<h1>

TEN ++ WAYS TO MAKE MONEY AS A DEVELOPER

</h1>

Florin Pop

Ten Ways to Make Money as a Developer

Version 1.0 - December 2020

Table of Contents

About this book

Coding is a fantastic skill. It opens doors to exciting new opportunities. This is what it did for me and for lots of people that I know!

In this book, I'm going to share everything I have learned along my journey about monetizing my development skills. Going from freelancing to having a job to becoming a content creator, writing articles to creating videos on my YouTube channel, and much more.

In this book, you're going to learn about TEN++ ways to generate income with your development skills.

I will walk through each of these methods, explain how you can apply your coding skills to succeed, and offer tips to help you get started.

You might be familiar with some of these ways, and that's perfect! But in case you'd like to try out

something else I want to simplify the process for you as much as possible.

It took me over 7 years to learn everything I'm going to share with you in this book, and above that, for some of the chapters, I've reached out to other experts in the field and got them to share their journey and experience as well.
I'm doing all of this because I want to make sure that you'll get the most value out of the money you spent buying this book!

I want to believe that this book offers a 100x Return on Investment (ROI) because it's definitely helped me earn over six figures in the past 7 years!

Was it easy? No. Was it fast? No. It took time and effort. I spent thousands of hours learning. I love to learn which made this journey even more enjoyable (and I hope you'll love it too!).

How to read the book?

As this book has unique chapters that let you read what you are interested in without having to read sequentially, you can start reading any chapter you wish. Feel free to jump to whichever section you seem fit.

If you want to read the entire book and get the most out of it, then just simply go page by page as with a usual book.

What's up with the "++" thing?

I don't claim to have all the knowledge in the world therefore I'm always learning and this book is in a "continuous development" mode. I'm going to keep adding more information to the book as I experiment and learn new things myself.

It might be just updating a chapter with new information or it might be even adding a brand new chapter in the future.

I'll continue to work to make this book better and better! That's what the `++` stands for.

Don't worry! You'll get access to all these book updates for free as a token of appreciation for trusting me and purchasing this book in the first place.

My programming journey

I started my career as a freelancer back in 2013, kind of. I say kind of because the projects I was working on were small and it was more of an experiment than something I planned and wanted to do. Even though these projects were small, they laid the foundation for my first "real" programming job as a professional developer.

I landed my first part-time development job during college, working as an XPages developer. I don't even remember what XPages is anymore! What I can remember is that we had some JavaScript too, which was cool. Nevertheless, I stayed in that job for around 8 months until the company had to cut the development department. I was one of the two people who remained but they wanted me to switch my focus to design and because I wanted to pursue my passion for programming, I resigned.

For the next few years I worked again as a freelancer, but this time I took it more seriously and worked on some bigger projects.

I got married after finishing college and soon after that I got a high-paying job offer and worked there for

almost 2 years as a ReactJS developer and CSS ninja! (The latter was not in the job description, I added it myself to look cool).

I accepted the ReactJS position for mainly two reasons:

- *First*, because I wanted to get a feel of how it is to be working in a big company with multiple co-workers and big projects. I've learned a lot along the way and met some amazing developers!
- *Second*, the steady paycheck. We saved up some money and after 2 years I resigned and started my journey as a content creator (blogger, YouTuber and streamer).

To be honest, I didn't feel that I could be employed for the rest of my life and I wanted to try my hand at being self-employed.

Note: *Just because standard employment wasn't for me, it doesn't mean that it's not for you either. Try it out and who knows, maybe it is the right thing for you as it is for a lot of people. You never know until you try.*

Since I began my journey as a content creator I have grown a following of over 150k followers across different social media platforms (Twitter, YouTube, blog) and had the chance to help hundreds of thousands of people through my blog posts, YouTube videos, live streams and in some cases directly through DMs and emails.

Doing all of the above brought me a lot of joy and I'm very grateful for everything that has happened to me since I started this journey!

As of now I can't imagine doing something else than learning and sharing what I'm learning with those who are following me. Seems like the perfect job for me.

Speaking of sharing what I've learned, let's get into the business, shall we?

CHAPTER 1

WORKING AS A PROFESSIONAL DEVELOPER

Working for a company as an employee is the most common way to make income as a developer. Even though this eBook is focused mostly on the other ways to make money as a developer, I recommend you to at least give this one a try for a while. Working within a team of other developers on large scale software projects is an effective way to improve both your technical and non-technical skills.

Why traditional employment?

There are a couple of reasons why I would recommend it, especially when you're starting out.

- Having a stable income while you're still learning and experimenting
- Health insurance, Pension, Paid vacation
- Learning to work on larger projects with large codebases (experience operating at scale)
- Learning from co-workers (formal + informal mentorship)
- Having a structured career progression

- Company reputation can be a useful signal even after you leave (e.g. FAANG)

Preparing for the job

There are numerous paths to learn how to code. Some options can be expensive, such as college or a bootcamp, while others are available for free online. Regardless of the path you choose, it is important to dedicate time to your studies and form a habit of learning and "hacking" (daily if possible!).

I recommend starting with freeCodeCamp. It's free and a great resource for beginners to get into web development. YouTube is another resource I used daily to learn coding. When learning from YouTube (or any course) it is important to not only watch the tutorial, but to also get your hands dirty by coding the project(s) yourself.

Build, build, build! That's the best way I found which helps you understand how the code works.

Whenever there is something in a tutorial that I don't fully understand, I pause the video and start changing the code myself to see how it affects the output. That's when I get my "Eureka!" moments.

When it comes to online courses, there are, again, thousands of them. The good thing about courses is that they give you a structured format, which is easier to follow.

There isn't one "best" course out there to teach you everything you need to know. You can learn something from any course, and at the end of the day, it might just come down to how much you like the teaching style of the instructor.

In my case, I pretty much like every instructor as long as I can understand clearly what they are saying.

Another thing I usually do is follow multiple instructors on the same topic. Once you do that you start to notice a pattern of the most important concepts they teach which are usually the fundamentals.

Coding Bootcamp and College

I've heard both sides of the story when it comes to Coding Bootcamps. Some find it very useful, and others not so much. There are multiple factors when it comes to deciding on a successful Coding Bootcamp. In case you want to attend one, do your research on your preferred bootcamps, and see what others are saying

online. You might find more information about it on Reddit, Twitter, or even simply by searching for "[Bootcamp name] reviews" on Google.

On the other hand, I went to college. College taught me the basics of coding (conditional statements, loops, functions, etc.) in C/C++ and Java, and I haven't used these languages since. Nonetheless, the basics are important and I can't deny that it helped.

The web development curriculum, on the other hand, was very basic and by the time we started learning something, I already knew my way around the topic as I'd already learned it on my own.

College was not a complete waste of time but I estimate that over 90% of what I use now comes from self-teaching, and I'm not the only one who says that. Remember the "Build. Build. Build." method we discussed earlier? That remains the best way I've found to attain skills and retain knowledge.

Another benefit of college is that it can help you make friends with similar interests and it gives you extra time, which you can leverage to your advantage, to learn more and gain experience during a time of no financial pressure. Once you are out of college you are expected to contribute financially (in most cases).

It might not be the case for all colleges, but we had a lot of extra time outside of class and studying. It was during this time that I leveraged and sped up my learning process by doing side projects.

This is when I started to dabble with freelancing and it helped tremendously after I finished college. Why? Because when I graduated and started applying for jobs, I already had experience working on real projects, while the majority of my colleagues had to start from scratch. Guess who got a bigger paycheck?

This doesn't mean that all your free time in college must be spent coding 24/7 and working on your side projects. What I'm saying is that you might have some extra time that you can use between college and making friends. Use that time wisely instead of just watching funny cat videos.

Other ways to speed up the learning process

- Review other people's code. You can find countless projects on GitHub and Codepen. Take a look at the source code and try to understand how it works.

- Be consistent. Set aside some time daily (if possible) for coding. It doesn't have to be 8 hours every single day. Just pick a timeframe that works for you and stick with it.
- Share your journey on Social Media and/or a blog. Let others give you their feedback. Some people might be harsh but focus on what matters: you improving and learning how to code.
- Be part of a community of like-minded people. The journey is much more fun if you're not alone.
- Find a mentor. Having a more senior developer to periodically meet with can be an effective way to ensure you continue your skill and career progression.

Create a Portfolio

Having a great portfolio is important for anyone applying to jobs or if you are looking to find freelance clients. It is one of the first things they see about you, so you have to stand out! You know what they say: "First impressions matter".

Here are some tips I wrote a while ago in a tweet on creating a great portfolio. I'm going to keep the same checklist format as it's faster to go through:

- Have a clean, attractive, and simple portfolio
- Make it Responsive, it's 2020
- Introduce yourself and tell your story (sell yourself). Try to keep it straight to the point while making it easy for people to scan through it
- Sections you should consider having:
  - About you
  - Best projects showcase (3-6 max)
  - Skills / Technologies you know
  - Contact form - make it easy to be contacted
  - Social links / GitHub
- *DO NOT* add percentages and skill bars. What do they even mean?! *80% JavaScript*?!...
- Wanna spice things up? Have a blog
- Add a live demo for the projects (if possible) not only images
- Update the projects you build from courses, tutorials, or freeCodeCamp. A lot of people have

them already so make them unique by adding some extra features and changing the design

- It would be best if you could build the portfolio yourself; however, if design is not your strong suit you can use a template OR you can get inspiration from sites like Dribbble, Behance, or other portfolios (DO NOT copy everything!)
- Add an SSL certificate
- Buy a custom domain. It's ~$12 per year and it will make your portfolio look more professional
- Be honest, don't add a language or a library or a framework just because you watched a few tutorials on the topic
- Add specific projects in case you are applying for a specific job

Write a Resume

Even though I believe that a nice personal website accompanied by a GitHub link showcasing your work should be enough, recruiters (a good majority of them) still ask for a resume. Hopefully, this will change in the future!

Many of the tips in the portfolio section above are also applicable for writing a great resume. Additionally, make sure to:

- Keep it short and to the point
- Add relevant things to the job that you are applying to
- Make it easy on the eye by helping those who are reading your resume to be able to quickly find the information they are looking for

Most recruiters are reviewing dozens of resumes for the same position and they will appreciate a simple yet comprehensive resume showcasing the relevant skills you have for the job.

Do **not** lie in your resume! You don't want to be put in an awkward situation once you get to the interview and someone asks a question related to something you fabricated on your resume.

Applying for jobs

One thing I've noticed happening a lot (and it happened to me too when I first started job-hunting) is that people are afraid to apply for jobs unless they can

cross off all the requirements listed in that job specification

Let me tell you one thing: *You DO NOT have to know everything they put in the job specifications in order to apply*. It is ok if you learn those new things on the job. I'd even go one step further and say that it's bad if you don't have anything new to learn on the job as that is a way to quickly become obsolete. Landing a challenging job is a great way to stay relevant in the marketplace.

With that in mind, don't be afraid to apply for jobs. The "worst" that can happen is that you won't get the job. I added "worst" in quotes because that's not necessarily a bad thing but it's rather a step forward in your journey of getting a new position.

Why? Well, because once you fail an interview you can assess why you failed and thus will know how to pass it next time. Because of this, I recommend you ask the interviewer how you could have improved, as well as what went well during the interview. Accept any recommendation or constructive criticism and use it to your advantage.

Build up your core skills and then start applying to jobs. I recommend applying for several roles until you get at least one interview, and then you can apply the

strategy mentioned above. Learn from your failures and use that to improve for the next interview.

Interviewing

I remember when I failed my first interview, followed by a couple more after that. I even failed my first coding interview with TopTal but I didn't give up. They gave me a second chance after about two weeks, and I spent those weeks playing CodeFights (now CodeSignal) - this gave me the required experience to ace the next coding interview. It was an amazing feeling!

