Google

Interviewing @ Google!

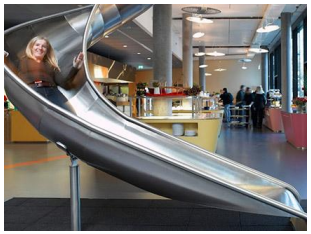
This guide will help your prepare for your interview as a **Front End or Mobile Software Engineer at Google**. You'll find out more about working at Google, some specifics on this role, what to expect in the process and then dive into technical prep to help you get ready. If you have any questions, please don't hesitate to get in touch.

THE OFFICE, PEOPLE AND PROJECTS

THE ROLES AND RESPONSIBILITIES

THE RECRUITMENT PROCESS

PREPARING FOR YOUR INTERVIEW



EXTRA PREP YOU MIGHT FIND HELPFUL

Google | Interviewing @ Google Mountain View

The Office

The Googleplex is home to scores of buildings, each with its own personality. Around the offices, you'll find Googlers sharing cubes, yurts and "huddles"; video games, pool tables and pianos; cafes and "microkitchens" stocked with healthy food; and good old fashioned whiteboards for spur-of-the-moment brainstorming.

You can find out plenty more about what it's like to actually work here by checking out our Life @ Google Youtube channel.

The People

It's really the people that make Google the kind of company it is. We hire people who are smart and determined, and we favor ability over experience. Although Googlers share common goals and visions for the company, we hail from all walks of life and speak dozens of languages, reflecting the global audience that we serve.

Our Mountain View campus is also home to some fellow engineers you may have already heard a bit about. From our founders, Larry Page and Sergey Brin, to engineers that created programming languages, operating systems and text editors you may use every day -- whether C, Java, Python, Unix or VIM is your preference.

The Projects

Our products include Google Search, Google+, Gmail, Calendar, AdWords, AdSense, Android, Chrome, Maps, Music, Earth, News, SketchUp, Wallet, Blogger, Scholar, Trends and more.

Our search engine processes a billion search queries per day. More than 250 million Android devices have been activated worldwide.

Still want more info?

Office location & Streetview
Office pictures
California information
HOA: Launching your Career
Life @ Google
Meet Rob, Pablo & Raman

Google | The Roles and Responsibilities of a Software Engineer

The Role

You can start at our Google Careers Page, which is broken down per location and has real-time and detailed requirements for our needs across all of the projects.

Ultimately, you're going to help us build great Google products!

The Responsibilities

Software Engineers at Google can be responsible for the whole lifecycle of a project. Depending upon the project you join, you could be involved in research, design, planning, architecture, development, testing, implementation, and release phases. Our front end engineering is more akin to the industry's "full-stack" engineering than anything else.

Get a Googler's Perspective:

"If you are passionate about engineering and web technologies, there are very few companies that can offer Google's scale and resources. When you launch a web product at Google, it instantly has millions of users and is supported by hundreds of engineers and data centers. It is very exciting to be able to affect so many people on a day to day basis with features I help build or improvements that I make."

To Summarize

We hire people with a broad set of technical skills who are ready to tackle some of technology's greatest challenges and make an impact on millions, if not billions, of users. Engineers at Google not only revolutionize search, they work on massive scalability and storage solutions, large-scale applications, and entirely new platforms for developers around the world. From AdWords to Chrome, Android to YouTube, Social to Local, Google engineers are changing the world one technological achievement after another.

Still want more info?

A Google engineer's career

Ask a Google engineer

Research @ Google

Life @ Google

Wait, you develop for iOS at Google?

How we Hire

Interview Process

Google's interview process is consistent globally. This approach allows engineers the opportunity to develop their career within Google across geographies and projects.

Project Matching

As we assess your skills and strengths throughout the interview process, we consider potential project matches. We take into consideration your interests as well as the current needs of the various Google projects before making a project match. It's not uncommon that your skills would fit a number of projects, and sometimes we'll arrange a call with a Tech Lead on a project to help find the best match for you and your interests.

The Process

Phone Interviews
Up to 2 x 45 min interviews with a Google Engineer. A Google doc will be used as a virtual whiteboard for coding and algorithmic problems.

