

**Dr. Andrey Esaulov**

# HIGHEST PAID PROGRAMMA MMER

---

**3 Days Plan To Start Your Career In Programming**

---

# Foreword

## Barcelona, Spain: Mobile World Congress

The spring of 2015 was an exiting time for me.

After months on planing, calculating the costs and revenues, building the prototypes and learning from the first user feedback “my” App was about to launch.

Here, in the middle of the most looked-upon venue for mobile industry - I was scheduled to have my first major meeting with the senior executive of Amazon to lay down the main features of the App.

When I was finished showing off the MVP, the man in his mid-fifties looked at me - and said: “Well, if you manage to do this - you would have found the *holy grail* of modern computing”. To turn the pathos of this phrase down - I remember saying “Well, I guess this PhD in German Literature I spent 4 years working on - was worth it after all”. We both laughed - but on my way out I was

asked to tell what I meant earlier. How did the someone with PhD in Literature ends up being Head of Mobile in a big german company?

It's 2017 now - the app has successfully launched, we're enjoying huge number of downloads, retention is high and people are spending more and more time in the App. And every time I get to talk about the App - on the conferences, [workshops](#) or in [interviews](#) - and get to the point about my background - the same question arises all the time.

So in a sense - this guide had to be written. I'm not an expert by any stretch of the imagination. I'm just someone who wanted to build something really cool. And learned how to acquire the skills to do it.

The approach I'm sharing is unique. I'm not focused on the particular programming language, or platform, or technology stack. Instead I try to precondition you to think differently. In a way to learn programming - you have to stop thinking about learning to program - and

start thinking about building products, services, cool stuff.

By shifting your focus - you will look quite differently at any online programming course out there. You will never face motivation problem again. And you will never have this nagging feeling - you have to learn a bit more - to get your first jog, build your first App, create your first website.

It's all about focus. And this guide will create this learning focus for you.

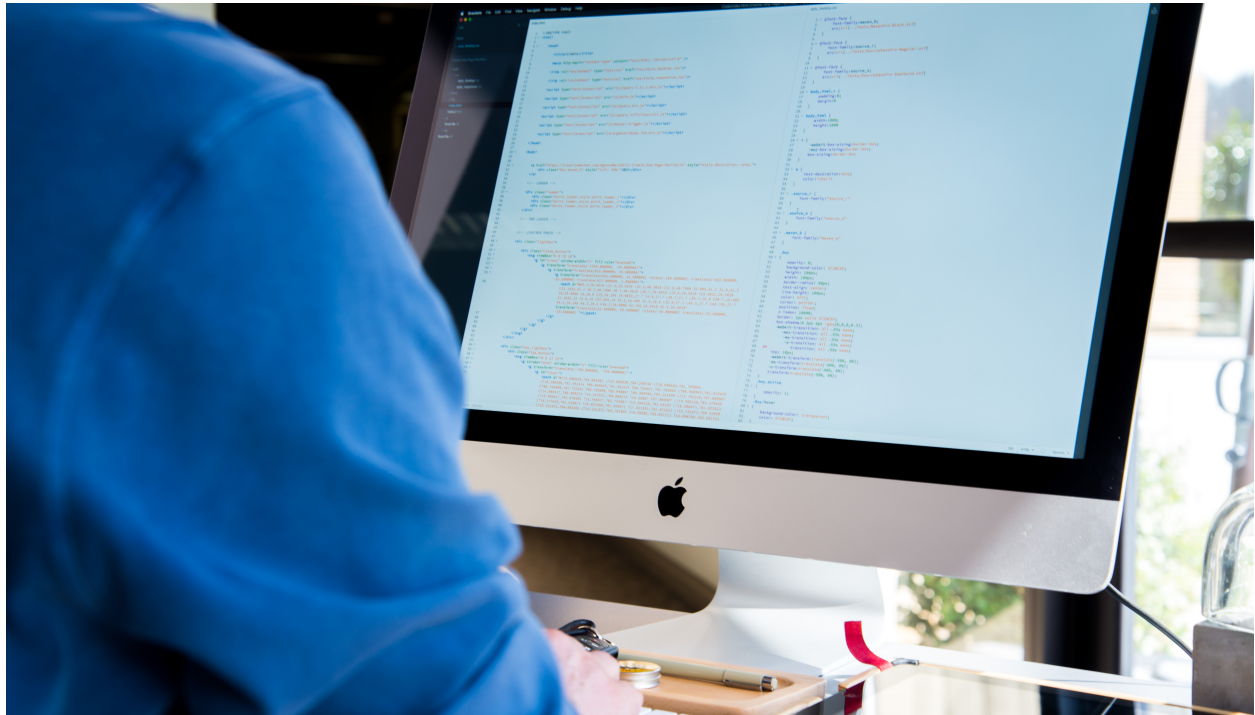
In 3 days.

Let's get started!

P.S. Like this book and want to get some free coaching every week? I designed a weekly guide just for my students - and you can subscribe to it for free.

Day 1

# Programming Language to Learn



## The Most Asked Question

“What programming language should I learn?” - is the most frequent question I get from my students. There are tons of “experts” on the internet - and on YouTube - that are trying to give the ultimate answer.

They either try to analyze the language design, or throw numbers around - like market demand, jobs in this language, Google search trends etc.

