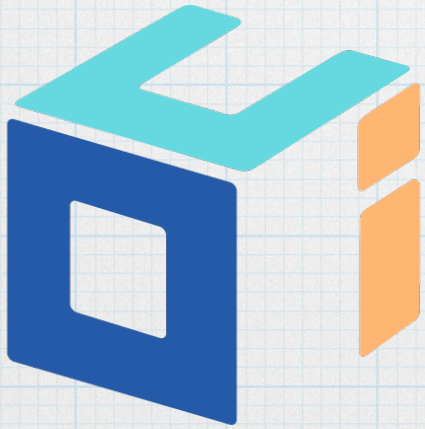
Experiences, Lessons, & Opinions from 38 years of Software Development

R. Mark Volkmann
Object Computing, Inc.
mark@objectcomputing.com



slides at https://github.com/mvolkmann/talks

OCI

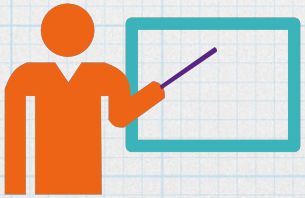
* Consulting and training company headquartered in St. Louis
* https://objectcomputing.com
* Started in 1994
* Strong focus on open source software
* Home of Grails framework
* Monthly SETT articles
  * https://objectcomputing.com/resources/publications/

38 Years?

* B.S. and M.S. in Computer Science
  * while having two kids

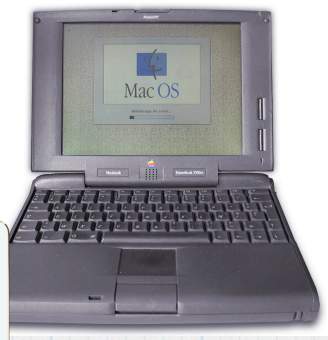
* Three employers
* Many languages including
  Fortran, Basic, PL/1, Mumps, Python, C, C++, Pascal, Java, Ruby, JavaScript, and dabbling in many others

* Too much XML, XML Schema, and XSLT
* Consulting for past 21 years
* Lots of reading, writing, teaching, and speaking



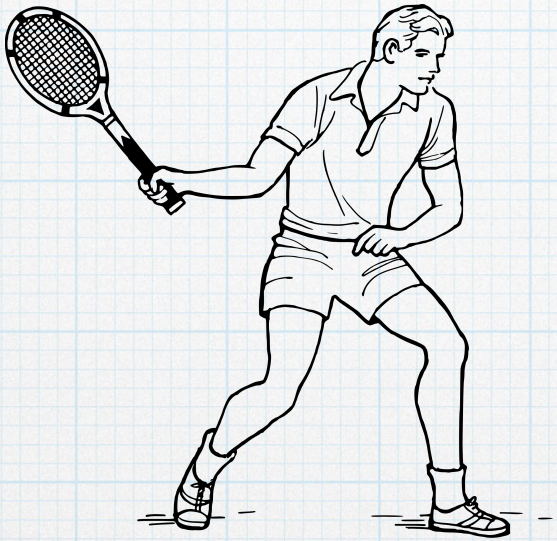
Computers I've Owned

* TI-59
* Atari 400 & 800
* Gateway PC
* Power Macintosh 7200
* PowerBook 5300
* Mac Mini
* iMac 21" & 27"
* series of MacBook Pros
* three iPads



