SURFING SCRATCHER

MENU

What is Coding? The 2020 Ultimate Guide for Beginners

January 20, 2020 | Surfing Scratcher

Open tab. Close tab. Another fruitless answer.

You hear about coding everywhere, but you're still not sure what it is or even where to begin. I'm sure you suspect coding has something to do with computers, but in what way?

Some posts get into the nitty-gritty too quickly. I know when I'm learning about a topic and it's pitched above my level, it can feel overwhelming and maybe even hopeless.

Momentum halted.

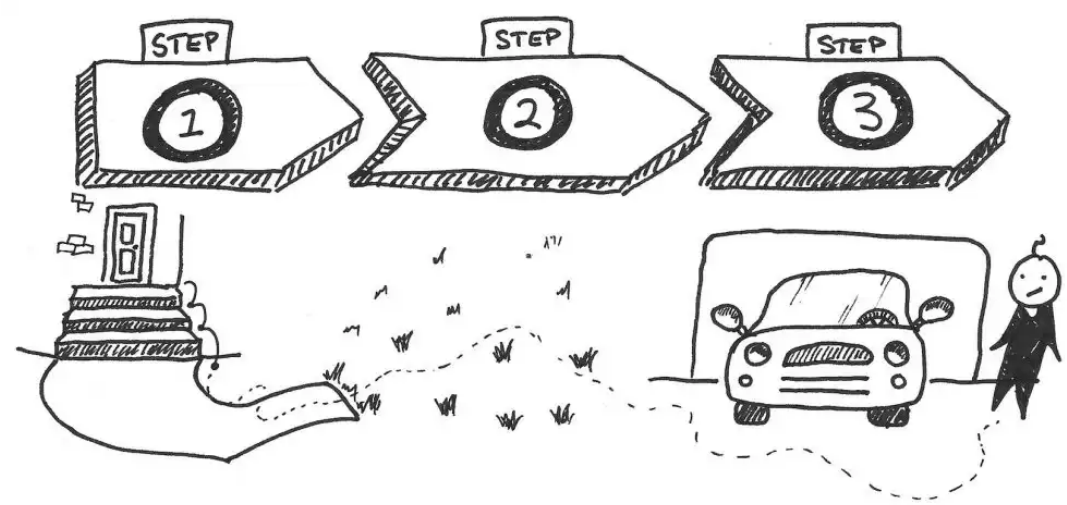
Coding might seem like this magic black box that runs the world, and in some ways that's true. Coding is kind of magical. But its magic can be learnt and understood by anyone- and quickly too.

To answer “What is coding?”, we’ll use some helpful metaphors, drawings and examples to build your understanding. Once we’ve arrived at that understanding, we can make connections to what coding is used for and the types of coding that are out there.

Lastly, you’ll leave with some actionable next steps you can take. That’s the exciting part: you get to choose your own adventure.

Okay, let’s turn mission impossible into possible.

Coding 101: What is Coding and How Does it Work?



Let’s go for a walk. Not an actual walk, but an imaginative one. You know that to walk, we must place one foot in front of the other. That’s called a step. When we repeat this action, we take multiple steps. So far, so good, right?

We walk to get from one location to another. You might walk from the front door of the house to the door of the car. Now imagine that on this walk, you must walk down some stairs, over some pavement, across some grass (for those shortcut days) and around the front of the vehicle to arrive at the driver’s side car door.

In this fictional example, we've just described a process to get from one location to another through a ***sequence of steps***. That's the essence of coding.

Coding is describing a sequence of steps or **instructions** to tell a computer what to do.

Computers are wonderful servants (but can easily become cruel masters if the dynamic is flipped). As wonderful servants, computers love to receive and perform instructions from us. Think of training a dog, except this dog listens to everything you tell it to do.

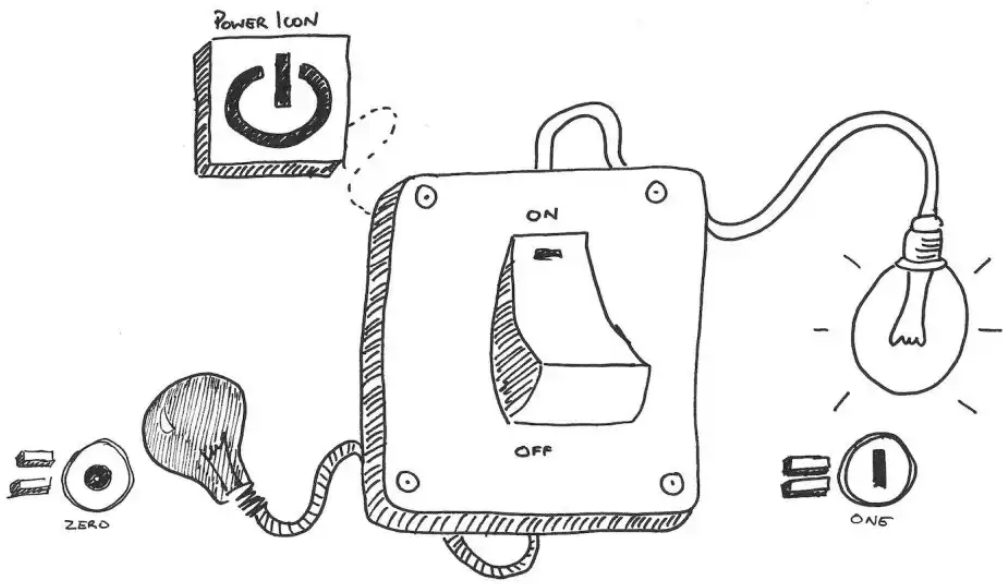
Kind of powerful isn't it?