**They're all wrong.**

Learning Programming is hard. You will be investing a lot of time - and often money - in the process.

So I treat this question from the ROI perspective.  
What is ROI?

**ROI** stands for “**return on investment**” and is used by virtually any serious organization to plan long-term project.

You should apply this principle to your programming career as well. After all - why would you not? Is your time or your money less valuable than those gentlemen on Wall Street? No? Well - then use their tools - to reach your goals!

What follows is the complete plan to calculate ROI while planing your programming career.

# Calculating ROI

You must take the following points into consideration, while deciding what programming language you should learn and master.

## Learning vs. Building

How fast can you build something useful in this language? It could be an App to track your sleep time, or it may be a website to post pictures of your pet. It doesn't really matter - it must be a useful product.

The goal of this question is to change your mindset from **learning** into **building**. The only reason to learn a programming language in the first place - is to build something with it.

Here is an analogy for you. Take a quick test below.

**What is the most efficient way to learn to swim?**

1. **Answer A:** Study human body anatomy. Then study physics. Then buy a theory course on swimming from olympic sports champion.
2. **Answer B:** Go to the pool!

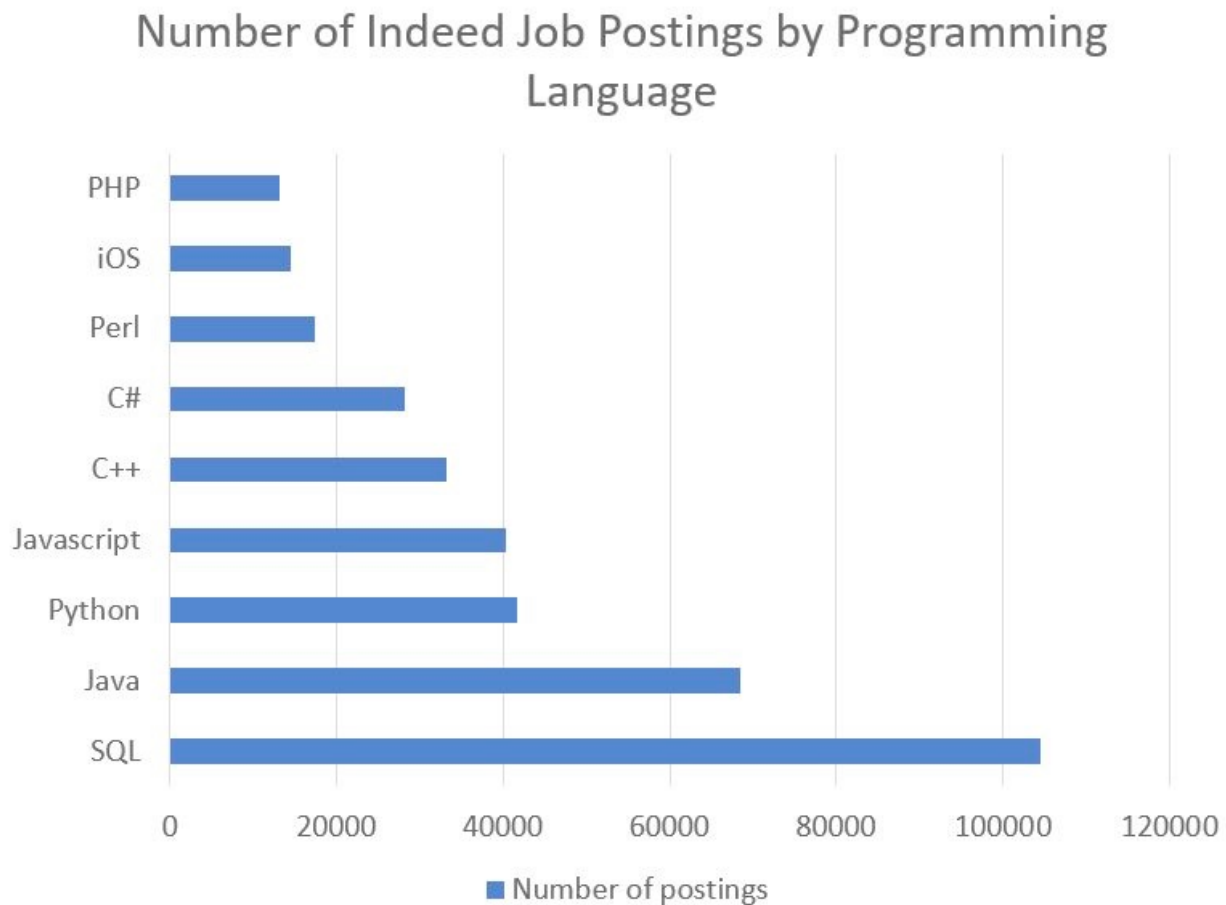
Way too many students are focusing on **learning** instead of **building** - and fall into the trap of multitude of online courses out there.

Those courses cover very narrow topics of how to start developing in Language X, or use technology Y - flooding with 60+ hours details - that are absolutely inapplicable in your day-to-day development career.

Change your mindset! And look at the technology from this perspective: how fast can I build something useful with it?