Here are a few of things you can do to prepare before an interview:

1) **Look up interview questions** related to the job position. For example, if applying for a ReactJS position search for possible ReactJS questions that might be asked during the interview (there are several resources online, for example search for "ReactJS Interview Questions" on Google). Once you find the questions, ensure that you understand the key concepts very well (props, state, components, hooks, etc) and also make sure that you have some practical experience using these concepts. This makes it easier when it comes to

explaining the concept in case an interviewer asks a question about it.

2) **Practice Coding Challenges** because you might get a few of them during the coding interview, depending on the company. There are several websites you can use to practice: CodeWars, HackerRank, LeetCode, and CodeSignal, to name a few. I personally like CodeWars as I prefer their UI, but other than that, any of the above can work well.

3) **During the actual Coding Interview, ask as many questions as you can** in order to make sure that you understand the problem correctly. Also, explain your entire thought process out loud. You can get visual feedback from the interviewer in case you are slipping off a bit, which is good to get you back on track.

Last but not least, **be confident**! It makes a huge difference in the entire interview process.

The mental trick I use to feel more confident during the interview is to not allow myself to think that the job I'm applying for is one of those "all-or-nothing" situations ("You either get this job or you're busted!"). That's a scary thought and I want to avoid it at all cost. There are other jobs out there that I can get so it's fine if I won't get this one.

Use LinkedIn to get job offers

Several years ago when I was on a vacation in Germany, out of nowhere I started sending LinkedIn invitations to a lot of people in order to get my connections to count up to 500. That way my profile will show that I have 500+ connections and for some reason, I thought that would make me look “cool”.

Well, I don’t know how cool I was with 500+ connections but I know one thing. Over time those connections I got on LinkedIn managed to put me in front of several recruiters, and it got me dozens of job offers over the years. Not all of them were worthy and some of them were even spammy, but I got a few great opportunities among those. One of them ended up in a high paying job at a company I worked at for almost 2 years.

With that in mind, I highly suggest creating a LinkedIn profile and spending some time making it look great. You never know when this will lead to your next big opportunity.

CHAPTER 2

FREELANCING

Disclaimer: This chapter was written in collaboration with Kyle Prinsloo. Kyle is an expert in freelancing, having worked as a freelancer for many years and helped hundreds of others start freelancing.

How to Start Freelancing

There's a lot to get into, but first, let's discuss expectations.

Some people think freelancing is easy, but for most people, it takes time to get good results. You may have kids, a spouse, a lack of resources, no time, a stressful job...

We will get into a few strategic approaches to help you overall, but either way, you need to push through and work towards your goal of earning a side or full-time income with freelancing.

Let's get into the basics:

What do I need to know to start freelancing?

You need to know how to create websites. When you know how to do this well, you are ready to start freelancing.

That could be HTML and CSS or a CMS like WordPress or WebFlow – the HOW doesn't matter. What matters is that you just know how to create websites.

Do I have to register a business?

At some point, yes. It's very beneficial to do this. Every country has different rules and regulations, so if possible, do speak to a professional regarding this.

My (non-legal opinion) is:

- If you *have money,* then pay a Lawyer/Accountant to register a business from the start.
- If you *don't have money*, just get a few clients in your personal capacity, then pay a Lawyer/Accountant to register a business.

Why would you want to freelance?

Here's the nugget overview of why you should freelance:

- You're not getting paid enough
- You want more control

- You want more freedom
- You want an additional income
- You want more time with your family
- You don't enjoy your current job

... then it's a no-brainer. You owe it to your future self to try freelancing.

What services can you offer as a freelance developer?

It's easy to offer once-off services like creating a website, but can you do one better? Yes.

How?

Offer web design services as well as marketing services. Yes, you can offer maintenance and hosting packages, but most of the income lies in monthly marketing services.

Here's why I recommend offering web design and marketing services:

- It helps you avoid the "Valley Incomes" of one-month earning a great salary and the next month, having no client, so your income goes to zero.

- You're providing a full-service package to clients.

The best solution to this is a monthly recurring income, and you can get this by offering marketing services too.

What type of marketing services?

- Content Writing
- SEO
- AdWords Management
- Social Media Management
- Social Media Ad Management
- Email Marketing
- Designs

Don't know marketing? You can either learn it or outsource it

How do you outsource? Look for someone on UpWork or Hubstaff Talent.

Outsourcing is what helped me grow to a full-time team of 5 people with a few contractors.

Let me illustrate this point with 2 scenarios:

Scenario 1:

You want to earn $2,000 on the side each month.

- Get 4 clients paying you $500 per month.
- Get 2 clients paying you $1,000 per month.
- Get 1 client paying you $2,000 per month.

Scenario 2:

You want to earn $5,000 each month.

- Get 5 clients paying you $1,000 per month.
- Get 2 clients paying you $2,500 per month.
- Get 1 client paying you $5,000 per month.

We'll get into the pricing section later on, this is just an overview.

Now that the fundamentals are out of the way, let's turn up the heat.

Where to Start Freelancing

My first recommendation is your own portfolio website *(we'll get into that later)*, but there are plenty of platforms you can try as well.

Now, I know it's not for everyone, but it's worth a try and it does work for many freelancers as it worked for me: **Fiverr**.

I offered a Web Design Analysis Report where I critiqued their website for $5.

Then I upsold them to a $100 Wireframe.

Then I upsold them to a $1,000 Website.

Then I upsold them to a Monthly Marketing Retainer.

One client resulted in 8 websites, a monthly marketing retainer of $1,000, and referral clients! And that was when everyone told me "it's too saturated." So don't worry about what you hear. Try this strategic approach and see what happens.

Platforms to freelance on

- Fiverr
- UpWork
- HubStaff Talent
- EmployRemotely
- RemoteOK
- TopTal

Your own portfolio website

A portfolio website should showcase who you are and how you can help a business grow. Stay away from the % skills bars and keep it simple.

Haven't decided on a niche yet? No problem. Your heading could be:

"Focus on running your business, while I focus on growing your business."

How to Charge for a Website

To get straight to it, these are the main ways to charge for a website:

- Hourly
- Project-based
- Value-based

Hourly pricing

Take your desired annual income and divide it by the hours you'd like to work in a year (keeping in mind vacation days).

It might look something like this:

Working **37.5 hours** a week: **$75,000** (desired income) / 1,950 hours = **$38.46** per hour

So you might charge **$45 per hour** in this case.

Project-based

This is when you estimate your hours worked and add a buffer to it. Might look something like this:

You estimate the project will take you 50 hours. **$45 (hourly rate)** x **50 hours** = **$2,250**

So you might charge **$3,000** (includes a $750 'buffer' amount).

Value-based

This is when you charge your fee based on the potential return the business can make after using your services.

This is my preferred method for pricing projects.

Before I get into the details, we need to understand that you're not just selling a "website" – you're selling a solution to increase business sales. Once you understand this, you'll understand the value behind it.

Here's why I like Value-based pricing:

- You don't sell hours (like everyone else does) – you sell results (or the potential results).
- There's an incentive to stay up to date with the latest technologies, software, or tools, in order to make your job easier and to become more efficient.
- It allows you to create something amazing and not to worry about going over the client's desired budget.

- There are no hidden financial surprises to clients. You take all the risk in delivering the project within the total cost you've informed the client about.
- You can work with less clients and provide a better service because you are often earning significantly more.

On the other hand, here's why I don't like Hourly billing:

- It discourages efficiency. It's in your best interest to drag on a project. Why finish it in one hour when you can bill for ten? Why buy a plugin or code snippet to save time when you can bill for that time?
- Most client disagreements are always disputes about payment. Proving you were working or why you took so long, etc. The admin hassle and stress, even with tracking software, it's not worth it.

Yes, it does work for a lot of people (mostly only those who charge large hourly rates), but for the most part, I prefer Value-based pricing.

Here's how to determine a value-based price:

The main thing you need to do is to figure out how much the site is worth to the business.

Example: A business sells 3D Printers and they want a website.

The two main things you need to know is:

1. How much the average 3D printer sells for
2. How many 3D printers the business sells every month on average

With this information, and after looking to see if I can make improvements (from their website to their marketing) I'd be able to figure out how much to charge for the project.

So if the business sells an average of **ten** 3D printers at an average of **$2,000 each** per month (**$20k** sales **per month**) and after calculating that I could potentially increase the sales by **30% every month** (based on experience or research of improvements), it equals to an extra three sales per month (or **$6,000**).

Now it's important to present this potential return over 12 months to price anchor your fee.

I then calculate that even if we work on just 2 extra sales per month, their sales increase adds up to an extra **$48,000 per year**.

Then you take around 10% of this value and that is your price to work from.

Therefore, spending around **$5,000** once-off for the website to potentially increase sales by almost **$50,000** in one year is a no-brainer.

This pricing principle applies to marketing services as well.

Figure out what they are currently spending per month, how their results are, and base your pricing off of this and how you can help them improve sales.

I've often started on a low monthly marketing fee and as the client gets results, increase it and offer more services.

How to send website proposals

What is a proposal template?

It's a document showing that you understand the project, the business needs, and that you can provide the solution for them.

It should include:

- Intro/Cover Page
- Project Overview (what they would like)
- Quote Options (always provide 3 options)
- Terms of Agreement (payment and delivery terms)

Here are 10 questions to ask your prospective client:

1. What does your business do?
2. Who is your ideal target market?
3. What is the desired action you would like a website visitor to do?
4. What makes you unique from your competitors?
5. What is an average client/customer worth to your business?

6. How do you currently get new clients/customers?
7. How many clients/sales do you get on an average month?
8. Do you have professional images and brand identity?
9. When would you like the project to be completed?
10. Is there anything else you would like me to know about?

🔥 **Bonus Tip:**

If you feel like the client has a very limited budget, you can ask them:

"Do you have a budget set aside for this project and is it at least over $xxx (insert your absolute minimum desired amount)?"

This will then justify further questioning, and help you decide whether you should bother sending a proposal through.

Once you have all this information, you'll be able to create the perfect proposal.

To save space in this section, you can download a free copy of the Proposal Template (and how to write it) here.

How to get clients

It's easy to get clients, but you want the **right** clients. Clients who value your input and pay you what you're worth. One of the best ways to do this is to niche down.

In contrast to targeting everyone who needs a website, niching down means you'll create websites for a specific niche like health and fitness, coffee shops, online courses, etc.

Although niching down is not a requirement, I do find it more beneficial in the long-term.

Successful examples and inspiration:

- SmileMarketing.com – Web Design and Marketing for Dentists
- PlumberSEO.net – Web Design and Marketing for Plumbers
- PaperStreet.com – Web Design and Marketing for Lawyers

- OuterBox – Web Design and Marketing for eCommerce

It's good to start out as a generalist (working with any clients), but at some point, do consider niching down based on interest, passion, enjoyment, opportunity, and location.

With that said, let's move on to some tips on how you can get clients.

Besides telling your friends and family you can create websites for them, these are some of the best outreach strategies to try:

- Google Ads
- Facebook Groups
- LinkedIn Ads
- Your Own Platforms
- Freelance Platforms
- Manual Outreach

Google Ads

Google AdWords is an effective tool to get your first client. It's the ads that you see on top of the search results:

You pay every time your ad is clicked on.

The average Cost-Per-Click is **$1 - $3+** depending on the industry competitiveness.

If you're selling a website for $500, would you be willing to spend $50 to close a deal with a client whom you can potentially upsell with marketing services?

Absolutely. What about $100? $200? $300

You have to decide how much you're willing to pay in relation to what you can potentially earn.

3 search terms you should consider to bid on are:

- "web design agency"
- "web design services"
- "freelance developer"
- "freelance web designer"

🔥 **Bonus Tip:**

Make sure to add a few negative keywords like: "courses", "ideas", "inspiration", "jobs", "learn" (so your ads don't show up with these words) and remember to target your desired locations only.

To succeed in Google AdWords, you need to target the right audience, have a high converting landing page, structure the campaign properly, and more.

The learning curve is steep, especially to learn it well, but it's worth your time. I like it a lot because it not only helps you get clients, but you can also offer it as a service to your clients, so you get a monthly retainer income.