First Applications

* Tennis Serve

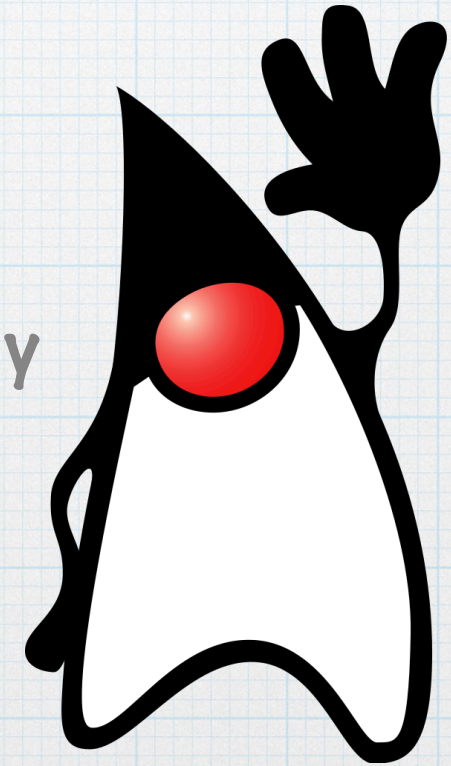


* Telephone Cable Resistance

Making a Bad Performance Review Good

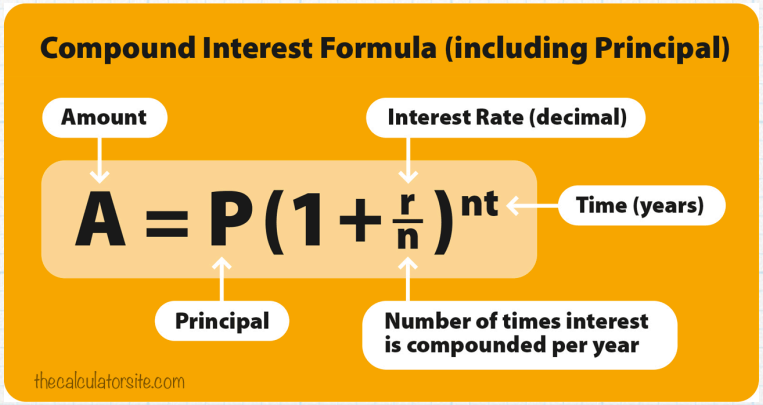
* Not focused enough on business

* Too interested in technology



Teaching

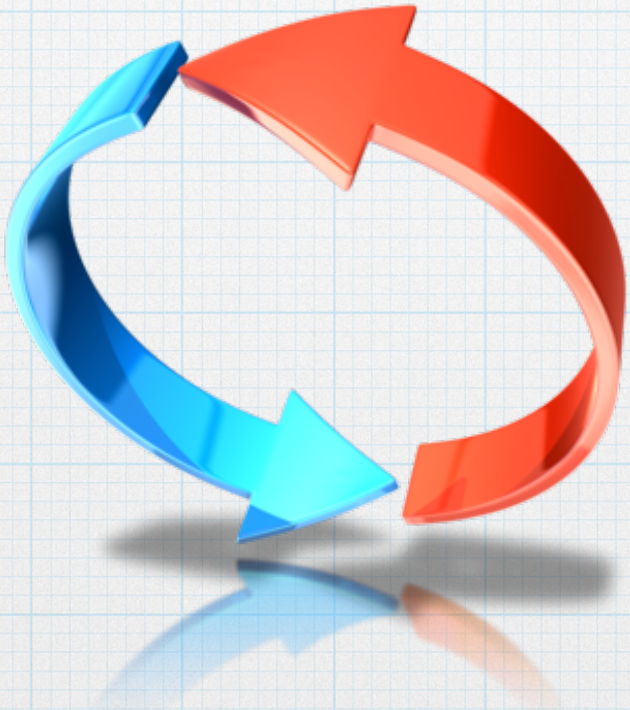
* Business Math
* Word & Excel
* QuickBasic
* C++



* many Java courses
* many XML courses
* Ruby
* HTML, CSS, JavaScript, Node, jQuery, Angular, React

Continuous Learning

* Surely you didn't think this would end after high school, bootcamp, or college!
* But how?



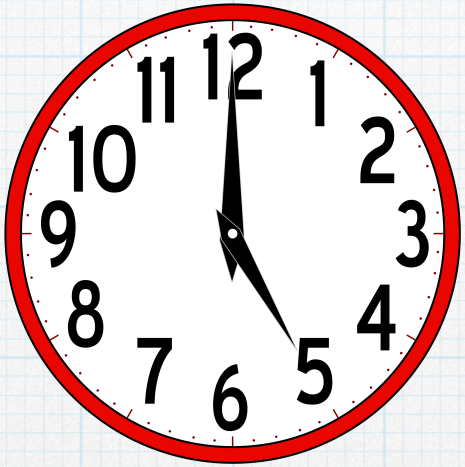
Learning Resources

* Books - paper or e
* Blog posts / websites
* Twitter
* Screencasts / Videos
  * YouTube and others
* User Groups
* Conferences
* Volunteer to give a talk
  * fear of embarrassment will cause you to dig in

Why Highlight?