# Job Market Analysis vs. Language Momentum



Another trap many fall into is using job market research as a starting point in deciding what language they should learn. It's only natural to think - "Well, let me

see what jobs are available in tech - and learn to do the technologies they mention”.

Why is this a trap?

Because 90% of the job listings refer to the **old** languages and **obsolete** technologies. The vast majority of companies who are hiring these days - have their stable businesses running. Those businesses were built some 5, 10 or may be even 20 years ago. Lots of them are running on old and obsolete technologies - like Java or PHP or C++ or MySQL.

Because those businesses are profitable - their owners want them just to stay this way. Any kind of change - is a potential risk. There is an old IT-saying “Never Change a Running System”. Unfortunately for you - if you start working in this environments - you will never learn anything new. All you will ever do - is patch up old, inefficient and terrible code.

Another analogy for you: imagine it's 1904 and you are looking at the following jobs listings

# what college degree should You get?

1. **Listing A:** For our London coachman service we are looking for experienced horse-caretakers and veterinarians with college degrees in order to make our horses even faster!
2. **Listing B:** For our first plant in Canada we, Ford Motor Company are looking for engineers with the college degrees in order to build the first automobile in Canada.

The screenshot shows the Ford website's navigation menu with 'OUR STORY' selected. Below is a timeline of key events:

- 1901:** Henry Ford defeats the top racecar driver of the era. Ford designed the 26-horsepower Sweepstakes and defeated Alexander Winton's 70-horsepower Bullet in 10 laps around the one-mile oval of the Detroit Driving Club. The victory led to Henry Ford's second short-lived attempt at auto manufacture, the Henry Ford Company.
- 1903:** The Ford Motor Company is incorporated. With 12 investors and 1,000 shares, the company had spent almost all of its \$28,000 cash investment by the time it sold the first Ford Model A on July 23, 1903. But by October 1, 1903 Ford Motor Company had turned a profit of \$37,000.
- 1904:** Ford Motor Company of Canada is founded. Ford's first international plant was built in Walkerville (now Windsor), Ontario, right across the Detroit River from Ford's existing facilities. The company was a separate organization with its own set of shareholders, financing, and so on.
- 1907:** Ford introduces the scripted typeface of its trademark. Childe Harold Wills designed the Ford logo. He used his grandfather's stencil set, which was based on the style of writing taught in schools when Ford and Wills were children. However, the Ford oval would not be featured on a car until the 1927 Model A.
- 1908:** Ford introduces the Model T. Henry Ford's Model T put the world on wheels with a simple, affordable, durable design. By 1927, Ford had sold 15 million Model T production in May 1927, making it the best-selling vehicle of all time.

What would you choose as your career path? London or Canada?

It's very important to look into the future, instead of the past. Programming language momentum is much more important than the current job listings. You want to aim for the job market that will come, not for the one that will past.

That is why momentum, or even hype - is so important when you're deciding what language and what technology your are going to master.

# Action Exercises to complete TODAY

Following the guidelines discovered in this chapter, put down in writing:

1. What is the thing/product/service you want to create yourself?

Is it an iOS App?

Android App?

It is a Website?

1. Depending on your choice - find the technologies, languages, frameworks that have the most momentum on that platform? Write down the list.

Good way to “feel” the momentum of the programming language is by looking at GitHub [repositories created in that language](#):


# Programming languages


A list of programming languages that are actively developed on GitHub.


48 repositories 33 languages Last updated 14 days ago


Stars Language

Search showcases

 **apple / swift**  
 The Swift Programming Language  
 C++ ★ 38,469 🍴 5,748 Updated 5 minutes ago



 **golang / go**  
 The Go programming language  
 Go ★ 28,156 🍴 3,758 Updated 16 minutes ago



### Related showcases

**Great for new contributors**  
 These projects have a history and reputation for being welcoming to new ...

**Projects that power GitHub**  
 Here are some of the great open source projects that GitHub is using to ...

Alternatively here are some keywords example you can use to Google it:

- iOS Development 2017 frameworks
- Android Frameworks 2017
- hype web technologies 2017
- javascript frameworks 2017
- web development trends 2017
- iOS programming trends 2017
- Frontend development trends 2017
- Backend development trends 2017

1. Study the documentation pages for this technologies. Don't try to **understand** - just try to get the feeling - how approachable they are.

Things to watch for:

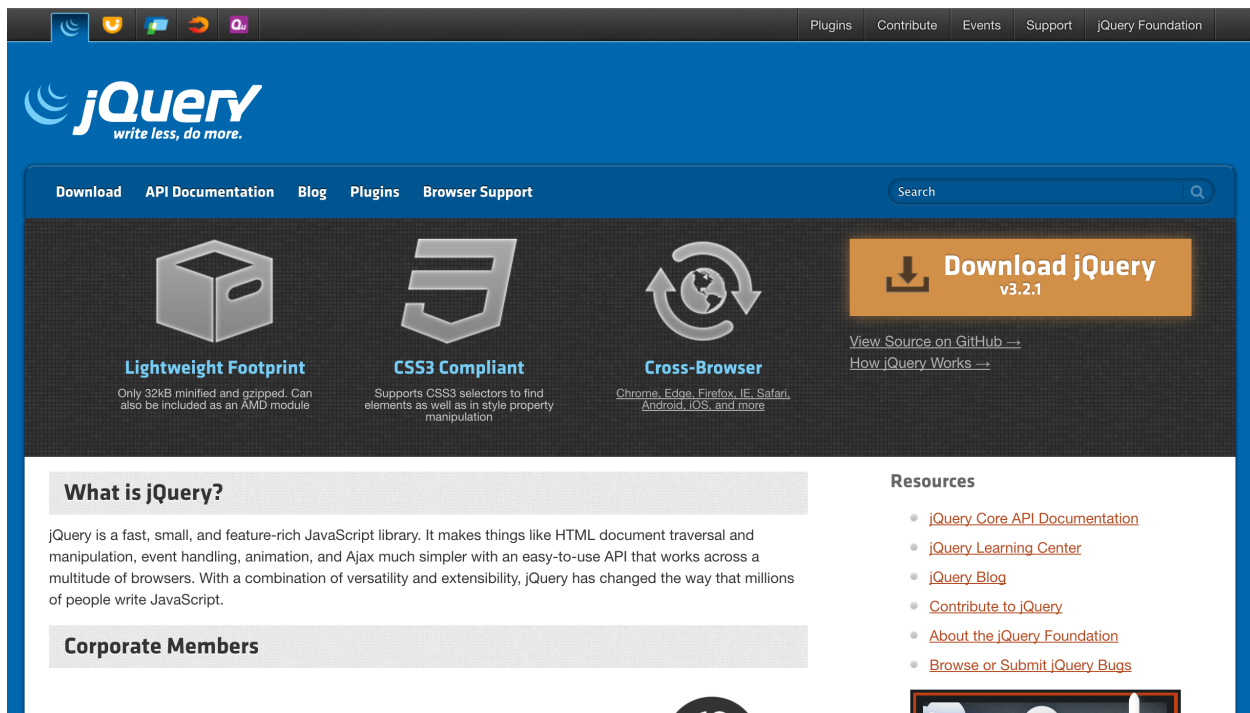
Does the website looks modern?

When was the last update?

Are there many practical examples?

Are there any videos online that are called "How to build an App with ... in 1 hour"?

Are there any *\*new\** coursed on that technologies on Udemy?



Site looks dated. So is the technology behind it



## The Progressive JavaScript Framework

[GET STARTED](#)[GITHUB](#)

### Approachable

Already know HTML, CSS and JavaScript? Read the guide and start building things in no time!

### Versatile

Simple, minimal core with an incrementally adoptable stack that can handle apps of any scale.

### Performant

19kb min+gzip Runtime  
Blazing Fast Virtual DOM  
Minimal Optimization Efforts

---

Site looks modern. Chances are so is the framework.

Sort your list by the approachability on this technologies. Starting from the most approachable - that typically will show how to build something in 10 minutes on their first page. All the way down to very complicated.

This list will be your study-template for the next days.



Day 2

# Choosing Right Learning Resources



## Creating a Goal

“Goals are the fuel in the furnace of achievement. A person without goals is like a ship without a rudder,

drifting aimlessly and always in danger of ending up on the rocks. A person with goals is like a ship with rudder, guided by a captain with a map, a compass, and a destination, sailing straight and true toward a port of his own choosing.”

– Brian Tracy, “Maximum Achievement”

If you completed the action exercises by the end of the day you should have a list of technologies you could build the product you wanted, sorted out by approachability.

What you’re going to do with your list today—is to take the first item on that list—and find the resources where you can learn about the technology: books, articles, tutorials, online courses.

I will provide you with the example list of resources below. In order for your to come up with your OWN list is to focus.

Remember the first principle you learned yesterday— Learning vs. Building. When you'll be looking at the resources— evaluate them by this one simple rule:

“Will I have built something after I finished reading this book, watching this tutorial, completing this course?”

Now, it must not be exactly the product/service/app you decided to build yesterday. It just needs to be a finished product. The book, tutorial, course— must specifically mention that by the end of it—you'll be able to build: an App, a Website, a Service etc.