But with great power comes great responsibility. Computers are incredibly literal and unforgiving. They'll do exactly what you tell them, even when you mess up your instructions. No matter what, computers will give us immediate feedback on how we're communicating with them.

Okay, coding so far is a list of instructions that tells the computer what to do. That's cool and all, but *how* do we tell a computer what to do?

To do that, we must first know a little about the language of computers.

Why Computer Code is a Language you don't want to learn



You can understand English (for which I'm grateful since you're here reading this post). Sadly, computers don't understand English (although artificial intelligence is changing that landscape).

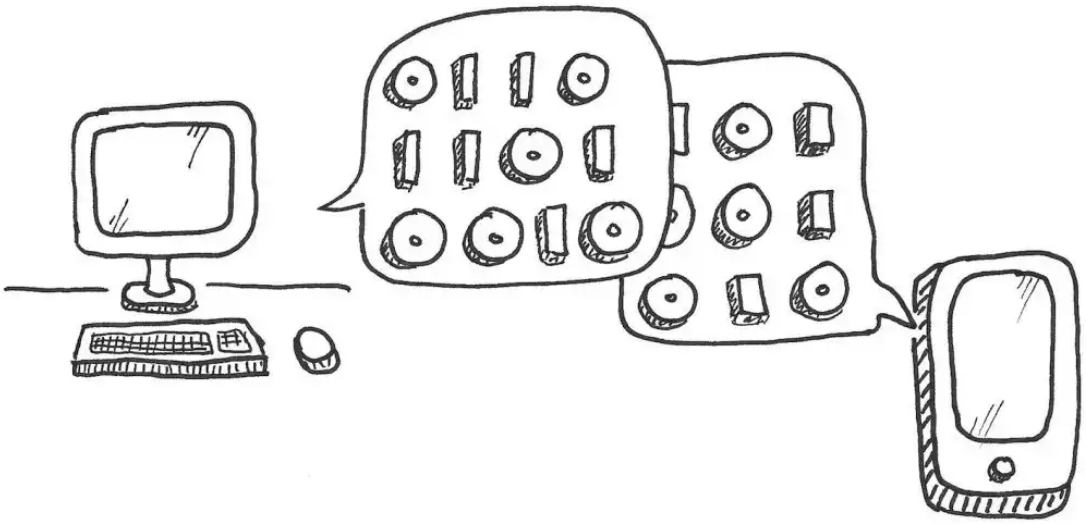
Computers have their own machine code language. To understand their language, I want you to think of a light switch. Nothing crazy here, just a switch that illuminates a light when flicked on.

The light can either be off or on. This single ***bit*** of information is like a letter in the computer's alphabet. So just what are the letters in the computer's alphabet? Well, you know them already: a zero (0), and a one (1).

Check out the power button icon on your computer (sorry mobile users, just Google a power button icon). Did you notice that the icon is a circle with a vertical dash that cuts through the top of it? That's a combined zero and one. The next time you're looking at a power board and wondering which is on and off, remember that zero means off and one means on.

Back to the computer's language.

WHAT IS CODING IN A COMPUTER? (BINARY)



A computer uses these bits of information to represent its language, which is called ***binary***. That's neat because binary literally means something that has two parts.

Now I don't know about you, but I'm going to have a hard time communicating with a computer using zeroes and ones. In order to communicate with each other, we need help.

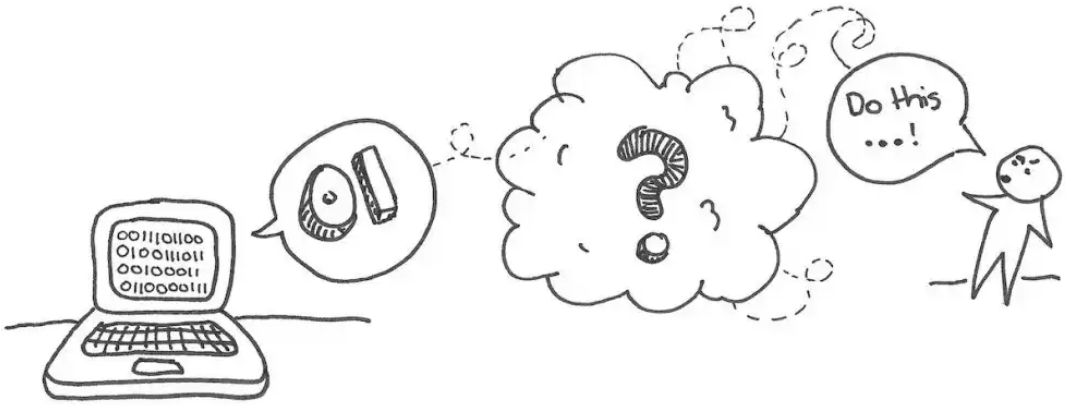
If We Don't Talk Computer Speak, Then What is a Coding Language?