On Site Interviews
Up to 5 x ~45 min interviews. You will use a whiteboard for discussions and coding tasks. You will also have lunch with an engineer (a great time to ask questions) and have a tour of the office.

Committee Review
An independent committee of Googlers review the interview feedback. They ensure our hiring process is fair and that we're holding true to our "good for Google" standards as we grow.

Offer
Once everything is reviewed and approved, we'll extend the offer, send you the details, and talk through getting started at Google!

Still want more info?

Interviewing @ Google

How we hire

Google | Preparing for your Interview

Before you Start

We highly recommend that you do your research about the interview process at Google. A great place to start would be finding out how we hire. Then check out our technical coaching video to see the format of our interviews.

Note: We also recommend checking out Steve Yegge's blog post covering everything from technical prep to mental preparation.

Interview Questions

Be prepared to write **code on a blank Google Doc or whiteboard**. Computer Science fundamentals are prerequisites for all engineering roles at Google, regardless of seniority, due to the complexities and global scale of the projects you would end up participating in. Interview topics may cover anything on your resume, especially if you have stated that you are an expert.

You may want to take a look at some practice exercises here.

Where to Focus

Demonstrate your problem solving skills as applied to the question asked. If it's a coding question, providing efficient prototype code in a short time frame to solve the problem is key. If it's a design question, work with your interviewer to create a high-level system, at times perhaps diving deeper on particularly salient issues. If it's a general analysis question, show that you understand all particularities of the problem described and (where applicable) offer multiple solutions, discussing their relative merits. Every interviewer ultimately wants to answer the question of whether they could successfully work on a Google Engineering team with you.

Still want more info?

About us
The Google story
Google Developers
Interview Tips from Recruiters

Main Focus: General Technical Preparation

Coding: Expertise in at least one programming language is required. Know a fair amount of detail about your favorite programming language. You will be expected to know about API's, OOD/OOP, how to test your code, as well as come up with corner cases and edge cases for yours and other people's code.

- We expect variable declaration/initialization, error checking, and effective use of appropriate data structures. Coding exercises may require synchronization primitives and concurrency libraries. Make sure your code is clean and bug-free, and avoid writing pseudo-code unless directed to do so.
- We focus on conceptual understanding rather than memorization. We are *not* looking for memorization of all Java APIs (or the equivalents in other languages). It's fine if you forget a method or a class name.
- Some interviewers care more about syntax than others - if you're not sure, ask.

Algorithms: You'll likely be asked to design an algorithm and write code. You'll be expected to know the complexity of an algorithm and how you can improve/change it. Be comfortable with big-O analysis and running time complexity. Big-O notation is also known as the run time characteristic of an algorithm. Be prepared to explain the running-time complexity of algorithms you know and are asked to devise.

- When writing a more complicated algorithm, make sure you have a solid idea in your head (or outlined in bullet points) before jumping into coding.
- If you get a chance, try to study up on fancier algorithms such as Dijkstra and A*.

Data structures: Study up on as many other structures and algorithms as possible. You should especially know about the most famous classes of NP-complete problems, such as traveling salesman and the knapsack problem. Be able to recognize them when an interviewer asks you in disguise. You will also need to know about trees (including basic tree construction), traversal and manipulation algorithms, hash tables, stacks, arrays, linked lists, priority queues.
Only implement standard data structures if your language's standard library does not have them. Show you understand their performance characteristics and know how to use them!

Expert tips:

Open up and review your Data Structures & Algorithms textbooks.

For more information on algorithms, you can also visit TopCoder and GitHub.