My top learning recommendation is the Google Ads Course by Isaac Rudensky on Udemy.

Facebook Groups

Not a lot of people realize how powerful Facebook groups can be as marketing tools. And that's partly what makes it so appealing.

There are two ways to use Facebook groups to sell your services:

1. Join existing Facebook groups of your target clients.
2. Create your own Facebook group and add your target clients as members.

Join Existing Facebook Groups

The idea here is joining a Facebook group where your target clients are members of and adding value by commenting, answering questions related to websites or your other services, giving suggestions, etc.

For example, if your target client is business coaches, you can search for groups related to this niche. Join as many of them as you can, introduce yourself, and start adding value by engaging with them. If you do this consistently for a few months, they will recognize you as someone who is helpful and can add value to their community.

Avoid directly selling your services the first time or every single time you engage because it will come off as spammy. You have to give value first before they trust you enough to want to know more about your services.

Once you become a trusted member, then you can introduce your services to them. Sometimes you won't even have to initiate it because as you help people out, they will begin to trust you, become curious, and ask you for help.

It sounds so simple, but it works and you'll get a good amount of client leads.

Create Your Own Facebook Group

If there are no existing Facebook groups with your specific target clients as members or if your niche is too specific (e.g. limited to an area or a hyper-specialized niche), then create your own Facebook group and add them as members.

Creating your own Facebook group helps you build lasting relationships with targeted clients and positions you as an expert in the industry.

For example, if you're targeting business coaches in "X" city, add them to the group and begin adding value.

Write a post about clever marketing tips for business coaches or the top mistakes in websites of business coaches.

Here are some tips for your group:

- Interact with your members. Don't just comment or post only when you need something from them. It's a community, and people want to build relationships.
- Keep things interesting. Maybe you can come up with a simple contest where they can win exciting prizes like a free landing page template. It can also be a chance to promote your services.

- Establish group rules. Forbid foul language or hateful speech.
- Build a relationship. Once you have a relationship with them, you can send a message or post occasionally to the group about how your services can help grow their business.
- Be yourself. People can tell if someone's not being genuine. Share your stories and let your personality show.

Don't be overwhelmed with the thought of creating and maintaining your own Facebook group. You don't need 1,000 members here. 100 is great. As long as it's targeted, you'll get results.

It may take months to get paying clients, but it will get better and more effective the longer you do it. Just focus on converting 2-3% of the members to paying clients and focus on consistently growing the group.

I've had a lot of students do incredibly well with this strategy.

LinkedIn Ads

If you know who your ideal client is, LinkedIn has a lot of potential for you to get clients. One of the most

effective ways to get the most out of it is to use their ads.

Their targeting is accurate and it's where you get access directly to professionals, so if you can create an ad that gets their attention, you're on to a winner.

We use LinkedIn to get leads and the results have been very good.

Here is an overview of my suggestions:

- Use LinkedIn Post Ads
- Try a lead-form ad and a landing-page ad
- Create a funny ad, question ad, and informative ad
- Run it for a month, then use the best performing ad

Another often neglected strategy is to connect with 20-30 profiles of your ideal client every day.

Before you know it, you'll have 500+ connections and you can add value to your posts, then eventually reach out to them to offer your services.

The other approach is to use LinkedIn InMail (Message) Ads and target your ideal client.

Try many different messages, but one message could be:

> *Hi [name],*
>
> *I was wondering if you knew your website isn't converting as good as it should be?*
>
> *There are various things you can do to increase your leads and ultimately, increase your sales.*
>
> *I'd love to chat and provide more solutions for you (at no charge).*
>
> *Schedule a call here or reply with any questions.*
>
> *Regards,*
>
> *[your name]*

Your Own Platform

Having a blog and/or YouTube channel is a non-negotiable in my view.

Learn **SEO** (to rank it on Google or on YouTube) and create valuable content your target client would find helpful.

Then promote it.

Freelancing Platforms

This is not my preference, but it does work for some people (as it did for Florin).

Try it, and if/where possible, try the Fiverr strategy I mentioned previously.

Manual Outreach

This strategy involves more work than the others, but there's no getting around it. Look for businesses in your niche on:

- Yellow pages
- Google/Google maps
- Local online business directories
- Facebook (search your niche keywords)
- Local print media like newspapers and flyers

Let's say you're targeting Business Coaches in Australia. On Facebook, navigate to the Pages section and search for "business coaches Australia". Then go through relevant Pages and look for the following:

1. Do they have a website? If yes, can you improve it?
2. Do they show up on Google for their business?

If you answer no to both questions, that means they can benefit from your services. Now it’s time for you to sell it to them.

It's a numbers game. The more prospective clients you reach out to, the more chance you have of succeeding.

You can't contact 7 businesses and say it doesn't work.

There are a few ways to approach and contact a business:

- Email (use a tool like FindThatLead.com)
- Phone
- Social Media: LinkedIn Connections (connect with your ideal clients), Twitter, Instagram, Facebook, etc
- Website Contact Form

A general message to send could be the one I mentioned above.

For Instagram, you can search for a relevant hashtag, view their profile, click on their website, and if it’s not mobile responsive or it’s poorly designed, you should DM them. Rinse and repeat.

Do this every day and contact as many businesses as possible, and eventually, you'll get your first few clients and scale from there.

I hope this freelancing section helped you. If you're interested in learning a lot more on this topic, invest in your freelancing career by getting The Complete Freelancing Bundle (use coupon POP25 for 25% off).

(This page intentionally left blank, for print)

CHAPTER 3

BLOGGING

Why start blogging?

There are multiple reasons why having a blog is a great idea and why we can see more developers going down this road. For some, it might be daunting at first but let's go over some of the benefits and you might change your way of thinking about it.

Improve your own skills

Whenever you write about a topic, you will most likely find yourself in need to do some research as you want your article to be great so that's the perfect moment to learn the topic more in-depth. They say that *"the best way to learn something is to teach it to others"* and that's exactly how it works when you're blogging.

It also encourages you to step up your game and stay up-to-date with what happens in your niche. It might just be the push you need to stay relevant and even be one step ahead of the crowd.
Guess who's the one that's going to be contacted when the next new, fancy startup which is working with the newest technology, will be hiring? Well... it could be you!

Market yourself

Blogging can set you up as an expert in your field and it allows you to get in front of various networking and business opportunities that you might not find otherwise. This can help you find your next big opportunity, be it either a great job or even the start of a successful collaboration with someone.

Blogging helped me meet some amazing developers. With some of them, I'm currently working on some exciting projects. This would have not been possible if I wouldn't start blogging.

Extra $$$

Although blogging might not be paying your bills when you're starting out (spoiler alert), it can definitely turn into a good source of income down the road. Consistency is the key for that to happen.

We'll talk more about the ways to make money as a blogger in a bit.

How to start blogging?

First, let's clarify one important thing. Do you have to be an expert programmer to start blogging? **No!**

Even if you just started out on your journey there are people who haven't started their journey yet. They can definitely learn from the things you are writing. Everyone started at some point, right?

Also, don't worry if the topic you want to write about was already covered. You are unique and you have your own unique perspective. Some people will love your way of writing and explaining concepts while others won't, but that's fine! We can't please everybody!

As long as you're helping at least one person, I call that a win. With this out of the way, let's see what you need to start blogging.

Pick a niche/topic

When starting out it's highly recommended to focus on one niche. That way you can build an audience of people who are interested in the topic and they will more likely start to perceive you as an expert in the field.

You should not jump from writing C++ tutorials to CSS hacks mixed with some articles about Go. At least not in the beginning as that might just confuse people.

Also, people who would read your CSS hacks probably wouldn't be interested in reading about C++ which will slow down your momentum.

As soon as your blog gets more popular you can start diversifying your content. Then you can try out all the "crazy" ideas mentioned above.

How to pick a niche

In order to find that out, you might want to answer a couple of questions:

What are you good at?
What are you passionate about?
What do you want to learn more about?

These are some of the questions you can ask yourself when picking a niche.

It wouldn't hurt if you did market research in order to discover what people would like to learn. Then you can find the "winning combination" of what people want to read about and what you want to write about.

Although if I would have to pick between the two, I would more likely go with the niche I'm passionate about. Why?

Well, I find it easier to write a topic I'm excited about and it's more rewarding. Also, even though at the beginning it might look like people don't care about a topic, I'm pretty sure that you can find an audience for it. You might just have to search a little harder.

How to create a blog

Once you have a niche in mind, it's time to create your blog. When it comes to creating a blog, there are so many ways you can approach this. As a developer, you can create your own blog from scratch. This is what I did.

I created my own blog using **Gatsby** which is a free, open-source framework built on top of **ReactJS**. It's a powerful static site generator and on top of that, it's also blazing fast!

It has a wide range of plugins that can help you add all kinds of features to your blog. Markdown support, analytics, meta tags for SEO, image optimization, just to name a few. Because it generates a static website you can simply create a GitHub repository to push your files there and then use **Netlify** to deploy your website without having to purchase a server to host your files on.

This is what I did and ended up saving the $120 which I was paying for hosting per year and now I don't have to worry about maintaining the server either. That's a win-win. All I have to do now whenever I want to publish a new article is to create a new markdown file, write my post, and then push it to GitHub and it's done! In a few minutes, the live website is built by Netlify and people can start reading the article. As simple as that.

Other ways to create a blog

If you don't want to build your blog from scratch you can use some existing platforms like **WordPress** or **Ghost**.

WordPress has a gazillion themes and plugins available online. Some of them are free but if you want something good you might need to pay for it. I personally was never a big fan of WordPress but I can't deny the fact that it might make blogging easier, especially for those who don't know a lot of coding.

Blog on existing platforms

There is also another option. This option is getting more popular nowadays. You can start blogging right away

using an already existing platform. All you need is to create an account and you're good to go.

Some of these platforms are:

- Dev.to
- Hashnode
- FreeCodeCamp

Dev.to keeps growing in popularity. Here you can find all sorts of dev related articles and it's used by almost 500,000 developers. It's a great way to get your first articles in front of an already existing audience.

Hashnode allows you to start your personal dev blog very fast. You can publish articles directly on your own domain while also being promoted in their community. It is like a platform of blogs with the added benefit of helping you find your audience as each article you publish on your domain gets shared on their homepage.

FreeCodeCamp is probably the biggest tech blog out there. It gets millions of views monthly and it can help jumpstart your blogging career as it did mine. Keep in mind that their curators and editors are working hard to provide the best content they can for their audience and because of that you need to get accepted before

publishing an article on freeCodeCamp. There's a form you have to submit and you need to have some previous work examples to showcase. But once you are in, the sky's the limit!

Finding an audience

If you are starting out on your own blog you need a way to find an audience. Ranking on Google is great but it requires time. It might take up to 6-12 months until you can see a steady flow of traffic coming from Google. There are ways to help speed up the process.

One way is to share your article on Social Media platforms like **Twitter**, **Linkedin,** and even **Reddit** (be careful of Reddit, there are a lot of trolls lurking there). This is how I started and although I didn't get a lot of attention at the beginning, I slowly built an audience there and more people started visiting my blog.

Another way to get in front of an audience is to **cross-post** your articles to other blogging platforms like **Dev.to** or **Hashnode** or to write content for **FreeCodeCamp** and let people know that you have your own blog they can visit. It will take a little time to find and grow a community around your blog but the payoff is definitely worth it!

Growing your audience

Now you have a blog and you also get visitors, GREAT! But there are still some important things you should do in order to increase the chances of having a successful blog.

Have a schedule

This helps on two fronts.

First, it helps you build a habit which will make it easier to get into a "productive state" and you'll end up producing content easier and faster.

Secondly, this will act as a reminder for your audience to check out your blog because there should be a new article on day X. This should turn into "free traffic" as some people will visit your blog without you having to do something extra for it.

