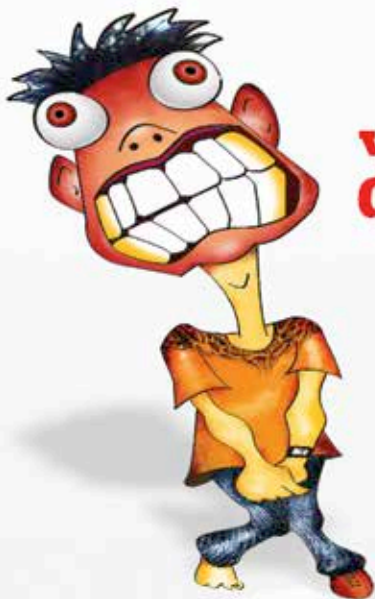


To

# HOTTEST PROGRAMMING LANGUAGES TODAY

- Assembly
- C
- C#
- C++
- D
- Processing
- Erlang
- Haskell
- Java
- Javascript
- PHP
- Python
- R
- Ruby
- Rust
- Swift



**www.  
digit.in/forum**

Join the forum to  
express your views  
and resolve your  
differences in a more  
civilised way.

**digit.in  
FORUM**

Post your queries  
and get instant  
answers to all  
your technology  
related questions



One of the most active online technology forums  
not only in India but world-wide

**JOIN  
NOW**

**digit.in**



# HOTTEST PROGRAMMING LANGUAGES TODAY

---

powered by

**digit**  
YOUR TECHNOLOGY NAVIGATOR

# CHAPTERS

HOTTEST PROGRAMMING LANGUAGES TODAY  
JANUARY 2015

06

PAGE

## Assembly

Assembly is the closest you'll ever come to actually "talking" to a computer.

14

PAGE

## C

No language can claim to have had as much of an impact on computing as C has since its inception.

21

PAGE

## C#

C# was developed with the aim of creating a language that suited just about every development need

27

PAGE

## C++

A language allowing higher level abstraction but still close enough to the hardware is what C++ aimed to be.

33

PAGE

## D

Find out why Facebook uses copious amounts of D to code some of its key features.

37

PAGE

## Processing

Processing is a programming language that's visually delightful to work with

42

PAGE

## Erlang

Erlang is a programming language built for distributed systems from the ground up. Scalable, robust that's Erlang for you.

## CREDITS

The people behind this book

### EDITORIAL

#### Executive Editor

Robert Sovereign-Smith

#### Assistant Editor

Siddharth Parwatay

#### Writers

Ankit Mathur

Daniel Dmello

Kshitij Sobti

Mehakinder Oberoi

Mithun Mohandas

Nicole Anklesaria

Paanini Navilekar

Vaibhav Kaushal

Varad Choudhari

Vishal Patil

#### Contributing Copy Editor

Infancia Cardozo

#### Manager Test Center

Jayesh Shinde

#### Reviewer

Mithun Mohandas

### DESIGN

#### Sr. Creative Director

Jayan K Narayanan

#### Sr. Art Director

Anil VK

#### Associate Art Director

Anil T

#### Sr. Visualisers

Shigil Narayanan

Sristi Maurya

#### Visualiser

Baiju NV

47  
PAGE

## Haskell

We take a look at one of the most popular functional languages out there, Haskell.

52  
PAGE

## JAVA

The language that truly propagated the era of “write once and run anywhere”

58  
PAGE

## JavaScript

If you are interested in Web Development, JavaScript is one of the languages you must learn.

65  
PAGE

## PHP

The power behind the world's most powerful websites

71  
PAGE

## Python

Python is probably one of the best languages to start with if you are new to programming and it continues to be useful later as well

76  
PAGE

## R

Everything you need to know about the R programming language.

81  
PAGE

## Ruby

Ruby is a language that takes the term object-oriented very seriously and was built from the ground up to be object-oriented.

87  
PAGE

## Rust

Rust is very much in development, but if things go well it will have a bright future.

92  
PAGE

## Swift

An alternative language to develop iOS and OS X applications removes the C(omplexity) from Objective-C

### © 9.9 Mediaworx Pvt. Ltd.

Published by 9.9 Mediaworx

No part of this book may be reproduced, stored, or transmitted in any form or by any means without the prior written permission of the publisher. The Arduino™ boards, shields, IDE and parts of the code depicted in this book are by Arduino™ and are covered under the CC BY-SA license.

### January 2015

Free with Digit. If you have paid to buy this Fast Track from any source other than 9.9 Mediaworx Pvt. Ltd., please write to [editor@digit.in](mailto:editor@digit.in) with details

### Custom publishing

If you want us to create a customised Fast Track for you in order to demystify technology for your community, employees or students contact [editor@digit.in](mailto:editor@digit.in)



COVER DESIGN: PETERSON P

# Introduction

A lot of conversation surrounding programming languages is often, unfortunately, mired in declaring one programming language better than the other. There will probably never be just one single programming language to rule them all, and more importantly there shouldn't be. There is need for diverse kinds of programming languages to meet the diverse needs of software. As a professional developer you can never stop after you have mastered one programming language, you need to be as programmable as the computers you deal with.

Fact is, there is no 'best tool', just the best tool for the job. The same can be said of programming languages. Often the best programming language for the job isn't just a factor of the technical merits of a language, but of the nature of the software itself, the goals of the software and even how the software will be developed and deployed.

The perfect example of this is OwnCloud. It's an open source project aimed at giving people control over their own data. The software has apps for cloud storage, calendar, music streaming, photo storage and more. It can also sync local files to the cloud like Dropbox. Best of all you can host it where you want, so you control where your data goes.

The developers of OwnCloud choose PHP for the project despite there being better and easier languages available. They had some very good reasons for their choice of PHP despite its poor reputation. PHP is often supported by even the cheapest of hosting providers, and in the cheapest of packages. This is great given OwnCloud's goals of reaching a wide audience. It's also great at attracting developers, since it is a popular language known by many developers, and easy to learn for others.


Just like the iPhone is a bad app platform if you want to reach Indian farmers, sometimes technical decisions need a non-technical context. For a software aiming for popularity it's important to look at the lowest common denominator; in OwnCloud's case it was PHP. Choosing Ruby or Python or even Node would be antithetical to the goals of the software.

The key is in finding the best language for the job. In some cases it chosen for you – or strongly suggested – as is the case with Java for Android, or

Objective-C / Swift for iOS. In other cases it is about the restrictions of the target environment, for instance embedded systems programming is often done in C and Assembly. For web development, there is little stopping you from using C / C++, but that's usually a bad idea, you'd be better off with Python, Ruby, JavaScript or even PHP and Java. For scientific computing, you'll probably want MATLAB, R, or Python; here MATLAB and R were designed for this purpose, but Python is useful because of the plethora of scientific computing packages available for it. In Facebook's case they started with PHP, and once they were in too far, rather than recode everything, they decided to improve the PHP ecosystem with a PHP compiler, and now a new language called Hack based on PHP.

In this book we have featured both old and entrenched languages such as C, C++ and Java, and also newcomers such as Rust and Swift. We have languages covering most of the popular paradigms listed above. For each language we will explore how it came to be, its strengths and significance, popular and common tools for the language, and we mention a few places you can go to learn more about them. For each language we also include small code samples, one of a regular 'Hello World' program, and another of a simple insertion sort algorithm where ever possible, so you can have a richer taste of the language.

So with that in mind, what do we expect you to do with all the programming languages covered in this text? Explore them all of course! It is important to not tie oneself to a single paradigm, functional or imperative, static or dynamic, interpreted or compiled. They all have strengths and weaknesses that make them shine in their own domain. Fun fact, there is no such thing as an interpreted language or a compiled language; whether a language is interpreted or compiled is dependent on the nature of its implementation. It is quite possible to compile Python or Ruby, and also quite possible to interpret C / C++

Having a preference is fine, but being flexible is more important. No matter your feelings about a given language, it might have strengths that make it most suitable for the software you are developing, even if that strength is merely its popularity. On that note, happy coding! 

# ASSEMBLY

Since assembly is the closest you'll ever come to actually "talking" to a computer, it's only appropriate that we start our journey into programming languages with it.

Programmable devices have long fascinated humans, even before the advent of computers. As long as two centuries ago we had music boxes, tiny mechanisms that produced music encoded as pins on a cylinder. It wasn't long before it was possible to switch out cylinders and have the same mechanism play new melodies. These aren't even the oldest programmable devices.

Bits may have replaced pins, hard drives may have replaced cylinders, and transistors may have replaced cogs and gears, but the principle of programmable machines remains the same. Essentially it was a device created once, but capable of following instructions that were possibly not even conceived when it was created.



A music box with interchangeable cylinders for different melodies.



You may wonder what this drivel has to do with Assembly, so here goes: Programming Assembly is like sticking individual pins on the cylinder that will later turn the gears that produce music.

Before there was assembly, computers needed to be programmed by speaking their language directly with individual 0s and 1s. For instance the sequence 10110111 00001011 might instruct the computer to store the value 11 in a particular part of its memory.

For instance, in the above example, the first 4 bits of the instruction (1011) might stand for “move value”, the second set of 4 bits (0111) might be the specific location in memory where the value is to be moved, and finally the last 8 bits (00001011) are just the binary encoding of the decimal number 11. The first part of the instruction is called the opcode, or operation code. It's like telling the computer which note to play.

What Assembly does is simply give this machine-speak a human face by translating what the computer is doing into instructions that are somewhat more understandable.

An Assembly representation of this instruction might be:

- `mov ax, 11`

Here ‘ax’ is the name of a memory register in your processor. In a higher-level programming language you might write this as:

- `var ax = 11`

At some level though, the above assembly instruction is the kind of thing that the CPU is finally executing.

If you want to, you can take a piece of assembly code, and look up its equivalent binary machine instruction (although usually represented in hex) and you will see exactly what the computer sees when that instruction is run.

Of course, the examples we have given above are mostly made up. Those aren't the exact opcodes and binary representations for these instructions. In fact these instructions aren't universal, and neither are their opcodes. There is no single Assembly language syntax, since Assembly is closely tied to the architecture of the machine on which it is intended to be run. As we mentioned before, it is essentially machine language put into English letters and symbols.

A knowledge of Assembly therefore requires a knowledge of the machine on which it will run, unlike C or Python code which just needs to be compiled on different platforms to get it running there.

This is part of the reason why higher-level languages have almost entirely replaced Assembly code. Assembly language is daunting and there is no getting around that fact.

## Significance

No matter which programming language you use, the end result is machine code, or assembly language. Knowing something about the end result of your code is of obvious benefit.

Think of the difference between knowing how to drive a car, and knowing how an engine, and other components of the car work. It's possible to know how to drive a car without knowing a thing about spark plugs and brake oil, but understanding how a car works puts you in a better position, especially when the car breaks down. In our case, while debugging software you can now peek at the instructions behind your instructions when you learn assembly.

Given how low-level assembly language is, it is also possible to improve performance when coding directly in assembly. However with the complexity of software today it's infeasible to develop in assembly, especially since such software would need to be rewritten for every platform it is intended to run on.

What is commonly done is that, performance-sensitive portions of higher-level code is written in assembly. These are usually sections of the code that are called thousands or even millions of times.

For example, a game might need to constantly evaluate the distance between two points. This piece of code might be called thousands of times in rendering a single frame. It makes sense to write this in assembly and gain a huge boost in performance as a result. This code will need to be rewritten for each platform though. Speaking of platforms and their differences, there are two major schools of CPU design that have a large impact on their assembly language as well. These are Reduced Instruction Set Computing (RISC) and Complex Instruction Set Computing (CISC).