* Highlight important parts of books and printed articles
* Forces thinking about what is important
* Forces re-reading which solidifies learning
* Makes it faster to refresh your memory in the future, only re-reading the highlighted parts

Finding Time to Learn



* Exercising
* Driving
* Kid activities
* Eating
* Waiting in lines, waiting rooms, ...

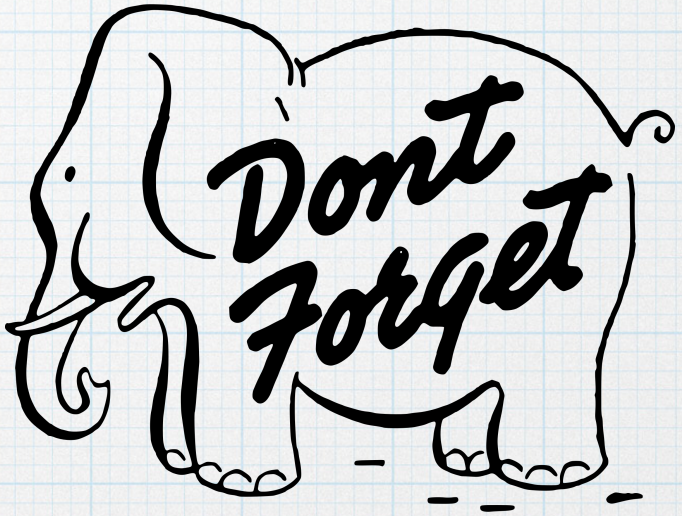
Always have something with you to read. You never know when spare minutes will appear.

Deciding What to Learn

* Interesting, but rarely used tech?
* Widely used tech?
* Does it have corporate backing?
  * Angular -> Google
  * React and Flow -> Facebook
  * TypeScript -> Microsoft
* Do you care?

Documenting What You Learn

* Keep notes by topic
  * in easy to edit files,
    like plain text or Markdown
  * provides a place to record new knowledge
  * becomes the basis for ...
    * blog posts / articles
    * slides for talks to give at
      user groups and conferences
    * books you will write



* If you don't document it, you'll forget it
* Just doing this will help you learn it more
  * you'll have to express it clearly

Why Share What You Learn?

