



C++ Beginner's Mistakes

There are many pitfalls the C++ developer can encounter, especially as this is a more complex and often unforgiving language to master. Beginners need to take C++ a step at a time and digest what they've learned before moving on.

VOID(C++, MISTAKES)

Admittedly it's not just C++ beginners that make the kinds of errors we outline on these pages, even hardened coders are prone to the odd mishap here and there. Here are some common issues to try and avoid.

UNDECLARED IDENTIFIERS

A common C++ mistake, and to be honest a common mistake with most programming languages, is when you try and output a variable that doesn't exist. Displaying the value of x on-screen is fine but not if you haven't told the compiler what the value of x is to begin with.

```
File Edit View Search Tools Documents Help
test1.cpp x
#include <iostream>

int main()
{
    std::cout << x;
}
```

STD NAMESPACE

Referencing the Standard Library is common for beginners throughout their code, but if you miss the std:: element of a statement, your code errors out when compiling. You can combat this by adding:

```
using namespace std;
```

Under the #include part and simply using cout, cin and so on from then on.

```
#include <iostream>
using namespace std;

int main()
{
    int a, b, c, d;
    a=10;
    b=20;
    c=30;
    d=40;

    cout << a, b, c, d;
}
```

SEMICOLONS

Remember that each line of a C++ program must end with a semicolon. If it doesn't then the compiler treats the line with the missing semicolon as the same line with the next semicolon on. This creates all manner of problems when trying to compile, so don't forget those semicolons.

```
#include <iostream>

int main()
{
    int a, b, c, d;
    a=10;
    b=20;
    c=30;
    d=40;

    std::cout << a, b, c, d;
}
```

GCC OR G++

If you're compiling in Linux then you will no doubt come across gcc and g++. In short, gcc is the Gnu Compiler Collection (or Gnu C Compiler as it used to be called) and g++ is the Gnu ++ (the C++ version) of the compiler. If you're compiling C++ then you need to use g++, as the incorrect compiler drivers will be used.

```
david@mint-mate ~/Documents
File Edit View Search Terminal Help
david@mint-mate ~/Documents $ gcc test1.cpp -o test
/tmp/ccA5zhtg.o: In function `main':
test1.cpp:(.text+0x2a): undefined reference to `std::cout'
test1.cpp:(.text+0x2f): undefined reference to `std::ostream'
/tmp/ccA5zhtg.o: In function `__static_initialization_and_destruction_0':
test1.cpp:(.text+0x5d): undefined reference to `std::ios_base'
test1.cpp:(.text+0x6c): undefined reference to `std::ios_base'
collect2: error: ld returned 1 exit status
david@mint-mate ~/Documents $ g++ test1.cpp -o test
david@mint-mate ~/Documents $
```

COMMENTS (AGAIN)

Indeed the mistake of never making any comments on code is back once more. As we've previously bemoaned, the lack of readable identifiers throughout the code makes it very difficult to look back at how it worked, for both you and someone else. Use more comments.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

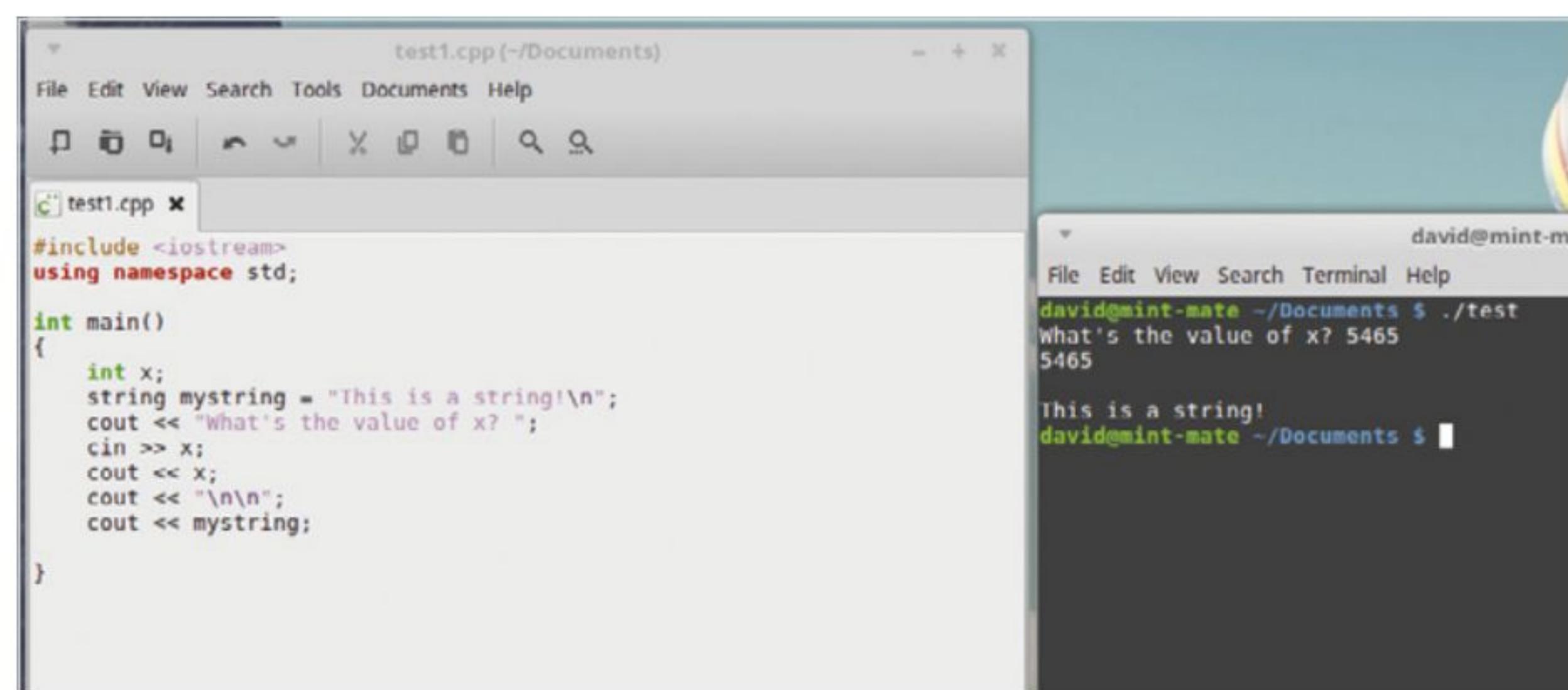
int main()
{
    vector<double> v;

    double d;
    while(cin>>d) v.push_back(d); // read elements
    if (!cin.eof()) { // check if input failed
        cerr << "format error\n";
        return 1; // error return
    }

    cout << "read " << v.size() << " elements\n";
    reverse(v.begin(),v.end());
    cout << "elements in reverse order:\n";
    for (int i = 0; i<v.size(); ++i) cout << v[i] << '\n';
    return 0; // success return
}
```

QUOTES

Missing quotes is a common mistake to make, for every level of user. Remember that quotes need to encase strings and anything that's going to be outputted to the screen or into a file, for example. Most compilers errors are due to missing quotes in the code.



EXTRA SEMICOLONS

While it's necessary to have a semicolon at the end of every C++ line, there are some exceptions to the rule. Semicolons need to be at the end of every complete statement but some lines of code aren't complete statements. Such as:

```
#include
if lines
switch lines
```

If it sounds confusing don't worry, the compiler lets you know where you went wrong.