The RISC design strategy is to have fewer possible instructions, but have them execute faster. With CISC the aim is to have fewer instructions that are more powerful. For example, a CISC processor might have an instruction



You can still buy an 8085 microprocessor training kit that lets you program the CPU by sending it single instructions.

to add data in one memory location to data in another memory location. A RISC processor on the other hand would require multiple steps, first instructing the CPU to load data from both memory locations onto registers one by one, and then instruction to add them, and finally an instruction to place this data back in memory.

It might seem like CISC is the better bet here, as it seems faster doesn't it? But the fact is, the CISC CPU still has to do all the things the RISC CPU would do. The fact that it exposes them as a single instruction doesn't change the amount of work behind that instruction. The CISC add operation might end up taking as much time, or even more than the equivalent RISC instructions.

The most famous examples of a CISC CPUs are those by Intel and AMD, x86 in other words. RISC is well represented by ARM, MIPS, AVR, PowerPC and most other CPU designs.

It is more an accident of history that most desktop and laptop computers use Intel / AMD CISC CPUs while most mobile platforms use ARM or MIPS RISC CPUs. In fact a number of TOP500 supercomputers in the world are of RISC design. The PlayStation 3s Cell processor too was a RISC processor.

Over time as the x86 RISC processors have advanced, there has been an increase in the number of instructions, and their capabilities. Commonly performed sequences of instructions have been converted into optimised single instructions. Think of buzz words like MMX, 3DNow!, SSE, SSE2 etc. They are all extensions to the instruction set of x86 CPUs. Many of these are powerful instructions that can efficiently perform repeated operations on larger data sets.

Knowledge of assembly also means you can take advantage new developments before compiler designers have taken them into account, and optimised code output to use them. In fact some special CPU instructions are only available in Assembly!

A programmer well versed in assembly is also one with good knowledge of the underlying hardware, and this pays off even when using higher level languages. The way hardware is designed means that often minor changes in code (such as switching the inner and outer loops in a nested loop) can have a huge impact on the speed. Why? How? When? These are the question that can only be answered by learning more about machine architectures.

Despite the proliferation of higher-level languages that hide all the complexity we just talked about, there is still great value in Assembly language, and knowing it can be very beneficial even if you don't ever need to code in it.



The IBM Sequoia is the 3rd ranking supercomputer in the TOP500 list, and is made up of over 1.5 million RISC PowerPC A2 processor cores

## Hello World

```

• global main
• extern printf
• section .data
•     fmtStr: db 'hello, world',0xA,0
• section .text
•     main:
•     sub     esp, 4           ; Allocate space on the
stack for one 4 byte parameter
•     lea     eax, [fmtStr]
•     mov     [esp], eax      ; Arg1: pointer to format
string
•     call    printf          ; Call printf(3):
•                               ;     int printf(const
char *format, ...);
•     add     esp, 4          ; Pop stack once
•     ret

```

Attribution: [http://en.wikibooks.org/wiki/X86\\_Assembly/NASM\\_Syntax](http://en.wikibooks.org/wiki/X86_Assembly/NASM_Syntax)

Above is the code for a Hello World program written for NASM, and targeting Linux.

Writing something as simple as a Hello World program can be quite a challenge in assembly. The above code cheats slightly by simply calling the C printf function for the bit that actually prints to screen. The rest of the code simply prepares to call printf and pass on the message 'hello, world'.

The 'section.data' segment of code we see above instructs the assembler to store 'hello, world' in memory and put its location in 'fmtStr'. After that the sub (subtract) statement, the lea (load effective address) statement, and the mov (move) statements put data and pointers in the right place so the printf function knows where to find them.

This is by no means the only assembly version of Hello World, not even for NASM.

```

• Sort
• isort:
•   %define a [ebp + 8]
•   %define n [ebp + 12]
•   enter 0, 0
•   pusha
•   mov ecx, 1
•   for:
•       mov ebx, ecx
•       imul ebx, 4
•       add ebx, a
•       mov ebx, [ebx]
•       mov edx, ecx
•       dec edx
•       while:
•           cmp edx, 0
•           jl while_quit
•           mov eax, edx
•           imul eax, 4
•           add eax, a
•           cmp ebx, [eax]
•           jge while_quit
•           mov esi, [eax]
•           mov dword [eax + 4], esi
•           dec edx
•           jmp while
•       while_quit:
•       mov [eax], ebx

```

- `inc ecx`
- `cmp ecx, n`
- `jl for`
- `popa`
- `leave`
- `ret`

The above code is insertion sort implemented in the NASM dialect of assembly. The same code implemented for a different architecture, such as ARM or MIPS would look completely different. Attribution: [http://dgit.in/NASM\\_Wiki\\_Attr](http://dgit.in/NASM_Wiki_Attr)

## Tools and Learning Resources

As we have made it abundantly clear, assembly language is highly influenced by the target platform on which it will run. So the best way to get information about assembly language straight from the source, the CPU designers themselves.

Whatever hardware platform you are trying to target, just search for the CPU architecture followed by ‘instruction set manual’, and you’re likely to find detailed PDF manuals with everything you’d need to know. For example, search for ‘ARM instruction set manual’ or ‘Intel instruction set manual’.

While these manuals are great, they are mostly targeted towards people who already have some knowledge of assembly. Getting into Assembly language is tough, and require a different approach than other languages.

Firstly you will need to learn about machine architectures, and then the actual code. It can be hard to figure all this out on your own. Luckily there are many websites available nowadays that offer college course lecture videos, projects, and assignments online and for free. One such great course is titled “The



At 1500 pages Intel's Software Developer's Manuals are the deepest look you can get at their processor architecture and instruction set.

Hardware/Software Interface” and is available on Coursera.org [<https://www.coursera.org/course/bwswinterface>].


Another great resource of free full courses on Assembly is available at [opensecuritytraining.info](http://opensecuritytraining.info). They have courses on x86, x86-64, and ARM at an introductory, intermediate and advanced level.

Instead of jumping straight into Core i7, you should start by looking into programming the 8-bit Intel 8085 processor. It might be nearly 40 years old, but it has a design that you can conceivably hold in your head all at once. From there you can build up to 16, 32, and finally today’s 64bit processors.

Once you know a bit of Assembly language, you can start playing around with your own code. For this you will need an Assembler, a piece of software that converts assembly code to machine code. A popular assembler is the Netwide Assembler (NASM) that can run on Windows, OSX or Linux.

The popular Gnu Compiler Collection (GCC) includes support for assembly language, and even even compile higher-level code to Assembly, so you can see what your code looks like in assembly. Visual Studio too ships with an assembler called MASM (Microsoft Assembler).

Another way to interact with Assembly code is by disassembling existing code! Decompiling code into its source code is hard to impossible, but getting assembly code for compiled software is easy. On POSIX platforms like OSX / Linux / Unices you can simply type `objdump -d /path/to/compiled/binary` to get the Assembly representation of compiled code.

Finally, if you want to play with running code and see as it executes in Assembly, you have gdb possibly the most powerful debugging tool in existence. To end we’d like to mention that Assembly is a good place to start at a conceptual level, it should be the last place you come to to actually get your hands dirty with code. Come back to this chapter again once you get familiar with higher level languages that follow. 

# C

No language can claim to have had as much of an impact on computing as C has since its inception.

**L**ike most programming languages, C was developed out of dissatisfaction with existing programming languages. Back in 1969 – when the idea of developing an operating system using a higher-level programming language was still novel – Dennis Ritchie was beginning to realise the shortcomings of the B programming language as he was working on recoding the Unix operating system to a new architecture in a higher-level language.

B was designed for a different computer generation – by Ken Thomson, who also contributed to C. The language was word-addressed instead of byte-addressed, and lacked floating point support. Since the early colonists of the digital era were not spoilt for choice as we are these days, they decided to create a programming language which better suited their needs.

Thus C was in use as early as 1972-1973, however this was the C before any serious efforts in standardisation were made, and would look odd to today's C developers. It was only with the publication of the 'The C Programming language' by Brian Kernighan and Dennis Ritchie in 1989 that C finally got a widely available manual for its syntax.

The first 'official' C standard that was ratified by the ANSI was published in 1989, and is still available in popular C compilers such as the GCC C compiler. This edition of the language is so popular that it is still the default





Dennis Ritchie (standing) and Ken Thomson working on the first computer for which they designed C (The PDP-11).

syntax of the current (4.9) version of GCC. It is commonly referred to as C89 or C90. Since then C has had two major revisions, once in 1999 (known as C99), and again in 2011 (known as C11). The upcoming 5.0 release of GCC will finally adopt the latest C11 edition of C as the default.

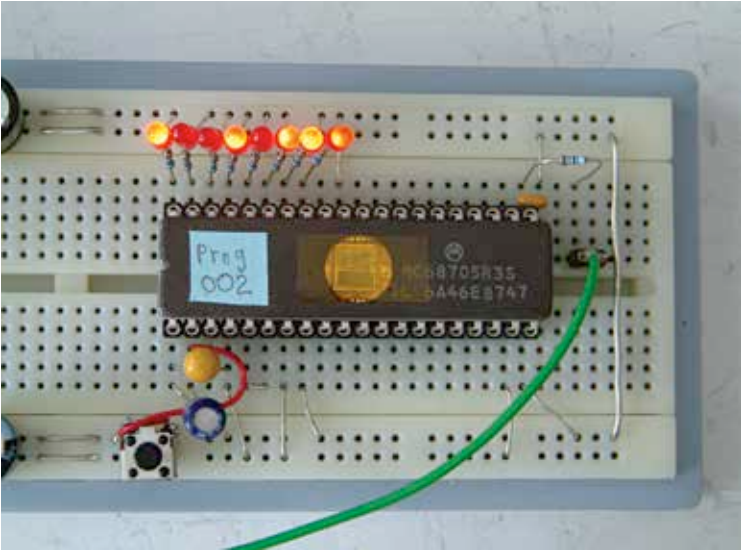
Given how old it is, it is a testament to the design of the language, and the skill of its designers, that it is relevant to this day.

## Significance

It is hard to overstate the significance of C in the landscape of computing today. Nearly every popular programming language used today, whether it be JavaScript, Java, C#, PHP, Objective-C, or even Python is inspired by conventions set by C.

Compared to languages it inspired though, C is significantly low-level. C improves the readability, maintainability and ease-of-development of code significantly compared to Assembly, and is significantly more portable despite giving developers nearly as much power as Assembly does.

C gives developers low-level access to machine hardware, with minimal abstraction and pointers in C let the developer directly work with raw



Whether its a hobby project or mission critical code running on an embedded device, chances are it was written in C.

memory locations. As you can imagine, this makes it the perfect candidate for system programming.

Few other languages are as suitable, or as highly used in writing low-level code such as operating system kernels or device drivers. One only needs to look at the three major operating system families to see proof of that. The Linux kernel, the OSX kernel (Mach / Darwin) and the Windows (NT) kernel are all mostly written in C, with some Assembly thrown in. In fact when one encounters a kernel not written in C, it is often a point to highlight.

Another field where C has a significant market share is in programming embedded systems. Low-level systems such as the ones that control various systems in cars, from ABS to managing fuel / air mixtures etc. Since these systems are mission critical, and you wouldn't want a blue screen of literal death, the version of C used in such systems is often a restricted one. After all, you can't really patch you car to car 1.0.1 if they detect a bug later on!

It is also an incredible point in its favour that C is the language in which the interpreters of popular languages such as Python, PHP, Ruby and Lua are written.

The deeper you go into computer programming, the more likely it is that you will find copious amounts of C. Nearly every electronic device you touch, that has some kind of embedded microcontroller or microprocessor in all likelihood runs at least some code that was once stored in a .c file.