Find out what schedule works best for you *and* your audience and try to stick to it. It can be once a week or twice a week or whatever you find to be working for you. Start by testing it and see how it goes.

Create an email newsletter

There are a lot of visitors you might lose forever if you don't have a way (a reminder) to contact them to let them know that you have a new article on your blog. That's what you can do by sending an email to your email subscribers.

When it comes to platforms, I've used Mailchimp and Convertkit. Started with Mailchimp but I moved to Convertkit now. Try them out and see which one works best for you.

Having people subscribe to your newsletter is also an important step in building a good relationship with them. A relationship that can bring a lot of benefits down the road.

A following on Social Media is not a guarantee that you will get heaps of visits to your articles because everyone wants to keep people on their own platform. Only a fraction of your followers will even see your articles. This is where having an email list shines (one of the reasons). People who already subscribed to your mailing list are more likely to want to read your articles because they already gave you their consent.

Also, by sending out an email you can drive sales to your products and/or services or if you don't have those yet, you can promote affiliate products. This is a great way to start earning money through your blog.

How to make money blogging

There are multiple ways you can monetize your blog:

- Ads
- Affiliate
- Sponsored Posts
- Promoting your own products/services
- Getting paid to write for other publications

Keep in mind that it is a slow grind. You might not see a lot of progress in the beginning, but consistency is key!

I still remember when I first got contacted to write a post for a publication. They offered me **$35 USD**. After a month or so another publication contacted me. This time they offered **$100**. Then **$200** and then **$350**.

Slow grind. Keep pushing and you'll see results.

Have a schedule. For example: **2-3 articles / week**. Something simple enough that you can manage without burning out.

You don't want to be pushing 7 articles a week for a month and then suddenly go to 0 because you burned out, but rather try keeping it at 2 articles per week for a year.

Ads

I'm sure you've seen them. They're pretty much everywhere and for a reason. It provides a bit of extra income.

A bit? Extra? Why are you saying that?"

, I wouldn't rely on ads as my main source of income from a blog. You need a high viewership in order to make a decent amount of money. In my experience, you'll be getting between **$1-2** for **1000** impressions.

An **impression** is when an ad is fetched from its source and is countable. So basically whenever someone sees an ad. Now let's say you want to make **$1000** a month from blogging. That means you need between **500k-1M** impressions. This is not an easy feat.

There are also the **PPC ads** (Pay-per-click) which can pay between **$0.25-$1** per click. These types of ads pay a bit better but not a lot of people click on them.

On my blog, the two ad types mentioned above pay almost the same. Half for clicks and half for impressions.

Please keep in mind that ads might be obtrusive so don't overdo it. Don't sacrifice the viewer's experience in exchange for some money, might not be worth it.

In case you want to try out ads, I'm using **CarbonAds** and I'm pretty happy with them. They provide ads specifically for developers and/or designers.

Before applying for them make sure you have around **10k Monthly Pageviews** on your blog. Last time I checked this was the amount you had to have in order to get accepted.

Affiliates

You can make money by promoting other people's products and/or services and in exchange, they'll give you a commission. The commission can range a lot. From **5%** up to **60%** or even more. It really depends on the service or the product. This can also be negotiated

with the owner(s). If you manage to bring in a lot of sales you can ask them to raise your commission.

"*What should I promote?*"

Find products and/or services that you used and enjoyed before and send the owners an email. Ask if they would like to collaborate and see if they have an affiliate program.

You can also contact individual creators as they might have an affiliate program too. That's what I did and it worked out well.

After they set you up with an affiliate link you can find a spot on your blog where to place it. In the sidebar, above-the-fold, after the article. Test and see what works best for you. You can use a URL shortener like bit.ly to track the number of clicks and based on that decide which is the best spot to place them.

I'd also suggest having a *Disclaimer* on your blog somewhere, letting your readers know that some of the links on your blog might be an affiliate and that you'll earn a commission at no additional cost for them. Let them know that this is a way for you to continue to work on your blog to provide great content.

Sponsored Posts

When your audience grows you'll notice that you'll get contacted by different companies to write about their products in your articles. There are "scammy" looking products out there so make sure you check them out really well before accepting anything.

Your test should be: *"Do I like this product? Would I use it?"*

If the answer is not 100% yes then it's a no.

Remember: you are in for the long run which means that keeping your audience is more important than some quick bucks you could make. Appreciate your audience and they will appreciate you in return..

When it comes to: *"How much should I charge for a sponsored post?"*. It depends.

It depends on the number of readers you have. It depends on the topic. It depends on the type of collaboration you have with the company that wants to sponsor a post.

Do they want to sponsor only one post? Charge more.
Do they want to sponsor a series of posts? Charge less per post as you'll get a bigger sum overall.

At the end of the day, it's all negotiable. There isn't a fixed amount I can give you. Do what feels right for you.

Promoting your own products/services

This one goes without saying, considering that we talked about affiliates and sponsored posts. Once you created your own products you might want to drop some of the affiliates and promote your own products instead.

Keeping 100% of the income from your product(s) is better than the 50% commission you might get from affiliate products.

Affiliates are great when you are starting out but the goal should be to have your own products, sooner or later. That's when the "real money" comes in.

"Real money" because this is your product.

By now you should probably have an audience that knows and trusts you and you also know what they need so the product would most likely be tailored for them.

A great way to sell your own products is through **Gumroad**. This is what I'm using to sell this ebook and

it's great! They take care of many things like payment, VAT, sending the product once someone bought it, and much more.

We'll be talking more about having and creating your own products in the **Digital Products** section.

Getting paid to write for other publications

After your articles start getting some traction you might soon be contacted by different publications that would like you to write for them.

These publications are always looking for new content as their income is based on that so they will often reach out to people to write for them.

You can leverage this to:

- Get in front of a new audience
- Get even more experience for writing for a publication as they might have their own "methods"
- Make some money

Again, the amount you can be paid will depend on the publication, the topic, your experience, etc. Do what works best for you. Keep in mind that money isn't the

only thing you might be getting. The opportunity to reach more people is valuable too as some of those readers might just stick with you longer, which is a great thing!

When it comes to payment, you can start by charging less at the beginning ($200-300) and as you're getting more experience, grow your prices.

(This page intentionally left blank, for print)

CHAPTER 4

RUNNING A YOUTUBE CHANNEL

Why YouTube?

YouTube is the second-largest search engine in the world, after Google, which makes it a great place to teach people and grow an audience online.

According to a poll I did on Twitter, over 90% of developers have used YouTube to learn something related to programming. Some use it to learn programming from scratch, others use it to learn something about a new feature while others use it to find solutions for their programming issues.

This is great news. This means that whenever you push a video to YouTube there is a good chance that people will find and learn from you. People who never heard of you can discover your work through YouTube.

Here are some other interesting statistics about YouTube. It has:

- 2 billion monthly active users (source)
- 1 billion hours of watch time per day (source)

And it keeps on growing.

Another reason why you should consider YouTube is because it gives you the freedom to create whatever content you want, whenever you want it. Do you like

Web Development? Mobile App Development? Game Development? Great! You can start in any of these niches.

Do you want to upload once, or twice, or 7 times per week? Perfect. You can pick the schedule which works best for you. Just keep in mind that YouTube is a long game, a marathon, not a sprint! You need to be consistent, so be sure to set your mindset right.

You might not notice any big traction in your first year or two depending on how much work you are willing to put in. But, if you manage to grind through the tougher beginning, YouTube can become a valuable source of income in the long term. Some would even say *passive income*, although you still need to put in a little bit of work after you reach that point in order to keep it going.

YouTube allows you to gather the rewards of your hard work months and even years after you created a video.

How I started on YouTube

For a long time, I thought about starting a YouTube channel but something was holding me back. That something was the fact that I wasn't a native English speaker. I didn't know how people would react to

hearing my funny Romanian-English accent. Just the thought of it was terrifying...

one day I found a video on youtube with the following title:
"**MAKE 100 CRAPPY YOUTUBE VIDEOS (Like Mr. Beast and PewDiePie Did!)"** by Roberto Blake.

Mr. Beast? PewDiePie? Wow... These are some huge names, right? What does he mean by "Crappy videos"? So I went to check out Mr. Beast's first videos and Roberto was right. They were indeed "crappy"!

Audio quality. Video quality. Storytelling. All of them were pretty bad compared to what they are now. So I told myself: *"If they could do it, so could I!"*

But I still had my fear because of my accent. How did I overcome that?

I asked on Twitter to see what people's opinions were regarding it and to my surprise, they were very supportive! The conclusion was that it would be all fine as long as I am able to transmit the message properly. And as a bonus, some people even said that having an accent will make you more unique.

And that's how I started. I created my first video on 7th November 2019 spending 8 hours recording, editing,

and publishing the 4-minute video. But it was worth it! I had my first video up on YouTube and my journey to create 100 crappy videos began!

Fast forward to today, after one year (at the time I'm writing this), the channel has over **75,000** subscribers and it brought me a little bit over **$450** in the last 28 days. (We're going to talk more about monetizing a youtube channel below.)

So, as a small conclusion to this section, if I could do it despite my fear, I'm very positive that you can do it too!

Starting your YouTube channel

In order to start a YouTube channel you might want to do a couple of things, but don't overthink it. It's much more important to start than having everything figured out from the beginning.

The best approach is to start and then try to improve a little bit in every video. Practice makes it better!

Picking a niche

As we discussed in the **Blogging** section, it's important to have a specific niche (or at most 2) when starting

out. Do not jump around topics as you will only confuse your audience.

Create a series of videos on the selected niche. This will slowly set you up as an expert in the field and YouTube will start to promote your videos to those who are seeking similar content.

Recording Software

While making a coding video you might want to record your screen. (I know, it's obvious).

The software I use and recommend is **OBS**. It is free and very simple to get up and running. You just have to create a scene and add a Display Capture as a Source. This will record your entire screen and the mouse.

The good thing with OBS is that you can add all sorts of sources and it's great for streaming too, but one thing you might also want to add to your videos is a webcam source. Easily done with OBS.

You might be asking: *"Should I also have a webcam and record myself?"* and the answer is... only if you want it to. It might create a more personal connection with your viewers but there are tons of videos in which the creator never appears so it's not a must.

Also, you can add your microphone as a source in OBS to record your voice. Some creators like to do voice over post-recording while editing the video. I personally find it a bit tedious and I'm just recording all of it at once.

Editing Software

Speaking of editing... You might want to crop off the "umms" and "ohs" and the awkward pauses during recording and for that, you need a video editing software. The program I used was **DaVinci Resolve**, which is a free Video Editing Software. It can do a lot of stuff like animations and such but I was mainly using it to crop out the things mentioned above.

I have to admit that I personally don't like editing. I find it time-consuming and since I started making money with my YouTube channel I decided to hire someone to do this for me whenever I need it to.

I also switched my focus to live streaming and I usually crop off parts of it and reupload them as standalone videos on YouTube. There were several people appreciating the fact that I didn't remove those moments when I had to google something or when I

had issues and was forced to debug my code, so I kept it in there.

At the end of the day, it's up to you if and how much you want to edit your videos. Feel free to try and see what works best for you.

And... START!

Now that you have the basics up and running it's time to start creating your first videos.

Do not worry too much at the beginning. Remember the "Make 100 Crappy videos" rule? As long as you start and you are doing your best to improve a little bit on each new video you make... you are on the right track!

How to make money on YouTube

YouTube Ads

We've probably all seen Ads on YouTube and that's because this is one of the "easiest" ways to make money on YouTube. I wrote "easiest" in quotes because there is a little catch.

In order to be able to place Ads on your YouTube videos, you need to be part of the YouTube Partnership Program and in order to apply for the program you need:

- **4,000 public watch hours** on your videos in the last 12 months
- **1,000 subscribers** on your channel

In my personal experience, the 1000 subscribers mark was easier to achieve than the 4000 public watch time hours. But for others, it was the other way around.

I got 1000 subscribers in 1 month (I can attribute this to my already existing Twitter audience) and I reached the 4000 public watch time hours after roughly 3 months.