```
// Program to print positive number entered by the user
// If user enters negative number, it is skipped

#include <iostream>
using namespace std;

int main()
{
    int number;
    cout << "Enter an integer: ";
    cin >> number;

    // checks if the number is positive
    if ( number > 0)
    {
        cout << "You entered a positive integer: " << number << endl;
    }

    cout << "This statement is always executed.";
    return 0;
}
```

TOO MANY BRACES

The braces, or curly brackets, are beginning and ending markers around blocks of code. So for every { you must have a }. Often it's easy to include or miss out one or the other facing brace when writing code; usually when writing in a text editor, as an IDE adds them for you.

```
#include <iostream>
using namespace std;

int main()
{
    int x;
    string mystring = "This is a string!\n";
    cout << "What's the value of x? ";
    cin >> x;
    cout << x;
    {
        cout << "\n\n";
        cout << mystring;
    }
}
```

INITIALISE VARIABLES

In C++ variables aren't initialised to zero by default. This means if you create a variable called x then, potentially, it is given a random number from 0 to 18,446,744,073,709,551,616, which can be difficult to include in an equation. When creating a variable, give it the value of zero to begin with: x=0.

```
#include <iostream>
using namespace std;

int main()
{
    int x;
    x=0;

    cout << x;
}
```

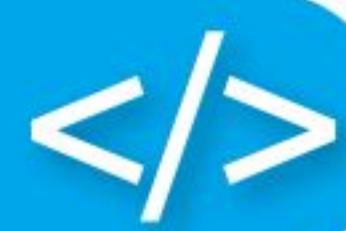
A.OUT

A common mistake when compiling in Linux is forgetting to name your C++ code post compiling. When you compile from the Terminal, you enter:

g++ code.cpp

This compiles the code in the file code.cpp and create an a.out file that can be executed with ./a.out. However, if you already have code in a.out then it's overwritten. Use:

g++ code.cpp -o nameofprogram



Where Next?

Coding, like most subjects, is a continual learning experience. You may not class yourself as a beginner any more but you still need to test your code, learn new tricks and hacks to make it more efficient and even branch out and learn another programming language.

#INCLUDE<KEEP ON LEARNING>

What can you do to further your skills, learn new coding practises, experiment and present your code and even begin to help others using what you've experienced so far?

TWITTER

Twitter isn't all trolls and antagonists, among the well-publicised vitriol are some genuine people who are more than willing to spread their coding knowledge. We recommend you find a few who you can relate to and follow them. Often they post great tips, hacks and fixes for common coding problems.

KEEP CODING

If you've mastered Python fairly well, then turn your attention to C++ or even C#. Still keep your Python skills going but learning a new coding language keeps the old brain ticking over nicely and gives you a view into another community, and how they do things differently.

```
#pragma once
#define EXPAND_ME __FILE__ " : " __LINE__
constexpr auto pi = 314LL;
thread_local decltype(pi) rage = 0b10;

[[deprecated("abc")]] char16_t *f() noexcept { return nullptr; }

template <typename T> struct Foo {
    static_assert(std::is_floating_point<T>::value,
                 "Foo<T>: T must be floating point");
};

struct A final : Foo {
    A() = default;
    [[noreturn]] virtual void foo() override;
};

