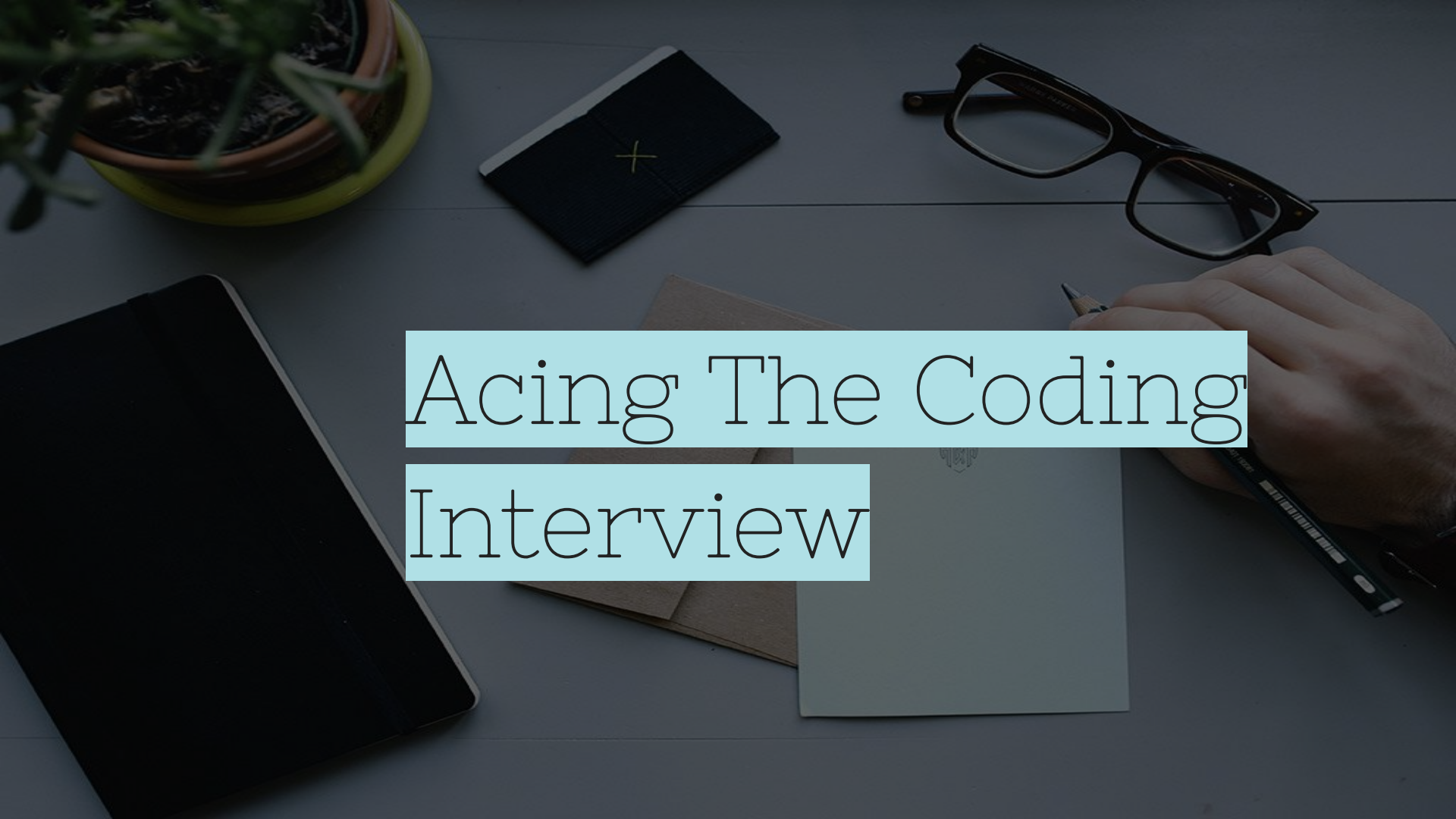
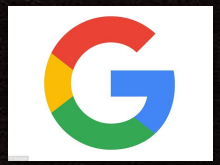
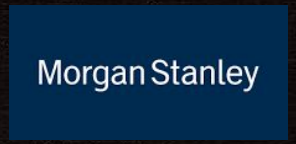

Acing The Coding
Interview

Background



Rumors/forum posts from other people on:

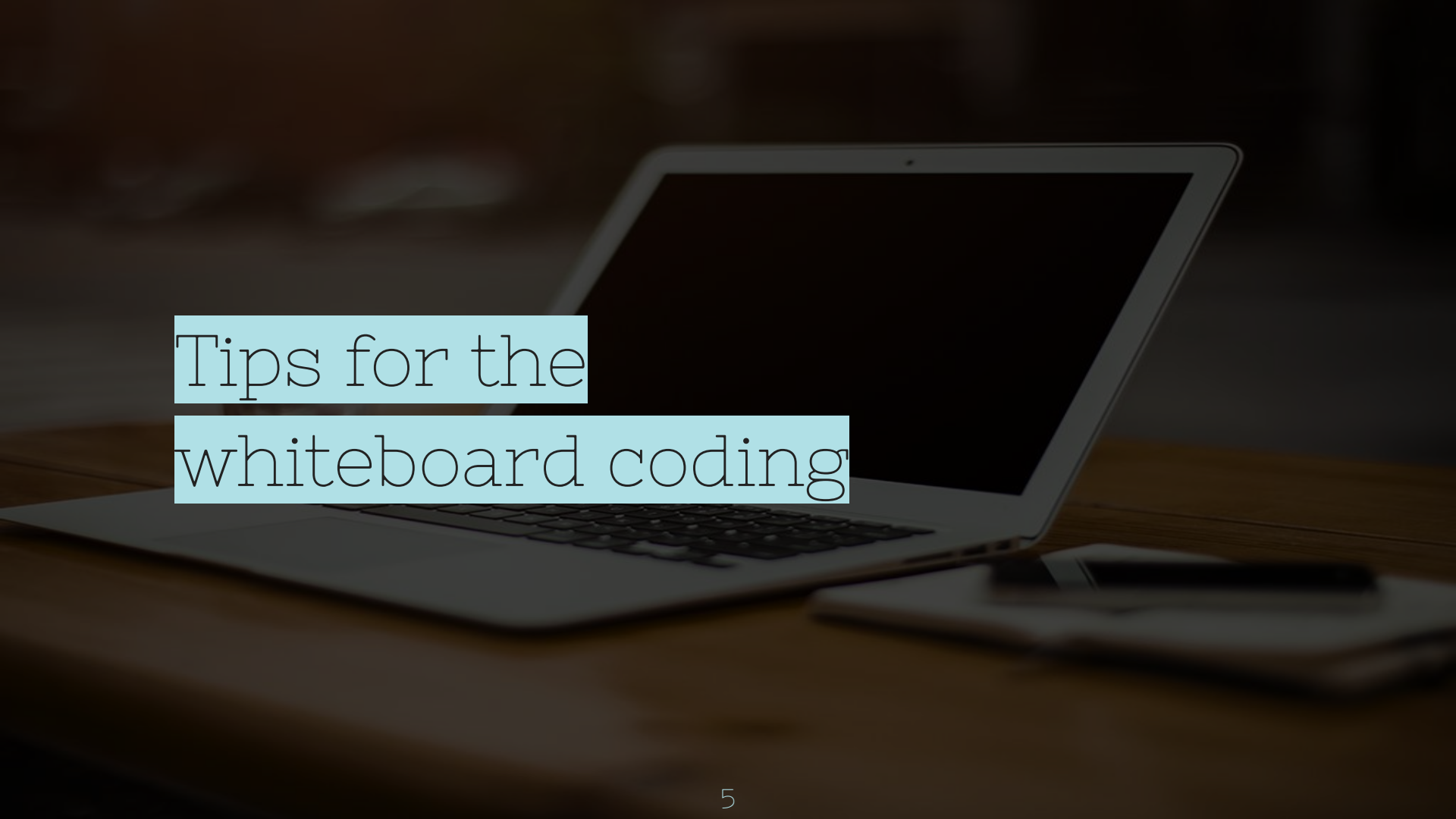

Book: Cracking the Coding Interview

Resume / CV

- 10 second rule
- Highlight the important experiences
- Maximum 2 page (1 for most of you)
- Profile picture?
- Everything it says SHOULD be true

Interview types

- Hackerrank / Whiteboard coding / Google docs (main focus)
- Algorithms / Data Structures, but why?
- “Homework”
- Code review together, Find a bug in a codebase
- Programming language quiz (?)
- HR programming round - definitely the best



Tips for the whiteboard coding

Technical preparation

- Algorithms: Big-O analysis, sort, sash, large amount of data
- Sorting: at least know one n*log(n), efficiency meaning
- Trees: construction, traversal, manipulate (BFS, DFS, Pre-, In-, Postorder) + Heaps
- Graphs: representation, cycle & connectivity detection
- Recursion + dynamic algorithms
- Operating Systems: threads, concurrency, lock, mutex
- Mathematics: Discrete (combinatorics), probability

Communication

Verbalise your thoughts!

- Make sure you understand the question
- Ask clarifying questions! - underspecified on purpose
- Try to think out loud, check for corner cases

Listen to your interviewer!

- Are they providing a hint?
- They want you to succeed! Are they?
- Try to view it as a “discussion” not as an “interview”

Presentation

Summarize the solution(s)!