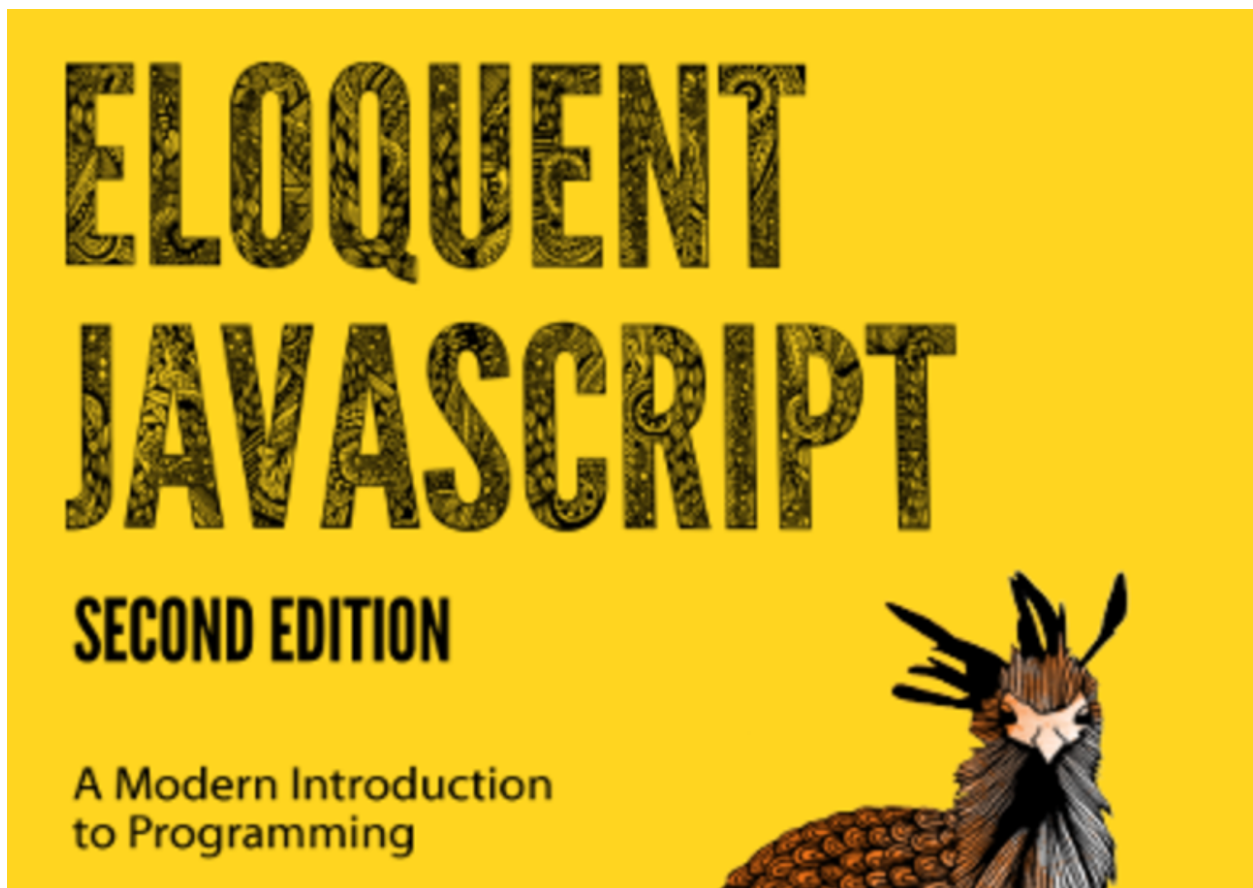
# How to Choose Right Learning Resource?

## Four Points Evaluation System

1. **Building** clones of exiting products: Instagram-Clone, Facebook-Clone, Twitter-Like Service
2. The resource must have explicit **timetable** – how long does it take to finish it. It's very, very important! You must understand upfront – how much time you're going to invest into building it. A goal without a deadline – is not a goal at all.
3. **Ratings**, reviews and comments of the courses / books / tutorials
4. Very important!!! Is there a way to **contact** an author directly? Find on Facebook? Twitter?

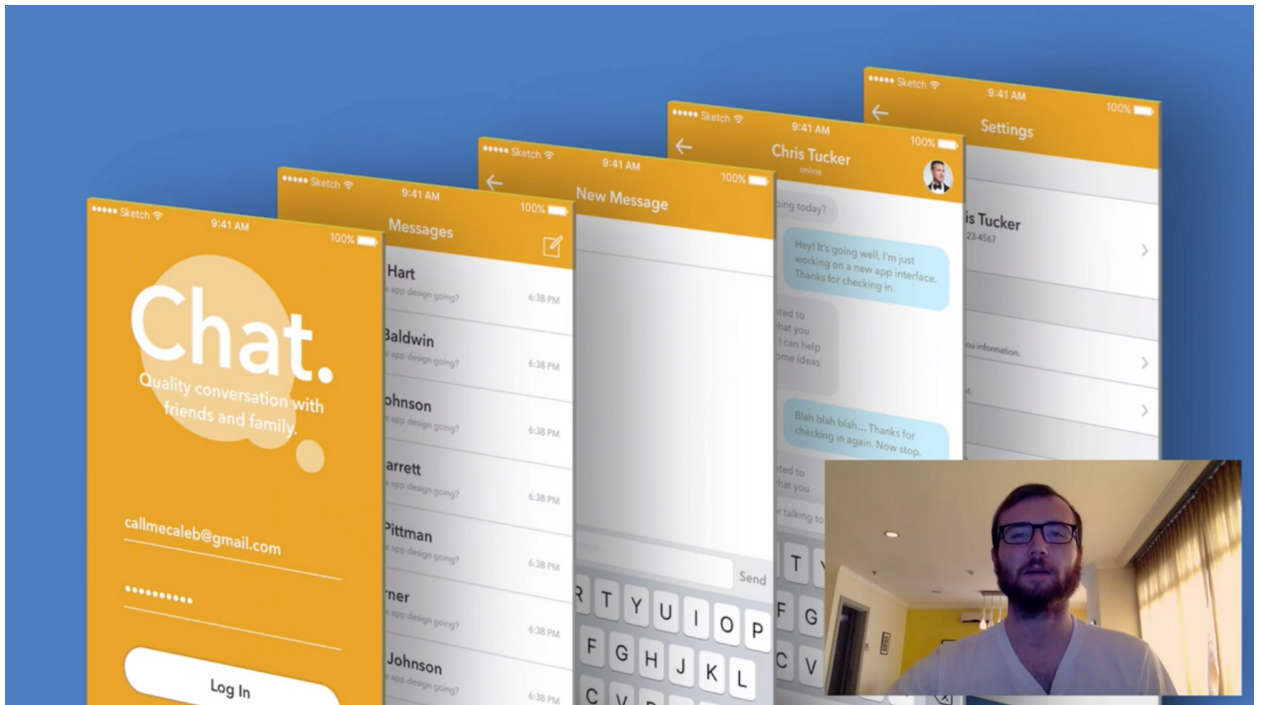
# Example List of Good Learning Resources