template <typename T> concept bool EqualityComparable = requires(T a
{ a == b } -> bool;
```

OPEN PROJECTS

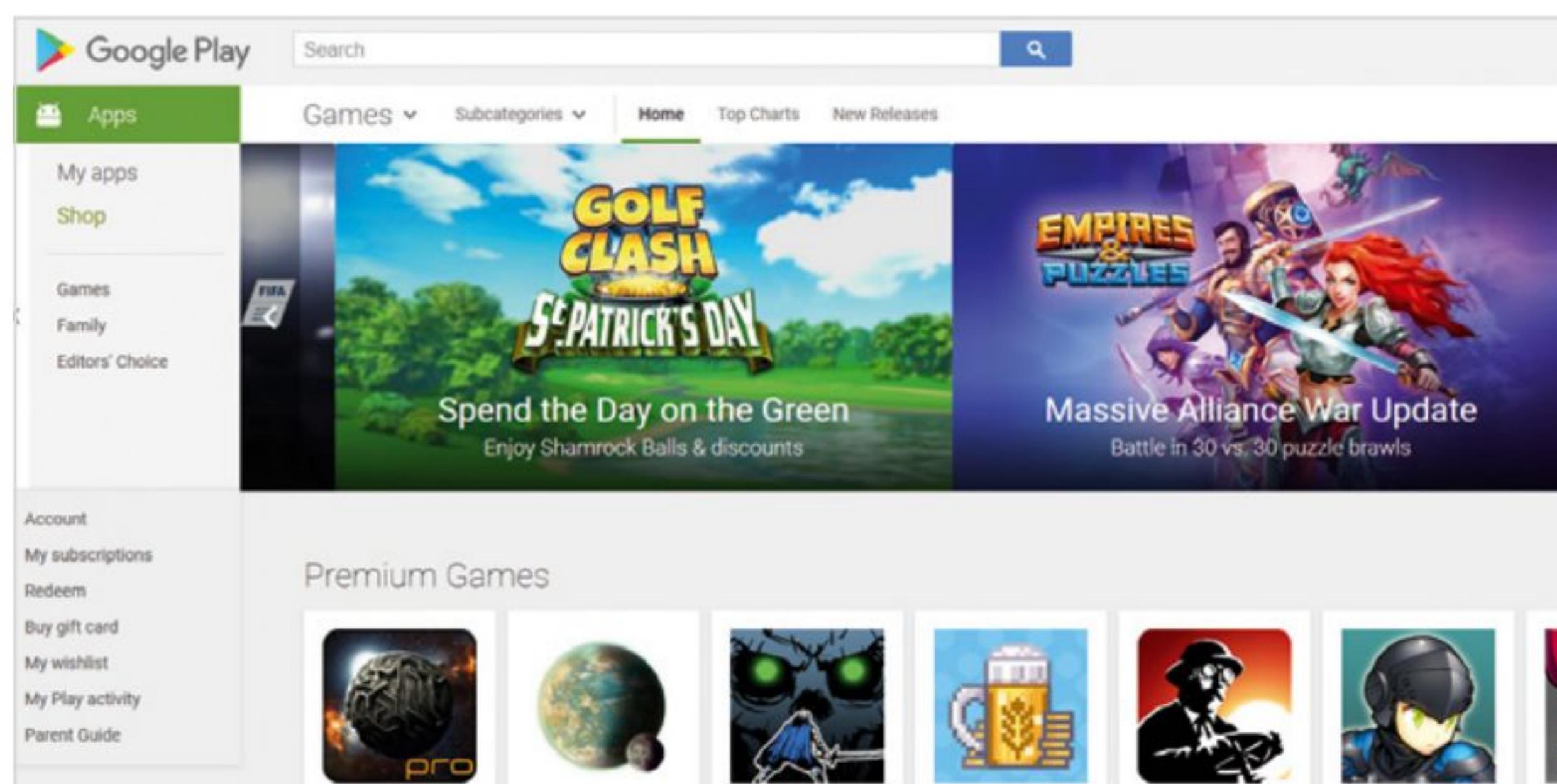
Look for open source projects that you like the sound of and offer to contribute to the code to keep it alive and up to date. There are millions of projects to choose from, so contact a few and see where they need help. It may only be a minor code update but it's a noble occupation for coders to get into.

SHARE SKILLS

Become more active on coding and development knowledge sites, such as StackExchange. If you have the skills to start and help others out, not only will you feel really good for doing so but you can also learn a lot yourself by interacting with other members.

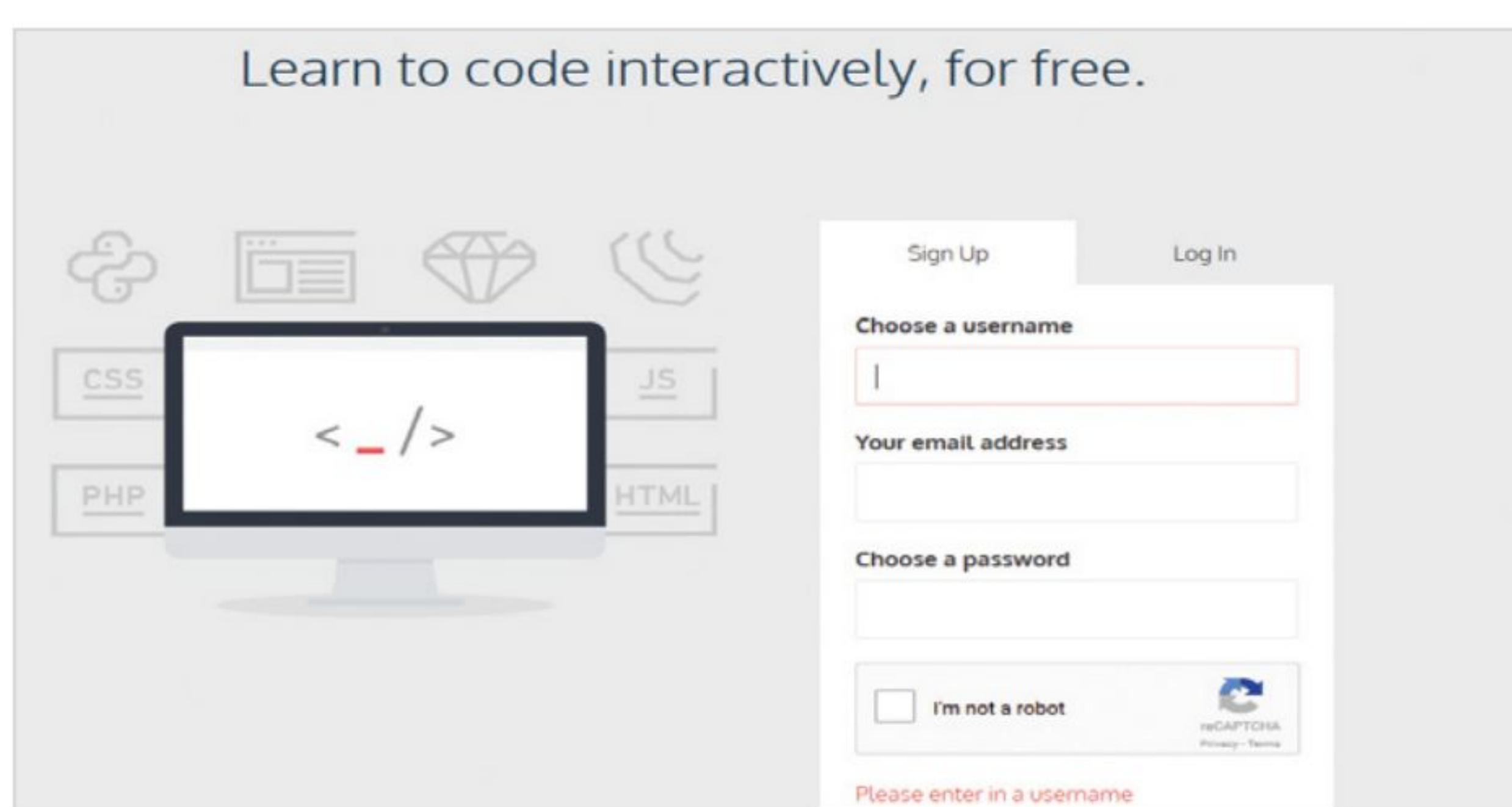
GOING MOBILE

The mobile market is a great place to test your coding skills and present any games or apps you've created. If your app is good, then who knows, it could be the next great thing to appear on the app stores. It's a good learning experience nevertheless, and something worth considering.



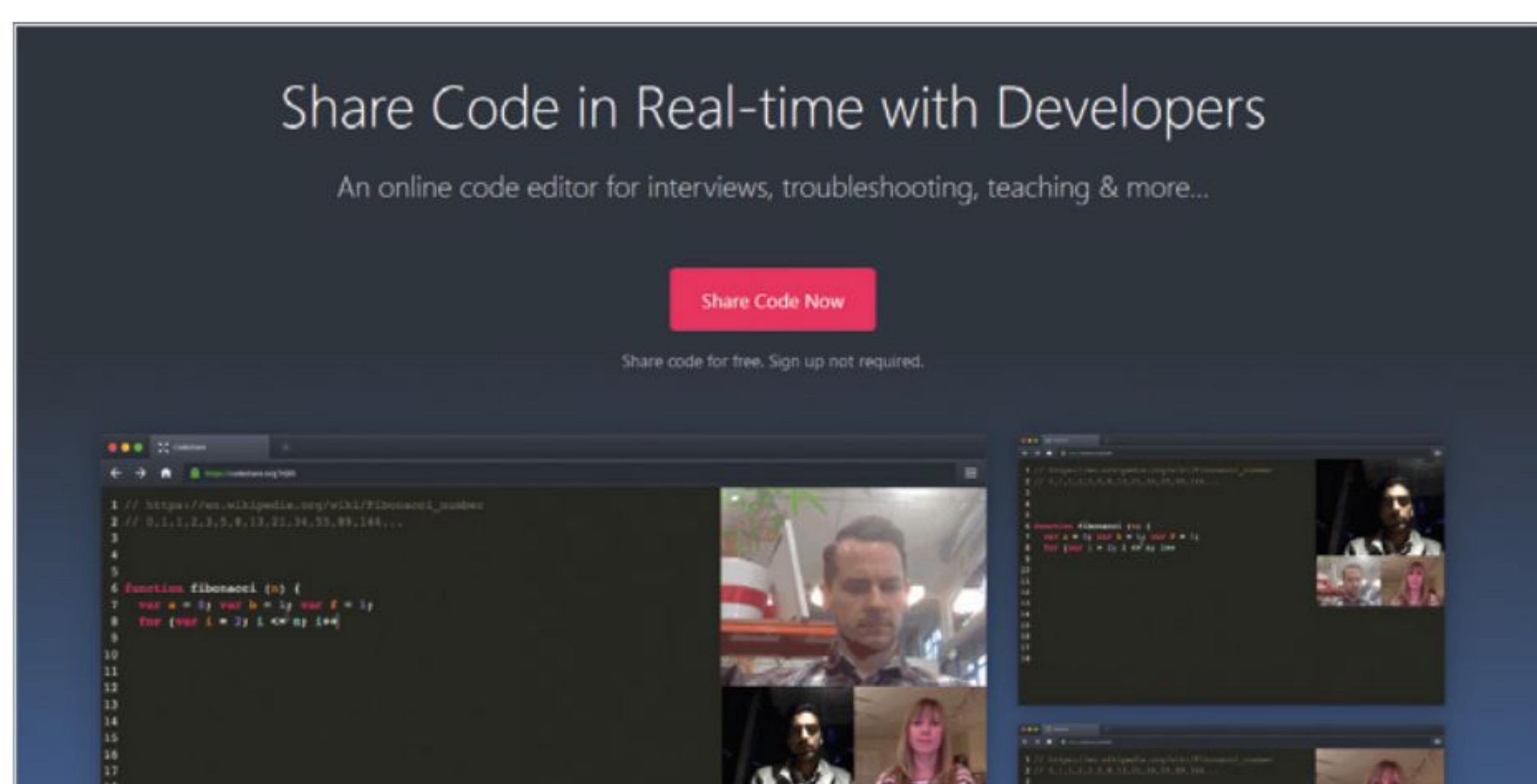
ONLINE LEARNING