Language is fascinating. It exists to communicate understanding between two things. When two people speak different languages, they'll need someone to help translate in order to reach a common understanding.

Heck, even when two people speak the same language, they might need a mediator to help them communicate with each other. That's both fascinating and slightly hilarious.

How do we reach this common understanding with a computer?

CODE AS A TRANSLATOR



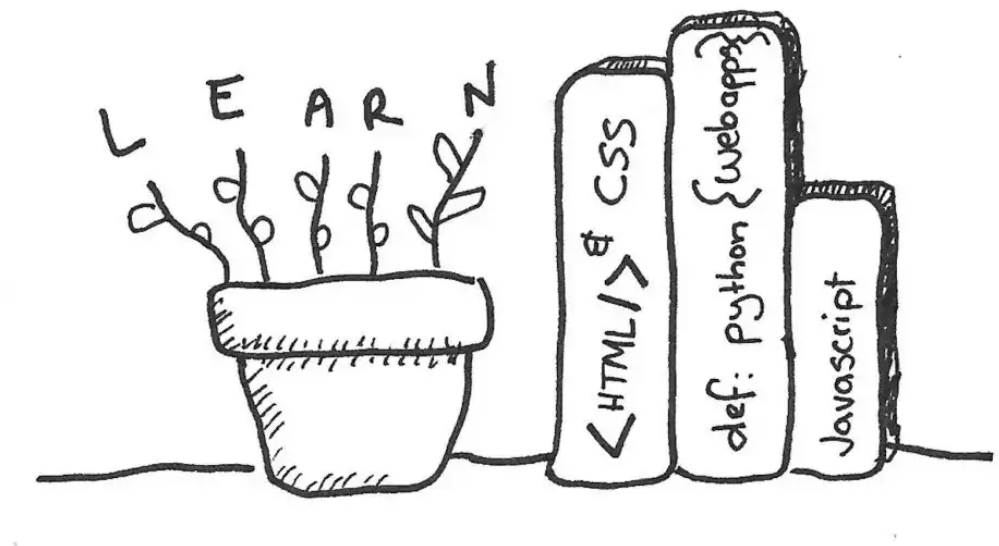
Since we don't speak binary, we need a translator so we can pass on our instructions to the computer. Who is this middleman? Enter *coding languages*. Coding languages are a mix of a little bit of English (or your own language) sprinkled with a little bit of computer grammar.

When we communicate using a coding language, what we communicate is packaged up or ***compiled*** into a form that can be understood by the computer. That's the seemingly magical part to this process. It works.

Code is a form of writing that both computers and humans can learn, understand and interpret.

So what languages can our translator speak?

Types of Coding Languages (including the Friendliest)



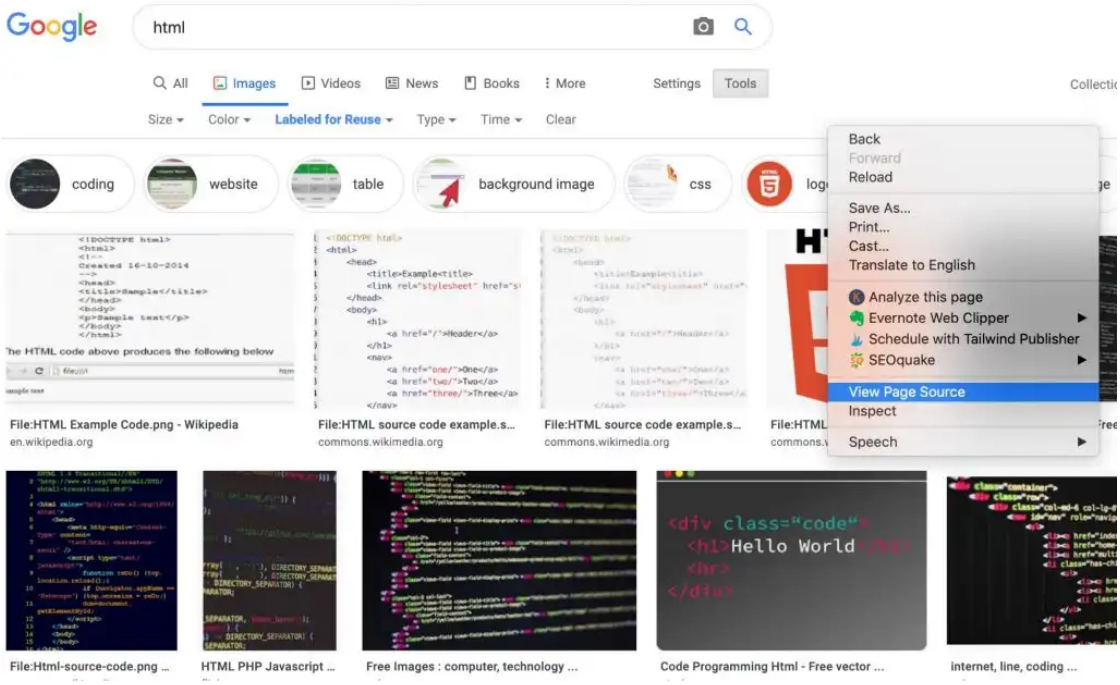
A quick Google search tells me that there's somewhere between 700 to over 2000 coding languages. Now before you go and get disheartened by those monstrous numbers, find comfort in knowing that you only need to speak a couple.