The good thing is that once you are accepted into the Partnership Programme you are ready to make your first money on YouTube! From that moment on you will get money from all of your monetized videos.

Disclaimer: Don't forget that you need to turn on monetization on your videos in order to have ads on them. The good thing is that once you are accepted into the program, you will be able to turn on monetization for all of your previous videos.

Now let's see how much money you can make from ads...

no easy feat to calculate exactly how much money you can make from ads, but here is a rough formula that you can use:

Number of views / **1,000** * **RPM** = **Income**

The **RPM** stands for **Revenue Per Mille**. More specifically: how much money you are getting for **1,000** views on your videos (after YouTube takes its cut).

As you can see, the main two variables which affect your income from ads are **views** and **RPM**.

The view is an easy metric to track, you can find it on your Analytics page but **RPM** depends mostly on the

topic of your videos and the country from which your audience is watching the videos.

It basically depends on how much advertisers are willing to pay in order to appear in front of your audience and the purchasing power of your audience.

For example, in the web development niche, I have an **RPM** of **$2-3**. This means that for **1,000,000** views on my channel I'm getting between **$2,000-3,000** (see the formula above).

This is not a high **RPM**.

I've seen channels on design having an RPM of **$5-6** and channels on financial and personal growth going as high as **$10-15**; some even more.

Despite the RPM playing a big role in how much money you can make, I highly advise you to do your best to create great content that people want to watch in order to increase the amount of money you'll be getting from ads.

Another metric that is affecting your revenue is **CTR** - click-through rate. How many people click to watch your videos whenever YouTube is promoting your content to them. A high **CTR** results in more views which brings in more money.

In order to increase the **CTR,** you need to write great video titles and have a great looking thumbnail. These are the things that the viewers see initially and it's the deciding factor on whether they click on your videos or not.

Once they click on your video you need to keep their attention for as long as possible. This is where the **Average Percentage Viewed** metric kicks in. **APV** is how much percentage of the video are viewers are watching. High **APV** results in viewers potentially watching more ads and in you making more money.

The ads which are displayed during the video are called **Mid-roll ads** and you can enable them only on videos longer than **8 minutes**. It was previously 10 minutes but YouTube recently changed that to 8.

This is why you might have seen a lot of videos with a duration of just above 10 minutes. The creators were trying to get the mid-roll ads juice squeezed in.

Short recap

In order to increase your ad revenue, you need to **create great thumbnails** and **have catchy titles** (avoid false clickbait as it'll ruin your reputation), and once

people click on the video you need to do your best to **keep them watching**.

This not only increases your ad revenue but it will be a great flag for YouTube that your content is worth watching and guess what? It will promote your videos to more people!

Keep in mind that although ads are a great way to start making money with a YouTube channel, they are merely a fraction of the income a bigger channel makes off of YouTube. There are other, more profitable ways to make money from it.

Membership / Donations

Another way you can make money directly from YouTube is to enable **Memberships**.

Memberships are a way for people to support your channel on a monthly basis in exchange for different perks that you can offer.

The sky's the limit when it comes to what you can offer to someone who becomes a member of your channel. Here are a couple of the things you might consider:

- Access to videos before releasing public release
- Members-only live streams

- Source code of the projects
- Discord access with special roles
- 1-on-1 mentorship with you
- Course discounts (if you have any)

You can also set multiple **Price Levels** and offer higher perks for those who decide to subscribe to a higher tier. You can even ask your audience what they'd like to receive in exchange for their monthly support. Be creative.

Currently, in order to be able to open memberships on your channel, you must have more than **30,000 subscribers** and you need to be located in one of the available locations.

In my case, because I was doing a lot of live streams, I suddenly woke up one day and realized that memberships were enabled on my channel. I had less than 20,000 subscribers when that happened so apparently it's not a set rule that you need 30k. Depends on what you are doing, apparently.

Fixed donations are also a way for people to support your channel. On YouTube, they are called **Super Chats** and **Super Stickers** and they're available while you are live streaming. These will appear as special colored

messages in the chat. Different colors, depending on the amount someone is donating. These messages last for longer if the amount donated is bigger.

Keep in mind that for both **Memberships** and **Super Chats**, YouTube will take a **30%** cut. Also, if someone is donating from their iOS or Android devices, there will be an extra cut. Once, I ended up receiving only $50 from a $100 donation someone made using the YouTube app on his iPhone. Ouch.

Nevertheless, it's still good that YouTube allows people to support your channel but there are some better ways to collect donations if you want to start live streaming. More about this in its own chapter.

Affiliate Marketing

As mentioned before in the Blogging section, Affiliate marketing is when you review and/or recommend a product or a service and earn a commission whenever someone purchases that product/service. Whenever you find a product or a service that you like to use, find out if the owner has an affiliate system and ask them to partner with you.

Or you can try Amazon's Affiliate Program - Amazon Associates which you can use to promote physical

products. Maybe there's a keyboard you bought off of Amazon and you can recommend or a monitor or even a Desktop. Whatever Amazon sells you can sell too.

Just keep in mind that your commission from Amazon is not too high. As they say on their official page: the commission is *"up to 10%"*. That's quite low if you ask me. You need a high volume in order to make anything significant.

That's why I recommend you to look after some other Affiliate Programs like courses, eBooks, Bundles, or some services for developers which you use. Gumroad has a great marketplace where you can find digital products that you could potentially use.

Find a product you like, contact the owners, and ask if they would like to create an affiliate link for you. Most of the time they would be happy to because you are giving them the opportunity to reach a new audience. This is what I'll be doing for this eBook too. In case you are looking to become an affiliate, feel free to reach out.

Brand Deals / Sponsorship

The more your audience grows and the more the engagement of that audience grows, the more emails

you'll get from different companies that will want to promote their products and services in your videos.

This can be a great thing!

I say "can" because you must be aware that there are shady products out there. I wouldn't advise promoting anything you are not comfortable using yourself. But, after cleaning out all these "scammy" products, you might find yourself a very lucrative way of making money on YouTube.

How much can you make? There isn't really an exact formula on how to calculate how much to charge a sponsor. It depends on multiple factors:

- The size of your audience
- The engagement of your audience
- The brand who's offering the product/service
- The deal you are making with them
- How specific your channel and video is to the product ("The riches are in the niches" - if you are specific, you are worth more money)

Companies are looking to make a profit and based on your audience you need to estimate as closely as

possible how much money they're going to make off of your promotion.

If you charge too much, they will probably not want to do another deal with you.

If you charge too little, well... you will be losing out on money which you could have made.

The deals can range from **$100-200 per video** when you're starting out with a smaller channel, and it can go up to **$1000** (or even more) once you have an established brand and a bigger audience.

Keep track of what is the click-through rate on different brand deals and, if possible, try to get the conversion rate (although not a lot of brands will give you this information). This will help you estimate more accurately how much to charge for future deals.

Also, there are different types of deals. Instead of giving you an amount of money some brands can offer one of their products in exchange for a review. This is how I got a standing desk which I use all the time now.

It is also very important the way you present a product in the video. Try to find clever ways to do that without sacrificing the quality of the video. Don't forget that

your audience is much more important than a one-time deal.

You can place the promotion at the beginning, in the middle, or in some cases, at the end of the video. The duration of the promotion is also something the companies might mention. Something around 20-30 seconds is what you should aim for if possible, although some brands might ask for more.

At the end of the day when it comes to brand deals, it is also important how much you want to make. You shouldn't sell yourself short. Work on growing an engaged audience and the opportunities will arise.

Disclaimer: Don't forget that you are required to let YouTube know if your video has a paid promotion. There is a simple checkbox you have to check when uploading the video to let YouTube know that it contains a paid promotion.

Promote your own products

As we mentioned in the **Blogging** chapter, promoting your own products might be the most lucrative way of making money on YouTube too. Why? Well, because you already have an audience to sell to and if done properly, you should already know what your

audience's needs are in order to create the perfect solution for them.

Read more about creating your own products in the **Creating Digital Products** chapter.

CHAPTER 5

LIVE CODING & STREAMING

Live Coding is probably one of the best ways to interact with your audience and build a strong community.

Going "Live" is becoming the new buzzword. Be it gaming (which is extremely popular due to people like Ninja) or just chatting with your audience and sharing your knowledge via Live Q&As. Or even better, in our case, live coding.

The live streaming ecosystem will continue to grow more over the following years and it will continue to expand beyond the gaming niche for which is very popular now. So if you decide to hop on the wagon now it will most likely pay off big time in the near future.

We know that behind every content creator there is an actual real person and we love it whenever we get to interact with our favorite creators. We want to get to know them better and maybe even get the chance to see some of the "behind the scenes".

How to start

There are a lot of similarities between doing live streams and creating videos so I won't repeat the tools you need in order to get started. Go and check the **YouTube** chapter if you want more details.

What is different than creating videos and it's worth pointing out is the fact that you need to be more engaging while doing live coding. When it comes to making videos you can easily cut off the boring parts or the parts where you don't say anything, but when you're live... you can't do that.

You have to find a way to be interacting with the community and avoid having lots of silent times because people might get bored and leave. We got the attention span of a goldfish, sometimes even less, and we easily get distracted if the content is not engaging enough.

If you are an introvert this might seem a bit tough but there is a simple trick you can do: Just keep talking. Whenever you are thinking of something, let's say: "How do I solve this bug in my code?", say it out loud. The same way you would do in a coding interview. You will be surprised at how many times you'll get helpful advice from the chat which means engagement and that people are interacting with you making them part of the solution. Win-win situation.

Also, make sure that you acknowledge everyone who says "Hi" or "Bye" or "Why?" or whoever donates, subscribes, etc. We have a good feeling whenever

people acknowledge us, especially those who we look up to.

You can also ask questions to get the chat to interact with you and to keep the ball rolling.

When you're starting out, just think of it as you having fun coding among friends. Most of the time the community is very supportive and this is just how it feels, coding with friends, so just relax and let those fingers write some code!

YouTube VS Twitch

The two big competitors in the **live coding** niche right now are **YouTube** and **Twitch**. In this section, we're going to take a look at the differences between these two platforms.

YouTube is a great platform to grow on. Their search algorithm helps people find your content more easily, especially when you are live as it will place your content on top of the other search results.

The downside is that the streams will slowly get buried after you go offline which doesn't make it great for long-term content. It doesn't happen to all of the live streams, but to most of them.

One of the reasons is because people who are watching the recording of a live stream might not fully understand what's happening the same way those who were live in the chat would as they were with you the whole time. Those who are watching the recording are missing the interactivity part of it.

Another downside of YouTube, as of the time I'm writing this, is that people aren't as open to supporting the creators as they are on Twitch. Even though I had more viewers on YouTube than on Twitch the income from the supporters (donations, subscriptions, etc), was smaller. The reason for that is the fact that YouTube is still known more for its VODs (Video-On-Demand) and people aren't yet fully aware of all the extra features a live streaming session has.

I believe that this will slowly change and we'll see more people engaging in live streams on YouTube too, but as of now, Twitch is leading the niche of live streaming.

Speaking of which...

Twitch is a great platform to build engagement with the community during live streams and, as discussed above, people are more likely to support the creators there. It's been around for a while and it established

itself as the leading streaming platform so those who go on Twitch are expecting live streams.

The majority goes on **YouTube** to watch shorter, straight to the point, videos. Especially in our development niche which is still evolving when it comes to live coding.

The downside of Twitch is the search feature. It's hard to grow on Twitch alone if you don't have an audience outside of it because the platform doesn't help smaller creators that much as their stream is buried down the list while the big streamers are on the top of the list.

We're still waiting for this to be fixed but until then a combination of YouTube videos for reach and growth and Twitch live streams for engaging with your audience seems to be the go-to.

Ways to make money while Live Coding

Making money while live coding it's pretty much the same as making money on **YouTube**, with a couple of exceptions.

We won't cover all the methods again here (see the **YouTube** chapter), but we'll rather focus on the ones which are more specific to live coding.