Online courses are good examples of where to take your coding skills next, even if you start from the beginner level again. Often, an online course follows a strict coding convention, so if you're self-taught then it might be worth seeing how other developers lay out their code, and what's considered acceptable.



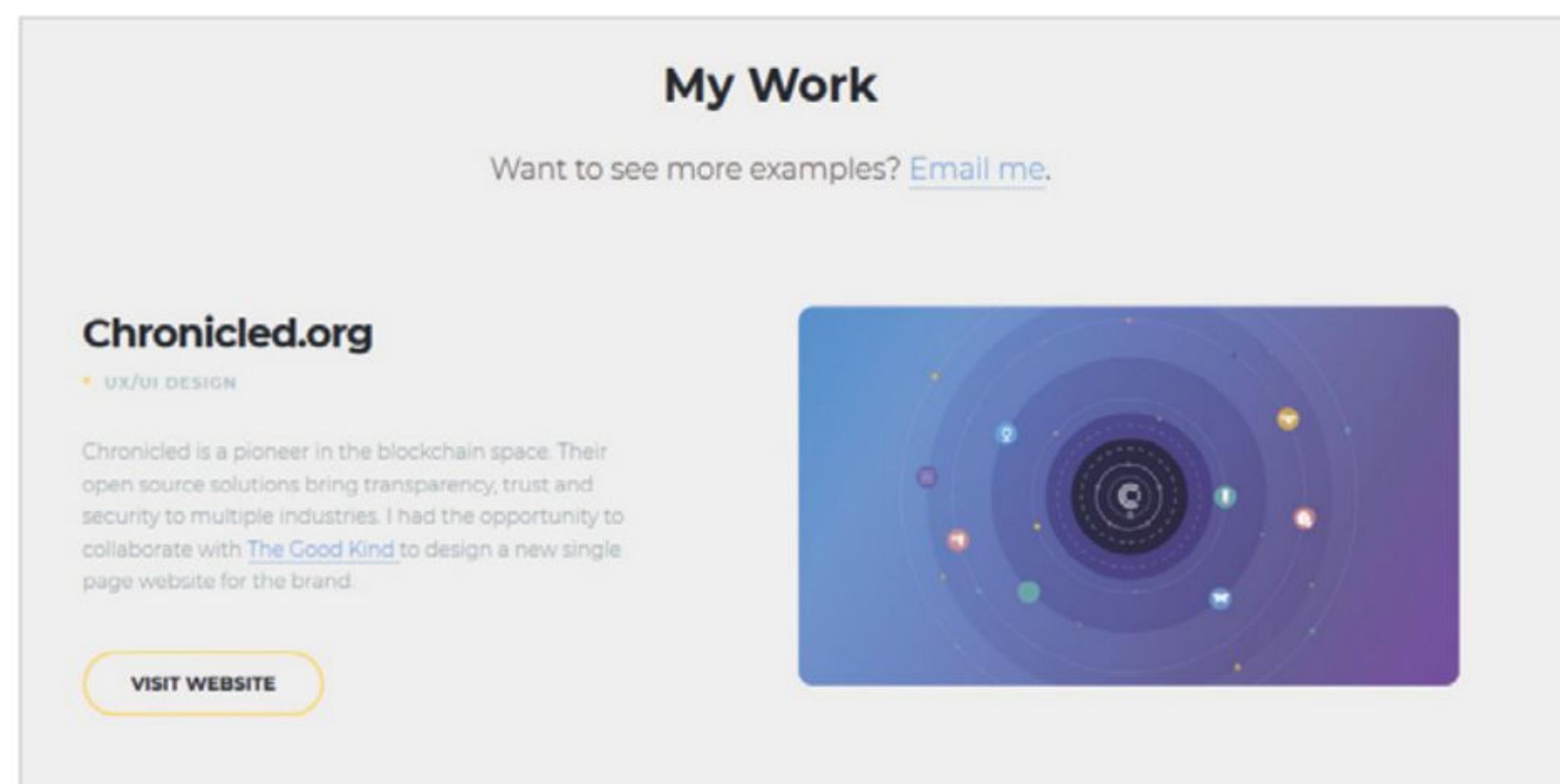
SHARE CODE

Get sharing, even if you think your code isn't very good. The criticism, advice and comments you receive back help you iron out any issues with your code, and you add them all to your checklist. Alternatively your code might be utterly amazing but you won't know unless you share it.



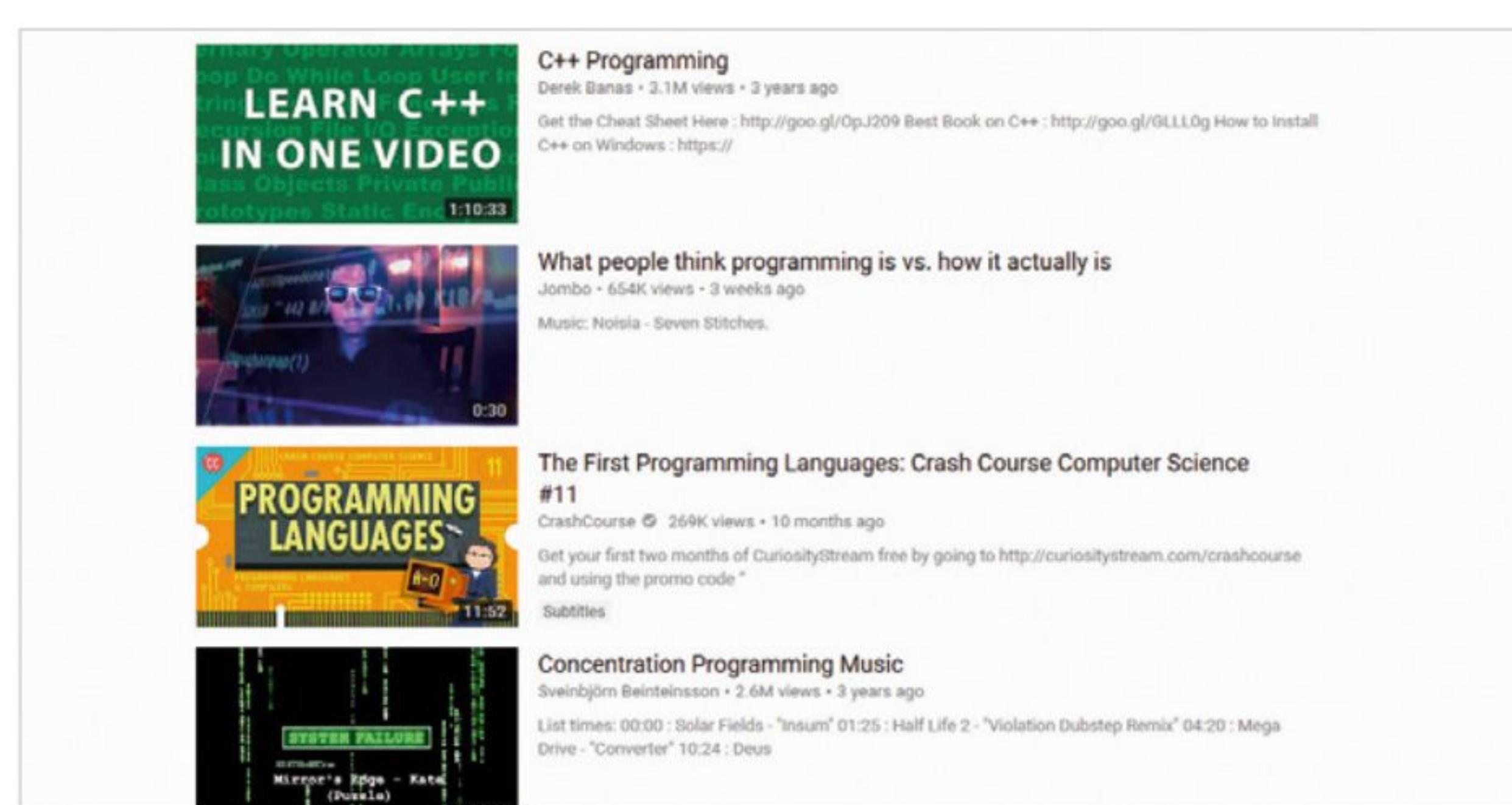
PORTFOLIOS

If you've learned how to code with an eye for a developer job in the future, then it's worth starting to build up an online portfolio of code. Look at job postings and see what skills they require, then learn and code something with those skills and add it to the portfolio. When it comes to applying, include a link to the portfolio.



TEACH CODE

Can you teach? If your coding skills are spot on, consider approaching a college or university to see if they have need for a programming language teacher, perhaps a part-time or evening course. If not teaching, then consider creating your own YouTube how to code channel.



