

---

## Pointers for Software Engineers build unknown

*A curated list of topics to start learning software engineering*

Pointers for Software Engineers is a complete open-source computer science curriculum, designed to cover the fundamentals and practical topics of software engineering most relevant to the industry today. Think of it as an alternative or supplement to college or bootcamp education. It aims to serve as a guidebook for beginners planning to make a career transition into software engineering, computer science students looking to expand their practical knowledge, and experienced engineers curious to seek references for other subject matters.

Each topic in this curriculum includes only a single reference for readers to gain familiarity and understanding in a short period of time. Tutorials with clear content structure and hands-on examples are selected over other references. Some would say it is taking a breadth-first search approach to computer science learning.

This curriculum is divided into four major sections: fundamentals, advanced, tracks, and subjects. Fundamentals cover the core part of computer science and are often required as the lower-division courses in universities. Advanced goes more in depth on various computer science subjects and are usually topics discussed in upper-division university electives. Tracks outline the most common software engineering roles and provide relevant topics for introductory understanding. Subjects offer content for specialization fields researched in corporate research labs and startup initiatives.

If you are just starting, it is recommended to go through the fundamentals in order, then pick a subset of advanced topics to read at will. For the curious ones, use tracks and subjects as references when choosing career options that most interest you.

### Table of Contents

#### 1. Fundamentals

- CS 101
- Data Structures & Algorithms
- Operating Systems
- Shell, Git, editors, and more

#### 2. Advanced

- Web Applications
- Databases
- Programming Languages

- 
- Compilers & Interpreters
  - Networking
  - Security
  - Miscellaneous

### 3. Tracks

- Systems Engineering
- Product Engineering
- Machine Learning & Data Engineering
- Security Engineering
- Game Engineering
- QA Engineering

### 4. Subjects

- Artificial Intelligence
- Blockchain
- Bioinformatics
- Cybersecurity
- Human-Computer Interaction
- Theory

## 1. Fundamentals

### CS 101

- Course: <https://www.edx.org/course/cs50s-introduction-to-computer-science>
- Python: <https://www.codecademy.com/learn/learn-python-3>

### Data Structures & Algorithms

- Course: <https://www.coursera.org/learn/algorithms-part1>
- Book: <https://livebook.manning.com/book/grokking-algorithms>

### Operating Systems

- Course: <https://www.udacity.com/course/introduction-to-operating-systems-ud923>
- Book: <http://pages.cs.wisc.edu/~remzi/OSTEP>

---

## Shell, Git, editors, and more

- Course: <https://missing.csail.mit.edu>

## 2. Advanced

### Web Applications

- HTML & CSS: <http://learn.shayhowe.com/html-css>
- jQuery: <http://jqfundamentals.com>
- Frontend development: <https://www.freecodecamp.org/news/from-zero-to-front-end-hero-part-1-7d4f7f0bff02>
- Backend development: <http://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>
- Design: <https://medium.com/hh-design/design-resources-5071be5f2e43>

### Databases

- SQL: <https://www.khanacademy.org/computing/computer-programming/sql/sql-basics/v/welcome-to-sql>
- SQL practice: <https://pgexercises.com>
- NoSQL: <https://www.openmymind.net/mongodb.pdf>
- Database systems: <http://www.redbook.io>

### Programming Languages

- JavaScript: <http://speakingjs.com/es5/index.html>
- Python: <http://pymbook.readthedocs.io>
- Go: <https://gobyexample.com>
- C++: <https://www.learncpp.com>
- Lisp: <http://www.gigamonkeys.com/book>
- Ruby: <http://rubylearning.com/satishtalim/tutorial.html>
- Java: <https://leanpub.com/aprimeronjava/read>
- Scala: [http://twitter.github.io/scala\\_school](http://twitter.github.io/scala_school)
- Rust: <https://doc.rust-lang.org/book>
- PHP: <https://phptherightway.com>
- Haskell: <http://learnyouahaskell.com>
- Clojure: <http://www.braveclojure.com/clojure-for-the-brave-and-true>

---

## Compilers & Interpreters

- Compilers: <https://www.amazon.co.uk/Compilers-Principles-Techniques-Alfred-Aho/dp/0201100886>
- Interpreters: <http://craftinginterpreters.com/contents.html>

## Networking

- Network programming: <http://beej.us/guide/bgnet/html>

## Security

- Cryptography: <https://www.crypto101.io/>
- Network security: <https://training.apnic.net/wp-content/uploads/sites/2/2016/12/TSEC01.pdf>

