Whiteboarding

Introduction

Goals

- Overview of whiteboard interviews
- Process for answering whiteboard challenges
- Live demo
- Practice

Whiteboarding Interview

- An interview style that poses a coding challenge …
- Which you do, live, at a whiteboard
  - or, sometimes, on paper or a computer

Why Do They Do This?

They want to assess

- Your understanding of algorithms
- Your problem-solving techniques
- How you communicate your thought process
- How you work under pressure

Process

Listen Carefully

> *"Write a function that is given a list of numbers.*
>
> *Find all the even numbers in the list and return the average of them."*

Repeat it Back

> *"Ok, so you want me to write a function that's called with a list of numbers, and returns average of the even numbers?"*

Ask Clarifying Questions

- Do I need to handle other kinds of things in list?
  - For example, if a string were in the list?
- Will these all be integers?
- Do I just skip over odd numbers?
- By "average," do you mean the mean? Median? Mode?
- Do I print the result or return it?
- Am I allowed/not allowed to use certain built-in methods?

Why?

- To buy more time.
- To understand the challenge details
- So you write bug-free code

Write Down the Requirements

- Make a short, bulleted list of requirements on whiteboard
  - So you can't forget any details
  - Gives you a moment to think with less pressure
- For example:
  - function given integers
  - just skip odd numbers
  - get mean of even numbers
  - return mean

Write Down a Test Case

```
[1, 2, 4, 5] => 6 / 2 => 3
```

Any other test case you'd want?

Perhaps one with non-integer average

```
[1, 2, 4, 8] = 14 / 3 => 4.6666
```

Stop and Think

Don't just start writing code!

Think about your strategy

> *"I'll loop over the list, skipping odds and non-numbers. I'll keep the sum of the evens, and the number of them. Once I finish looping, I can divide the sum by the count."*

Pseudo-Code

This can keep you from getting lost in the weeds

```
for number in list
    if not even, skip
    add number to sum
    increase count by 1
return sum divided by count
```

Code

- Start at top-left of the board
  - You want space to fit code!
- Write neatly and evenly
  - In Python, you may find it helpful to show indentation with lines

```
function avgEvens(nums) {
    let sum = 0;
    let count = 0;

    for (let num of nums) {
        if (num % 2 === 0)  continue;

        sum += num;
        count += 1;
    }

    return sum / sum;
}
```

Test Your Code

```
function avgEvens(nums) {
    let sum = 0;
    let count = 0;

    for (let num of nums) {
        if (num % 2 === 0)  continue;

        sum += num;
        count += 1;
    }

    return sum / sum;
}
```

- Go slowly. Be the computer.
- Keep track of vars (use a table)
- We're skipping *even* numbers!
- Dividing ***sum*** by ***sum***, not ***count***

```
function avgEvens(nums) {
    let sum = 0;
    let count = 0;

    for (let num of nums) {
        if (num % 2 !== 0)  continue;

        sum += num;
        count += 1;
    }

    return sum / count;
}
```

nums = [1, 2, 4, 8]

number	sum	count
1	0	0
2	2	1
4	6	2
8	14	3

return 14 / 3

Things to Think About

Whiteboarding Is A New Skill

- It's not the same as programming
- The first few times, your brain will fall out
- Like any skill, it takes time — practice!

Partial Credit

- It's not pass/fail
- Do what you can,
  - even if it's only pseudocode
  - even if it's a simpler case
  - even if it's just 1 part of the problem
- They want to see how you think
- They want to see how you handle pressure
- Sometimes, the questions are *really hard*
- They typically don't want you to solve it with a built-in function
  - eg, for "find max number," you can't use `Math.max()`
  - You can get partial credit/bonus point by knowing ***Math.max()*** exists

Don't Go Radio Silent

- It's fine (good, even!) to stop and think
  - Don't go entirely silent for too long — let them know where you are
- Use the whiteboard for scratch space
  - Helps keep you organized
  - Helps them see where you are

Hints

- It's fine to ask for a hint
- Some questions are designed so that's expected
- If you know part of the answer, say that before asking for help

Good Variable Names

- Think for a second before writing down
  - You want something short but helpful
- Good rules of thumb:
  - For *indexes of list*: ***i***, ***j***, ***k***
  - For *items in list*: ***a***, ***b***, ***c*** (or ***x***, ***y***, ***z***)
  - Use mnemonics: ***n*** for number, ***s*** for string, etc

Test, Don't Hand-Wave

- Some parts are hairy and you might feel shaky
- It's easy to try to "hand-wave" past them
  - "And now I recurse and find the longest string"
- Resist that temptation
  - The parts you're less sure of need the slowest testing
  - Be the computer

It's Not an API Quiz

- Try to remember the very most common operations
  - eg, to add to an array, it's ***myArray.push()***
- But whiteboarding isn't an API pop quiz
  - It's ok to ask what a method is called
  - It's ok to use a best-guess name (***mySet.addItem()***)
- They want to test your thinking, not memorization of APIs!

Take Your Time

- Interviewers will not be checking watches
- They want you to think deeply
- Don't let nerves speed up your speech

Remember

- You have a useful, new skill
- They're hungry for people they can hire – they want you to succeed!
- Think of them as a "pair programming partner", not a "test proctor"
- Think first, go slow, code out loud, test your work