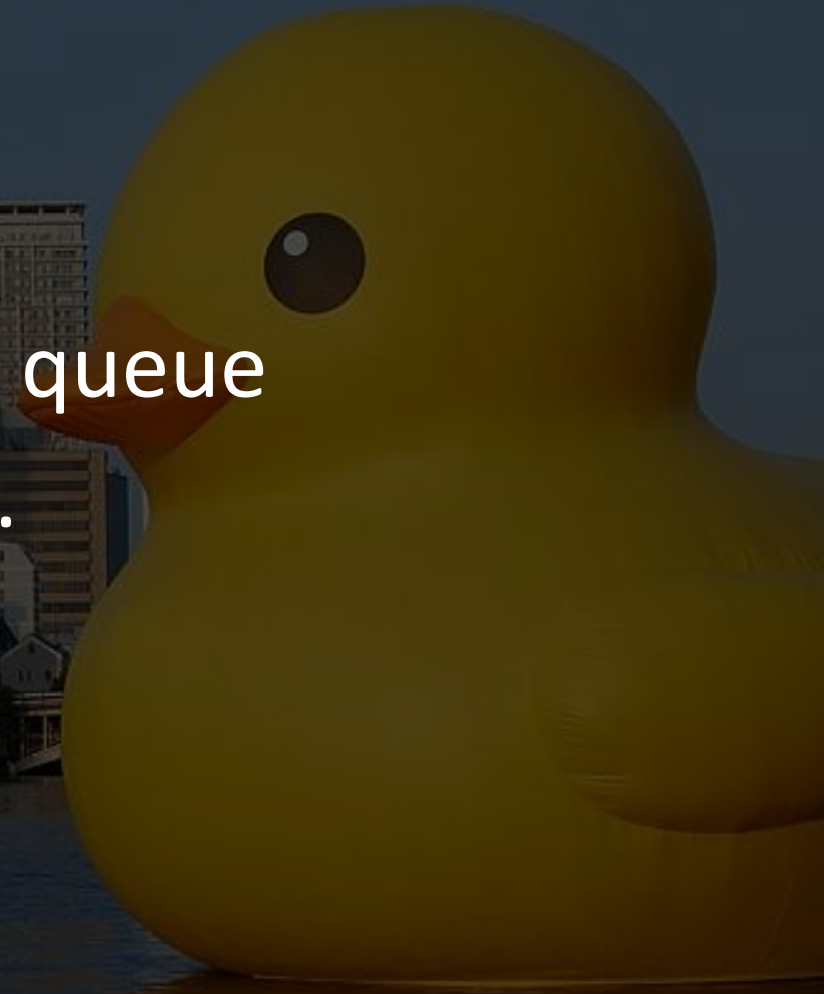
abcd123 is the first bad commit

netcat

```
[codell@athena ~]$ nc -vz localhost 80
Connection to localhost 80 port [tcp/http] succeeded!
[codell@athena ~]$ nc -vz google.com 22
nc: connect to google.com port 22 (tcp) failed: Connection timed out
nc: connect to google.com port 22 (tcp) failed: Network is unreachable
[codell@athena ~]$ █
```