## Miscellaneous

- Interview preparation: <https://leetcode.com>
- Design patterns: [https://sourcemaking.com/design\\_patterns](https://sourcemaking.com/design_patterns)
- Linux: <https://linuxjourney.com>
- Bash: <http://www.tldp.org/LDP/Bash-Beginners-Guide/html>
- Markdown: <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>
- LaTeX: <https://www.latex-tutorial.com/tutorials>
- Algorithms: [http://mimoza.marmara.edu.tr/~msakalli/cse706\\_12/SkienaTheAlgorithmDesignManual.pdf](http://mimoza.marmara.edu.tr/~msakalli/cse706_12/SkienaTheAlgorithmDesignManual.pdf)
- Competitive programming: <http://www.stanford.edu/class/cs97si>
- Mathematical programming: <https://projecteuler.net>

## 3. Tracks

### Systems Engineering

- System design: <https://github.com/donnemartin/system-design-primer>
- Site reliability: <https://landing.google.com/sre/sre-book/toc/index.html>
- Data-intensive systems: <https://www.amazon.co.uk/Designing-Data-Intensive-Applications-Reliable-Maintainable/dp/1449373321>
- Microservices: <https://microservices.io>
- Distributed systems: <https://github.com/aphyr/distsys-class>
- AWS: <https://adayinthelifeof.nl/2020/05/20/aws.html>

---

## Product Engineering

- React: <https://reactjs.org/tutorial/tutorial.html>
- React Native: <https://facebook.github.io/react-native/docs/tutorial.html>
- API design: <https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api>
- iOS: <https://itunes.apple.com/gb/course/developing-ios-11-apps-with-swift/id1309275316>
- Android: <https://www.udacity.com/course/developing-android-apps-ud853>
- Ruby on Rails: <https://www.railstutorial.org/book/beginning>
- Node: <https://www.airpair.com/javascript/node-js-tutorial>

## Machine Learning & Data Engineering

- Machine learning: <https://www.coursera.org/learn/machine-learning>
- Data mining: <http://guidetodatamining.com>
- Data visualization: <http://alignedleft.com/tutorials/d3>
- TensorFlow: <https://www.tensorflow.org/tutorials>
- PyTorch: <https://pytorch.org/tutorials>

## Security Engineering

- Security engineering: <https://www.cl.cam.ac.uk/~rja14/book.html>
- Security checklist: <https://www.sqreen.com/checklists/saas-cto-security-checklist>

## Game Engineering

- Game programming: <http://www-cs-students.stanford.edu/~amitp/gameprog.html>
- Game design: [http://dl.booktolearn.com/ebooks2/computer/gamedevelopment/9781449337933\\_designing\\_g](http://dl.booktolearn.com/ebooks2/computer/gamedevelopment/9781449337933_designing_g)

## QA Engineering

- Test design: <http://dahlan.unimal.ac.id/files/ebooks/2004%20A%20Practitioner's%20Guide%20to%20Software>
- Refactoring: <http://silab.fon.bg.ac.rs/wp-content/uploads/2016/10/Refactoring-Improving-the-Design-of-Existing-Code-Addison-Wesley-Professional-1999.pdf?lang=latxzf>

---

## 4. Subjects

### Artificial Intelligence

- Neural networks and deep learning: <http://neuralnetworksanddeeplearning.com/index.html>
- Reinforcement learning: <https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html>
- OpenCV: <https://www.pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv>

### Blockchain

- Bitcoin: <https://bitcoin.org/bitcoin.pdf>
- Crypto: <https://a16z.com/2018/02/10/crypto-readings-resources>

### Bioinformatics

- Computational biology: <https://open.oregonstate.education/computationalbiology>

### Cybersecurity

- Digital defense: [https://books.google.co.uk/books?id=UlvDCgAAQBAJ&printsec=frontcover&source=gbs\\_ge\\_s](https://books.google.co.uk/books?id=UlvDCgAAQBAJ&printsec=frontcover&source=gbs_ge_s)

### Human-Computer Interaction

- User interface: [https://www.amazon.com/gp/product/0321537351/ref=as\\_li\\_ss\\_tl?ie=UTF8&camp=1789&creat](https://www.amazon.com/gp/product/0321537351/ref=as_li_ss_tl?ie=UTF8&camp=1789&creat)  
20

### Theory

- Academic paper review: <https://blog.acolyer.org>
- Corporate research: <https://research.google/research-areas>

## Appendix

### Why is a reference chosen over the others for a topic?

References are selected using the following order of criteria:

- 
1. Free and available online
  2. Primers over textbooks
  3. Learning by examples
  4. Content coverage on practical applications

**How do I contribute?**

*Pointers for Software Engineers* is meant to be an open-source curriculum under continuous updates by the community. All contributions are welcome by simply creating a pull request for review.

**May I modify or distribute this project?**

This project is released under MIT License. See [here](#) for more information.