A couple? That's more than one. Hmmph!

Yep, but you'll start with a single one and expand from there. Most languages say the same thing in a different way, so it's more about learning the types of grammar than learning new words and vocabulary.

Coding, or *programming languages*, can be loosely categorised into different types. Those types aren't as interesting for a beginner, so we'll just focus on some entry-level languages. Let's start with the language of the web pages that you browse. To be completely honest, it's not even classed as a language, but it's a nice starting point.

HTML — HYPERTEXT MARKUP LANGUAGE



The word language even appears in its name! Sadly, HTML is seen as just markup. So what is markup? If you're on a computer (sorry mobile users), you can right-click your mouse and select the option that says "View page source". With this, you'll glimpse a peek into HTML.

If it looks a little overwhelming, that's okay. You don't need to understand it. What's going on, is that coders have used something called ***tags*** to tell your browser how to organise the page.

A tag is a word or letter that is sandwiched between the less-than and greater-than arrows (<, >). Tags are kind of like the frames of a house.

However, to decorate the house, we'll need something else. Hello CSS!

CSS — CASCADING STYLE SHEETS

Example

```
p {
  font-family: "Times New Roman", Times, serif;
}
```

Screenshot taken from w3schools.com

CSS is the home decorator of the Internet. Instead of decorating walls and rooms, CSS styles fonts, colours, links, headings and more.

Like HTML, it's not classed directly as a language, but it still tells the computer what to do.

Both CSS and HTML are quite simple to learn and work hand in hand with each other.

JAVASCRIPT

Example

```
function myFunction(p1, p2) {
  return p1 * p2;   // The function returns the product of p1 and p2
}
```

Screenshot taken from w3schools.org

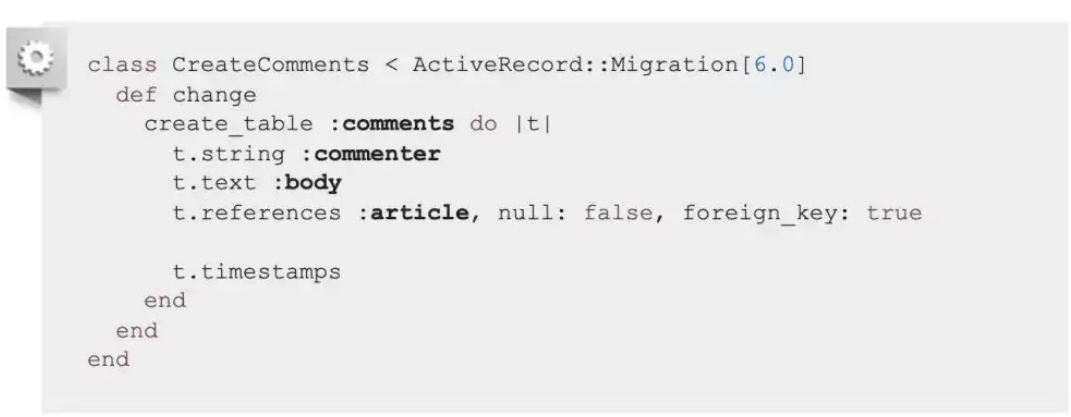
Javascript is a little like adding home automation to your house, such as turning on the lights at sunset, or turning on the hose at 8 am. Javascript performs the actions on webpages, and is responsible for controlling video players and animations to create interactive displays.

So far, all of these languages have referred to the 'what you see' part of the Internet, the web pages that you use when you're on your computer or mobile device.

But even before your house is even built, it needs plans. These plans tell the builders how to construct the house.

So which languages form these plans?

RUBY AND PYTHON



```
class CreateComments < ActiveRecord::Migration[6.0]
  def change
    create_table :comments do |t|
      t.string :commenter
      t.text :body
      t.references :article, null: false, foreign_key: true

      t.timestamps
    end
  end
end
```

Screenshot taken from guides.rubyonrails.org

If HTML, CSS and Javascript control what you see, then Ruby and Python control what you don't see.

The simplest way to understand this is to think of data. Whenever you sign up to a website, you enter some of your personal data. That data needs to be stored somewhere, so Ruby and Python help move that information to its storage location.