## Hello World

```
• #include <stdio.h>
•
• int main(void) {
•     printf("Hello, world!\n");
•     return 0;
• }
```

Hello World is possibly one of the simplest programs you can write in any language, and C is possibly one of the more verbose ones. If you just want something that compiles – with warnings – and throws “Hello, world!” to the screen though, you can even get away with just the following (with some compilers):

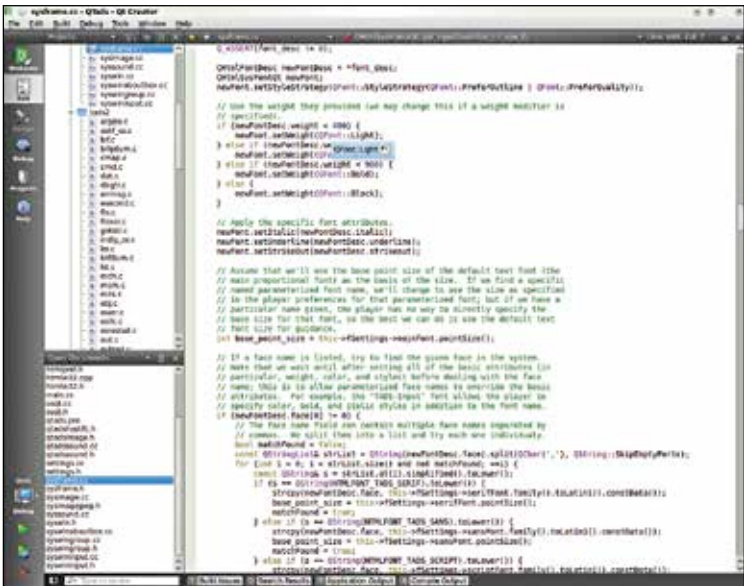
```
• main() {
•     printf("Hello, world!\n");
• }
```

## Sort

```
• void insertion_sort(int *arr, const size_t arr_len)
{
•     for (size_t i = 1; i < arr_len; i++) {
•         int elem = arr[i];
•         size_t j = i;
•         for (; j > 0 && elem < arr[j - 1]; j--) {
•             arr[j] = arr[j - 1];
•         }
•         arr[j] = elem;
•     }
• }
```

Insertion sort in C isn’t exactly complex. Here in the above example we are the newer syntax which allows us to define variable as they are used rather than in the beginning of the function. This has been supported since C99.





QtCreator has great debugger integration as well as support for targeting Android.

tools and basic libraries for C are already installed, if not it can definitely be found in your distro's repositories and is just a command or two away. Also, if you're using Linux you should probably know how to take it from here!

Another good IDE for C and C++ is QtCretor by Digia. It is part of the Qt project, but doesn't require you to use Qt itself. Best of all, it is free, open source, and cross-platform.

If you are using Linux / OSX, C documentation is basically built into your operating system! The documentation for any function is just a single command away. Linux (and Unix and OSX) include the 'man' or manual command that can be invoked in the command line to open documentation on any topics. Many people are familiar with the command for looking at the usage instructions for command line programs, but it works equally well for C functions! Try it, if you are running a \*nix OS, type 'man printf' and find a instant usage guide to that function in C. Most C libraries you install will also install a documentation package, so this should work for any development library you have installed.

If you are using Windows, or otherwise would prefer to view this information in a GUI, you can just try searching for 'man <function>' in your

favourite search engine, and you will likely find one of the many online man-page repositories. This will be less useful for Windows users though.

If you're already reaching for your browser though, there is no end to good learning resources for C. Simply a search will reveal great starting resources, so rather than reiterate those here we will share a three resources that teach something interesting.

### **Create your own Kernel Module:**

If you ever wanted to get into really low-level programming, building your own driver for the Linux kernel, here is a guide that will help you. Instead of jumping into the deep end, this tutorial showcases a very simple kernel module that simply creates a `/dev/reverse` device in Linux that will reverse the word order of strings sent to it. Where you take it from there is up to you.

<http://www.linuxvoice.com/be-a-kernel-backer/>

### **Learn about Memory Allocators, possibly write your own:**

Believe it or not, someone actually has to code the bits of software that actually allocate, deallocate and keep track of the memory used by variables you use in your programs. Since creating and destroying variables happens a LOT in any software, speeding up this process can have a large impact on software performance, especially if the way you use variables is special, for instance if you are developing a game. These articles give an overview of this bit of software.

<http://jamesgolic.com/2013/5/15/memory-allocators-101.html>

<http://jamesgolic.com/2013/5/19/how-tcmalloc-works.html>

### **Add a new feature to Python by modifying its code:**

If you've ever wondered how the features of a programming language map to actual code in its compiler or interpreter, this article is what you are looking for. It's written by a core Python contributor, and takes you through adding a new statement to the Python syntax by modifying its C code.

<http://eli.thegreenplace.net/2010/06/30/python-internals-adding-a-new-statement-to-python/> 

# C#

C# was developed with the aim of creating a language that suited just about *every* development need

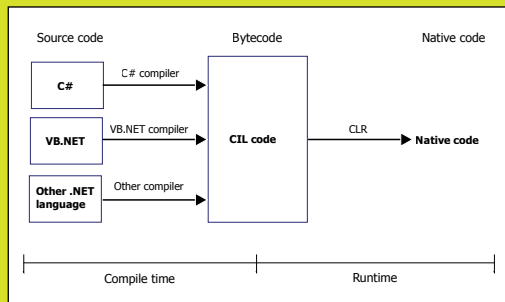
## Introduction

When languages such as Java and C++ already existed, was C# really needed? Some say it was only developed by Microsoft to take on Sun's Java?

C# was developed by Andres Hejlsberg and his team at Microsoft. In one of his papers, Hejlsberg commented that most programming languages had flaws. This led him to develop CLR (Common Language Runtime) – which is an environment that promotes the conversion of compiled byte code directly to machine instructions for a CPU to execute (that's execute as in run, not execute as in hang unto death) – and later led to the to the creation of C#.

## C# and Java

C# was released in the year 2000, and was quite similar to Java in functionality. Though Andres Hejlsberg always said C# shared more similarities with C++ than



CLR in use

Java, James Gosling (the father of Java) and Bill Joy (Sun founder) labeled C# an “imitation of Java”. Some went as far as saying that both were just “boring repetitions of each other.”

## Why C#

Originally, the language was named ‘COOL’, which was, well, not so cool. COOL stood for ‘C-like Object Oriented Language’. However, it was later renamed C# (pronounced ‘C sharp’), which is a musical note that is a semi-tone higher than the ‘C’ note. Thus, C# is a higher version of C.

## What’s So Sharp About C#

C# is one of the most popular languages today and there are many reasons for this. It has many qualities that make it suitable for development for users at almost all levels. Fifteen years after its advent and following the release of many new versions, C# today has a whole slew of features which make it unique and its use inevitable. Let’s discuss some of them here.

## COM (Component Object Model)

COM, or Component Object Model, provides C# with capabilities such as platform independence and integration of objects, which need not be in the same development environment or even on the same computer. COM along with the DotNet framework promote interoperability. Versioning helps in easy deployment, while support for web-based integration is provided by XML.

## Dynamic binding

C# is a strongly typed language, which illustrates the fact that all the variables are resolved at compile time. But then the question arises: How does C# simplify access to Document Object Model (DOM) or COM API’s? Dynamic Binding does this for C#. Dynamic Binding delays the process of resolving variables from compile time to runtime.

## Scalability

Scalability allows C# to be suitable for handling enterprise applications and projects on a large scale. MVC’s based on DotNet are used to design web constructs that are scalable and updatable as well as to establish the assembly version, digital signature, identity, culture, etc. Updation is useful when you need to delete or update old files to scale your application.



Other features include robustness, which can be attributed to Garbage collection and type safety attributes provided by C#. Security in C# is assured through intrinsic code trust mechanisms. Also, the replacement of pointers (as in C++) by automatic memory management adds to this feature.

## Syntax

### Hello World

This is the way Hello World is programmed in C#.

```
• using System;
• class Hello{
•     static void main() {
•         Console.WriteLine("Hello World");    }
• }
```

C# is a good language to start with for the novice programmer, but you'll need to get some experience with Object-oriented programming before exploring C#. This is why people usually get to C# only after learning C++, or other OOP languages.

Now, back to the program listed above, class name doesn't need to be similar to the file name (a feature observed in Java). The output is written on the console and 'main' is accompanied by a 'static' keyword.

## Windows and C#

While developing software, the most important factor for choosing a language is the environment in which it will be working.

Probably one of the major reasons for the success of C# is the fact that it's the best suited language for Windows servers, Windows clients and now even Windows phone applications. The reason is obvious: its development is being overseen by Microsoft.

## Learning C#

Some benefits of learning C#:

- ◆ Promotes easy learning of IIS for server programming
- ◆ Promotes easy deployment of applications made in C# in a Windows centered environment.
- ◆ Is best suited for working in DotNet framework and with its API's.
- ◆ Provides the best graphical user interface along with Windows Presentation Foundation and Winforms.

Thus, Windows being the most widely used operating system further encourages programming in C#.

## Where's C# used?

C# is used across many applications. Let's look at the scope of the language in the industry today.

- ◆ **Game Engines:** One of the most popular uses of C# has been its contribution in the development of the game engine Unity. It's currently one of the market leaders, and is used to develop games for a wide range of devices – desktops and mobile devices to consoles and websites.
- ◆ **Cross Application/Platform Development:** The interoperability discussed earlier facilitates the growth of applications across various platforms and even across different applications, which can later be integrated as one application for use by an enterprise.
- ◆ **Client Applications:** One of the primary uses of C# is development of client applications such as chat or email.
- ◆ **Backend Service:** C# is used for programming at the backend in a variety of applications. The combination of XAML and C# is quite famous in the industry for designing mobile apps.
- ◆ **Components and Controls:** The main functionality of the components and controls is their reusability as code. They can also be included as libraries to build something that's sharable. Examples of this are built-in tools in a game engine or GPS libraries used for building navigation systems. The developers can further use these components as API's, saving time and making it easy to code.
- ◆ **Multiplatform application (mobiles):** The word “multiplatform” has been intentionally used here. C# is no longer limited to Windows Phone applications and has infact extended its reach to the IOS and Android markets too with the help of Xamarin Studio.
- ◆ **Web Applications:** JavaScript along with C# (in the DotNet framework) is one of the most widely used combinations to create web applications. Go to <http://bit.ly/1JzYqv> for more about the web development side of C#.



Gloria Victis: A game developed on Unity

- ♦ **Cloud Computing:** One often hears that Cloud Computing is the future of Computer Science. Infact, it's one of the fields that hasn't reached its full potential yet. Cloud-based applications can be easily designed with C# (Visual Studio toolkit).

## Tools and Learning Resources

As discussed before, the support by Windows gives C# one of the best programming environments available for it – there are numerous IDE's available.

The most suitable development environment is undoubtedly Visual Studio because of the toolkits it provides and the ease of design and deployment one experiences with it. But, apart from Visual Studio there are several other IDE's such as SharpDevelop, MonoDevelop, xacc, cSharpStudio, QuickSharp and Fireball.CodeEditor available for C#. We'll briefly discuss some of them.



C# is used in mobile application development

'Sharpdevelop' is an open source IDE for C#. 'MonoDevelop' aims at developing support for C# in the Linux environment and aims at introducing all the features currently present in Windows in Linux too. 'xacc' is mainly used because of its debugging support as well as its support of various languages (27 of them).