- Draw the conclusion from the discussion
- There is no one true answer
- Present all possible solutions and trade-offs
- Make sure the interviewer understands your intents
- Draw pictures/graphs to visualize/show examples
- Maybe write pseudo code to make it more clear

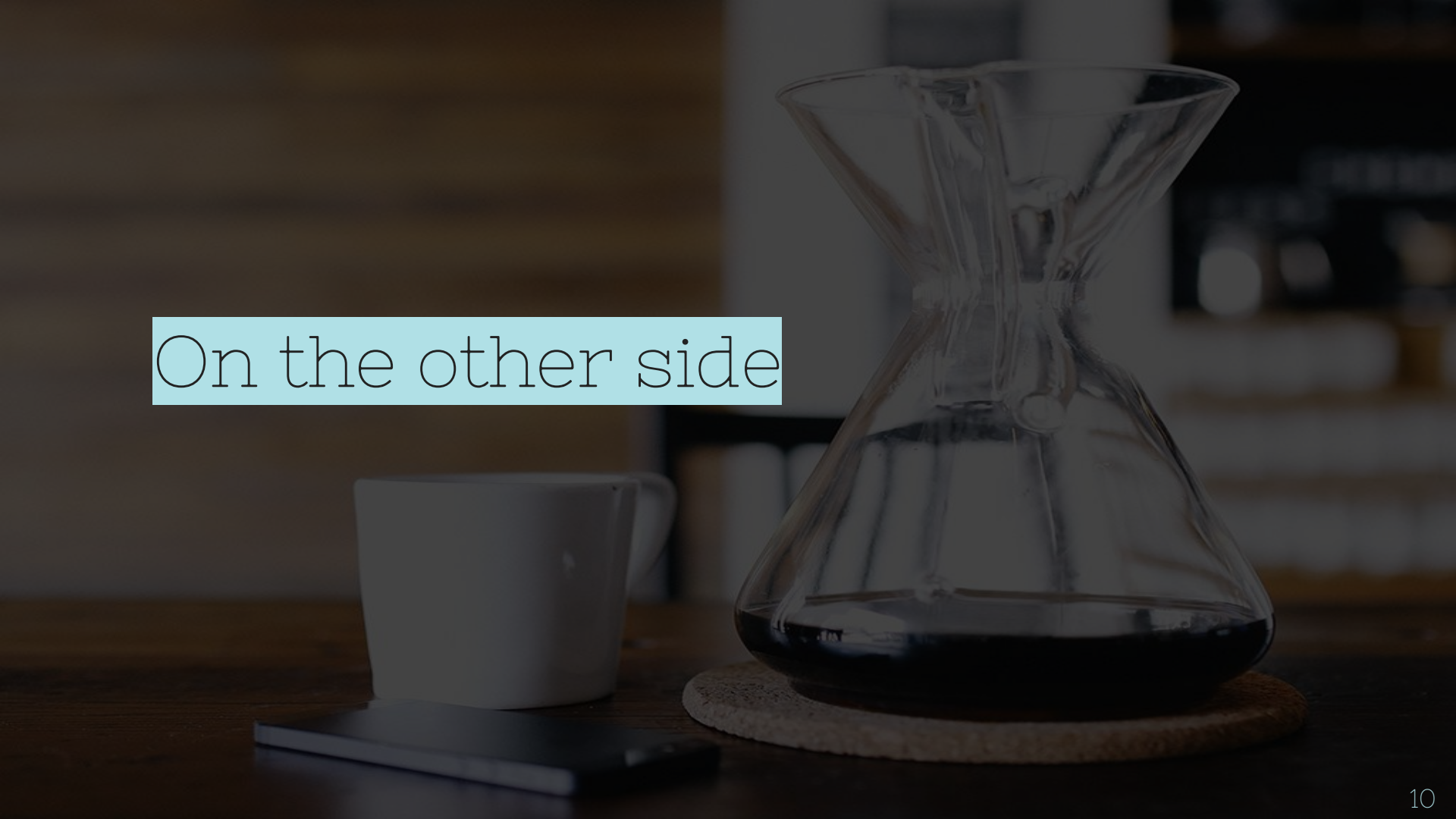
DO NOT WRITE ANY CODE UNTIL NOT ASKED

The CODING part

You have already done the most important part, forged and presented your proposed solution to the problem!

Now you just have to implement it - the easy part.

- Make sure to have a deep understanding of at least ONE coding language
- Algorithms/data structure knowledge is important
- Coding routine/experience is more important
- Check it line-by-line on some example input you have already created!


On the other side

Evaluation

- How did the candidate analyze the problem?
- Did she cover the special cases?
- Does she have a strong foundations in CS?
- Produced working code? Tested it?
- Is it maintainable?
- How he presented/explained the solutions?
- **Would you like to work with the candidate?**

Other methods

- Win a Code Jam competition
- Commit for a project/research
- LogMeIn case (Xively)