I'm oversimplifying it here, but Ruby and Python are a little like British English and US English. They essentially do the same thing with minor differences. The differences between Ruby and Python have more to do with grammar than word spellings, but you get the idea.

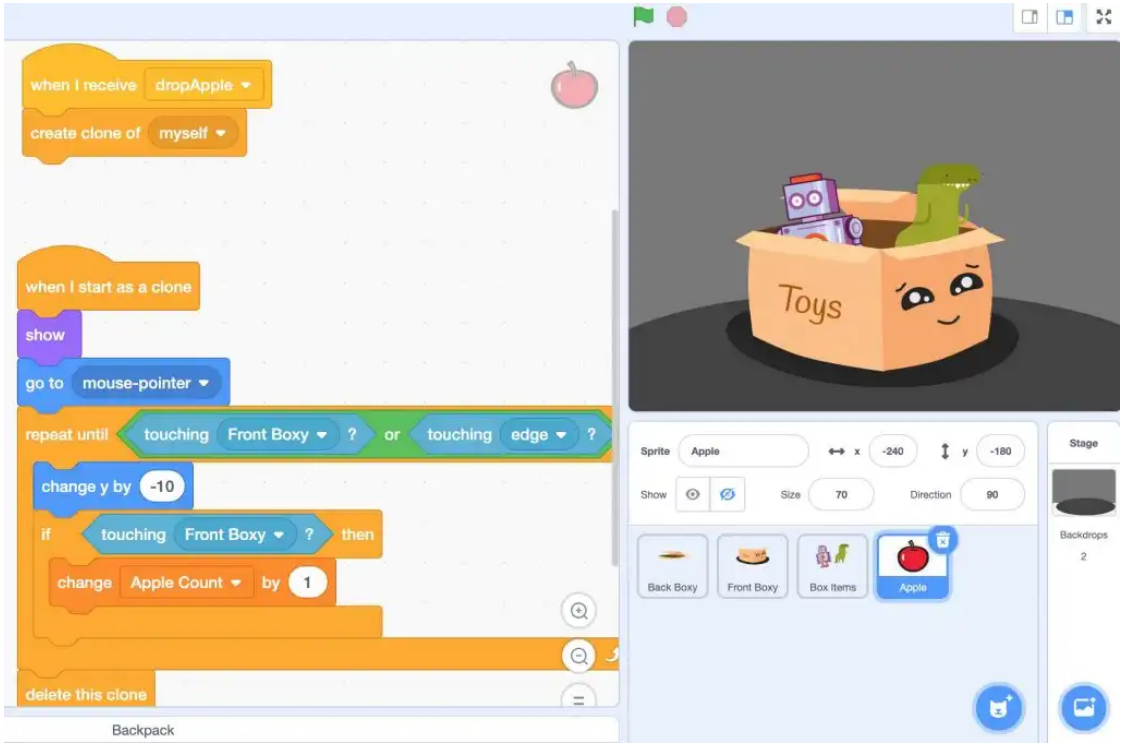
These languages are the foundation of web apps. Web apps are complex applications that are constructed to do a certain thing. Whatever it does is built into the website's underlying code.

The combination of HTML, CSS, Javascript, and Ruby or Python result in all the content and functionality you experience while browsing the web.

These languages require you to write text-based code. That can be an issue for beginners, as computers will only do what you tell them exactly to do, and they won't accept spelling or grammatical errors. Harsh, but that's just how it is.

The easiest way to get started is to use a visual programming language.

VISUAL PROGRAMMING LANGUAGES (FRIENDLIEST)



When I first encountered visual programming languages a couple of years ago, my mind was blown.

No longer do beginners have to struggle with the burden of grammar when learning to code. You can just start speaking it. But how does this work?

Visual programming languages are a lot like fitting Lego blocks together to build something. Each block represents a piece of code that's associated with it. In other words, the code is hidden inside the block.

Some visual programming languages include Scratch and Blockly. They are wonderful entry-level points that get you creating right away.

Now that you have some familiarity with programming languages, let's explore who uses them.

What is Coding Used for: Types of Coders and Their Superpowers

FRONT-END DEVELOPER (HTML, CSS, JAVASCRIPT)

Front-end developers build the part of a website that you, the user, can see and interact with. In other words, they are the builders and decorators of the house.

They use markup and scripting languages to do this.

Typically, front-end developers take designs and translate them into web pages. They'll make sure that a website is functional and is easy to use. It's also the job of the front-end developer to make sure pages load quickly. Nobody wants to be waiting around for a long time just to view content.

A front-end developer has many roles, and learning them is quite fun. Because it is linked to the visual aspect of the web pages, you can instantly see your handiwork, which is a rewarding experience.

BACK-END DEVELOPER (RUBY, PYTHON)

```
>>> from django.apps import apps
>>> apps.get_app_config('admin').verbose_name
'Administration'
```

Screenshot taken from docs.djangoproject.com

The architects and planners are the back-end developers. The role of the back-end developer is to construct functioning web applications. For example, when you purchase something online, that transaction results in many contact points between

various third parties. It's the job of the back-end developer to make sure that all communication happens securely and swiftly.