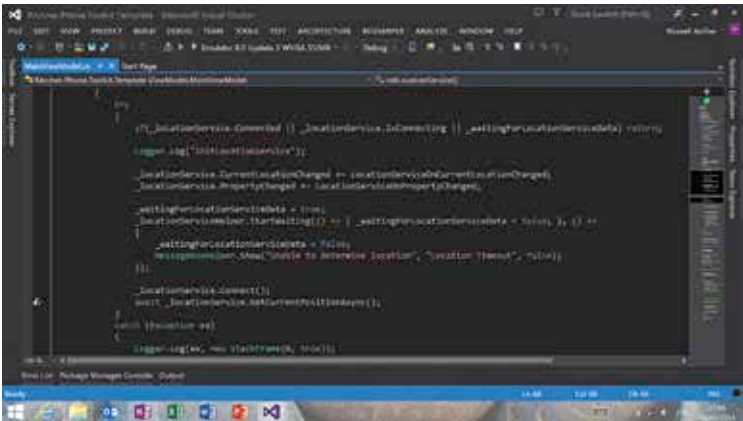
There is a tremendous amount of online resources for anyone from beginners to advanced coders to learn C#. There are MOOC's such as 'Game Programming with C#' (Coursera) available, and also websites such as red-hoop.com or noexcuselist.com that can help you stay updated with various MOOC's available and pick from the best of them.

Websites such as 'pragimtech' and many others keep posting well-crafted videos on their YouTube channels, so make sure to check those out.


Book worms should start with this list::

- ◆ Component-based Development with Visual C# (M&T books) – Ted Faison
- ◆ C# Essentials, 2nd edition (O'Reilly) – Ben Albahari, Peter Drayton & Brad Merrill
- ◆ A Programmer's Introduction to C#, 2nd edition – Eric Gunnerson
- ◆ Inside C#, 2nd edition (Microsoft Press) – Tom Archer
- ◆ C# in Depth – Jon Skeet

Given the wide expanse of the language, we suggest you build certain projects, or at least go through these links for some knowledge on how to get started with them.



That's the way development on Visual Studio happens.

- ◆ <http://www.w3schools.com/aspnet/> introduces the web development side of DotNet framework to you. Going through it would provide you with an idea regarding website building with the help of C#. Apart from this, we recommend actually implementing, maybe, a small website just to get your feet wet with C#.
- ◆ <http://www.codeproject.com/kb/cs/> has numerous projects available covering almost all the fields of C#. Going through them would make you aware of the expanse of C#. You can pick any of them and start working on it to try your hand at C#. 

## C++

A language allowing higher level abstraction but still close enough to the hardware is what C++ aimed to be.

### Introduction

What was C++ introduced for? Drawbacks in languages such as C, Simula, ALGOL, BPCL made Bjarne Stroustrup, a Danish computer scientist begin his work on C++ in around 1979, the era when these languages already existed and had varied uses. The idea was to use C as a base and inculcate features of Simula (An OOP language), that would promote features such as code reusability, general level abstraction etc.

Why the name C++? According to Stroustrup, the ‘++’ (post increment operator) in C++ symbolizes the evolutionary changes made from C.



Bjarne Stroustrup, the brilliant creator of C++.

### Why C in C++?

C was used as a base as it was reasonably popular at that time and also was fast, portable and easy to understand. Overall, in C++, using a general purpose language we could now do general level abstraction (Simula) and at the same time stay close to the hardware, which made it easy to perform highly demanding computer tasks with great speed and ease.

## Is it still needed today?

All right, enough of 1980's. The important question is, with the advent of languages such as Java, is C++ still needed? The fact that C++ is still very popular should answer that question. Why? Read on we'll cover that later in this chapter.

## The killer feature (RAII)

C++ is the only widely used language in the industry which supports the feature of RAII. For those, who aren't familiar with what RAII is, it basically stands for Scope Bound Resource Management. The control C++ provides over the lifetime of the objects it generates is something that garbage collection or any other management method lacks. In fact, in other languages, a common problem encountered is that objects exist beyond their lifetime. In C++ the objects can be acquired as well as be released in a deterministic manner.

## What about performance?

C++ performs better than all recently made languages such as Python or Java, which despite having large libraries are no match for C++. The reasons for this unbeatable performance are brevity, modular programming and, of course, closeness to the hardware.

## Brevity? Modular programming?

**Brevity:** Brevity refers to compactness of code. In other words, how short is the code, measured as LOC (Lines of Code). The more lines of code the more a project costs and performance takes a hit as well. This is why many companies such as Adobe, Amazon, Apple, Facebook, Quora etc., all have some parts of their basic codes written in C++.

**Modular Programming:** One of the most important aspects of Software Engineering is the concept of Modularity, which basically means the ability to build different, independent modules and later link them together in a program during compilation or run time. C++ not only promotes modularity within itself, but also with other languages, such as Assembler or C.

## Syntax:

### Hello world

- `#include<iostream>`
- `Using namespace std;`

- `Int main()`
- `{cout<<"HELLO WORLD";}`

That sums up the Hello World program. As you can see the simplicity in the write-ability of the program is similar to C. In fact, in this program the only change is the use of **cout** operator instead of **printf()** function.

## STL: The Cool Library

In the chapter about C, we discussed the insertion sort program in C. You can do the same in C++, using the same logic, but insertion sort is rarely used and some of the sort functions are difficult to write every time. Apart from **sort()**, the programming of ADT (Abstract Data Types), functions for linked lists, queues, stack etc. can make programming a really cumbersome job. This led to the introduction of STL (Standard Template Library). STL not only helps you program, but also introduces brevity along with a reduction in compilation errors.

If you want some additional details on what STL is, refer to its wiki page – <http://bit.ly/1GFfMVe>

## Where Is C++ Used?

Remember the question we asked before, about whether it's still relevant? We'll answer that now:

First of all let us consider the popular Operating Systems today. Windows, Apple OS, Symbian OS are all developed in C++. In fact popular Windows editions such as Windows 95, 98 and XP have been written prominently in C++. C#, a language developed by Microsoft for its DotNet framework is significantly based on C++. In fact, knowing C++ makes it very easy to learn C#.

Apart from the use in OSes, popular Internet browsers such



Both Linux and Mac OS-X use C++

as Firefox and email clients such as Thunderbird are made using C++.

Go to <http://bit.ly/13NYgjP> to find a list of game engines – 80% to 90% have been built on C++. Advanced multiplayer gaming, 3D gaming and management of resources are some of the features which make C++ imperative. The infamous ROOT framework created for the high energy physics problems by CERN uses C++.

Plus, here are some software you probably have used: Adobe Systems use C++ for various applications such as Adobe Premier, Photoshop, Illustrator, InDesign, etc. Google Chromium and File systems have been reportedly been using C++ too. Even in the field of Database Design, MySQL which is probably the world's most popular database software, has parts written in C++. CAD systems in Civil Engineering by companies such as 12D have been programmed in C++

Head to <http://bit.ly/13oztSD> for a longer list of stuff that's made using C++, and it's a trustworthy source as it's a page written by the creator of C++ himself, Bjarne Stroustrup.



Companies like facebook and Adobe use C++ in their products.

## Tools and learning resources:

The best part about C++ is the programming environments available for it. There are numerous IDE's available, depending on the OS you use:

**Windows:** Codeblocks, Microsoft Visual Studio, EMacs, QtCreator, DevC++ and many others.

**Linux:** DevC++ or NetBeans.

**Mac OS:** Xcode and Eclipse along with CDT will do.

Emacs, VIM and QtCreator can also be used on any of these OSes.



Code less.  
Create more.  
Deploy everywhere.

QT Creator

From the IDE's listed above, Microsoft Visual Studio and QtCreator are suggested for the development of projects because of the tools and resources they provide. Also, QtCreator is available for free, so



there's that. Another cool resource is [www.ideone.com](http://www.ideone.com) which is something you can use to compile code online.

Just for the knowledge of the reader, the compilers used in C++ are GCC(\*nix) & MinGw, the windows port of GCC.

Thanks to the popularity of C++ and the immense amount of online resources, videos, project repositories available, learning C++ these days is no big deal for anyone who knows how to use Google. We recommend sites such as redhoop.com or Noexcuselist for that matter to go through the MOOC's available.

Several sites and even YouTube channels offer C++ tutorials. Then there is the vast library of good books to choose from:

- ◆ The C++ Programming Language by Bjarne Stroustrup
- ◆ Effective and More Effective C++ by Scott Meyers
- ◆ Programming: Principles and Practice Using C++ by Bjarne Stroustrup
- ◆ C++ How to Program by H. M. Deitel, P. J. Deitel

A complete list of books can be found at: <http://bit.ly/1bOSliB>

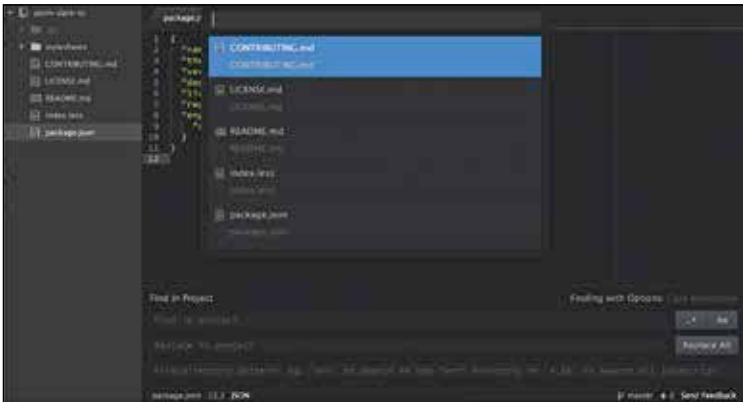
Apart from all these try some of the articles at the links below which detail the special features of more recent updates to C++:

<http://bit.ly/1olzLGY>

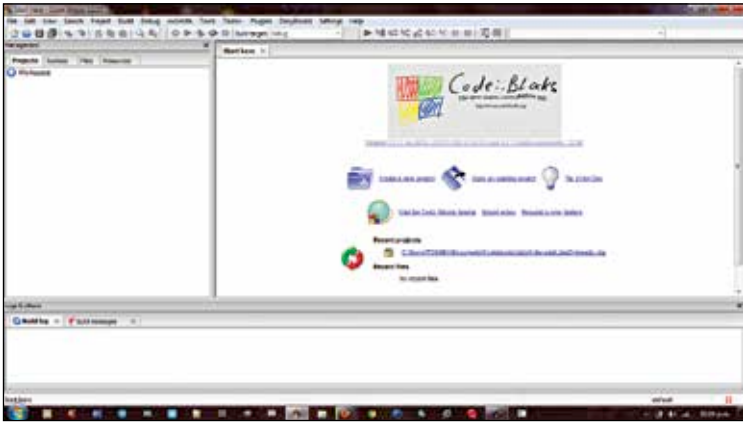
<http://bit.ly/1GFL1g>

Once you get started, we suggest working on projects. Here are a few:

<http://bit.ly/1j8KEmX> has about 150 different projects that can be built in C++ ranging from Networking to numbers to even stylizing HTML documents.



This is what a good text editor looks like



Codeblocks[Can be grouped under the name C++ IDE's]

Specific projects we recommend?

Build your own Text editor:

You don't have to build something as good as the picture above, but attempting to will show you the ins and outs of the language – by coding for functionality such as open, view, edit, save file and so on. Take a look at <http://bit.ly/1v6b7U1> which we feel is a good article for this project.

To give you an idea of the usage of C++ libraries in the Industries here's a link to a github repository: <https://github.com/facebook/folly/> This will take you to development of library named Folly, which is now being used and developed by Facebook. Folly aims at increasing performance of the system overall. Another example of C++ enhancing performance of the system.

Interested ones may try reading <https://github.com/L2Program/FlintPlusPlus> too. Another tool developed in C++ and used by Facebook. 📄

# D

Find out why Facebook uses D to code some of its features.

**D** is a new programming language that is object-oriented, imperative, multi-paradigm system programming language. It started off as a re-engineered C++ though over the years it has been redesigned quite a bit to incorporate features of modern day programming languages like Java, Python, Ruby and C#. So you have the same performance as C++, is less verbose and is memory safe but it has the expressive power of modern languages as mentioned above.



