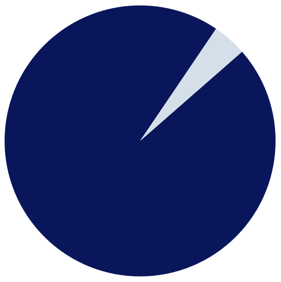
educative

12-Week Roadmap to Ace the Developer Interview

HOW DEVELOPERS SHOULD APPROACH THE INTERVIEW

Roughly 5% of all software engineering job applicants are selected for a preliminary screening. And of those that make it through the full interview loop, only 20% receive an offer.



1 *in* **5** *candidates pass the full loop*

Technical interview loops are one of the most dreaded aspects of applying for a software engineering role. They can take a lot of time to prepare for, and your performance can mean the difference of tens of thousands of dollars even if you receive an offer.

Going to an interview unprepared can mean losing out on the next big step in your career.

That's why we created this resource for software developers seeking a structured approach to interview prep.

> ***"Learn 2x faster with text-based courses"***

Educative is a hands-on developer learning platform for the future. Our library of 700+ courses is entirely text-based. This means no scrubbing, buffering, or rewatching. Since you can read faster than watching videos, you're able to cover more ground in less time.

Our courses have helped land learners secure jobs at top tech companies like Google, Facebook, and Apple. The questions you will learn are taken straight from interviews at top companies, with battle-tested prep strategies developed and validated by FAANG hiring managers.

First, we'll cover how to **prepare for an interview**, and then provide a **12-week example prep plan.**

Each week will break down the technical topics to revew along with questions to practice.

> Everything you need for interview prep, right here.

HOW DO I PREPARE FOR A TECHNICAL INTERVIEW LOOP?

With new technologies constantly emerging, and the competitive nature of the job market (especially at top FAANG companies), standing out from other candidates in the interview loop can be a challenge.

Without a structured plan in place, it can be difficult to know you're studying the right topics. When the time comes to actually interview, you don't want to be caught off guard by a question you didn't anticipate. Knowing you've done everything you can to prepare brings peace of mind when you're in the hot seat.

The Developer Interview Loop:

Coding Interview
- 60-90 minutes
- **Focus:** Data structures and algorithms

Design Interview
- 45-60 minutes
- **Focus:** System design, API/product design, or object-oriented design

Behavioral Interview
- 45-60 minutes
- **Focus:** Cultural fit, workstyle, logistics

Below we have provided an example timeline to help you prepare in a structured and efficient way.

> You may need more or less time; this is just a frame of reference. You should consider our review topics and example problems against your individual needs.

educative

12 WEEK INTERVIEW PREP ROADMAP

Week 1 **Review Chosen Language Basics**

- Refresh Coding Fundamentals
  Python | JavaScript | Java | Go | Ruby | C# | C++

Week 2 and 3 **Review Data Structures and Algorithms**

- A Visual Introduction to Algorithms
- Data Structures and Algorithms in Python

Week 4 and 5 **Practice with Data Structures and Algorithms**

- Data Structures for Coding Interviews Python | Java | C++ | JavaScript | C#
- Algorithms for Coding Interviews Python | Java | C++

Week 6, 7, and 8 **Coding Interview Practice**

- Grokking Coding Interview Patterns Python | JavaScript | Java | C++ | Go

Week 9 **Concurrency and Multithreading**

- Concurrency for Senior Engineering Interviews Python | C# | Ruby
- Multithreading for Senior Engineering Interviews in Java

Week 10, 11, and 12 **Study Design Interviews**

- Grokking Modern System Design Interview for Engineers and Managers
- FREE: System Design Interview Prep Handbook
- Grokking the API Design Interview
- Grokking the Low Level Design Interview Using OOD Principles
- Grokking the Behavioral Interview

ADDITIONAL RESOURCES:

- *Big O Primer for Coding Interviews Cheat Sheet*
- *7 Essential Data Structures for Coding Interviews Cheat Sheet*
- *5 Common Coding Interview Patterns*
- *System Design Interview Cheat Sheet*

For more Coding Interview resources,

click here

If you're preparing for your first interview in more than a year or two, we generally recommend three months of prep time. Yes, that is a long time, but planning far in advance allows you to cover every aspect of an interview loop (coding interview, design interview, and behavioral interview).

THE COMPLETE 12-WEEK PLAN

Educative serves an audience of over 1.7 million developers. We've helped learners from writing their first line of code to getting an offer at companies like Google, Amazon, and Meta.

This plan will work for most software engineering disciplines. Depending on your experience, target role, and seniority, your plan may look different.

Week 1

Brush up on the basics of your chosen language
Many technical interviews start with easy questions to raise the candidate's confidence. Don't get tripped up by something simple about your language of choice at the very beginning!

Topics to cover:
- *Splitting strings*
- *Parsing CSV or text files*
- *Declaring and using 2D arrays*
- *Reading and writing to and from files*
- *Processing command line arguments*

Weeks 2 and 3

Review data structures and algorithms
The more difficult questions in your interview will likely stem from these topics. Data structures and algorithms are the core of most high-level computer science concepts. It's entirely possible that you haven't thought about them since school, but they are integral to coding interviews.