Back-end developers requires a sound working knowledge of data structures and patterns in order to build web apps. They are more concerned with the underlying architecture of the application than with how things appear visually.

You probably wouldn't begin your journey as a back-end developer, but it's often a logical endpoint. You'll develop an understanding and level of proficiency with most types of roles on your coding journey.

MOBILE APP DEVELOPER (SWIFT, JAVA)

```
let apples = 3
let oranges = 5
let appleSummary = "I have \(apples) apples."
let fruitSummary = "I have \(apples + oranges) pieces of fruit."
```

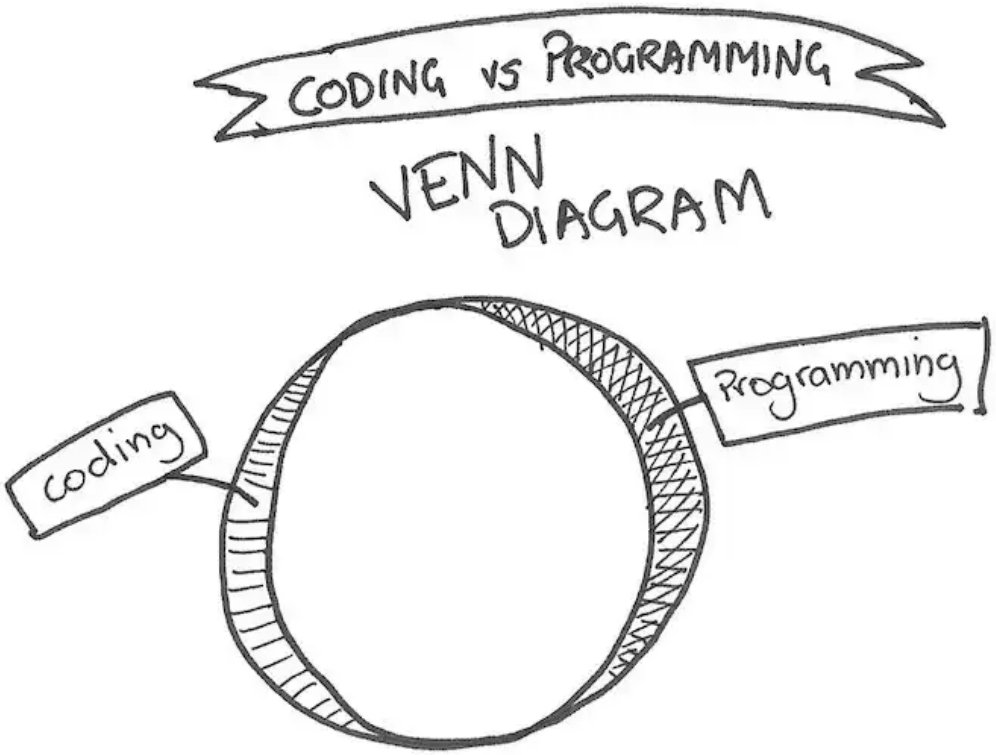
Screenshot taken from docs.swift.org

Mobile app developers build *native* apps for mobile devices. The word native means using the language that the device speaks. The native programming languages for iOS devices are Swift and Objective-C. Android devices, on the other hand, are built using Java, which is not to be confused with Javascript.

Whenever an app is downloaded from the device's store, it's most likely been constructed using the device's native language.

Whichever language you choose to pursue, once you've learnt one, you can transfer the principles across to another. The Rosetta stone is just learning one language first.

What is the Difference Between Coding and Programming?



You may have noticed that I started with coding languages and ended up using the term *programming languages* in the preceding section. I did so because the latter is the preferred term when talking about languages. But what's the difference between coding and programming?

Honestly, the terms are interchangeable. If you wanted to get a little more specific you could argue that coders just create code, while programmers create programs or apps.