Work began on D in 1999 when Walter Bright started off with improving C++ which lacked a few paradigms like OOP(object oriented programming) and metaprogramming that were staple of modern programming languages. A functional language was created in 2001. However, in 2005 Walter Bright happened to meet Andrei Alexandrescu who was also working on his own programming language(Enki). Both of their languages had significant upgrades and it seemed as if they were solving each other's problems.

So they started working together and by 2007, D1 or D version 1.0 was born. By this time the language had already been seeded into the community and they had a few modifications of their own to add to the mix. Phobos which was D's official runtime and standard library wasn't well received and the community came up with Tango to replace it.

Tango focused more on OOP and modularity. The existence of two libraries which were incompatible with each other early in the development cycle of a programming language would have spelled its doom but that was not the case. D2 was released six months after D1 and streamlined what could go into the main language and what couldn't, thus bringing the community together. Tango was ported onto D2 sometime in 2012. Soon enough D1 was sent to the grave and D2 was all that remained and is now simply called as D. If one were to put forth how exactly D pans out, you could say it's C++ done right.

Programming in D1 felt as if you were programming using a low-level-oriented Python. The analogy might seem weird but you have speed and there is focus on the task at hand rather than fixating on the aspects of the language. D2 brought more features but it also brought more complexity that C++ has. Concurrency is a built-in feature for D2 and that is what makes D2 more popular among programmers. It is still developing and there are plenty of people dropping their preferred language for D.



The D creators - Walter Bright (top) and Andrei Alexandrescu (above)

## Why do the D?

Let's take a look at the programming paradigms for D. These are the related to the style or manner by which one goes about creating functions and programs.

### 1. Imperative

This paradigm brings it in the same league as C. Functions, statements, data, and expressions work the same as they would in C. The differences

include the 'foreach' loop that allows for creating loops over a collection and nested functions that have functions built within another function in such a manner that the internal function has access to the variables of the outer function.

## **2. Object-oriented**

Object-oriented programming in D uses a single inheritance hierarchy. It does not have support multiple inheritance, however, it does make use of the interface which has been influenced from Java.

## **3. Metaprogramming**

Metaprogramming is obtained by the combination of tuples, templates, string mixins and compile time function execution.

## **4. Functional**

Function literals(the ability to use an alternate syntax for defining a function), closures, recursively-immutable objects(state of which cannot be altered between iterations) and higher-order functions are supported in D.

## **5. Concurrent**

Concurrency is when there can be multiple instances of the same computation process occurring simultaneously.

## **6. Memory safe programming**

Functions can be marked with `@safe` so that they are checked by the compiler during compile time to make sure that they aren't using any features which might lead to corruption of memory.

## **The Significance**

Every person that's developing his/her programming language wishes to do something that has never been done before. D is by no means any different... or is it? Modern languages allow you to either write code fast or write fast code. Bringing the two approaches together has been a daunting task. And this is exactly what D has decided to tackle. So it's been written for simplicity of programming while maintaining a high execution speed.

Another thing that Andrei (one of the creators) boasts of is D's modeling power. To elaborate, this means that programmers can easily replicate real world problems in D. So everything from a high volume transaction problem that banks face to automotive sensors and spark plugs can be replicated in D. So the simplicity allows the programmer to focus on the problem at hand rather than fiddle with all the constraints around approaching a problem.

## Key adopters

### Facebook

Yup, the big blue actually employs Andrei Alexandrescu. Andrei, along with a team of programmers use Hack (another programming language) along with D to built some of the programs behind the world's biggest social network.



For example, the program that goes through Facebook's code in order to find errors was created using D and the same goes for the preprocessor that generates the core code.

### Sociomantic

Sociomantic is an advertising agency that was recently acquired by Tesco for \$200 million dollars and their entire operation has been created using D.

### ABA Games

A lot of the games released by them make use of D1 which has been declared obsolete as of 2012.

### Remedy Games

They are the guys behind Max Payne, Max Payne 2, Alan Wake, Alan Wake's American Nightmare etc. and they have a massive codebase that is written in D.

## Lets get coding

So here's how a simple program in D looks like. You may type this down in any text editor and rename the file so that it has the extension '.d'

```
• import std.stdio;
• void main() {
•     writeln("Hello World!");}
```

Save the program (let's call it hello.d), then navigate to the folder in a command prompt window and type "dmd hello.d". Your console should have a new line that says "Hello World!".

## Intrigued? Here's where you can get more:

<http://dlang.org/download.html>

<http://dlang.org/spec.html>

[http://www.tutorialspoint.com/d\\_programming/index.htm](http://www.tutorialspoint.com/d_programming/index.htm)

<http://www.dprogramming.com/tutorial.php> 

# PROCESSING

Processing is a programming language that's visually delightful to work with

**A**lmost every programming language deals with text output in the form of words or alphabets, numbers and special characters. It's one of the easiest things to do when you start out learning a new language. However, the moment you start doodling with graphics it feels as if you've hit a brick wall. The learning curve becomes steeper and quite a few beginners might even hang up their boots. Even troubleshooting what you've created can be a daunting task as you start dealing with more complex constructs.

Processing makes it a lot simpler to work with graphics. You can draw a circle with just a single line of code and then colour it with another line. If you wish to resize it, that's another line. When it comes to Processing, each program is called a 'sketch'. It makes sense since you've actually sketched a visual element i.e the circle.

Adding another line to the above 'sketch' allows you to position the circle on your canvas. You can probably guess where this is going – animation. Thus, it not only helps in creating shapes but also allows you to animate those shapes with relative ease. Adding a little more to the mix, you can



even use Newton's laws of physics to simulate the flight parameters of an actual spaceship. That might be an overstatement but you know what we're talking about.



Bouncing bubbles simulated using Processing

Processing was developed by Casey Reas and Benjamin Fry from the Aesthetics and Computation Group at the MIT Media Lab in 2001. Their main goal was to create a tool to get beginners started with programming through the instant gratification of getting a visual feedback. The

language is built on top of Java but makes use of a simplified syntax. The same goes for the graphics programming model which depends on many libraries that help achieve the graphical interface.

## Significance

It is a programming language that uses a simplified Java syntax taking away the hassle of understanding the actual Java syntax which for beginners, is quite complex. And it still maintains the functionality necessary to make visualizations and perform other necessary operations. This makes it a great tool to start programming, to create graphics and animations with visual analysis of data which in turn helps create physics simulations.

The core Processing framework simplifies the use of basic multimedia libraries (OpenGL, PDF, camera capture), thus, removing the overhead involved in the setting up of basic applications. It uses an extensible code structure to allow the creation of dozens of useful libraries for everything from 3D import / export tools to ones that create complex 3D geometrical figures.

For scientific data analysis, it can act as tool that combines data from multiple sources and helps visualise the data. Basically, it makes for easy comparison of data from various sensors and other hardware.

## Processing the Sketch (Coding)

To work with graphics in Processing, we use the Coordinate system which is used for positioning every shape. The computer's coordinate system is different from that of the Cartesian coordinate system taught in schools.

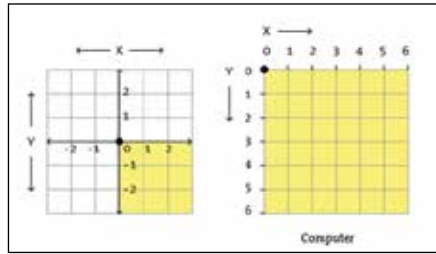
The difference isn't that big and the image on the left will help you understand the difference between the Cartesian and computer coordinate system.



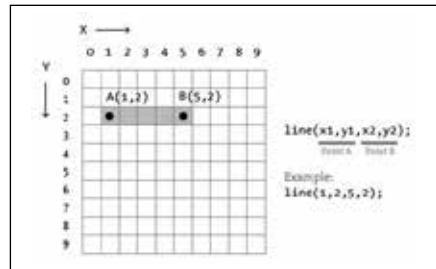
Let's start off by drawing a line using Processing. Everything after `'/'` is a comment to elaborate upon what that particular line of code accomplishes.

```
• void setup() {
•   size(200,200);}
• // Create a window
of dimensions 200x200
in which the line will
be displayed
• void draw() {
•   background(255);
// Set background
colour to white
•   rectMode(CENTER);
•   line(50,50,150,150);
}
```

```
• // Draw a line
starting at coordinates (x,y) = (50,50) and ending at coordinates (x,y) = (150,150)
```



Cartesian and computer coordinate system



A line in the computer coordinate system

That's it. you've created your very first program in Processing and it's now time to crank it up a little.

- Now let's try making a simple cartoon character using Processing.

```
• void setup() {
•   size(200,200);
• } // Create a window of dimensions 200x200 in which the
character will be displayed
```

```
• void draw() {
•   background(255); // Set the background colour to white
•   rectMode(CENTER); // Set the rectMode to centre which
is used to colour the objects. However, we won't be using
any colours in this example.
•   rect(100,100,20,100); // Draw the body of character
```

using a rectangle. Coordinates  $(x,y) = (100,100)$  is the centre of the top edge of the rectangle and it has a width of 20 and a height of 100

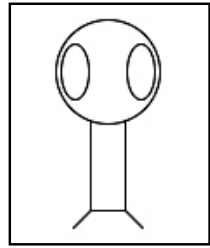
- `ellipse(100,70,60,60);` // Draw the face with an ellipse centered at coordinates  $(x,y) = (100,70)$  whose minor and major axis are of length 60. This creates a circle.

- `ellipse(81,70,16,32);` // Draw the left eye with an ellipse at coordinates  $(x,y) = (81,70)$  and minor axis of length 16 and major axis of length 32

- `ellipse(119,70,16,32);` // Draw the right eye in a similar fashion with the same ellipse but at coordinates  $(x,y) = (119,70)$

- `line(90,150,80,160);` // Draw the left leg, using line starting at coordinates  $(x,y) = (90,150)$  and ending at coordinates  $(x,y) = (80,160)$

`line(110,150,120,160);` } // Draw the right leg, using line starting at coordinates  $(x,y) = (110,150)$  and ending at coordinates  $(x,y) = (120,160)$



Cartoon character created using Processing

## Tools and Learning Systems

On the official website there are video tutorials classified into three categories viz beginner, intermediate and advanced. These video tutorials can be accessed at <https://processing.org/tutorials/>.

The Processing IDE is available for download at <https://processing.org/download/>. You can download the IDE for Windows and Linux environments in both 32-bit and 64-bit versions, and the Mac OS X platform hasn't been left behind either.

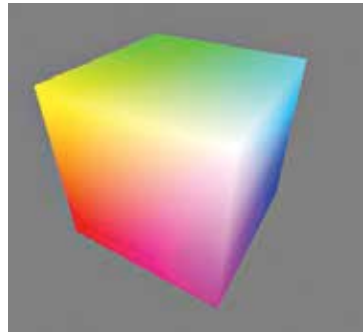
There are many libraries available for various functions like exporting PDF, transmitting and receiving serial data over an RS-232 port etc. Also, libraries have been developed for network related functionality i.e. to send and receive data



Processing integrated development environment


over the Internet between clients and servers. Multimedia libraries like the video library assists in reading images from a camera, playing movie files and creating movies. The sound library can play audio files, get audio input from input audio port, add effects and take it one step further and synthesize sound. The DXF Export library is used to create DXF files to save geometry data that can be used in other programs. You can check the documentation of these libraries at <https://processing.org/reference/libraries/>.

Like every programming language out there even Processing has user contributed libraries to make your life easier. Libraries for 3D, animation, data analysis, I/O, math, video, animation etc. helps speed up the development process. Let's take a look at some of them. One adds support for the Microsoft Kinect which comes with the XBOX 360 and XBOX One consoles. And there's even one for the ultrasensitive 3D motion detector – Leap Motion. Dare we mention Arduino? In fact the Arduino development environment is inspired by the Processing development environment. Some of the hardware on Apple's Macbook can be accessed using a community library.