Donations

One of the simplest and most commonly used ways to make money at the beginning while streaming is receiving donations from viewers who want to support your work.

In order to receive a donation, you need to set up a Third-party system like **Streamlabs** which gives you a special link where people can donate. Once they do you'll get the money directly to your PayPal account.

most of the time you cannot control when and whether someone is donating or not, there are a couple of things you can do in order to incentivize people to support your work.

You can offer different services or benefits for those who decided to do that. Below are a couple of examples:

- Review a portfolio
- Work on a requested project
- Solve a coding challenge

In the end, it all comes down to the relationship you have with your audience and the type of work you are doing while streaming. Is it directed towards teaching those who are watching you or is it more about you working on your own projects?

Also, you could set up **Goals** to appear on the screen and viewers can support your work in order to achieve your goal. Goals like buying a new PC or getting some new streaming equipment that will improve the quality of it might be something people would be interested in helping you out with.

Nonetheless, keep in mind that there might be also a downside to getting donations as some donors might feel entitled to ask for all sorts of things just because they donated $5. In order to avoid that let everyone know if there are things you do in exchange for a certain donation, or, if that's not the case, just mention that their support is just for supporting your work and nothing else.

Whatever the case would be, make sure you acknowledge every donation and read it out loud as that will make the donators happy. It will make them feel a little bit more special.

Bits

Bits are a virtual good that can be bought on Twitch. Simply put, Twitch's currency. People can buy bits and donate them to their favorite streamers.

For every **100** bits donated, the creator receives **$1**. It's not a lot but it can slowly add up to a bigger sum as more people are donating. Also, there is the incentive of other gifting bits when they see someone else do it, the snowball effect.

These bits will be automatically converted to USD, you don't need to do anything about them, just enjoy whenever someone gives you some.

Memberships and Subscriptions

Once you reach a certain audience size on Twitch (or YouTube), you have the option to offer different perks to those who decide to become subscribers (Twitch) or members (YouTube). We have covered the perks people can get for supporting your work when we discussed the YouTube Memberships (see the **YouTube** chapter).

Twitch subscriptions are pretty similar to YouTube memberships.

Note the fact that Twitch takes **50%** off of your subscription revenue. It's not a small chunk but they are providing us the platform so we need to go with that. I've heard people say that those who end up becoming Twitch Partners can negotiate their cut. I'm not a Partner yet so I can't tell if that's true or not.

Exclusive Live Streams

Another way to monetize streaming is to have special, members-only streams. In these streams, you can do special Q&A sessions where people can ask you anything related to a topic, or you can review their projects or anything else you can think of doing for those who are willing to support you on a monthly basis.

Keep in mind that all of those joining these streams are proven to be already willing to give you their money, and this is important. One person giving you $1 is better than 10 people saying they will give you $100 but never actually doing it.

It's not that the ones who are not supporting you are not to be appreciated. Those that are willing to "put their money where their mouth is" might just help you

pay the bills. So make them get their money-worth of... whatever you are planning to offer.

CHAPTER 6

1-ON-1 MENTORING

If you are good at what you are doing and you like helping others succeed, 1-on-1 mentoring would be something you could consider. It's great to get to know other people and to learn about their journey while expanding your own knowledge and perspective of that corresponding niche. It is also a good way to earn some extra money on the side.

Having a blog, a YouTube channel, and/or a Social Media presence (like Twitter) where you provide great value on a particular topic and share your own experience will slowly set you up as an expert in the eyes of those who follow you. Soon after that, you will most likely get contacted by those who would want to learn more from you. This is what happened in my case and I've seen it happen to others too.

Of course, some people might like to get everything for free, including your time and effort, but at the same time you'll find that others value you and your time (and indirectly value their own time too) so they will be happy to pay for the time you spend helping them get to their next step in their journey or maybe just to simply help them get "unstuck".

This is what paid mentoring is about. Exchanging money for time and experience. In my opinion, this is a

great exchange of value because our time is limited but money comes and goes. Whenever I have a chance to use the money I made to save time, I'm going for it! So far it worked out great.

It doesn't have to always be money in exchange for your time but you gotta make sure that it is a fair exchange. For example, I started with free mentoring because I wanted to gain experience and I wanted to practice my English speaking skills. So I basically offered my time and programming experience in exchange for exposure, growth, and English lessons.

My mentoring journey

I still remember one of my first paid mentoring sessions. I helped someone with a rather small ReactJS issue. For me, it seemed like a small issue because I had over 4 years of experience with React but he was just starting out and spent about 2-3 days trying to figure out why it didn't work. He gladly paid for my time because, in exchange, I helped him save some of his own time.

After that, we had multiple mentoring sessions. He even started to recommend me to someone else and that's how I started my paid mentoring journey.

Over the past year, I had dozens of people with whom I had at least a short 20 minute call. Some of them were free because, as I mentioned above, I wanted to gain more experience (and because I wasn't comfortable enough with my English accent) and some of them were paid.

As my Social Media presence grew, so did these requests for 1-on-1 mentoring, but I stopped doing them after a while as I started focusing on growing my YouTube channel.

I still have a Google Form with dozens of submissions from people wanting to have a paid mentoring session with them. I created the form a while ago which contained a couple of questions and I had the link up in my Twitter description for anyone who would be interested. They could fill out the form which made it easier for me to gather as much information as possible before the calls.

While doing these sessions, I couldn't say that I was an expert and I knew everything the mentees asked me during the calls, but due to my experience, I was able to find an answer faster than they would, or at least I was able to point them in the right direction.

How much to charge?

This is a common question that comes up, but as with every service we can provide, there isn't a fixed number. It depends on multiple variables. I can tell you what I did and hopefully, you can figure out your own amount.

I started by charging **$30** per hour. That was the amount I was happy to start with because it was around what I was getting paid as an employee which made it worth my time.

After a while, I got more experience and I became more confident about what I was doing. I also started to work more on other projects so I decided to raise my rate to **$50** per hour. I was expecting to get people turning me down because of this but it didn't happen.

I even got a message from another YouTuber telling me that I should raise my rate to at least **$100-200** per hour as that was the norm (or at least that's what he said) but at that time I felt that it was a bit too much for me. I still wanted to give a chance to those who can't afford to pay that much.

It's worth keeping in mind that in some countries people might not be able to pay $30 per hour while in other countries, $200 or even $500 per hour is alright as

long as you'll get your money's worth. This is where you should use your own judgment to decide the right price.

This is why I can't tell you how much to charge. You should find out how much is one hour worth of your time and charge that. I'm sure that if you provide great value people will be happy to pay for your time.

Where to find mentees?

I don't remember ever using a platform to find mentees because I leveraged my social media presence to do that and I encourage you to follow this path. Share great content, grow your audience, and then you can find mentees more easily.

You can however leverage other platforms. I haven't used them myself but I know some people who did so it could be worth checking out.

Coding Coach seems like a good way to start. It's a "free, open-source platform which aims to connect software developers and mentors all over the world". You can use this platform to get more experience and exposure.

Code Mentor seems to be a combination of a mentoring and freelancing platform. I have a friend who used to work through them part-time and he seemed happy about it. Give it a try and see if it works for you too.

(This page intentionally left blank, for print)

CHAPTER 7

CREATING COURSES

You might be wondering: *"Why should I create a course? There are so many out there already"*. And that's true. There are a lot of courses out there, but those were not made by you!

You have your own perspective and experience. You have your own way of teaching and you can create a course that is different and better than what is out there already.

Nonetheless, if you never created content before it might be a good idea to give it a try first. Create some videos and upload them to YouTube and see if you like doing that. This will prepare you for creating courses and it will also help you build a following. Win-win.

How to create a course

Pick a topic

This is one of the most important things you have to do at the beginning before even starting to work on the course.

There are a few ways which can help you decide what topic to pick:

- Choose something you can talk about all day. Something you are passionate about
- Ask your audience (if you have one) what they'd like to learn from you
- Research the market. Check out what people are looking for

Create the Course outline

You can start with "brain dumping" your ideas for the course. See what you can come up with at the beginning without actually getting into details yet. You will do this later. Figure out what you want your modules and lessons to be about.

Some videos you might consider having:

- Who are you?
- What is this course about?
- Why is it important for those who want to purchase it?
- What's the structure of the course?

Software and Tools

We covered everything you need to know about recording, editing, etc., in the **YouTube** section. Check it out for more information.

Having an audience

It is not required to have an audience in order to create and sell a course but it would definitely help to have one. This is why I recommend starting first with either a blog or a YouTube channel where you can grow an audience by providing quality content.

There are platforms out there like Udemy or Skillshare which can help you promote your courses to their audience, but as you might expect, they take a cut of your paycheck. It might be worth it if you want to get straight into making a course, but I would personally work on growing an audience first.

It's much easier to sell to those who already know you and the way you are teaching and it's also more profitable. Just as an example, if you sell your courses on Udemy and you bring in the sales you get to keep 97% of the sales but if people find you organically by browsing through their marketplace of courses you

keep only 50%. This is because they helped you to make the sale.

It would be better to sell to your audience directly on your platform, but more about that below.

Where to sell your courses

There are multiple ways you can sell your courses online. Look into all of them and pick what you think is best suited for you.

Course marketplaces

Udemy and **Skillshare** are two big marketplaces where people can buy courses on various topics including programming. You probably heard of them and even used them to learn a new skill.

As mentioned before, these marketplaces already have a big audience which makes it easier for you to sell your courses in case you don't have an audience yet.

Udemy is paying you a percentage of the sale. The percentage amount depends on whether you got the sale from someone using your coupon code or your referral link (you keep 97% of the sale) or if the sale was made by someone browsing through their marketplace

(you keep 50% of the sale). Another thing to note about Udemy is that most of the courses are sold for around $10-12 because they have sales all the time.

Skillshare is paying based on watch-time minutes. The more people are watching your content, the more you earn.

Course platforms

Podia, **Teachable,** and **Thinkific** are some platforms that allow you to create your own website on which you can host and sell your courses. They handle pretty much everything you need in order to get started:

- Hosting
- Course player
- Website builder
- Payments
- Email integration

and more.

Each platform has different pricing tiers that you can choose from depending on what features you want. Make sure you check them out and pick the one which is best suited for your needs.

The main reason for using a course platform is the control it gives you. You can charge as much as you want for your course and you also get the information of those who enrolled in your courses so that you can follow up with them more easily.

The worst part of self-hosting your course is the fact that you have to do the marketing yourself compared to Course Marketplaces like Udemy. This is not that bad if you already have an audience (a blog, a Youtube channel, etc).

Build your own platform

Maybe you aren't satisfied with what the above-mentioned platforms provide so there is always the choice to build the entire platform on your own from scratch. After all, you are a programmer, right?

I wouldn't personally go down this path but I've seen companies doing it as they needed specific features that were not provided by default by these other platforms. At the same time, they had a large number of students which made the investment-worthy.

Marketing

After you create and publish your course online it's time to spread the news. There is no point in creating the best course out there if no one knows about it.

As mentioned above, some platforms will do this for you but if you don't want to go down that road it would help to create a landing page to promote the course on.

On the Landing page you can have things like:

- The course outline
- Reasons why people should buy
- Testimonials (very powerful)
- FAQ section
- Information about you

Before the launch of the course, you can set up a newsletter form where people can submit their email. This is a very powerful thing to have as those who willingly give you their email address are more likely to buy than any other person who's just lurking around the landing page.

After the launch use the power of having an audience (if you have one) to promote your course. Consider

giving away parts of the course for free to incentivize people to buy the full course.

Setting up a discount for a short period after the launch is another good way to incentivize people to buy as we often don't like to miss out on good deals. It's called scarcity and it's used all over the place to push us (more or less) into buying things.

Promotion post-launch

Having a launch is great as in the first couple of days it will bring in the majority of the sales but it will eventually slow down. That's when you need some other ways to promote the course.