Eloquent JavaScript: A Modern Introduction to Programming



1. **Things you'll build:** artificial life simulation, a programming language, a platform game, a paint program, and a dynamic website.
2. **Timetable:** there is a finite number of pages, you can determine your pace when you start reading this book. It took me 15 days to finish the book.
3. **Ratings:** 4,5 stars on amazon based on 77 reviews
4. **Contact:** the author is active on [GitHub](#), [Twitter](#), and you can even email him via his [site](#).

# iOS 10 & Swift 3: From Beginner to Paid Professional



1. **Things you'll build:** social network (Facebook clone), chat app (WhatsApp clone), tutorial app (youtube clone).
2. **Timetable:** 71.5 hours of video. So, if you study hard you can become a real iOS Developer with existing working apps in 2 weeks.
3. **Ratings:** 4.5 stars based on 9,190 ratings

4. **Contact:** the creator of the course is active on [Twitter](#), but even more so—on Udemy itself—actively answering students questions. I’m sure if you dig deeper, for instance on his personal [learning platform](#), you’ll find more ways to contact him.



# Sails.js in Action



1. **Things you'll build:** video directory website, that will scrape youtube videos and display them on the page, with user authentication, management, 2 different ways to build frontend—using templates and using javascript frameworks. Complete full stack development.
2. **Timetable:** there is a finite number of pages, you can determine your pace when you start reading this book. It took me 10 days to finish the project.
3. **Ratings:** This one is from the creator of Sails.js himself—Mike McNeil! Trust me, it's brilliant.

4. **Contact:** Great way to start is Mike McNeil's [GitHub](#), his [Twitter](#) account and his [Blog](#).

# Action Exercises to complete TODAY

1. Choose ONE resource you're going to use to build ONE product.
2. Set a definite goal – in the future – one week, two weeks, three weeks – when this product will be ready. Plan your time – so that you'll get to study every day. The minimum amount of time you should be spending on this is 25 minutes a day. With the longer session once or twice a week – where you can really concentrate and get into "the flow".
3. Comfort challenge: Contact to the author BEFORE buying with the following message:

*Dear Mr./Mrs. \_\_\_\_\_,*

*my name is \_\_\_\_\_. Recently I've set the goal to learn programming. My internal goal is to build the following product: (Describe the Product/App/Service you decided on yesterday).*

*After some research I've done — I figured out using (Name of the Technology you decided on yesterday) would be the best way to build this product.*

*I've found your course/book/article/tutorial and would love to start my journey with it. My goal was to find something that would help me understand programming by building a product.*

*I have created my internal deadline: (name your deadline here). Do you think it's realistic?*

*Truly Yours,*

*Your Signature*

## Chapter 3

# Learn Programming From the Best



**Get To Know People Who Built The Technology  
Personally**

Everything around you that you call life, was made up by people that were no smarter than you.