HARDWARE PROJECTS

Contributing to hardware projects is a great resource for proving your code with others and learning from other contributors. Many of the developer boards have postings for coders to apply to for hardware projects, using unique code to get the most from the hardware that's being designed.





Black Dog Media

Master Your Tech

From Beginner to Expert

To continue learning more about your tech visit us at:

www.bdmpublications.com

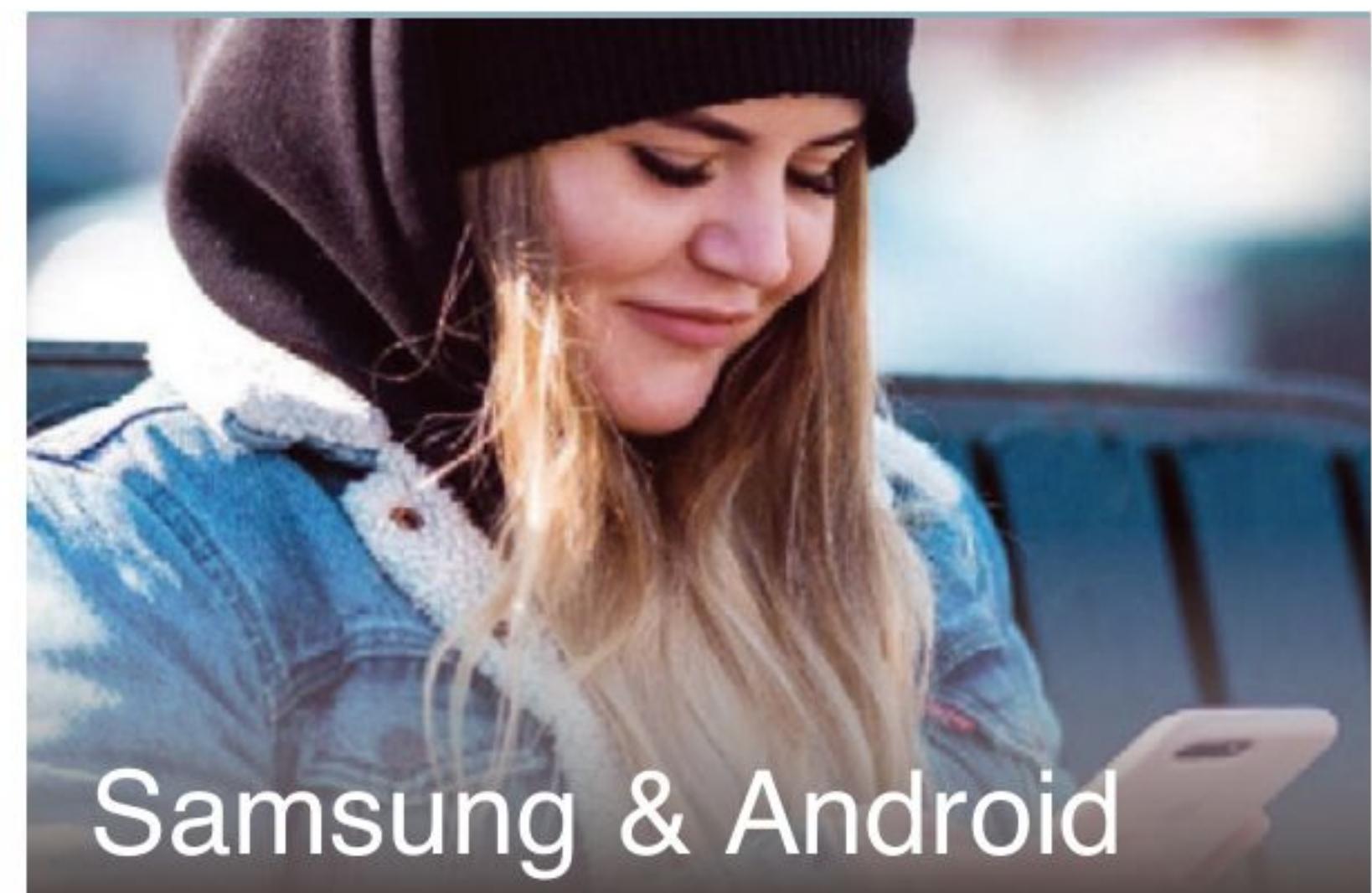
FREE Tech Guides



Apple iPhone, iPad,
Mac, MacBook & Watch



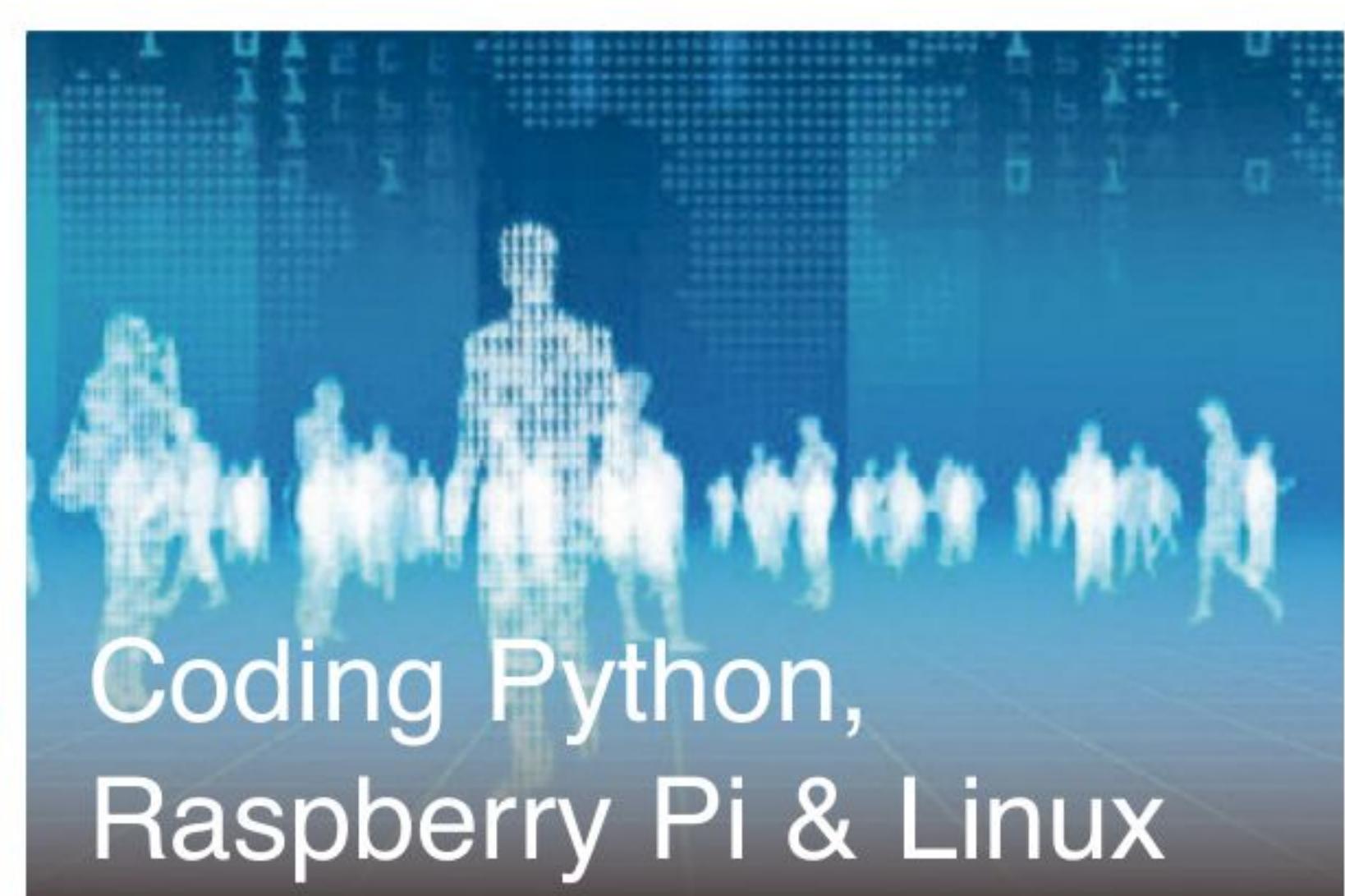
PC & Windows 10



Samsung & Android

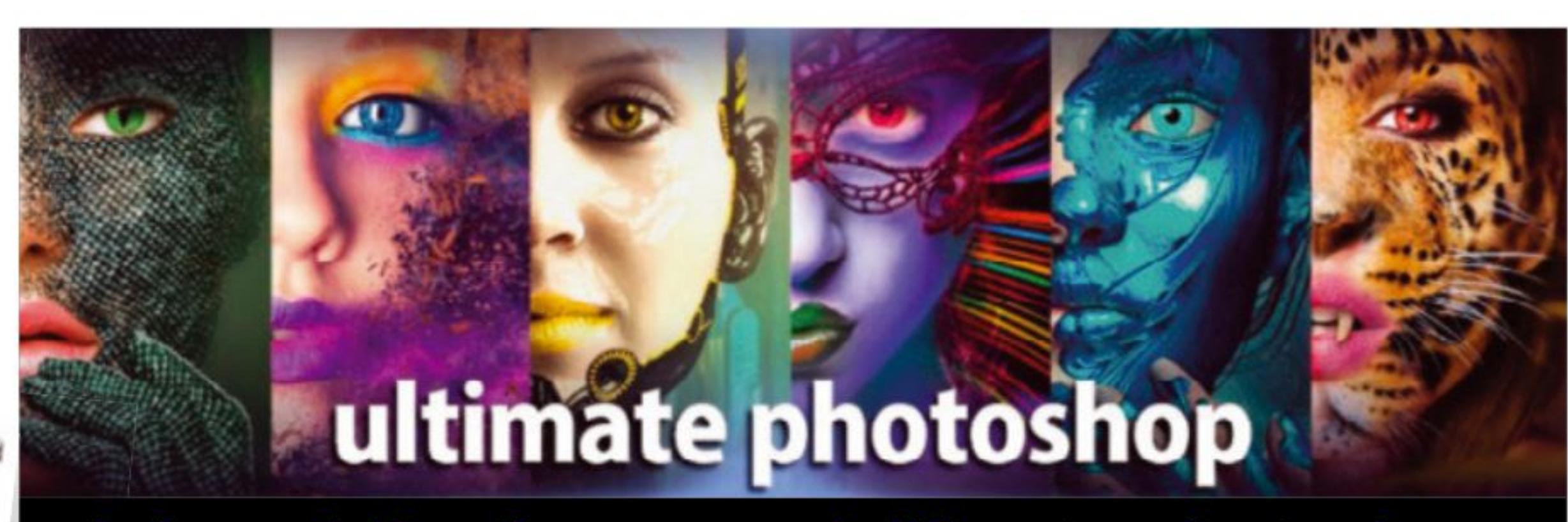
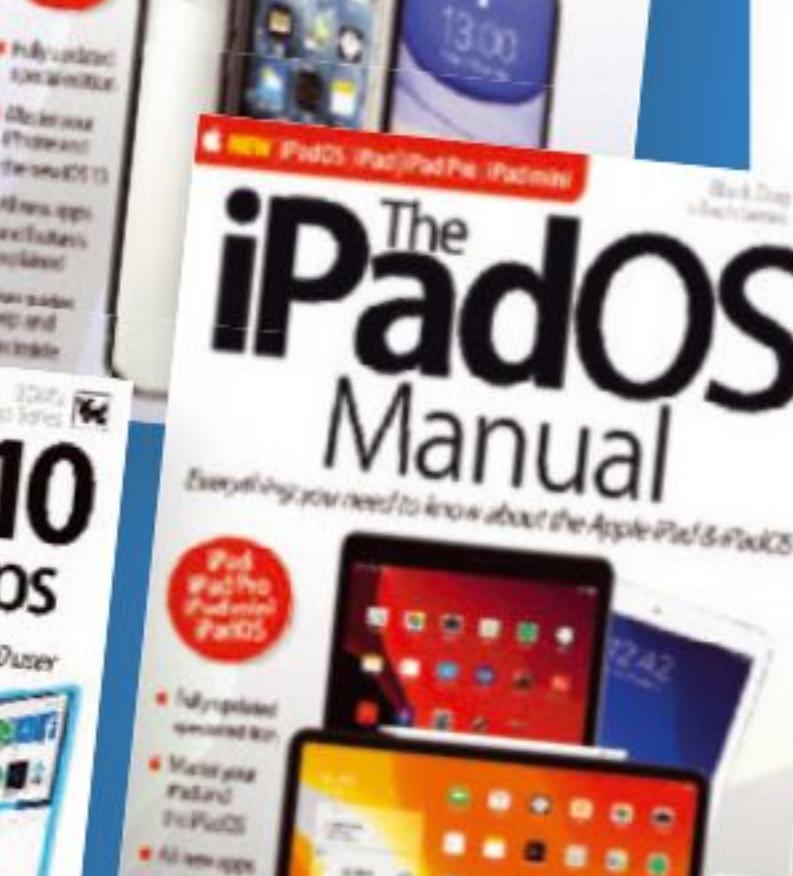
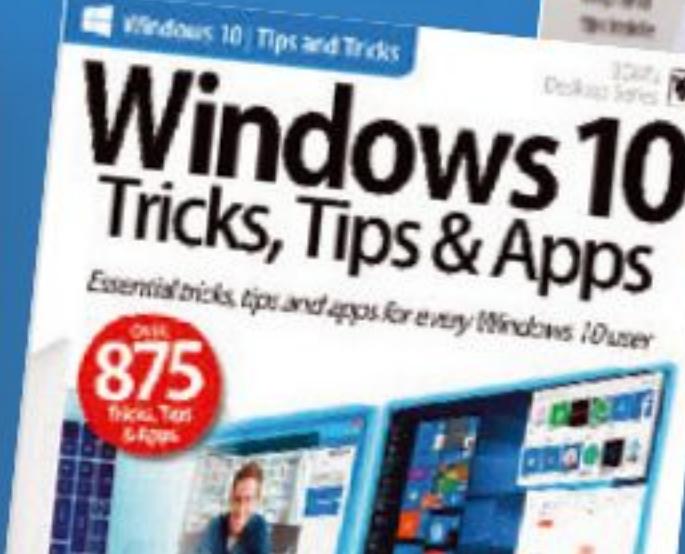
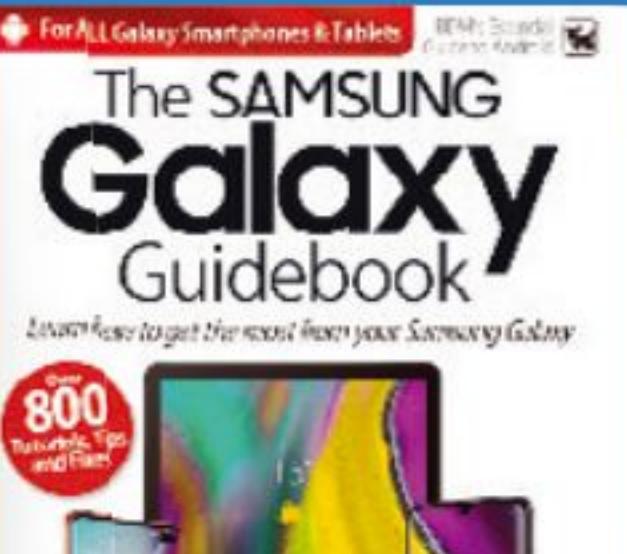
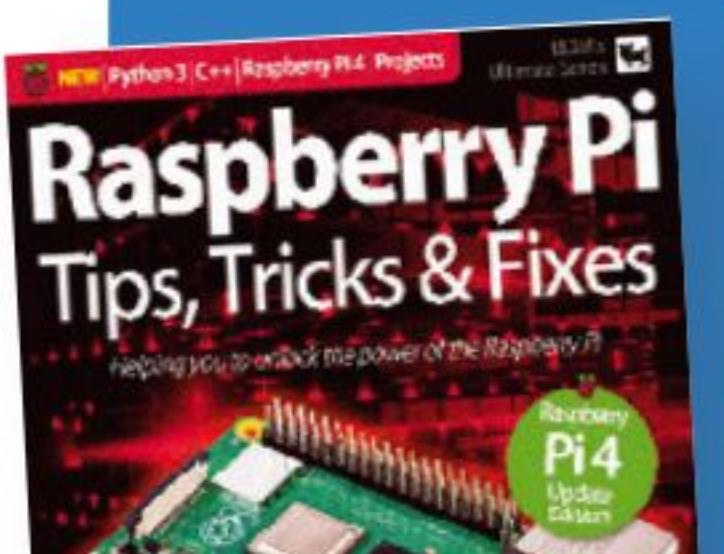


Photography,
Photoshop & Lightroom



Coding Python,
Raspberry Pi & Linux

EXCLUSIVE Offers on our Tech Guidebooks



bdmpublications.com/ultimate-photoshop

Buy our Photoshop guides and download
tutorial images for free!

Simply sign-up and get creative.