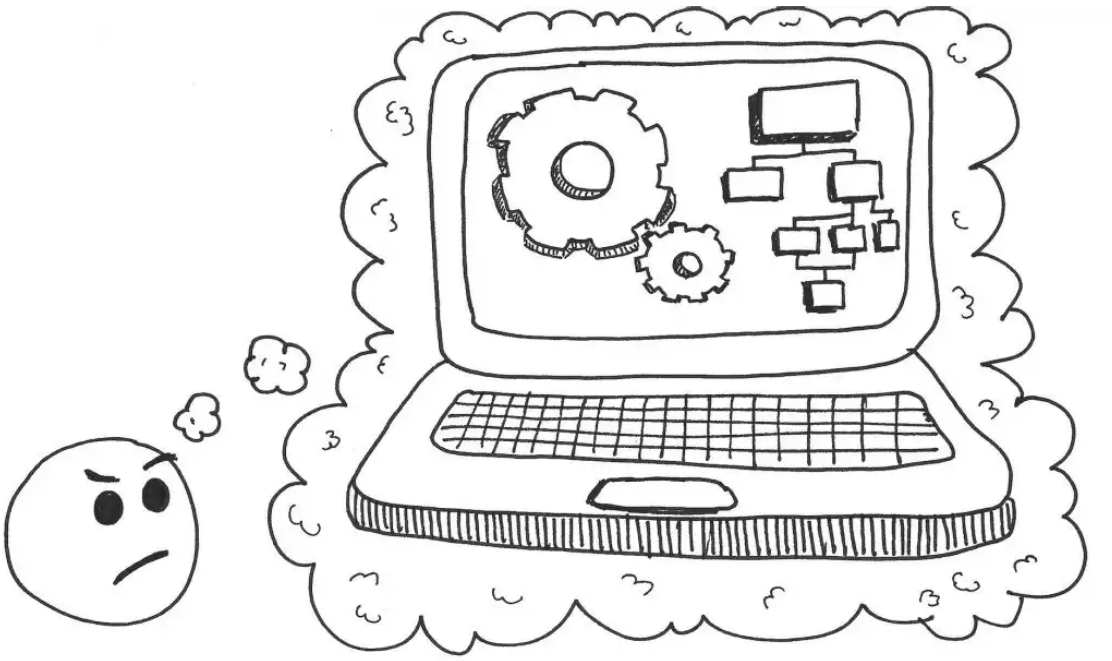
Can you distinguish the difference?

If you jump onto a website to complete a free coding course, you'll be coding.

However, if you take your coding skills and build a game or application, then you've programmed something.

That's how I would distinguish between the two, but don't waste too much brainpower on this one.

How Learning to Code Will Impact Your Life



I can't speak for you, but I can give you some insight into how learning to code has impacted my life.

Computational Thinking

Learning to code has equipped me with essential problem-solving skills that I can use in virtually any context. I can use these skills in the kitchen, in relationships (you read that right) and when learning a new skill.

Coding requires programmers to deconstruct large problems into smaller chunks. This process is referred to as **decomposition** and forms a part of **computational**

thinking. For more detail, head to this post.

Creator and Consumer

When you learn to code, you become a contributor to the process of creation. You will no longer be just a consumer of digital media and technology. You've become a part of the process.

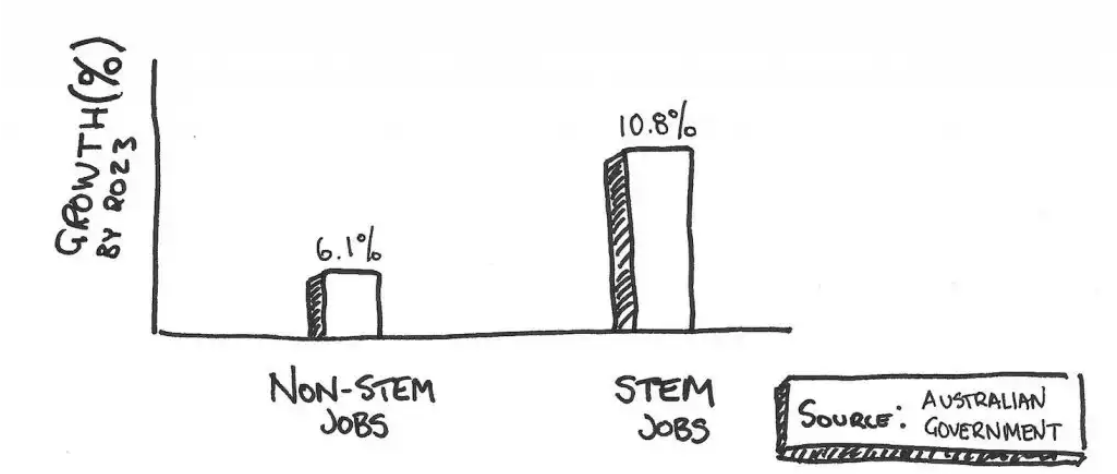
You'll develop a deeper level of understanding of the technology that you use. Once that understanding is acquired, you'll be able to see patterns and routines that were previously hidden.

Automation

When you zoom out, learning to code is more a way of thinking than just an act of doing. Once you become clearer with your thinking, you can better articulate that thinking. If that becomes crystal clear, you can create instructions that others can follow and remove yourself from the process. This is called automation.

Automating your processes means you'll free up precious time. Automation extends beyond coding through the act of delegation. We can delegate work we'd otherwise do to another person or computer.

Career Prospects



Source: employment.gov.au

Lastly, if you pursue a career in coding, you'll find that it's a stable field in high demand. If you persist with your coding endeavours, you'll end up in some lucrative roles. For some, that's perhaps the most important impact, but I would regard computational thinking as more impactful.

Whatever your flavour preferences, pursuing coding will have a positive impact on your thinking and stability. Sure, it's competitively early, but so are most professions. Persistence and determination will help you succeed.

What's the Best Way to Learn to Code?

Now that you have the answer to what coding is, it's time to take the next step in the sequence.