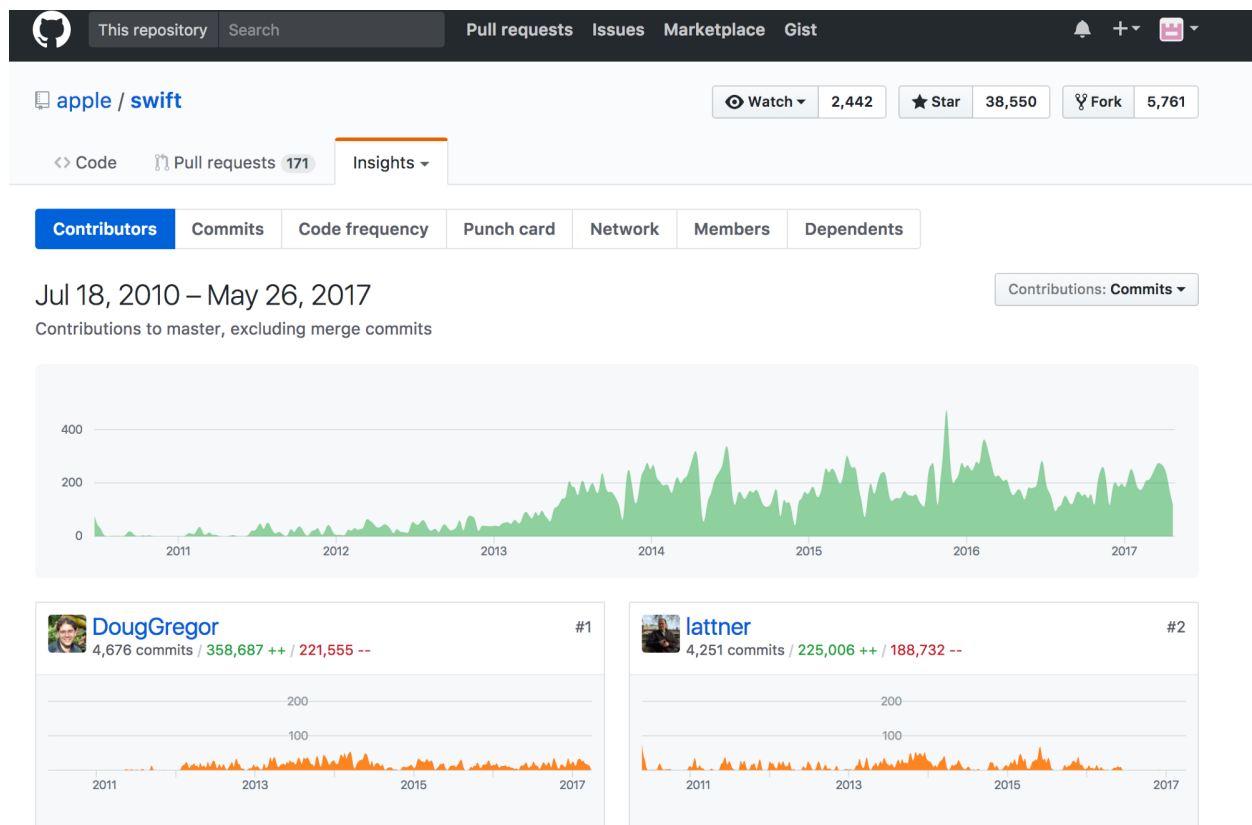
– Steve Jobs

The most challenging task I tell my students to do is, at the same time, the most rewarding one. By the end of the day you'll get to know and talk to people behind the technology / programming language you're about to learn.

It's essential to understand that even though you're just about to learn this hot new thing—people who built it are just like you. And they started exactly where you are now. The greatest thing about programming—is the community. The brightest programmers I've known—are at the same time—the humblest, easy-going and cool human beings.

So when I tell you that by the end of the day you will be essentially chatting the top programmers in the world—just take my word for it: it's not a secret club. There is no barrier to enter. And you are welcome to join.

# Go to the Source: Github



First of all—you need to go to the source. Chances are—the technology you’ve chosen—is open source. Meaning virtually anyone could take a look at how it’s created.

Your first step is to find the GitHub repository—where people are actively working on developing this technology.

Here are some GitHub repositories of hot stuff today:

[facebook/react](#): A declarative, efficient, and flexible JavaScript library for building user interfaces.

[apple/swift](#): The Swift Programming Language

[angular/angular](#): One framework. Mobile & desktop.

[vuejs/vue](#): A progressive, incrementally-adoptable JavaScript framework for building UI on the web.

[twbs/bootstrap](#): The most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web.

Open recent pull requests and just read couple of them. Essentially what happens here is that people are finding bugs and fixing them together. In the Insights tab you can discover the core contributors to the project—those are real people behind this technology. With real names. And twitter accounts. And sometimes even emails.



# Say Hi: Slack Channels / IRC / Mailing Lists

||| GITTER INTEGRATIONS APPS SIGN IN TO START TALKING

## Where communities thrive

Gitter is a chat and networking platform that helps to manage, grow and connect communities through messaging, content and discovery.

[CREATE YOUR OWN COMMUNITY](#) [EXPLORE OTHER COMMUNITIES](#)

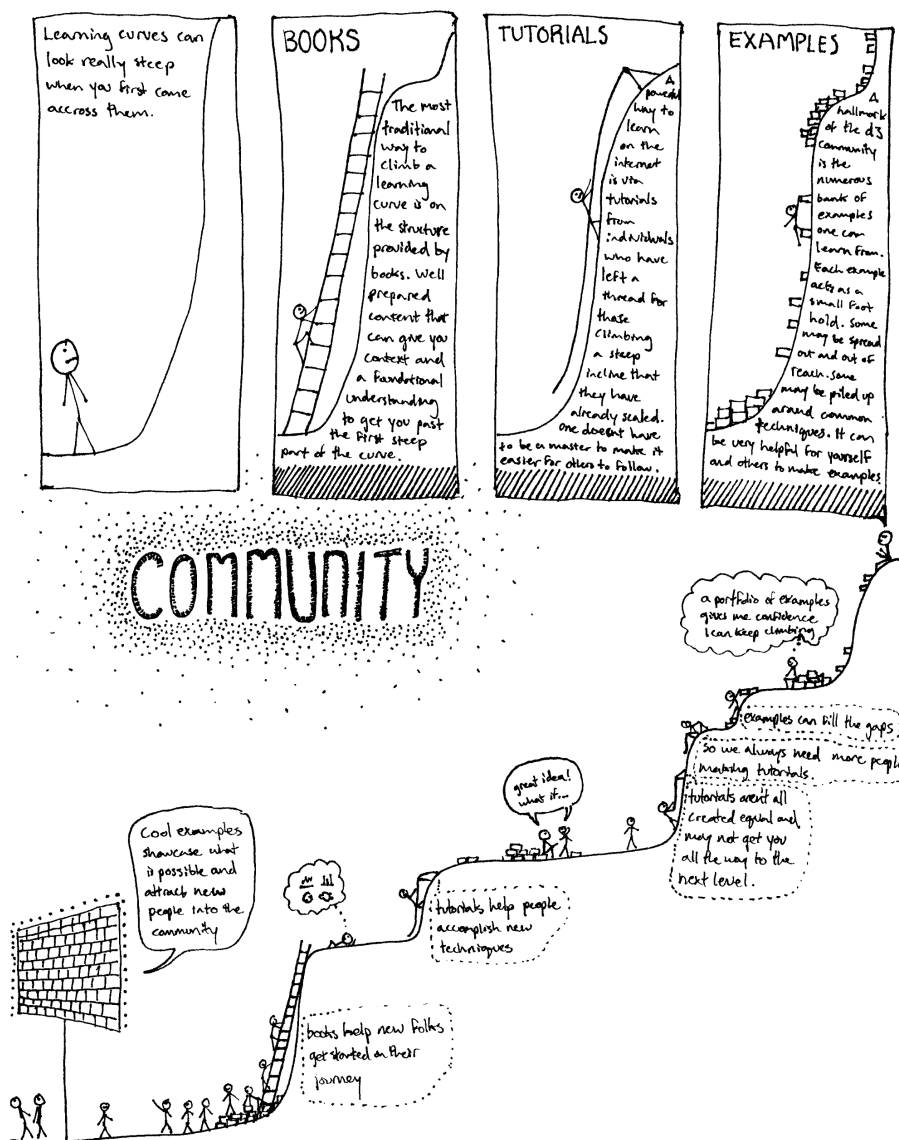
By signing up you agree to our [Terms and Conditions](#).

