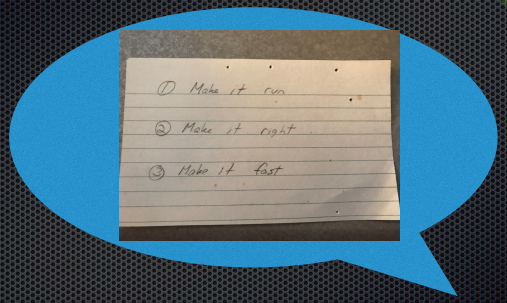
Avoid Troubleshooting Troubles: Effective Debugging Techniques to Help You Get Unstuck

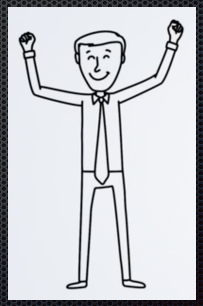
Paul Lefebvre
Xojo, Inc.



Kent Beck

Make it Work

- Get something to happen
- Proof of concept
- It compiles!
- It runs!
- It does what you expect!
- Once
- ~~Ship it!~~

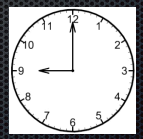


Make it Right

- Improve the code
- Error handling
- Edge cases
- Testing!

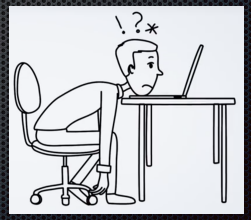
Make it Fast

- Not talking about this now
- Come to **Virtuous Code Optimization** session to learn more
- Friday at 9am



What if you get stuck?

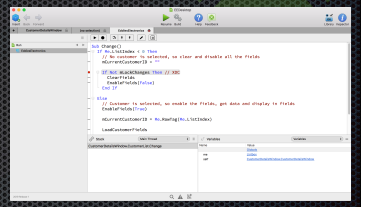
- Take a break, take a breath, take a walk
- Step away
- Get some sleep



What are expections?

- To know if something works, you have to know what you **want it to do**
- To know if something is fixed you have to know **what was wrong**
- Think about what you **expected to happen**
- Then read the code and see if that is **what is happening**
- Step through the code with the **Debugger**

Debugger



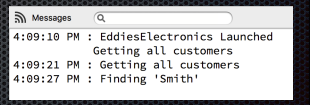
- Don't ignore the Debugger
- It's the first resort not the last resort
- Set breakpoints
- Use Break command
- Breakpoint in Computed Property
- Step through code to watch variable values change
- Step Over
- Step Out
- Verify expectations
- Getting a file error? Verify that the FolderItem actually refers to the file you think it does

Rubber Ducking

- Explaining the situation/problem to someone else
- This can literally be a “rubber duck” at your desk
- I have a “Darth Vader” stress ball for this purpose
- Explaining to another forces you to dig deeper into your understanding and can often reveal a solution that was previously overlooked

Logging

- The debugger is great, but not for everything
- Not as useful for
  - long-running tasks
  - shipping apps
  - graphics drawing
- Use a Log
  - System.DebugLog
  - View in IDE Messages panel

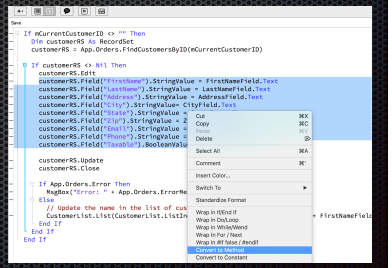


Simplify Code

- Remove “one-liners”
  - They are harder to debug
  - Use interim variables
  - Multiple things happening per line is more complex than one thing happening per line
- Split into multiple methods
- This is a great way to verify expectations

Isolate Code

- Long methods can be confusing
- Split things into smaller chunks of code and separate methods
- Once you’ve narrowed things down to a specific method you’ll be in a better position to find and fix a problem
- Code Editor **Convert to Method**



Consider Alternatives

- There are always multiple ways to solve a problem
- Try a totally different approach
- Don't stick with your original idea "just because"

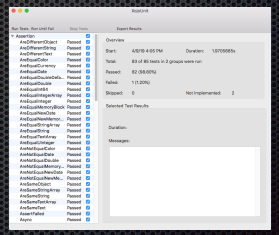
Check for errors

- Always check for errors
- Such as with Databases or File I/O
- Use Error property
- Catch Exceptions

```
Try
  xml.LoadXml(xmlFile)
Catch e As XmlException
  MsgBox("XML error: " + e.Message)
End Try
```

Unit Testing

- Repeatable tests that are run to verify code and any changes work as expected
- XojoUnit
  - Free and open-source
  - github.com/xojo/XojoUnit
- Or roll your own
- If you find a bug, fix it and add a test for it
- Test-Driven Development



Separate Project

- Create a separate project to test
- Isolating a problem can help you focus
- Easier to share and demonstrate with others

Version Control

- Free yourself with Version Control or Source Control
  - Git or Subversion
  - Use with Text project format (Xojo Project)
- You won't be afraid to make changes
- Don't be afraid to completely redesign something in order to fix it or make it better
  - You can always get back to the original version
- As my grandfather said:
  - "You can't break it twice, Paul."

Demo

Q & A

Paul Lefebvre

paul@xojo.com

Give us feedback on this session in the XDC app!