BDM's Manual Series

5th Edition

ISSN 2056-6662

Published by: Black Dog Media Limited (BDM)

Editor: James Gale

Art Director & Production: Mark Ayshford

Production Manager: Karl Linstead

Design: Robin Drew, Martin Smith, Lena Whitaker

Editorial: Russ Ware, David Hayward

Sub Editor: Alison Drew

Printed and bound in Great Britain by: Acorn Web Offset Ltd

Newsstand distribution by: Seymour Distribution Limited
2, East Poultry Avenue London EC1A 9PT

International distribution by:

Pineapple Media Limited www.pineapple-media.com

Digital distribution: Readly AB, Pocketmags, Zinio, Cafeyn, Magzter

For all advertising and promotional opportunities contact:
enquiries@bdmpublications.com

INTERNATIONAL LICENSING

Black Dog Media has many great publications and all are available for licensing worldwide. For more information go to: www.brucесawfordlicensing.com; email: bruce@brucесawfordlicensing.com telephone: 0044 7831 567372

Copyright © 2020 Black Dog Media. All rights reserved.

Editorial and design are the copyright © Papercut Limited and are reproduced under licence to Black Dog Media. No part of this publication may be reproduced in any form, stored in a retrieval system, or integrated into any other publication, database, or commercial programs, without the express written permission of the publisher. Under no circumstances should this publication and its contents be resold, loaned out, or used in any form by way of trade without the publisher's written permission. While we pride ourselves on the quality of the information we provide, Black Dog Media Limited reserves the right not to be held responsible for any mistakes, or inaccuracies, found within the text of this publication. Due to the nature of the software industry, the publisher cannot guarantee that all tutorials will work on every version of Raspbian OS. It remains the purchaser's sole responsibility to determine the suitability of this book and its content for whatever purpose. Images reproduced on the front and back covers are solely for design purposes and are not representative of content. We advise all potential buyers to check listing prior to purchase for confirmation of actual content. All editorial opinion herein is that of the reviewer as an individual and is not representative of the publisher or any of its affiliates. Therefore, the publisher holds no responsibility in regard to editorial opinion and content.

BDM's Manual Series is an independent publication and as such does not necessarily reflect the views or opinions of the producers contained within. This publication is not endorsed or associated in any way with The Linux Foundation, The Raspberry Pi Foundation, ARM Holding, Canonical Ltd, Python, Debian Project, Lenovo, Dell, Hewlett-Packard, Apple, Microsoft and Samsung or any associate or affiliate company. All copyrights, trademarks and registered trademarks for the respective companies are acknowledged. Relevant graphic imagery reproduced with courtesy of Lenovo, Hewlett-Packard, Dell, Samsung, FUZE Technologies Ltd, and Apple. Additional images contained within this publication are reproduced under licence from Shutterstock.com. Prices, international availability, ratings, titles and content are subject to change. All information was correct at time of print. Some content may have been previously published in other volumes or BDM titles. We advise potential buyers to check the suitability of contents prior to purchase.



Black Dog Media Limited (BDM)
Registered in England & Wales No: 5311511

Discover more of our guides today...