- Free without limits**  
Enjoy free public communities with unlimited people, message history and integrations.
- Simple to create**  
Simply create your community and start talking - no need to set up any invitation services.
- Quick to grow**  
Grow your community in no time with our easy sharing tools.
- Out in the open**  
Let your community be discovered! With Gitter, everyone can find your community in our directory and via search engines.

Second thing you want to do—is no engage with the community. To go to the place where the developers hang out, discuss the technology of your choice, ask questions.

You want to search for direct communication—not Stack Overflow or Forums. What you need is a chat-format—where you can go to—and essentially say hi.

The reason for this is simple. From the first days of your learning experience—you have to realize—there is an infinite amount of information on the subject out there. But there is no better way to get it—then to ask from the real human being.



You can only get so far by yourself. You need a place to get directions when you get stuck. You need a place to ask your questions. To show your code. To clear the things you didn't understand reading documentation.

Your essential goal is to find a mentor or many mentors—that will help you out—and lead you once you need this the most.

This is imperative that you FIND those channels—and subscribe to them.

Here is the list of some of them you might be interested in:

[Where To Get Support — React](#)

[swift.org — Communiy Communication](#)

[angular/angular — Gitter](#)

[vuejs/vue — Gitter](#)

# Grab a Beer: Local Meetups

meetup Invite Get the app

meetup

Find a Meetup

2 Meetups in your groups • 270 Meetups nearby

JavaScript within 50 miles of Köln, DE

Groups Calendar

MONDAY, MAY 29

7:00PM OKLAB KÖLN MEETUP  
**OpenData Time @ Lokal\_K**  
6 Mitglieder going

THURSDAY, JUNE 1

6:30PM WEB ENGINEERING DÜSSELDORF  
**Prometheus Monitoring System && Replacing MongoDB with PostgreSQL**  
trivago GmbH  
70 Engineers going • 5 spots left!

7:00PM COLOGNE WEB PERFORMANCE OPTIMIZATION GROUP  
**CGNwebperf #14 with Malte Lantin (Microsoft) and Christian**

All Meetups  
My Meetups & suggestions  
My Meetups  
I'm going

Today

May 2017

SU	MO	TU	WE	TH	FR	SA
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Third thing you want to do—is search for local meet ups—on the technology you’re about to study.

The reasoning behind this is simple: you get all the benefits we’ve discussed in the previous section—but then you also get some human contact. And first contacts with like-minded people—are imperative.

The best place to start your search is [meetup.com](https://www.meetup.com) or Facebook Local Groups.

No group in you area? CREATE one. Just go online and create a group. Be clear about it: you are studying technology X and you want to meet other people who are interested it.

# Action Exercises to complete TODAY:

1. Find one of the creators of the Language / Core contributors and write him personal note via Twitter:  
**"I'm just starting to learn X and wanted to thank you for creating it!"**
2. In the Slack/IRC channel you've found post following message:  
**"Hey, just starting to learn X. Choose this one to start: 'COURSE/BOOK/ARTICLE NAME'. Any suggestions how / where to start – in order to build something useful as a result – not just theory. Thanks in advance!"**
3. Register to participate in the next meeting in you local Meetup group. No group in your area? Create a group yourself!

# Conclusion

*By now you should have reset your mindset - and be thinking like a real programmer. You have all the tools and tricks at your disposal to start learning this wonderful craft.*

*It's all about continues behavior change - you need to commit to learning - and sometimes it's a challenging thing to do.*

*Would you like to receive a weekly guide, just to make sure - you stay on track?*

*The guide is free - and you can subscribe to it here.*

*I'm excited to see what you're going to build next!*