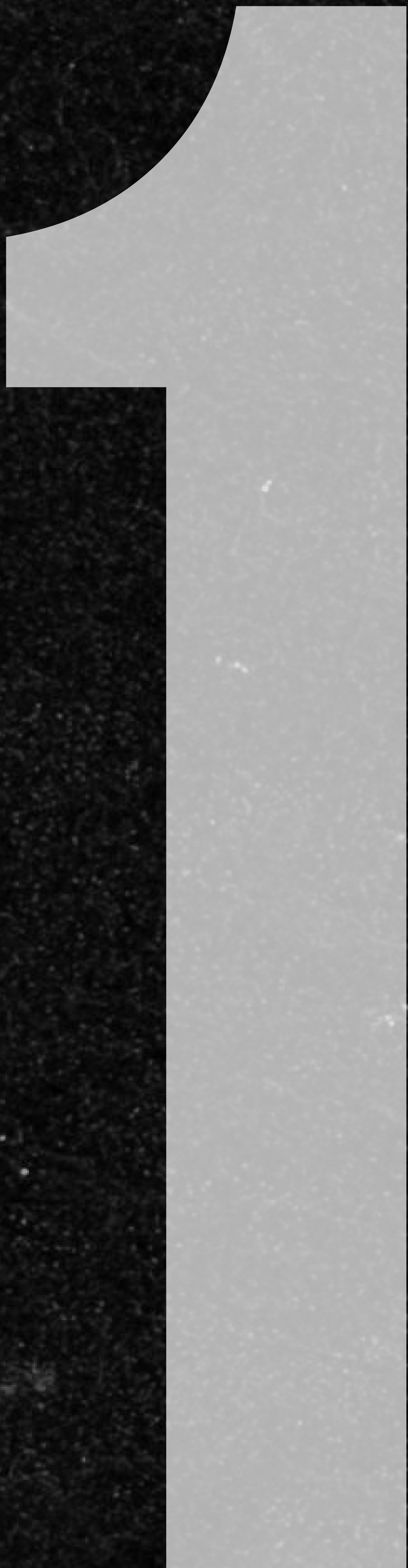


AVOID THESE CODING MISTAKES

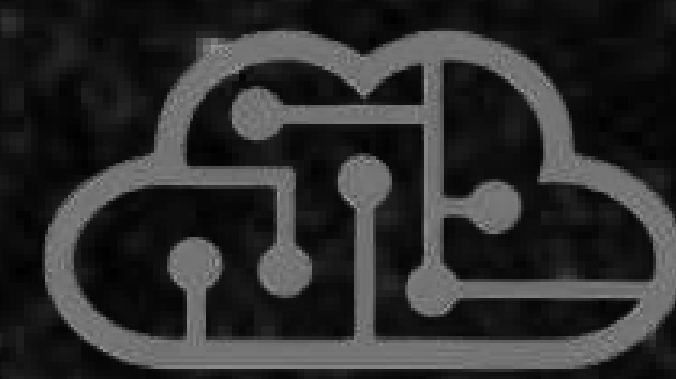





NO PROPER VARIABLE NAMES



- Small variable names work when you're writing a 10-20 line program or snippet, but not when working on projects.
- Improper variables are a menace to readability and productivity.
- The basic rule of naming your variables is that they should be self explanatory.