Main Focus: Front End and Mobile Technical Preparation

Web Front End: You should be ready to cover topics like front end latency and implementation of standard CS algorithms using idiomatic Javascript. You should be able to articulate Javascript strengths and shortcomings and ready to cover any of the following:

- Web security issues: XSS, XSRF, etc.
- Prototypal inheritance
- DOM API & manipulation
- Browser / DOM events & event handling
- XHR requests & HTTP headers
- CSS manipulation
- JavaScript closures

Native: Be ready to cover implementation of standard CS algorithms using idiomatic mobile Java, ObjC, or Swift, and have awareness of chosen language strengths and shortcomings

- How to split tasks in a UI-friendly way (ie threading/GCD, not stopping the UI thread, etc.)
- Structuring APKs for a large application, managing permissions (Android)
- Building offline functionality (Android)
- Leveraging Intents and Intent Filters (Android)

iOS: Focus translating ideas to code. You should demonstrate ability to grasp some basic data structures available in Foundation (NSArray, NSDictionary, NSSet) and block usage (no, you don't have the memorize the block syntax!) Show ability to write code to do client/server interactions and understand limitations and system behaviors. Understand key language features like memory management model; object-orientated features; protocols, delegates, categories; Grand Central Dispatch, and performance. Demonstrate ability to create screens in UIKit, understanding of View Controller concepts, UIView structure and best practices. Understand how to create a view controller and layout subviews.

Expert Tips:

Try completing exercises at:

AlgorithmIV
AngularJS tutorial.

Nice to Know: Technical Preparation

Sorting: What common sorting functions are there? On what kind of input data are they efficient, when are they not? What does efficiency mean in these cases in terms of runtime and space used? E.g. in exceptional cases insertion-sort or radix-sort are much better than the generic QuickSort / MergeSort / HeapSort answers. You should know the details of at least one n*log(n) sorting algorithm, preferably two (say, quick sort and merge sort).

Searching: Familiarize yourself with some common search algorithms. This could be anything from simple binary search, over depth-first/breadth-first tree search, combinatorial search over solution spaces, substring searching, etc.

Mathematics: Counting problems, probability problems, and other Discrete Math 101 situations surrounds us here at Google. Spend some time before the interview refreshing your memory on (or teaching yourself) the essentials of elementary probability theory and combinatorics. You should be familiar with n-choose-k problems and their ilk – the more the better.

Graphs: To consider a problem as a graph is often a very good abstraction to apply, since well known graph algorithms for distance, search, connectivity, cycle-detection etc. will then yield a solution to the original problem. There are 3 basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list); familiarize yourself with each representation and its pros/cons. You should know the basic graph traversal algorithms, breadth-first search and depth-first search. Know their computational complexity, their tradeoffs, and how to implement them in real code.

Expert Tips:

Break out the books and brush up on content in:

Elements of Programming Interviews

Nice to Know: Technical Preparation

Recursion: Many coding problems involve thinking recursively and potentially coding a recursive solution. Prepare for recursion, which can sometimes be tricky if not approached properly. Practice some problems that can be solved iteratively, but a more elegant solution is recursion.

Operating systems: Study up on concurrency and distributed systems. You should understand processes, threads, concurrency issues, locks, mutexes, semaphores, monitors and how they all work. Understand deadlock, livelock and how to avoid them. Understand how context switching works, how it's initiated by the operating system and underlying hardware. Know a little about scheduling. Know the fundamentals of "modern" concurrency constructs.

System design: Questions about system design or component design are common. They are typically formulated so that they are approachable to any person with a computer science background and some work experience. Sample topics include: features sets, interfaces, class hierarchies, distributed systems, designing a system under certain constraints, simplicity, limitations, robustness and tradeoffs. You should also have an understanding of how the Internet actually works and be familiar with the various pieces (routers, domain name servers, load balancers, firewalls, etc.). **Note**: University Graduate candidates without industry experience are *not* typically asked system design questions.

Expert tips on distributed system design:

Distributed systems & parallel programming

Scalable Web Architecture & Distributed systems

General Interview Tips

Substantiate

Show enthusiasm for whatever it is about the Software Engineer role that you are passionate about! Be familiar with your past experiences and comfortable communicating your work.

Explain

Explain your thought process and decision making as you work through problems. Approach the problem as if you were trying to solve it with a colleague at work, in which case it would be more of a discussion than a test. In Google's interviews, our engineers are evaluating not just your technical abilities but also how you approach problems and try to solve them. We want to understand how you think. This would include **explicitly stating and checking any assumptions** you make in your problem solving to ensure they are reasonable.

