How to Get Unstuck

Getting stuck in the middle of the technical interview/technical admissions assessment can be tricky. Here are some tips on how to get unstuck.

Don't give up

- Keep trying different strategies.
- Stay open to creative ideas.
- Try not to get frustrated.

Decompose the problem

- Break the problem into simpler parts.
- Analyze the parts of the problem.
- Concentrate on the parts of the problem you understand and that can be solved.
- Solve each piece by itself.

Rubber-ducking

Verbalize the problem to yourself and others.

Describe:

- what the problem is

- what you have done
- where you are stuck

Paraphrase:

- Re-describe the problem.
- Think of simpler ways to describe the problem

This is sometimes called "rubber-ducking" because there is an anecdote about an engineer who would keep a rubber duck (like what little kids use in the bathtub) at his desk. When he got stuck, he would talk it over with his duck.

Look for assumptions

- Look for hidden assumptions or information you have forgotten to use.
- After reading each phrase or sentence of the problem statement, ask yourself if any assumptions can be inferred from that phrase.
- Check again to make sure you are solving the right problem.
- Double-check all of your values, assumptions, and approaches. Make sure you haven't missed anything and that you are looking for the correct solution.

Re-evaluate your strategy

Try using a different strategy. There is usually more than one way to solve a problem, and you may find a method that you haven't considered is much easier than the one you're working on currently.

- Consider a different core data structure.
- Consider a different algorithm (a different way of processing the data)

- Consider using different helper functions (different built-in JavaScript methods, say)

Be the interpreter

- Walk through your code as though you were the interpreter.
- To "be the interpreter" means to evaluate *every* small step in a section of code. And by "evaluating", we mean actually evaluating the code all the way down to the literal value which are the end result of evaluation.
- `console.log` statements can help as well, but make sure to use them precisely.
- Then, when you see the concrete output, it's a lot easier to realize what is wrong.

Example:

```
var numbers = [5, 3, 8, 6, 9, 1, 0, 2, 2];

var oddEvenCounts = numbers.reduce(function(counts, number) {

  if (isOdd(number)) {

    counts[odd]++;

  } else {

    counts[even]++;

  }

}, {});
```

There's a bug in this code. Where is it? If inspection of the console output doesn't show the error immediately, then start walking through it as though you were the interpreter.Try it now, and see if you can spot the error!

Use your tests

- Write more tests.
- Re-examine the tests you've already written.
- Is there an edge case you've forgotten?