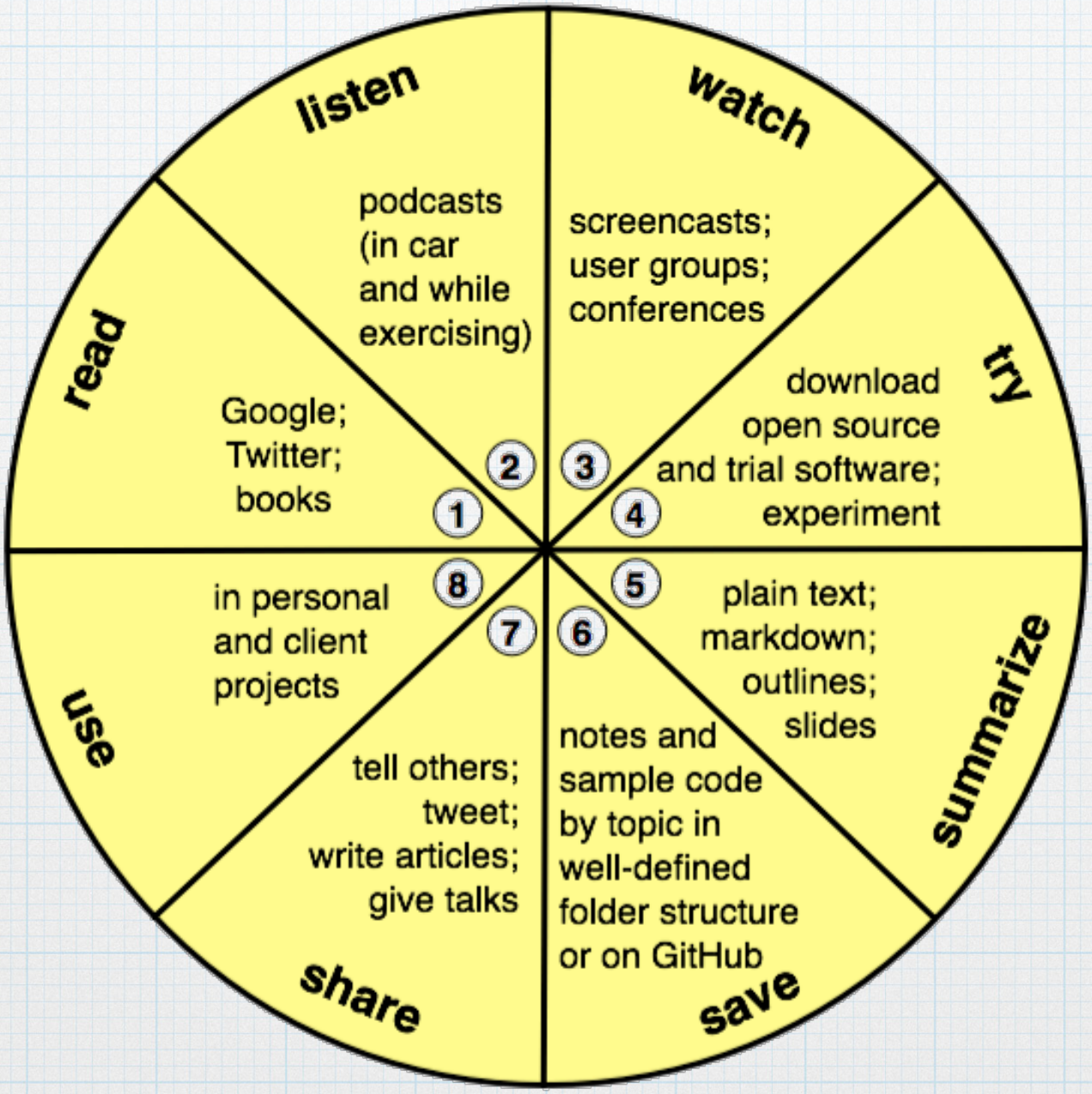
* To learn more deeply
* It's only fair
  * you've gained so much from what others have shared

Sharing What You Learn

* Create screencasts
* Write blog posts / articles
* Create web pages
* Publish sample code in GitHub
* Give user group talks
* Give conference talks
* Write books

Same as list of learning resources, but not you're creating the resources!

Wheel of Continuous Learning



Code Formatting

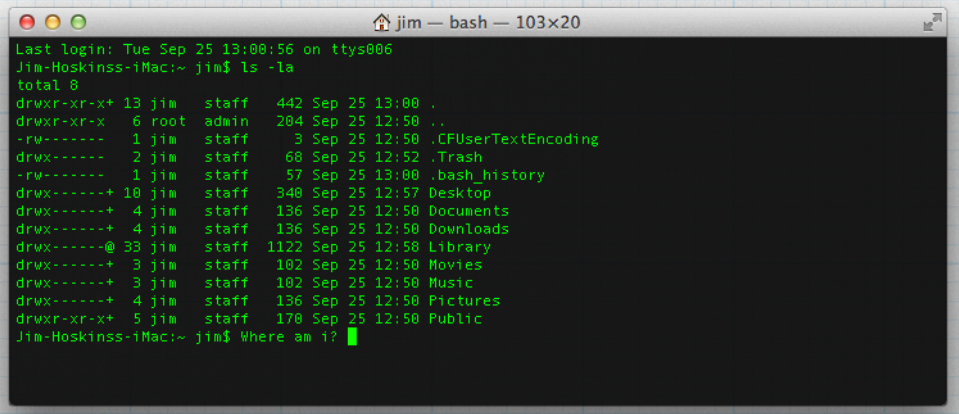
- Do we really have to talk about tabs vs. spaces and indentation?
- Yes! But why?
  - consistent code is less distracting to read
- Automate!
  - with tools like Prettier and ESLint (--fix)
  - each developer can have a different configuration that they apply to all code after creating a new branch
  - a prepush Git hook can format to project standard before every push