Clarify

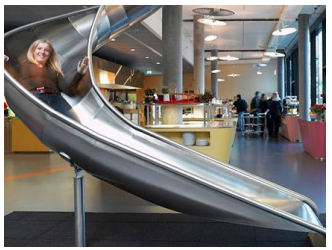
Ask clarifying questions if you need more information or do not understand the problem. Many of the questions asked in Google interviews are deliberately underspecified because our engineers are looking to see how you engage the problem. In particular, they are looking to see which areas you see as the most important piece of the technological puzzle you've been presented.

Improve

Think about ways to improve the solution that you present. In many cases, the first solution that springs to mind isn't the most elegant and may need some refining. It's worthwhile to talk through your initial thoughts with the interviewer. If necessary, don't be afraid to start with the brute force solution and improve on it - just let the interviewer know that's what you're doing.

Practice

Practice writing code without an IDE on either a whiteboard (for onsite interviews) or Google Doc (for phone interviews). After all, this likely isn't your usual development environment. Be sure to test your own code and be sure it works (e.g. test for edge cases).



Ask questions:

You are also interviewing Google, and you want to make sure Google is a good place for you.

Feel free to ask the same question more than once; this is a great opportunity to hear multiple engineers' perspectives on what matters to you in your work environment.

Additional Interview Tips from Googlers

Ask a friend to practice whiteboard coding with you, asking you questions. Step away from the problem and think out loud while answering.

When given a question, (i) make sure you completely and fully understand the question, (b) write down an initial set of edge cases, (c) verbalize every solution or idea you have, (d) code up a solution on the whiteboard or Google Doc, and (e) go through all your edge cases.

Approach the interview as a problem to be solved. No one is expected to do perfect on every aspect; the interview should explore the edges of knowledge and so occasionally may venture into "I don't know" territory. Don't let it fluster you.

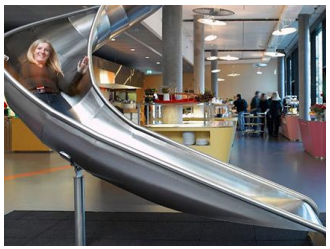
Listen -- don't miss a hint if your interviewer is trying to assist you!

Saying what you know can be helpful, specifically when it can convey a lot of knowledge while spending minimal time. (ex: "I would normally use StringBuilder, but in the interest of time, I'll use the + syntax because it is quicker to write.")

Consider inventing methods as you go. Like findFirstVowel(String s);
If the method is "interesting" the interviewer can ask you to implement.

Don't stress small syntactical errors. If you can't remember whether the method is a substring(start, end) or substring(start, length), just pick one and let your interviewer know. In the real world, you could just check the documentation or even autocomplete.

Test your solution with sample data/parameters. Discuss how your solutions scale. Discuss the time and size complexity of your solutions. Think about and express the Google scale implications of decisions.



Get some practice!
To practice for your interview, you may want to try:

Google Code Jam questions -- samples from Google coding competitions.

Take Dean's advice! Try practicing coding in a Google doc or on a whiteboard with a friend.

Prep Recommended by Googlers

Books

Most Highly Recommended:

Cracking the Coding Interview

Additional Resources:

Programming Interviews Exposed: Secrets to Landing Your Next Job
Programming Pearls
Introduction to Algorithms
The Algorithm Design Manual
Eloquent Javascript
Effective JavaScript
JavaScript Inheritance Models
Google Closure Tool Chain
Elements of Programming Interviews
Android Developer Materials
Android Developer YouTube Channel
NSHipster

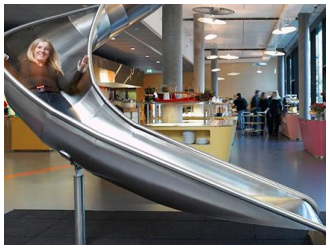
Online resources

Places to Find Practice Problems

Code Jam
Leet Code
Project Euler

Google Publications for Additional Reference:

The Google File System
Bigtable: A Distributed Storage System for Structured Data
MapReduce: Simplified Data Processing on Large Clusters
Google Spanner: Google's Globally-Distributed Database
Google Chubby



Still want more info?

Google books
Google publications
Google scholar