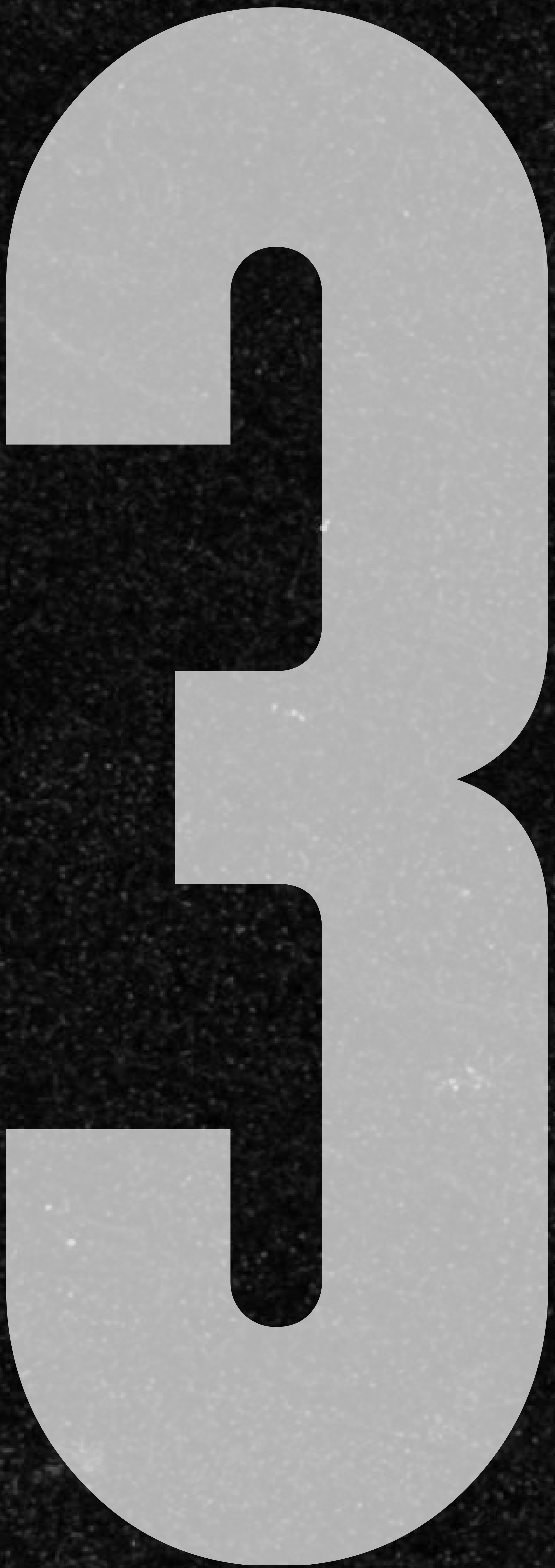


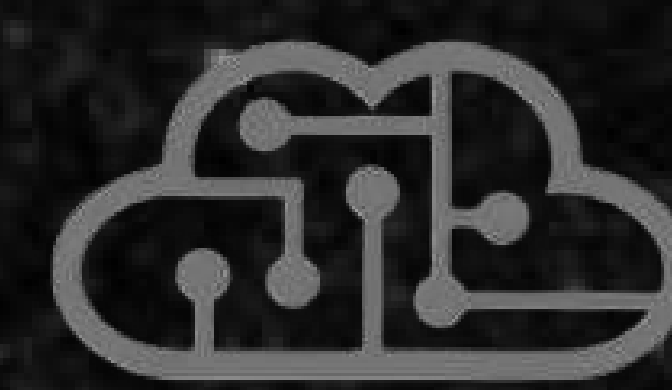
NOT DOCUMENTING THE CODE

- 
- We should always try to write properly documented code.
 - On the first overview, the reader should get a gist of what's happening.
 - This may involve proper code refactoring, using uniform syntax and proper variable names among other things.
 - Add comments wherever necessary.
 - Don't over comment.



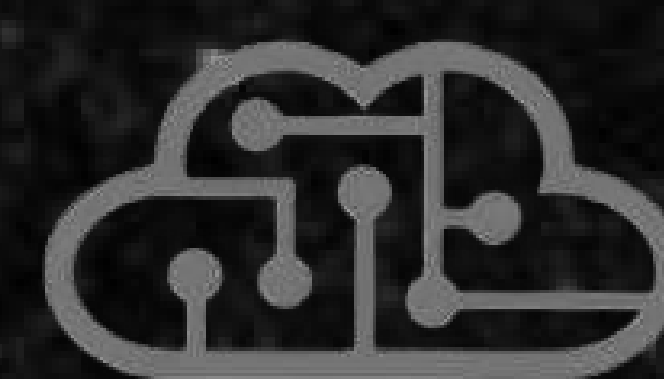
INCONSISTENT FORMATTING.

- 
- A large, light gray number '3' is positioned on the left side of the slide, serving as a visual separator for the third point in the list.
- Inconsistent formatting can greatly affect readability and productivity.
 - Don't change your style midway through the code.
 - Nothing puts people off more than a poorly formatted code.



NO ERROR HANDLING.

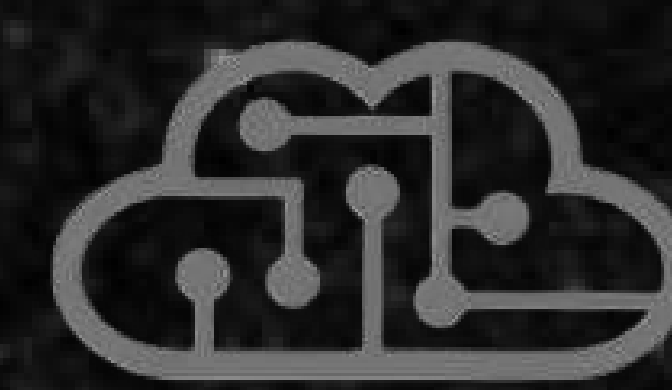
- Error handling can vary greatly depending upon the language you're working on.
- But a general rule of thumb is to be consistent with implementing null checks before accessing data.
- Therefore whenever doing a data related task, it's always advisable to enclose your code into try-catch blocks and handle the exceptions as required.



NOT TAKING ADVANTAGE OF DEBUGGING TOOLS

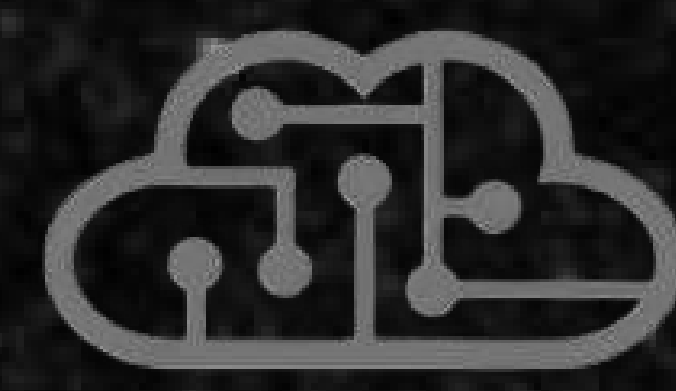
5

- If you're working in a statically typed language like Java, C# or ActionScript3, you should be using the debugger.
- These languages give you really detailed errors, which combined with a debugger, can help you track down the bugs in your code easily.



NOT BACKING UP YOUR WORK

- The phrase “I just lost [X] hours work” should not be in a developer’s vocabulary! There are so many good tools for automatic back-up and version control now, that there’s really no excuse to lose anything, even if you have a major computer malfunction, fire, burglary or other minor disaster.



NOT VALIDATE USER INPUT

- It is absolutely crucial to make sure you check for errors every time you read in a value from user input. If it's invalid, output an appropriate message and read in a new value.
- This will save your program from breaking when one single bad input is received.