Before diving straight into example questions, we recommend you study up on the principles of data structures and algorithms. If you're aiming for a specialized role, it is smart to consider what individual concepts are the most relevant to the work you'll be doing.

Regardless of your specialization, pay particular attention to ***Big O notation*** and other practices for complexity analysis. Understanding Big O notation helps in an intense technical interview, but it also teaches you to write programs that are faster and more efficient across the board.
Below is a reference guide to help you keep track of different Big O complexities.

> Algorithmic paradigms, perfected
Study every common algorithm and its use case in our hands-on courses Algorithms for Coding Interviews. Available in Python, Java, and C++.

Weeks 4 and 5

Practice with data structures and algorithms
As you're reviewing the basics of data structures and algorithms, start practicing simple problems with the resources listed below. Reviewing the basics will help you internalize these concepts and tackle more difficult problems later.

> Master data structures
Review all the common data structures in detail with our hands-on courses, Data Structures for Coding Interviews. Available in ***Python***, ***Java***, ***JavaScript***, ***C#***, and ***C++***.

Weeks 6, 7, and 8

Coding interview practice
By this point, you should be breezing through easier practice problems. Next, consider the problems interviewers are actually likely to ask.

Best practices:

- **Time yourself.** Try to solve your problem in 20 to 30 minutes, but don't be discouraged if some questions take longer at first.
- **Think about the runtime and memory complexities of your solutions.** Your interviewers will likely want you to articulate these complexities and how to optimize them.

Work on problems using *coding interview patterns.* Almost all questions for a coding interview are built on patterns that serve as a blueprint for solving related problems.

Depending on your seniority, software development specialization, and the individual organization, the actual questions you get asked may vary greatly. For many general roles, you can expect something related to algorithms and data structures, but it is impossible to say with certainty the actual questions you'll be faced with.
As a result, some of the best prep you can do is to study the overarching patterns

that inform each of these questions.

Drilling daily LeetCode problems may help keep your problem-solving skills sharp, but by studying the patterns representative of almost every technical interview question, you will be better prepared.

> Make coding interview patterns a habit
Learn how to solve any possible coding interview problem with our 24 essential patterns. *Grokking Coding Interview Patterns* is available in Python, JavaScript, Java, C++, and Go.

Week 9

Concurrency and multithreading
Interviewees aiming for a senior or staff engineering role especially should consider these concepts.

Certain organizations are more likely to ask questions related to concurrency and multithreading to junior roles. This is because some web applications need to serve data very quickly as real world conditions change, and they require engineers that are experienced in building systems that respond with extremely low latency.

Knowing the industry or the product niche that you're interviewing for can go a long way in predicting the technical questions you may be asked.

Example questions:
- *What is a BlockingQueue?*
- *What are some differences between a process and a thread?*
- *How can you interrupt a running thread in Java?*

> Unlock senior roles with Concurrency and Multithreading
Prepare your concurrency skills with courses specifically tailored to interviews. Concurrency for Senior Engineering Interviews is available in *Python*, *C#*, and *Ruby*.

Multithreading for Senior Engineering Interviews is available in Java.

Weeks 10, 11, and 12

The last three weeks of this plan are devoted to studying design interviews.

System Design Interviews

System Design Interviews (SDIs) are increasingly common in many software engineering interviews. These interviews are important because they help determine your engineering level. While many SDIs are for those applying to positions with more than two years of experience, design interviews have become commonplace in junior-level interview loops as well.

Most SDIs will ask you to design a large-scale web service. Typically, these systems mimic popular real-world applications so applicants can easily understand the functional and nonfunctional requirements.

Below is a guide to our structured approach to the System Design Interview.

> The best System Design course on the internet
Go under the hood with some of the most groundbreaking software systems of the last 15 years. ***Grokking Modern System Design Interview for Engineers and Managers*** is one of our most popular courses.

Object-oriented Design (OOD)

More senior applicants will be well aware of the process of object-oriented programming (OOP) through defining classes and objects. But this is still something that plays an enormous role in determining how strong of a problem-solver you'll be on the job.

Acknowledging how systems designed with an object-oriented approach work will help make you a better programmer and candidate.

> Understand real-world systems with OOD
Don't neglect design interviews just because you're applying to junior roles. Practice our bottom-up approach to design fundamentals with ***Grokking the Low-Level Design Interview Using OOD Principles.***

Product Design & API Design

API Design is a more recent take on a System Design Interview. Product design heavily centers around what APIs need to be capable of. This is a more zoomed-in take on how to build a software system capable of meeting certain functional requirements.

Considering the time limitations of a design interview, API design is often favored by interviewers and candidates who want to focus on the product-specific side of a larger system.

> Calling all web developers
Level up your career by learning what good API Design really means. *Grokking the API Design Interview* is a truly one-of-a-kind resource, with nothing like it anywhere else online.

START PREPARING TODAY

*Visit **educative.io/tech-interview-prep-roadmap** and master your next interview loop.*