LED cube responding to mouse input

Then there are tools like the applet maker and colour selector to help you with simple things that you might have had to spend some extra time creating from scratch. These tools and others can be obtained at <https://processing.org/reference/tools/>.

The creators of Processing, Casey Reas and Ben Fry have together written a book titled 'Processing: A Programming Handbook for Visual Designers' that can act as a companion guide for quick reference. You can request a preview copy of the book at <http://mitpress.mit.edu/books/processing>. Topics such as photography and drawing, and its relation to software have been covered with great detail. Even advanced topics like animation, performance metrics of applications have also been taken into consideration. 

# ERLANG

What if there was a programming language that was built for distributed systems from the ground up? Erlang is one such language which targets massively scalable systems along with robust fault tolerance in much less code than you can imagine.

**E**rlang is one of those early languages which were based on the functional programming paradigm which was developed back in somewhere 1986 internally within Ericsson for proprietary use. The immediate target for it, being a platform for distributed and fault tolerant telecommunications applications. This was one of the few places when Ericsson was seen in the software technology scene, and this is one of the reasons why this language was open sourced in 1998 after it was banned for further product development internally. The most important factor being that Ericsson intended to focus on developing hardware products, rather than being a producer of the software stack that powers them. But this turned out to be for the better in one way, specially considering the fact that the people who were responsible for developing and writing such a language i.e. Joe Armstrong, Robert Virding and Mike Williams, promptly

quit the organization due to this reason and worked towards its wider adoption once it was open sourced. However, they were rehired later and the ban was lifted as well. As of today, it has a thriving community which actively develops it for numerous production applications.

The name for this language is ostensibly inspired from the unit with the same name, which was used to measure the load on telecommunication switches and other devices. Originally, its usage was obviously restricted mostly to telecoms systems, but today it works under the hood for many large scale distributed systems.



Joe Armstrong, the co-founder of Erlang

While, it may not be as ubiquitous in the wider technology sector as are other languages like C or Java, its relevance grows day by day as our systems grow more and more complex and distributed at the same time. It's also getting more and more dominant because of the increasing focus and availability of multicore processors, which makes concurrency and multiprocessing even more important. Erlang's usage has been more than dominant in developing modern software which are used in systems like chatting, commenting, managing transactional data and networking devices.

The success of Erlang in modern distributed systems has been largely untold, despite the fact that companies like Facebook, Whatsapp, Amazon, Rackspace etc., along with popular open source software like RabbitMQ and CouchDB actively use it as part of the underlying stack in their software. It's also used as part of the Disco MapReduce framework and the YAWS web server.



Riak is a NoSQL database that uses Erlang



CouchDB is a NoSQL document store that has been developed in Erlang

## How, What and Why Erlang?

A conventional programmer would look at Erlang as a language which tackles concurrency, resilience and flow of data in a very unique manner and this may seem unusual or difficult to some of you. But some of these features along with excellent learning material makes the language very attractive once you are in the loop with all of these aspects. OTP or Open Telecom Platform forms the most important part of this language. It is a set of libraries and a full fledged framework that provides communication protocols, static analysis tool, interpreter, compiler and a distributed database called Mnesia. The language can be compiled and run on top of a virtual machine just like Java, or it can also be used in an interactive environment like Python. The compiler, known as beam, converts the source code into a byte code, which is then converted to threaded code at the time of loading. Some of the core principles behind how the language works can be summed up with the help of below points:

### ◆ **Concurrency and Inter-Process Communication**

Concurrency and distributed computing is one of the primary facets on which the language has been built from the ground up. A major factor that helps in concurrency is the fact that all data structures in Erlang are immutable and cannot be changed, this avoids having to focus on locks, semaphores and other ways of trying to ensure the sanctity of shared data. Processes are like the primary building block that form a part of an Erlang program, and each process communicates with another using an asynchronous message passing protocol. No data is shared between any two processes and this is irrespective of the fact whether these processes are running on the same node or are part of a larger distributed cluster. This also makes it more secure and predictable, because data is shared only when there is clear and explicit communication between the two processes.

### ◆ **Pattern Matching**

Pattern matching is another feature that is inherited from the concept of functional programming. This practice promotes declarative syntax and makes working with complex data structures much easier. List Comprehensions for example, which are inspired from Lisp and other such languages also form an important part of Erlang.

### ◆ **Robustness**

Erlang has been made to ensure fault tolerance and robustness of a system.

The messaging system in Erlang has been implemented in such a way that even if a process fails or stops executing, it can communicate the reason before it is killed, so that the remaining processes can bypass or take appropriate action. These error handling mechanisms allow the developers to create the program in such a way that processes are allowed to fail and this makes the code cleaner and shorter because most of these things are handled by the in built libraries themselves, rather than the programmer having to code them explicitly.

#### ♦ Hot swapping or code loading

Keeping the concept of distributed computing in mind, Erlang ensures that any part of its code or program can be replaced or updated without any problems or hiccups. This makes scalability an important feature of the language and you can take advantage of multicore processors, add new nodes or change things transparently without any impact on the rest of the system. This is made possible because of the fact that a process can contain a reference to an old as well as a newer version of the same module and can cleanly make the switch at a specific time with the help of an external call that is made to the module.

## The Next Step

Given that there isn't as big of a community around Erlang, as is for other languages, there are limited outlets where you are able to look at Erlang code and learn it from scratch. But some of these resources including the official documentation can help you get an overview. You can give it a test run at [www.tryerlang.org](http://www.tryerlang.org) in the form of an online interpreter where you can try and run some of the basic commands.

However, one of the best and easily the most popular Erlang book in our opinion is available online at <http://learnyousomeerlang.com/> written by Fred Hebert. It includes an excellent introduction and takes you step by step with the help of some funny cartoons. Some other good books include Programming Erlang by Joe Armstrong and O'Reilly's Erlang Programming by Francesco Cesarini and Simon Thompson. To get started you can either download it from <http://www.erlang.org/> or get it from your distribution's repository if you are on a linux system.

A good way to start learning would be to start looking at functional programming concepts, move towards various data structures and then look at distributed and concurrency features at the last. A sample

program to show you how to define the fibonacci function would be as follows:

- `fibonacci(0) -> 0 ;`
- `fibonacci(1) -> 1 ;`
- `fibonacci(N) when N > 0 -> fibo(N-1) + fibo(N-2) .`
- An example of immutable variables can be shown by trying to assign different values to a variable at different points in time: -
- `1> A = 456.`
- `2> A = 345.`
- `** exception error: no match of right hand side value 345`

Lists in Erlang are ever present and can be used for many operations. They can be defined by square brackets and can contain anything, even functions: -

- `3> [a,b,c,34,45,[1,2,3]].`
- `[a,b,c,34,45,[1,2,3]].`

Tuples are another data type which can be used for comparison operations and are defined with curly braces: -

- `4> {a,b,c,34,45,{1,2,3}}.`
- `{a,b,c,34,45,{1,2,3}}.`

Atoms are static string literals which are used internally for message passing and other such functions: -

- `5> hello.`
- `Hello`

This concludes our discussion and overview for this language but you can go through the links and documentation in the references section.

## References

- ♦ <http://www.erlang.org/course/course.html>
- ♦ <http://www.erlang-in-anger.com/>
- ♦ <http://erlangcentral.org/>
- ♦ <https://www.erlang-solutions.com/>
- ♦ Joe Armstrong's Thesis: [http://www.erlang.org/download/armstrong\\_thesis\\_2003.pdf](http://www.erlang.org/download/armstrong_thesis_2003.pdf)
- ♦ Erlang Handbook: <http://opensource.erlang-solutions.com/erlang-handbook/>
- ♦ <http://chimera.labs.oreilly.com/books/1234000000726/index.html>
- ♦ [http://en.wikibooks.org/wiki/Erlang\\_Programming](http://en.wikibooks.org/wiki/Erlang_Programming) 



# HASKELL

While largely ignored in favour of more traditional languages such as C, Java or Python, functional programming languages are beginning to make a comeback, particularly in the fields of analytics and big data. We take a look at one of the most popular functional languages out there, Haskell.

**H**askell is the new kid on the block of programming languages. It too follows the computing paradigm called 'Functional Programming' i.e. approach a problem in terms of what the solution should look like, rather than what steps should be taken to get to that solution.

Though Haskell has risen in popularity in recent years, the language itself is more than 20 years old. As far back as 1987, there existed at least a dozen implementations of a purely functional language, and at the conference of Functional Programming Languages and Computer Architecture (FPLCA) in Portland, it was decided to publish an open standard for functional programming to promote it as an alternative style of programming.



Haskell Curry, the mathematician whom the language was named after

## What is Functional Programming?

One example that's oft cited is that of formulas in Microsoft Excel. When entering formulas in a cell, the final value is always expressed in terms of the values of other existing cells. No attempt is made to specify the order of evaluating these cells; instead, we expect that Excel will figure out dependencies on its own. Similarly, no attempt is made at managing the memory used by the expression. Finally, we program using expressions rather than instructions - an expression merely establishes the relationship between the inputs and the outputs without explicitly listing the steps

required to fulfil that relationship.

As the name suggests, functions in a Functional Programming language (including Haskell) are first class citizens. This means that they can not only be invoked and called like their imperative counterparts, but also that functions can be passed as arguments to other functions, and can even be defined within the scope of other (parent) functions. This feature allows nearly all programming constructs to be expressed in the form of functions.

A hallmark feature of all functional programming languages is that they are said to be 'lazy'. So it only evaluates an expression when it is required, and not before. This is highly advantageous, especially when dealing with compound expressions (expressions where the operands themselves are expressions) or with infinitely large data structures such as infinite lists. Hence, these languages are far more memory efficient than their 'strict evaluation' counterparts. Furthermore, lazy evaluation reduced the number of computations by storing the result of all function calls against their parameters in the form of a lookup table which is used the next time the same function is called without bothering to re-compute it.

Take the example of computing the entire Fibonacci sequence. In Haskell, the sequence can be computed using the following one-liner:

- `fibs = 0 : 1 : zipWith (+) fibs (tail fibs)`

Here, 'fibs' is the name of the list that will store the sequence, '0 : 1' is the initialization of the list, and the 'zipWith' operator performs an operation (in this case, addition) on the corresponding elements of the two operators following it. The first operator is the list fibs itself, while the tail operator

returns the list without the first element. Notice how no attempt is made to restrict the list. Since we haven't actually tried to retrieve a value from this list, it will not be computed at all. If we were to call an element of the list (using `fib n !! <element number n>`), the list would then compute all elements upto and including the desired element, and return the `n`th element. Had the same logic been implemented in an eager or 'strict' language like as C or Java, it would've resulted in an infinite loop (or an exception, at best)

## Hello World in Haskell:

Put the following line in file called `hello.hs` and run it with `ghc` (included in the DVD)

- `main = putStrLn "Hello, World!"`

Run it with the command `ghc -o hello hello.hs`

## Features of Haskell

Defining functions in Haskell can be done in two ways, one where the single function takes multiple parameters (or arguments) and returns a value, and the other where a function acts on a single argument, which returns a function that will then take the second argument as its sole argument, and so on. The second approach is a more 'functional' way to approach programming and the result is more compact than the former. This approach is called currying, named after the American mathematician Haskell Curry.

Here is an example to illustrate the difference in the two approaches - the following function 'hyp' computes the value of the hypotenuse of a right angled triangle given the lengths of the two shorter sides.

Currying:

- `hyp :: Float -> Float -> Float`
- `hyp x y = sqrt (x*x + y*y)`

Non-Currying:

- `hyp :: (Float, Float) -> Float`
- `hyp (x,y) = sqrt (x*x + y*y)`



Relevant XKCD

As can be seen, the first approach is more compact (uses fewer characters), and is hence used as per convention. Similarly, Haskell also supports anonymous or lambda functions using the `\` operator.