Writing articles on your blog and/or having some videos on your YouTube channel talking about some of the topics you covered in the course is a great way to continuously promote it.

Like for example if you have a JavaScript course, you can make different JavaScript tutorials for your YouTube channel and inside these videos, you can mention the course. You never know when YouTube decides to make that a viral video and your sales can pick up again.

Let YouTube and/or Google (if you have a blog) work in your favor and help you set up that juicy stream of (relatively) passive income.

Setting up affiliates

Another way to bring in more sales is to set up affiliates for your course. The majority of the platforms we mentioned above allows you to do that and they take care of pretty much everything.

Once that's done, reach out to people who have bigger audiences in a similar niche and offer them to become an affiliate for your course, and for every sale they make, you both earn money. Win-win.

CHAPTER 8

CREATING DIGITAL PRODUCTS

Selling digital products is amongst the most lucrative ways of making money online, especially for Content Creators like bloggers and/or YouTubers. You don't have to have an audience in order to start but it will definitely help a lot.

What are digital products?

Simply put, a digital product is any product that you can sell online that is not physical - you can't touch it like you could touch a book, a chair, or a car - and it can be downloadable.

Some digital products can also be transformed into physical ones later down the road - for example, an eBook can be transformed into a physical book.

What kind of digital products are there?

When it comes to our niche, programming, there are a lot of different products you can create. If the product is valuable and it helps others save time and/or money, you got yourself a product to sell!

Some examples might be:

- Plugins (for WordPress, Shopify, Drupal)
- Components (in React, Angular, Vue, Bootstrap)
- Libraries
- Kits (Design Kits, Component Kits)
- Themes (WordPress, Shopify, Drupal)
- Templates (HTML5, Bootstrap, Tailwind)
- eBooks
- Courses (*we covered this in its own chapter*)
- Software
- Web Apps and Mobile Apps

and more.

You can find a lot of examples scrolling through marketplaces like Creative-Tim, Gumroad, ThemeForest, and CodeCanyon.

Why digital products?

Digital products have many advantages compared to their physical counterparts. With digital products:

- You can sell infinite copies while the cost of producing/replicating is inexistent (most of the time). After all, is just a copy of the initial product
- Customers receive the product immediately after purchasing it, either via email or via a custom link where they can download the product directly
- You can use the money which you would have paid to produce the physical product (manufacturers, labor, etc.) into advertising and marketing the digital product

Easy put: *Build once - Sell an infinite amount of times.*

Of course, some products might have to be updated from time to time but overall it's much easier to do that than to create a new product from scratch.

Where to sell?

As developers and creators, we always have the option of creating our own platform which we can use to sell our digital products. I know some people who did but I wouldn't go down that path.

The first reason is that I'm mostly specialized as a Frontend Developer and it feels like it would take me ages to create a platform that would suit my needs.

And the second reason (which is probably the most viable) is that there are already platforms that can do this job for me in exchange for a small monthly fee, which I'm happy to pay.

of these platforms, and the one I use personally, is Gumroad. Reasons why I chose Gumroad:

- It has a simple payment process
- Handles VAT for me (a huge pain selling from an EU country)
- I can build my own email newsletter and have control over it
- I can set up affiliates and have others help me sell my products in exchange for a commission
- Great Analytics for customers and sales

Another great thing about Gumroad is that you can easily integrate it with your own website/blog and customers can purchase directly, without having to leave it.

How to create a product?

Due to the fact that there are so many different digital products that you can create, the "HOW" comes down

to just exactly that. What do you want to build and sell? Is it an eBook? Is it a course? A theme? A template?

We covered the course in its own chapter.

When it comes to writing an eBook, the process is quite similar. The difference is that one is in text format and the other one is in video format. To write this eBook I used Google Docs because it made it easy to receive feedback by sharing the chapters with the reviewers.

When it comes to creating a theme or a template you have to put yourself in the shoes of the users. What would they expect if they would buy the product? Is it a faster development? Ease of use? Both?

You can also buy similar products and check out what they are offering and how they are structuring the product.

Build an MVP

Before going all berserk into building the most advanced product on the market with tons of fancy features I highly recommend you to test the waters first. Is this something people are really interested in? Will they buy it?

You can do this by creating an MVP - **Minimum Viable Product**. Figure out what are the basic functionalities your product should cover and build that in the first stage. Once that is done put it out on the market and see if people are really interested in purchasing it. If they are, then you can continue investing time (and money) into making the product even better and you can start marketing it to a bigger audience.

Market your product

When it comes to marketing, some platforms might help with that but in most cases, you have to do the marketing yourself. This is where having an online audience shines. It can be a blog, a YouTube channel, or a following on a Social Media platform (Twitter, LinkedIn, Instagram, Facebook, etc.), or all of them (the best case scenario).

People following you online are already acknowledged of your work and are more likely to trust and buy from you (*probably this is what happened to you too when you bought this eBook. Thank you for your trust!*)

Another great thing about having an audience is that you can focus and create a niched product specially tailored for them. Find out what is the biggest problem

the majority of your audience has and solve that problem for them. I wrote this eBook because most of the people who were following me weren't aware of how someone can use programming to make money, except for having a job.

Build an email newsletter

Blog visitors, YouTube views, Social Media followers - they are all great but you don't own anything. It might be very improbable but any day YouTube can disappear, so couldTwitter, and all of a sudden your big audience will just... vanish! You'll go from Hero to Zero. Which is definitely not something we want, right?

This is why having an email newsletter is a must! This is your audience and you have control over it (unless they unsubscribe, lol). Truth to be told I'm still not the best at this and it is one of those things where you should: "Do what the preacher says and not what the preacher does."

So, learn from my mistake and build an email newsletter. Now!

You can leverage your subscribers later on when you are ready to build your digital product in order to build

up hype and eventually get lots of sales once you launch!

Leverage other people's audiences

If you are at the beginning and you don't have an audience yet but you have a killer product, don't worry! You can still make money by leveraging other people's audiences.

Reach out to people in your niche and ask them to partner with you by promoting your product. You can either offer a fixed fee (for example $X amount per post/tweet/video) or you can sign them up to become affiliates and receive a commission for every sale they help you make. I consider the latter approach to be a win-win for everyone as it won't cost you anything upfront while avoiding the risk of not making any sales from the shoutout and at the same time, the sellers are more likely to promote the product multiple times as it's in their interest to make more money. Win-win!

Bundle it up!

In some cases, you might find other creators who have a complementary product that might go very well

combined with your product so you can team up and sell a bundle containing both products with a small discount. This way you can leverage both your audiences.

Example: Killer Design + Efficient Development = $$$.

CHAPTER 9

CONTRIBUTING TO OPEN SOURCE

Open source is software with publicly available source code, making it easy for anyone to contribute by adding, editing, or even removing parts of the code.

The OS movement is growing rapidly, and more users are joining daily. In 2019 GitHub had a total of 40M+ users, including 10M new users in 2019 alone. 44% more developers created their repositories on GitHub in 2019 than in 2018. (*As of the time I'm writing this the data for 2020 was not yet published. You can find the statistics here*).

Here are a couple of very popular Open Source projects you might have heard of or even used before:

- WordPress
- Linux
- VSCode
- Libraries like React and Vue

and much more!

Even though most of the time the work on an OS project is seen as "free work", there are some ways people can support the work of the maintainers/creators via different platforms. One notable example is **Patreon** via a monthly donation or, even better, GitHub recently

launched **GitHub Sponsors**, which is another great way to get support from those who are using these OS projects.

What is great about GitHub Sponsors is that there are *no commissions* which means that you get to keep the full donation, unlike Patreon, which keeps a percentage of every donation.

At the time I'm writing this, GitHub Sponsors is supported in 34 regions for individuals and organizations. You can find the entire list here. If you can't find your country on that list yet, make sure you sign up to get notified when it's available. In the meantime, you can sign up for a Patreon account and get support there.

Another great thing about these platforms is that you can set up different tiers that people can choose from in order to receive perks in exchange for their financial support. Some people might support the OS project without expecting anything in return, but having tiers containing different perks will most likely attract new supporters.

Also, setting a goal (like making x amount of money or having y number of supporters) will incentivize even

more people to support your Open Source contributions.

Extra ways to get a reward for your OS contributions

You might not be getting a direct financial reward when working on an Open Source project, but there are other great benefits you might get instead:

- Expanding your online presence and growing your personal brand
- Standing out from the crowd when applying for a job
- Impressing your potential clients when working as a freelancer

Another thing some people (and even companies) do after creating a successful Open Source project is to create a Premium version of the project containing extra features and benefits which they are charging money for.

One great example is with the creators of TailwindCSS - an Open Source utility-first CSS framework. Once the project took off and became popular, they created a

paid product called TailwindUI which contains several UI components built using TailwindCSS.

Last, but not least, you can also monetize an Open Source project by adding a sponsored banner in the README of the project, or in case you have a popular website which gets a lot of views, you can use all the methods we discussed in the Blogging section (ads, affiliate products, promoting your own products, and even selling merch).

(This page intentionally left blank, for print)

CHAPTER 10

CREATING SOFTWARE-AS-A-SERVICE

Disclaimer: *Before getting started, I want to state that I don't yet have much practical experience with building SaaS products, which is why I have reached out to Simon Høiberg, the founder of Sigmetic and starter of the #100DaysOfTechStartup challenge to help me write this chapter for you.*

What is Software-as-a-Service (SaaS)?

The term Software-as-a-Service (or SaaS for short) refers to a hosted software product that is licensed on a subscription basis. This means that the software product "lives" on a domain as opposed to being distributed and that the customers buy access to the software on a recurring basis, via a subscription.

Good examples of popular SaaS products are:

- Jira
- Trello
- Basecamp
- Figma

- Slack
- Dropbox

Another (more informal) definition of a SaaS product, is that it is used to solve a problem on-demand. By this definition, Netflix and Spotify are not classified as SaaS products, even though the users pay for access on a recurring basis. They are selling a license to access movies, tv-shows, music, and podcasts, but they are not considered tools that you use to solve a problem.

Software such as Adobe Photoshop, Sketch, and Pro Tools are not classified as SaaS either. Even though you pay for them on a recurring basis, they are not centrally hosted. Instead, they are distributed through executable files (i.e. they are software that you install on your computer). The definition of a SaaS product is a bit loosely defined, but the above is the most generally accepted.

Benefits of SaaS as a Startup

SaaS is becoming increasingly popular for a number of reasons. The barrier of entry is really low.

Anyone with some programming experience can set up their own SaaS product.

Social Media such as Twitter, LinkedIn, and YouTube makes promotion easily accessible, let alone all the popular sites such as ProductHunt and Betalist that are made specifically for early adopters of new SaaS Startups.

Given the rise of the internet and modern browser technologies, SaaS also introduces another huge advantage: Continuous Delivery (CD). It has never been easier to constantly roll out updates, bug fixes, run multiple versions of your product in parallel, and gain constant feedback and validation.

Technologies like WebAssembly - which pushes the boundaries of how heavy and sophisticated software you can run in a browser - makes the market of SaaS products extremely competitive and lucrative. Figma is a great example of this.

Most of the technologies needed to run a successful SaaS startup today are cheap, easy to access, and have a ton of resources on the internet on how to get started.

This also means that we can kill off some myths about running a Tech Startup:

You **don't** need to live in Silicon Valley, you **don't** need to be backed by multiple Venture Capitalists and you **don't** need to hire a full team.

In fact, you don't even need an office. You don't need to quit your regular day-job. You can start a SaaS company on your own, in your spare time, and simply let it grow, slow and steady.

On Twitter there are several great examples of this:

Simon Høiberg who created *Sigmetic* during the #100DaysOfTechStartup challenge, *Daniel Vassallo* who is currently building *Userbase*, and many others.

These are all people with software skills that created their own SaaS products, marketed them through Twitter, and are now making money on the side from their products.

Considerations of SaaS as a Startup

Unfortunately (for latecomers especially), as in all industries, where factors such as those previously discussed create a low barrier to entry, another potential obstacle arises - competition.