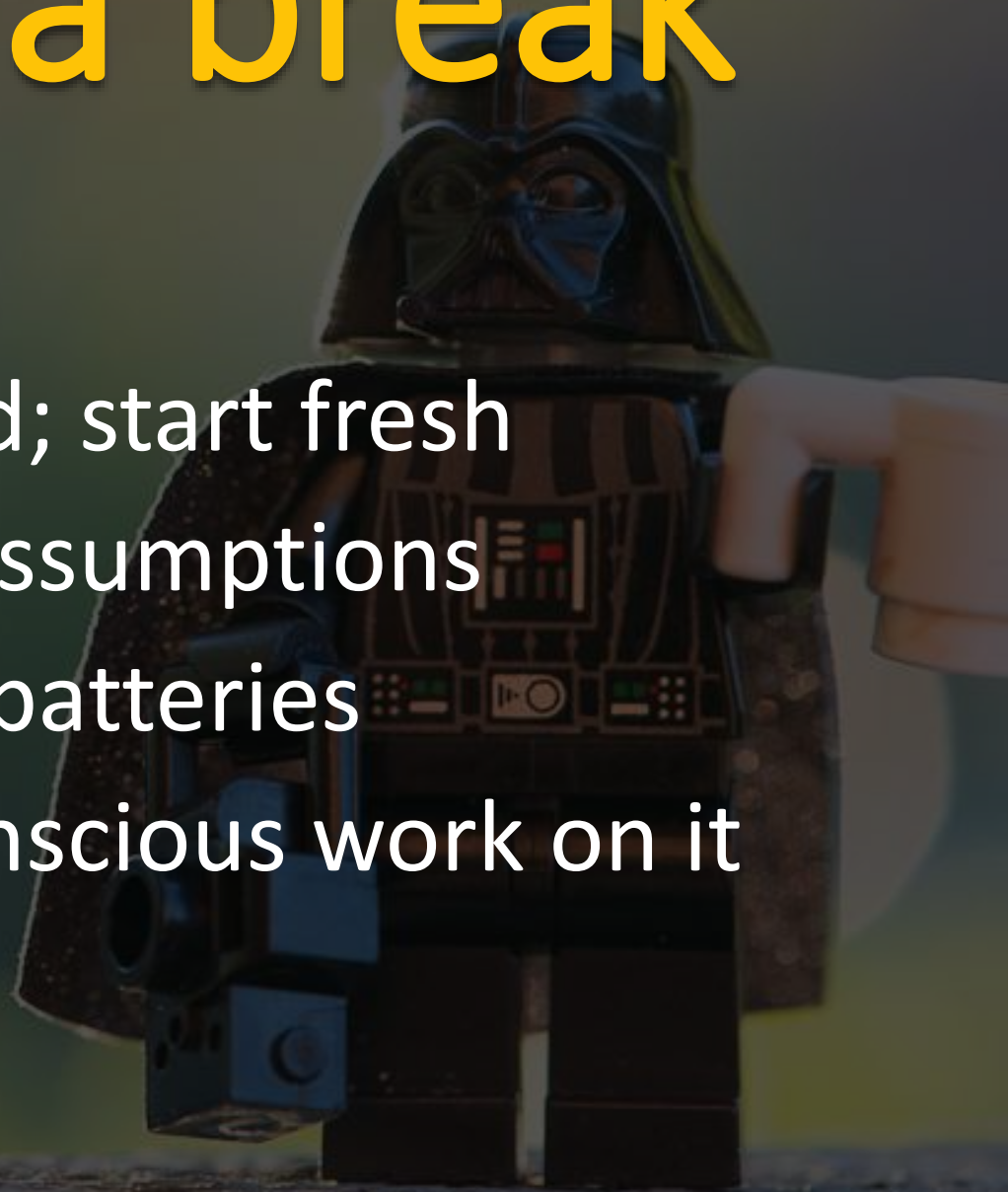
4. Get help

- RTFM / RTFD
- Project forums or issue queue
- StackOverflow, IRC, etc.
- Ask a colleague
 - Expert in that area
 - Senior developer
- Rubber ducking



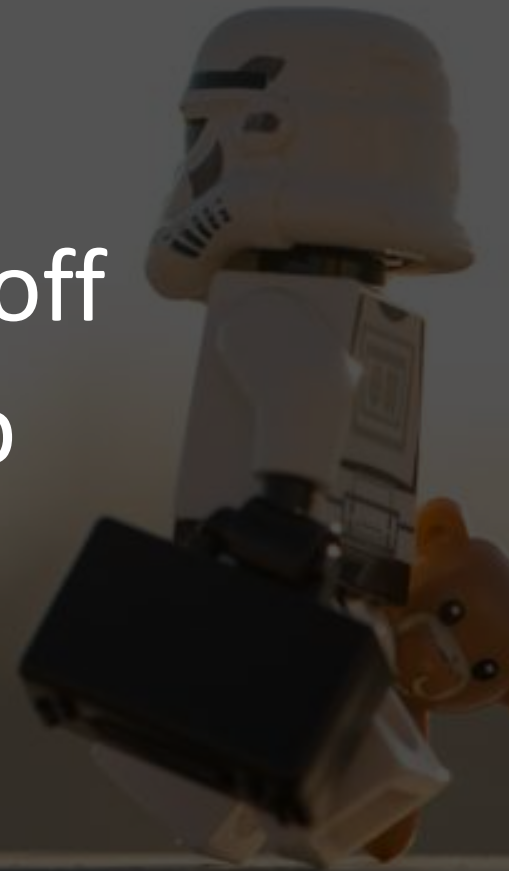
5. Take a break

- Clear your mind; start fresh
- Forget invalid assumptions
- Recharge your batteries
- Let your subconscious work on it



Four things to walk away with

1. Computers aren't random, and neither are bugs
2. Persistence will always pay off
3. Don't be afraid to dive deep
4. Don't make assumptions or take things for granted



Questions?



Learn More

- <https://web.duke.edu/cps001/notes/Debugging.pdf>
- <http://www.fiveminutegeekshow.com/20>
- <http://blog.codeunion.io/2014/09/03/teaching-novices-how-to-debug-code/>
- <https://www.jetbrains.com/phpstorm/help/debugging.html>
- http://symfony.com/doc/current/components/var_dumper/introduction.html
- <http://www.sitepoint.com/debugging-git-blame-bisect/>
- <http://unix.stackexchange.com/a/50099/80744>
- <http://codeception.com/docs/01-Introduction>
- <http://chadfowler.com/blog/2014/01/26/the-magic-of-strace/>
- <http://c2.com/cgi/wiki?RubberDucking>

Thanks!

Feedback?

- <https://joind.in/talk/f3c6c>
- @colinodell