Editors / IDEs

* Modal editing
  * command mode versus insert mode
  * allows more convenient keyboard shortcuts
* Editing macros
  * record a sequence of keystrokes including editor commands
  * assign a name
  * replay any number of times
* Repeating actions
  * ex. replace word under cursor with a specific word
  * best if triggered with a single key (ex. "dot" key)

Command Line

* Master it
* fish shell
  * better autocomplete support with history recall and completion of commands, options, file paths, and git branches
  * better error messages
  * better script control flow syntax
  * more colorful
  * can configure with a web UI
* tmux - terminal multiplexer
  * http://java.ociweb.com/mark/tmux/



Version Control

* Game over, Git has won!
* Learn how to use from command-line

Automation

- Command-line aliases
  - cdgitroot, br, status, co, ci, nr, push, klp, ...
- Scripts
  - findjs, findscss, ...
- Linting - ESLint
- Type checking - Flow, TypeScript
- Code formatting - Prettier
- Testing - Jest, Enzyme
- Git hooks - Husky
  - ex. prepush

Some Benefits of Types
finding errors
documentation
refactoring
code completion

Mono or Polyglot?

- Are you really using the "best tool for the job"?
  - or are you just using that as a justification to use a language you like?
- Is it worth the mental shift?
  - between client-side and server-side languages
- Is it worth the effort to learn?
  - for the whole current team and future developers that will maintain it?

Formal Education

- Is it important?
- Self-taught vs. Bootcamp vs. University
- Is it worth knowing topics outside of CS well?
  - math, statistics, physics, ...
- Learning how to learn
  - I use very little of what I was taught in college
  - but I did learn about data structures and how compilers work
  - and I learned "how to learn" which is important for continuous learning

Getting Along With Others

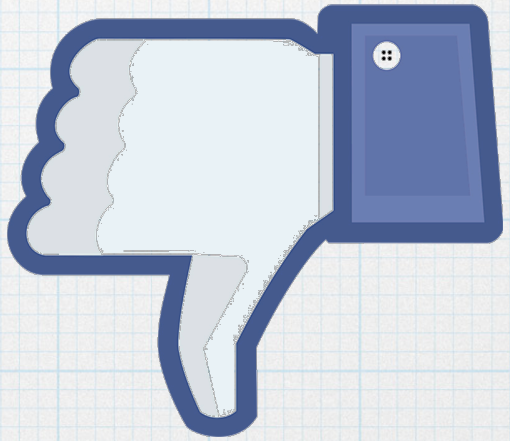
* Logic over emotion
* Manner in which you share your opinions
  * questions sometimes better than statements
  * "That will never work because ..."
    vs.
    "How would that handle ..."
* Cursing is not cool
  * can make others discount your opinions before they even consider them
  * don't be too lazy to choose better words

Being a Consultant

- Lots of variety!
- Get to recommend approaches and describe their pros and cons
  - don't get to decide
- Expected to be a good mentor
  - Do others frequently ask you for help or your opinion?
- Expected to find solutions that you didn't already know
  - "I'll get back to you on that", not "I don't know"
- Expected to quickly find bugs
  - in code written by other

Choosing Talks to Attend

* Pick talks that will
  * teach you something new
  * inspire you to learn more on your own
* Pick talks on tech you think you don't like
  * so you can better elaborate why you don't like it
  * document this so you don't forget



Wrapup

* Thanks so much for listening to my ramblings
* AMA
* Sit with me at lunch ... please!
* Email me at mark@objectcomputing.com
* Checkout my resources at github.com/mvolkmann
* Share your opinions with me
* Run with me sometime