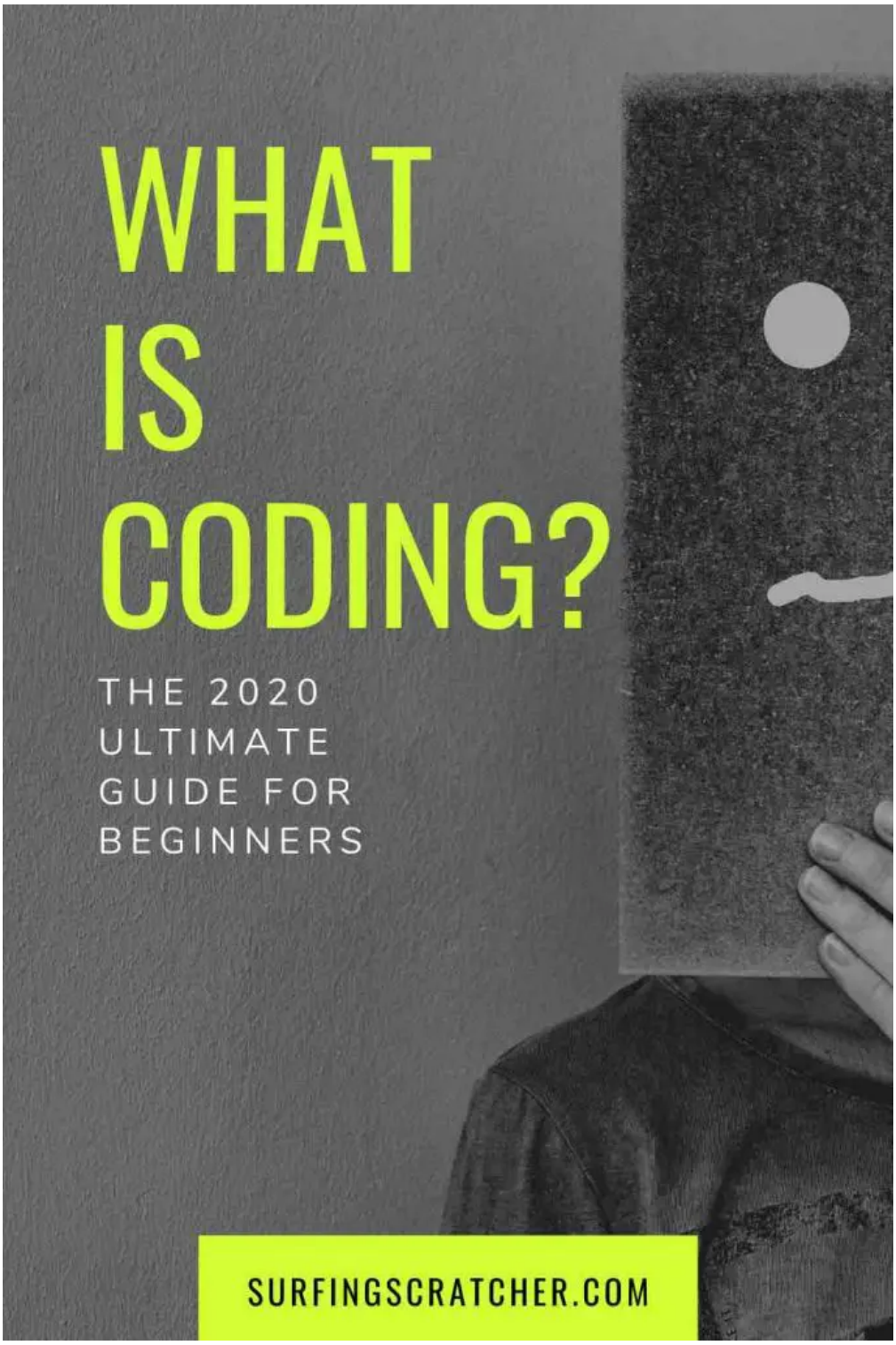
You've learned that coding is just telling a computer what to do through a programming language. Now, it's time to choose your own adventure picking a language to start learning.

If you're an absolute beginner, start with something like Scratch. Head to the website and follow some of their tutorials, or check out some of mine.

If you're committed and know what kind of coding appeals to you (front-end, back-end, mobile) then choose a language from those options. This will form your foundation and when you decide to study something more formal, you'll have momentum. That feeling of 'oh, I get this!'

That's hope, Satisfaction, and that's you on the path to becoming a coder. I mean programmer. In any case, the fun begins now.

Now get coding!



Pin it!

Beginner Guides, Ultimate Guides

coding for teachers, what is coding

< 4 Benefits of Why Every Child Should Learn to Code in 2020

> Coding for Teachers: The Kickstart Guide for 2020

LEAVE A REPLY

Enter your comment here...

JOIN THE SURFING SCRATCHER COMMUNITY

Join the Surfing Scratcher community to stay in the loop with resources, news and courses.

Email Address

YES, COUNT ME IN!

LOOKING FOR SOMETHING?

Home

Blog

About

Contact

GET FREE RUBRICS

FACEBOOK TWITTER LINKEDIN