The internet is flooded with SaaS products. The competition is extreme. There are developing countries that can deliver software at almost no cost, and where you might be the only one working on something, there could be many other companies working on the same service.

At this stage, the most obvious products have been built. It is really difficult to come up with a great idea that hasn't been considered already. If your idea is unique and has not been built yet, there's a good chance that it's because no one is interested in the concept.

9/10 Tech Startups fail. The most common reason is that the startup is building something that no users are interested in and that they - eventually - run out of money and time.

The market is tough. Therefore, it is of crucial importance to build a SaaS startup in small iterations, and include users as soon as possible.

You want to learn what you're doing right, and what you're doing wrong as fast and often as possible. We will look into that during this chapter.

How to Build a SaaS Product

Find an idea

As we just touched upon, finding an idea for a SaaS product can be a challenge.

You typically run into one of two situations:

1. Either the idea is taken, and there's a huge player out there already.
2. The idea is new, thus no market data is available.

Both of these situations present challenges. If there's a big player already out there, you will have a huge competitor. You will be met with a lot of "but *x* is already doing that" when presenting your idea, and the dominance of your competitor can become very demotivating and demoralizing at first.

On the other hand, if you happen to come up with something new, there's now a lot of uncertainty at play. You have no market data to rely on. You have no idea whether people even need or want your product, and it may become extremely difficult to validate.

Some of the greatest products in the world, created by visionaries, had the second situation to contend with. However, the safest bet is actually situation number 1.

Find a product that is already mass-adopted and then create an alternative to this product. Change small variables, approaches, and ideas. Innovate the way things are carried out. But aim at solving the same problem, overall. Typically, the market is big enough for another player and another product. Don't be too concerned about competitors.

If you can solve a problem, and solve it in a satisfying way, you are bound to get clients, sooner or later.

The 'LEAN' method

As we established earlier, it is of crucial importance to build a SaaS startup in small iterations, and include users as soon as possible. We want to do this in order to gather *validated learnings*. That is, either confirming or rejecting a hypothesis that we have.

When using the 'LEAN' Startup method, this process is called the *build-measure-learn* feedback loop.

Every time we build a feature, we build it based on an assumption that the user will be using it and eventually be willing to pay for it.

An assumption is also a "hypothesis", and if we build an entire SaaS product based on assumptions, we might completely miss it! That's why it's important to not get carried away and just build blindly.

According to the 'LEAN' method - every time we plan on building a feature, we must somehow validate that the feature is actually wanted. We can do this by building the smallest possible version of that feature and then testing it on real users.

If the feature turns out to be unwanted by the users (hence, we reject the hypothesis), we haven't spent too much time developing it.

If the feature turns out to be wanted by the users (hence, we confirm the hypothesis), we can now invest more time into improving upon it.

A SaaS product that only consists of these small and unfinished features, is called an **MVP** (Minimal Viable Product). It is a SaaS Product that is working only to the extent that we can gather *validated learnings* about it (confirming and rejecting our hypotheses).

Eric Ries wrote an award-winning book called *The Lean Startup.* I highly suggest you take a look at that.

Choosing the Tech Stack

Choosing the right Tech Stack for your SaaS product is very important.

You may like React better than Vue. You may prefer Node.js over Python. You may be more experienced with Azure than with AWS or Google Cloud Platform.

These are actually not that important. I suggest simply pick the technologies you know the best. What is important, is to keep scalability and flexibility in mind. Because building a SaaS product is based on a high level of uncertainty, you want the ability to make as little initial investment in running your platform as possible, but at the same time, you want to be able to scale it, as soon as needed.

For this reason, picking solutions based on a pay-per-use model is ideal for running a SaaS product.

Serverless architecture is probably the best choice here. Instead of running a server and paying for constant up-time, you run cloud functions where you pay per 100ms of runtime.

Instead of running a hosted database, where you also pay for constant up-time, you run a managed database that allows you to pay for individual reads/writes.

Both AWS, Azure, and Google Cloud platforms have great solutions to build serverless architecture.

AWS is by far the most advanced platform when it comes to serverless, so if you don't already have a preference, I suggest you start with that.

The Development Productivity Tool, *Sigmetic*, is built entirely on a serverless architecture using AWS. It uses the *'GRADS'* Stack, which is *GraphQL*, *React*, *AWS*, *Direflow,* and *Serverless Framework*. You can visit sigmetic.io/blog where you can find a lot of great resources on this topic.

Choosing the Pricing Model

When running a SaaS product, you need to choose a pricing model.

The following pricing models are commonly used.

Usage-based pricing.

With this pricing model, the price increases along with usage.
For example, this model is used by AWS when they charge per 100ms of runtime of a Cloud Function.

Per-user pricing.

With this pricing model, companies charge a fixed rate per month for each user on an account.
For example, Slack uses this for their paid version.

Flat-rate pricing.

This is the most commonly used model.
One product with one price. The price is the same for all users, and all users get the same set of features.

Freemium and Trial

Along with the above pricing models, it is typical to combine either with:

a) a *freemium* model where part of the software is free and unlimited.

b) a trial version of the software where the full software is free and unlimited for a period of time, most typically 30 days.

Picking the right pricing model and price can be hard, and most typically, this calls for multiple iterations on its own.

The popular Customer Feedback Tool, *Canny*, was experiencing these problems first hand. Founder of *Canny, Sarah Hum*, wrote a great article on IndieHackers about her experience. Take a look at the article as there is a lot to learn from their situation.

Stop thinking - start doing

The amount of decisions to take is deeply overwhelming, thus most people without any experience get stuck in analysis paralysis and never get much further than buying a domain.

This is one of the reasons why Simon started the 100DaysOfTechStartup challenge.

100 Days of Tech Startup is heavily inspired by the trendy challenge 100 Days Of Code, and - as the name suggests - it's about creating a Tech Startup in 100 Days.

Needless to say, it's a tough challenge, but it's also more centered on *learning* about SaaS, than attempting to build a successful one on the very first attempt.

100 Days is sufficiently limited that you don't get too hung up on the thinking process. It's about executing and making fast decisions.If you want to give it a try, check out 100DaysOfTechStartup.com. You'll find a nicely-outlined program for anyone to get started in the SaaS world.

How to gather Validated Learnings

There are many ways to gather *validated learnings*, and only creativity sets the boundary. However, there are some common strategies that you can rely on. Let's go through them here.

Prototyping

With modern technology, it has become really easy to create a prototype of your SaaS product. A prototype can range from *low fidelity* prototypes, which can be as simple as a mock-up sketched on a piece of paper or a

whiteboard, to *high fidelity* and *functional* prototypes that emulates an almost functional application.

Tools like Figma, Sketch, and Adobe XD are great tools to quickly set up a prototype. Now you have something to present and to show users.

Analytics

Gathering analytics is an essential part of running a SaaS startup. You want to leverage tools such as Google Analytics, Segment, and Amplitude to carefully track the behavior of your users on your landing page and on your app.

You want to track how many visitors bounce in less than 2 seconds, how many visitors scroll to the bottom, how many visitors check out the pricing page, etc.

Studying the demographics of your visitors is also important. Which country they are from, which device they are using, from where they are referenced, and so on. Most modern analytics tools ship with a complete package for this out-of-the-box.

Making a simple "sign up for a newsletter" form can also be a great way to get a sense of interest in your upcoming product.

User testing

User testing is a great way to gather validated learnings. Align with an initial round of users to try out your product. Make a Zoom call with each of them, and have them share their screen while using your product.

Do not guide them. Simply observe how they use your product to solve various problems. Record the sessions (with the user's consent, of course), so you can study it later.

Creating "personas" and "client profiles" is a must to formulate a hypothesis.

Make up a fictional persona that has the qualities of your ideal client. Make assumptions about how your persona will use your product, then see if you can find an actual user that matches the description of your persona.

Now observe how the actual user is interacting with your product, and hold that up against the hypotheses you made based on your persona.

Split testing

Asking users about their opinion can be an obvious way to gather information about the problem that you are

trying to solve. There is an issue with this, though. Most often, the user knows the problem really well but has very limited insight when it comes to the solution.

As a general rule of thumb, you should listen closely when the user describes the problem. When they try to suggest a solution, you should stop listening entirely. Instead, you want to test various solutions out yourself.

Split testing is a very powerful technique here. The idea is that you create multiple versions of a solution to a problem, and then you expose different groups of users to the different versions, and you track which one works the best.

Most Cloud providers have mechanisms that support this.

With AWS, you can use the Lamda@Edge feature to randomly assign an incoming visitor to a version of your webpage to present to them.

This is a great way to answer questions such as “Where should I put the signup form?”, “How should the Call-To-Action be formulated?”, “Should the user be allowed access before paying?”, etc.

After split testing such hypotheses, you will be able to see for yourself which version of something yields the best results.

Promoting your SaaS Product

A classical mistake among new SaaS founders is that they underestimate the importance of promoting their product. You should put at least the same amount of effort into promoting your product as you do into developing it.

Promoting your product is not something you do when the product is finished. You should be promoting your product from day one. How else would you gather *validated learnings?*

Let's take a look at a couple of ways to promote your SaaS product.

Build an audience

You should spend time building an audience around your product. Some of the most successful products in the world have a strong base of fans associated with their product. This audience will become advocates for

your product, thus this type of promotion is by far the most powerful.

Building an audience takes a lot of time. It's not something that happens overnight. You need to be fully invested in this, and - as mentioned - spend at least the same amount of time on this, as you do on developing your product.

There are great communities centered around SaaS and Tech Startups. Indie Hackers is a great example as you can start networking with other SaaS founders here and you can learn from their examples.

Paid ads

Another addition to getting your SaaS in front of an audience is using paid ads. Most Social Media platforms like Twitter, Facebook, Instagram, and YouTube offer an ad program.

You typically pay a fixed price per day, and the platform then takes care of exposing your ads to the right audience. It's neither complicated nor particularly expensive to get started.

Paid exposure doesn't have the same effect as having an audience advocate for your product, but it can be a great way to create awareness about your new SaaS.

Affiliate marketing

Affiliate marketing is placed somewhere between an audience and paid ads. This marketing technique can be a great compromise when you don't have an audience of dedicated fans of your product already. Offering a commission to users that are willing to advocate for your product can be a great way to create initial trust and reputation.

The best way to get started with affiliate marketing is to reach out to people with an already existing audience on Social Media. Be aware that the ones with a massive follower base receive many offers every day, and they can be hard to reach.

In the beginning, it may be better to find users that have a smaller following. The popular term being "Micro-Influencers" with audiences between 1,000 and 10,000 followers.

Selling your SaaS Product

Getting acquired

At some point, you may want to make an exit. If your SaaS product has done well, it might make sense to find a bigger company that wants to buy it.

Getting acquired isn't a simple process, and it often takes months of negotiating back-and-forth. Making the correct evaluation of your SaaS company is a huge topic on its own, and is beyond the scope of this chapter.

I do want to mention the platform, *MicroAcquire*, which allows you to list your SaaS product(s) so that potential buyers can find it easily and see that the owner is interested in selling it.

(This page intentionally left blank, for print)

Thank you!

Thank you for purchasing this book and taking your time to read it! I really hope that you will use the knowledge you gained by reading this book to accomplish everything you are planning to achieve!

What's next?

Don't forget that it's better to start now rather than waiting until you feel ready, because in most cases, you won't feel that way. You may not feel like an expert but if you don't start, you will never become one.

As Nathan Barry says: "To be successful you need to get in the habit of starting before you feel ready."

In case you're interested in learning more on this topic, I'm working on a new project on MakeMoney.dev where I'm going to write articles on the topic, interview other developers and keep you updated with my own progress by sharing my journey in-depth.

Don’t be a stranger

Come and say hi on Twitter, let me know how you liked my first book. In case you found it valuable, please leave a review on Gumroad and share it with your friends. I would really appreciate that!

I wish you all the best on your journey!