List comprehensions are a way of synthesizing a list using an expression, where each element in the list is logically related to a parameter. For example, to make a list of all numbers from 1 to 100, the following Haskell one-liner is valid:

- `[ f | f <- [1..100]]`

We can include powerful logic to modify our lists during runtime. For e.g., a list that contains the squares of all numbers present in a list `l`,

- `[f*f | f <- l]`

Similarly, a list that stores all the factors of a number `n` can be expressed as,

- `[f | f <- [1..n], n mod f == 0]`

(Here the mod operator returns the remainder after dividing `n` with `f`)

Comments in Haskell are of three types - single line comments start with `-` and end with a newline character. Multi Line comments begin with `{-` and end with `-}` and can include nested comments or code. Literate comments, a third type, aim to demarcate the code instead of the comments and start with `\begin{code}` and end with `\end{code}` as is commonly found in LaTeX. This allows Haskell programs to be easily typeset in LaTeX documents.

Monads are one of the most important features of Haskell, and can be thought of as the programmatic equivalent of an assembly line. Data that is input into a monad gets modified in stages as if it were to be on an assembly line. Formally, a monad is simply a way of describe any computation as a sequence of steps. Monads are used to define how a particular data type should behave when multiple operations on it are chained together, or how functions that are nested inside other functions should return values. Monads may be constructed for computations that handle I/O, change state of their resident variables or return multiple values.

## Implementations of Haskell

The most commonly used (and full featured) Haskell compiler is the GHC (Glasgow Haskell Compiler) and is available for Windows, Mac and Linux <http://dgkit.in/fitHaskell>. Other implementations of Haskell such as lbc, Gofer and Yale Haskell are also available, but GHC is the de-facto standard for Haskell compilers. You can find the latest version of GHC on the DVD accompanying this month's FastTrack issue. Text editors such as SublimeText, Vim, IntelliJ and Eclipse support Haskell.

## Haskell In The Wild


A number of real world projects are written either partially or entirely in Haskell.

- ◆ Darcs is a source code management system that relies on a sophisticated system of diffs rather than snapshots and supports spontaneous branching.
- ◆ Large corporations such as Google, Facebook, NVIDIA, Bank of America and AT&T use Haskell internally for a number of support tools.
- ◆ Chordify (<http://chordify.net/>) is a popular chord transcribing service that takes a song from YouTube, Soundcloud or an uploaded audio file, and displays the chords used in the song. Haskell is used for modelling the musical harmony between chords in the functional domain.
- ◆ The New York Times used Haskell's parallel array library to process images from the 2013 New York Fashion Week.
- ◆ Silk (<http://www.silk.co/>) a tool used to publish data visualizations in a beautiful way, uses Haskell to generate infographics from their data.



Chordify transcribes a song's chords and uses Haskell to do so

## Learning Haskell

There exist a large number of Haskell tutorials on the web and in print form. The most commonly recommended one is Learn You A Haskell For Greater Good, available as a free HTML ebook on <http://learnyouabaskell.com/>. edX, the popular online courseware site, offers a course called Introduction to Functional Programming which uses Haskell to illustrate functional programming concepts (<http://dgit.in/edXHaskell>). The School of Haskell is a series of interactive web tutorials on the Haskell programming language (<http://dgit.in/SchoolHaskell>). Try Haskell allows you run interactive Haskell commands right inside your browser without the need to install anything (<https://tryhaskell.org/>) 

# JAVA

## The language that truly propagated the era of “write once and run anywhere”

Java is usually the first programming language most coders start out with since it is easy to understand and runs cross-platform. In fact Java is often referred to as ‘lingua franca’ or common language since such a vast number of developers are using it. Since just about everyone has either used it in the past or is still doing so, troubleshooting couldn’t be simpler – you’re guaranteed to find someone who can help.

The developers of Java originally wanted to name it Oak. But the name was already trademarked by Oak Technologies, so a series of brainstorming sessions were held to come up with a dynamic and fun name that would attract coders of the next generation. The team came up with three options that were legally acceptable- Java, DNA and Silk. Finally Java was chosen.

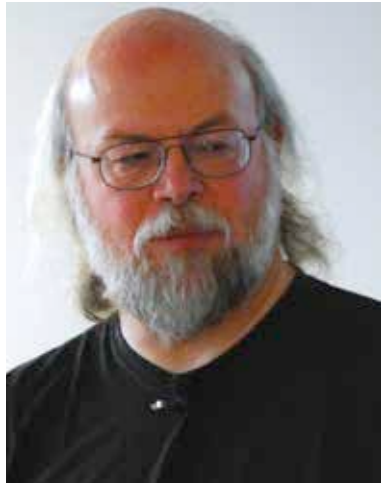
### **Write Once Run Anywhere (WORA)**

The term WORA is used to describe Java which means write once run anywhere. Once compiled the intermittent JVM or Java Virtual Machine takes compiled byte code and converts it into machine code which is understood by the machine. So once compiled your Java code can be run on any system irrespective of the underlying hardware or operating system. This feature of portability is what makes Java most attractive to programmers.

Java derives a lot of its syntax and features from C++ but adds a lot of its own capabilities in the process.

Java is used to develop enterprise-wide applications, web applications, games and now mobile applications.

In the mobile age of today Java is more relevant than ever. Android programming is largely done using Java. In fact the whole Oracle v/s Google lawsuit was because Oracle claimed Google infringed on their Java APIs to develop the Android operating system and APIs.



James Gosling

## Java Editions

There are four editions of Java that target different application environments. The Java APIs are segmented to belong to one of the editions.

- ◆ Java Card for Smart Cards
- ◆ Java Platform, Micro Edition (ME) for environments with limited resources such as mobiles.
- ◆ Java Platform, Standard Edition (SE) for workstation environments
- ◆ Java Platform, Enterprise Edition (EE) for large distributed enterprises or Internet environments.

There are also a number of lightweight languages built on top of Java that can use the JVM eg Groovy (a dynamic language with features of Python, Perl, Smalltalk and Ruby), Clojure, Jython (a Python interpreter), JRuby (a Ruby interpreter). Each of these were built for their own dedicated purpose.

## Java and Android

The Android operating system is built on the Linux kernel which is written in C. Android applications however are written in Java. Android does not use the Java Virtual Machine or JVM. It has its own Virtual Machine called the Dalvik Virtual Machine which does all the byte code conversions. The Java code is compiled into proprietary byte code and run on the Dalvik Virtual Machine. The Dalvik VM has a register based architecture unlike the Java VM which has uses a stack based architecture.

## Oracle's Java timeline

Oracle's Java Timeline makes for interesting reading if you want to know the history of Java. It shows how Java has evolved and grown to become an integral part of our lives. <http://bit.ly/13oEXgg>

## Hello world

The first line of code written in any programming language is a print statement of 'Hello World'

So here it is in Java.

```
• class FirstProgram{
• public static void main(String args[]){
• System.out.println("Hello World!");
• }
• }
```

## Object Oriented Concepts

Conventional or procedural programming consists of executing a sequence of commands which input, output and manipulate data. Functions and subroutine allow the commands to be called repeatedly from different parts in the program.

Object oriented programming consists of data and functions called methods which are bundled together into software 'objects' called 'classes'. A Class is the blueprint from which objects are created.

Object oriented programming makes it easier for programmers to structure and order the code. For example take a Racing game.

Here 'Car' will be defined as an object having data members like make, model and speed. Car will also have methods like accelerate, turn left, turn right.

```
• Class Car {
• String make;
• String model;
```



I see lawyers coming!



```

• double speed;

• public void accelerate(){
• //code to accelerate
• }

• public void turnLeft(){
• //code to turn left
• }
• public void turnRight(){
• //code to turn right
• }
• }

```

There can be two cars or there can be a hundred. Each will be called an instance of that object. Individual instances can be modified without affecting other parts of the program.

## Core OOP Concepts

### Data Hiding/Abstraction

When data is grouped into classes/objects, programmers have the option of making data private to certain classes. Therefore the outer world will see only what is required to use the class without having access to any unnecessary sometimes private information.

### Data Encapsulation

Wrapping up logically related data and functions that operate on this data into bundles called classes.

### Packages

Containers for logically related classes and interfaces bundled together.

### Inheritance

When one class can be based on another class, or reuse that implementation. It can also add its own features to that implementation. In our example above the class Car could have inherited certain features from another class (called its 'super class' or 'base class') called 'Vehicle'.

### Polymorphism

Polymorphism is the ability for a method in Java to be able to do different things depending on the object it is acting upon. For example a method called `accelerate()` in our Car class could have different implementations depending on the individual Car object or depending on the parameters provided to it.

## Java Memory Management

One of the Java languages finest achievements is its memory management. Java has built in garbage collection. Developers can create objects without worrying about explicitly allocating and de-allocating memory.

The garbage collector automatically reclaims memory it deems not in use. This eliminates memory leaks and other memory related problems. Garbage collection maintains program integrity and is an important part of Java's security strategy. It also stops heap fragmentation which occurs when memory is manually de-allocated freeing up a lot of intermittent space in between.

## Learning tools

The tools available for learning and programming in java are excellent. Not to mention free. Sophisticated Integrated Development Environments (IDEs) such as NetBeans and Eclipse as well as web servers (TomCat), application servers (Glassfish, JBoss).

## Software

The Oracle docs have specified a set of highly useful tools to learn Java and Java-like languages depending on both age and experience. Whether you are an experienced programmer or a complete novice there is something in there for you. Whether you are 5 years old or 80 they have an easy interface for you to begin with.

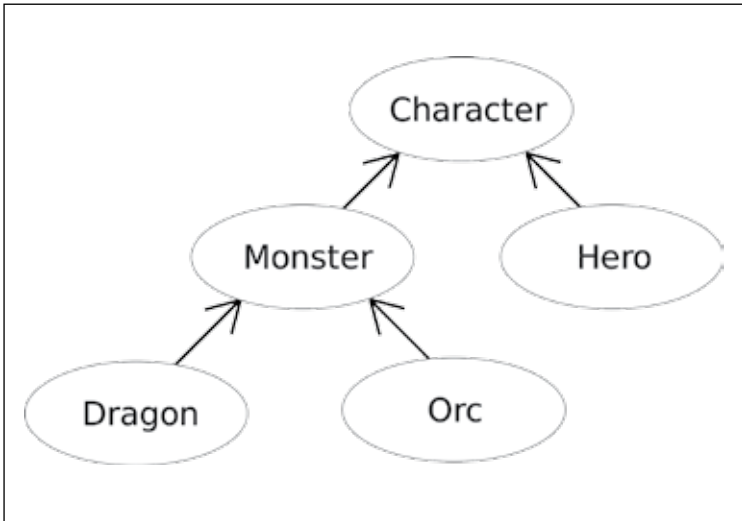
**Scratch:** A simple programming interface with drag and drop features. This tool was built for kids between the ages of 5-15 with no programming experience.

**Greenfoot:** A 2D visual software with a built-in code editor to create games and simulations. Again, this is for starters with no prior programming experience, but aged 13 upwards.

**Alice:** A 3D educational tool for animations with a drag and drop interface. No programming experience required. No age limit either.

**BlueJ:** This is a slightly more sophisticated tool where users are expected to write their own code. It is not an overwhelming IDE like Eclipse or NetBeans and provides the bare basics to run a Java program. A little coding experience is required to use BlueJ. High school kids usually start with this.

**NetBeans/ Eclipse:** These are fully featured Java Integrated Development Environments used in educational as well as professional environments.



How Inheritance works

## Documentation and Websites

The Oracle online java documentation has pretty much everything that you need to get started. All functions calls and classes are explained well.

## Tutorials

[www.mykyong.com](http://www.mykyong.com)

MIT OpenCourseware has Java tutorials

Coursera has not only courses for Java but courses that will teach you the basics of design patterns/principles and algorithm development.

## Troubleshooting

For doubts about code and questions about how best to tackle a particular obstacle StackOverflow is possibly the best quorum out there. They even have a section for Code Review where senior peers will review the code you have written and give you feedback.

<http://stackoverflow.com/>

<http://codereview.stackexchange.com/>

# JAVASCRIPT

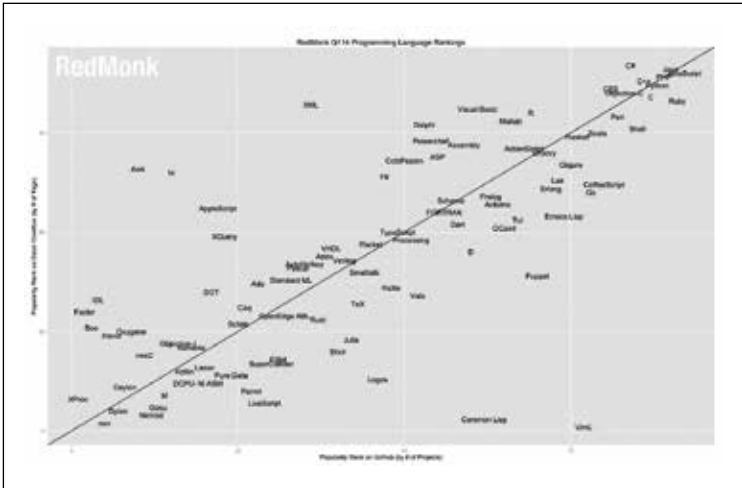
If you are interested in Web Development, JavaScript is one of the languages you must learn.

JavaScript was developed in May 1995 by Brendon Eich who worked for Netscape Communications Corporation. JavaScript was conceived from the need to make web pages more dynamic. Eich created it in just 10 days. JavaScript was originally named Mocha by Marc Andreessen the founder of Netscape. The name was then changed to LiveScript before finally receiving a trademark license from Sun to be called JavaScript. Though they did not want people to think the language was influenced by Java, they wanted to latch on to the hype and publicity Java was receiving in the early 1990s.

Environment-wise, JavaScript is probably the easiest language one can execute. Anyone with a computer or a smart phone is fully equipped to run JavaScript programs or even make their own. The only requirements are a browser and a text editor. No complex developmental environments to



Brendon Eich



RedMonk's popularity ratings for programming languages lists JavaScript right up there

install. No IDE's to get familiar with. Due to the ease with which JavaScript can be used and executed, many developers think of it as a toy or a weak language. However it is probably one of the most advanced languages developed to date. Many very advanced projects have been developed using JavaScript from office suites such as Zoho.com to social integration tools such as Facebook's JavaScript SDK.

## Most popular language

Every programmer worth his salt will have a Github account. So one can tell a lot about the current technology in use, by the number of Github projects based on it. JavaScript Projects have shown a steady incline every since the creation of Github in 2008.

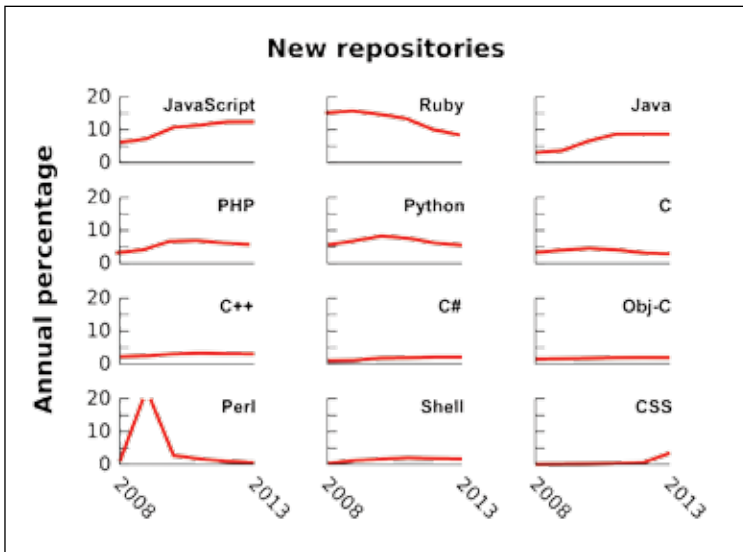
Ironically Ruby, the language in which Github was written has shown a steady decline.

It should be noted that NodeJS, a JavaScript framework (more on this later) has been a major contributor to the increase in JavaScript projects.

## Most impressive features

### Just In Time (JIT) compilation

Just In Time (JIT) compilation is compilation done at the time of execution. JIT Compilation is also called dynamic translation. This allows for high



JavaScript winning another popularity contest by a mile, and then some

speed execution. In modern browsers JavaScript is compiled, optimised and executed at runtime. The performance is so good that it is comparable to software written in C or C++.

## Object Oriented Programming

JavaScript uses all the object oriented programming concepts such as encapsulation, inheritance, polymorphism, etc., and adds to that as well. In fact some OOP concepts can be implemented in better ways using JavaScript features called 'lambda' and 'modules'.

## Syntax

JavaScript shares some syntax with Java, though its own syntax is easier, concise and more flexible. It has been adapted to become the dominant standard of client-server communication using JSON. JSON has since replaced XML since it is more compact and flexible.

## Asynchronicity

Everything executed in the browser occurs in a single loop hence JavaScript code is highly dependent on event handlers and AJAX calls. This allows

for a smooth and responsive UI no matter what the background processing may be.

## **Reusable**

A language wouldn't be worth learning if the code were not reusable. Especially for the web where tasks such as validation are required often. JavaScript allows you to write code that can be run both server side and client side.

## **Advantage of JavaScript over Others**

### **Built-in interpreters**

All browsers have JavaScript interpreters built in. No other language has this advantage and its highly unlikely that this will change any time soon.

### **Easy debugging**

All browsers also let you debug your code easily. If dissatisfied with this you can use a plugin or tool to help debug. These plugins are free and small in size. Firebug for Firefox and Web Developer Toolbar for Chrome are examples of this.

### **Functions as objects**

Functions can be used as objects and variables. This is a highly useful tool when it comes to reusability. You can literally mold old code on the fly.

### **Ready resources available**

JavaScript developers are privy to a vast collection of free and powerful Frameworks and libraries.

### **Ultra fast execution engines**

There are three ultra fast JavaScript engines today each competing with the other to become better. These are FireFox's SpiderMonkey, Google V8 and Safari JavaScriptCore.

### **Both Clientside and Serverside**

Javascript was originally made as a purely client side language but gradually became a server side implementation as well. NodeJS is one such example.

## **Hello World in JavaScript**

All JavaScript is written within HTML code and enclosed between two tags `<script>` and `</script>`.

- `<script>`
- `....your code`
- `</script>`

So let's go ahead and see what our standard snippet of code that says Hello World looks like in JavaScript.

- `<html>`
- `<body>`
- `<script>`
- `alert("Hello World!");`
- `</script>`
- `</body>`
- `</html>`

This bit of code, when executed will output “Hello World!” in the form of an alert box on your screen.

Alternately you could print it directly on the webpage by using the `document.write()` statement.

- `<html>`
- `<body>`
- `<script>`
- `document.write("Hello World!");`
- `</script>`
- `</body>`
- `</html>`

## Using JavaScript frameworks and libraries

Developing large software can be daunting and time consuming. To overcome this, developers resort to frameworks and libraries. A framework is basically a tried and tested design/configuration of your software components. The most popular JavaScript frameworks are AngularJS, Backbone.js and Bootstrap.

A library is a set of predefined functions grouped together that are reused in many places implemented in the most efficient way possible. The most popular libraries are jQuery, Prototype and Dojo. Libraries allow you to do anything from creating and displaying graphs to processing and displaying PDF files.



## Interesting Tools

### Grunt

Grunt is a JavaScript task runner that's used to automate repetitive tasks such as minification, unit testing, linting etc. Configure a Gruntfile and the task runner does everything else. Twitter and Adobe actively use Grunt.

### Mocha

Mocha is a testing framework for JavaScript that allows you to perform both synchronous or asynchronous tests. It is hosted on Github.

### MongoDB

It is a document based NoSQL database. It offers fast querying and updating, flexible aggregation and data processing. It is easy to scale and uses internal memory for storing current data which enables faster access.

### Sinon

Sinon is a Javascript library which provides spies, stubs and mocks for JavaScript. A Spy is a function that records metrics passed to it. A stub is a spy with some preprogrammed functionality. It can perform actions based on the metrics passed to it.

The four tools mentioned above only graze the surface of the vast amount of readymade tools available for JavaScript. There's probably a tool for almost every need, and if there isn't it's easy to write your own, but remember to make it free for others to use.

## Tools to learn JavaScript

### W3Schools for HTML and CSS

To learn JavaScript you must have some basic knowledge of HTML and CSS. For that w3schools provides some fairly good step by step tutorials for the bare basics. It should be used only as a quick reference tool though since as a learning tool it doesn't quite measure up. [www.w3schools.com](http://www.w3schools.com)

### The Mozilla developer network

The Mozilla Developer Network is a also good place to start. It always has freshly updated content, is well maintained and has thousands of documents on a variety of subjects such as HTML5, Node, Javascript etc. <https://developer.mozilla.org/en-US/>

## Code Academy

CodeAcademy has some free tutorials, and you don't even have to sign up to use them. They're slow though, so if you're an experienced coder, you won't like them. <http://www.codecademy.com/en/tracks/javascript>

## Douglas Crockford

For the slightly experienced, Douglas Crockford from Yahoo! has some good tutorials you can download and watch. He gets straight to the point, and focuses on how to get things done rather than delve into the minor features. <http://yuiblog.com/crockford/> or buy his book, JavaScript: The Good Parts.

## Tools for JavaScript frameworks

### AngularJS

AngularJS offers a powerful framework to build large apps, but has a steep learning curve. It is often described as a hockey stick – very easy initially and then increasingly hard thereafter. If you master it, AngularJS will help to build the most robust, large scale applications. <https://thinkster.io/angulartutorial/a-better-way-to-learn-angularjs/> and <https://angularjs.org/>

### Backbone.js


Backbone.js is highly useful for building Single Page Applications (SPAs). It is a small library, but gives you an enormous amount of functionality. <http://code.tutsplus.com/series/getting-to-know-backbonejs--net-24408> and <http://backbonejs.org/>

### Bootstrap

Bootstrap is a framework to develop mobile-first, responsive projects on the web. <http://getbootstrap.com/> and <http://www.tutorialspoint.com/bootstrap/>

### JSFiddle

JSFiddle provides a custom environment to test your JavaScript,HTML and CSS right in your browser.

It is described as an online playground for JavaScript. Users can select individual JS versions, libraries and frameworks with which they want to test their code. This is a must use tool. <http://jsfiddle.net/> 

# PHP

## The power behind the world's most powerful websites

### Introduction

In 2004, when Mark Zuckerberg sat down to create the site that needs no introduction, he chose PHP. Moving forward the code grew and it is currently one of the biggest websites the world has ever seen. If you know the early history of Facebook, you'd know that it was not meant to be the behemoth that

it is today. There had to be something wise about Mark to choose PHP for the task and something equally incredible for PHP to carry his website to create a world-wide-phenomenon. Probably it is the same thing in PHP which attracts or attracted anyone else to work with it.

Welcome to the world's most loved and most hated programming language – PHP.



PHP is the world's most popular server-side scripting language

### History

PHP was originally called PHP/FI (Personal Home Page/Forms Interpreter) in 1994 when it was originally designed and was renamed to PHP Tools in 1995 before it was renamed to just PHP (as a recursive acronym

meaning PHP Hypertext Preprocessor) in 1997 with the release of version 3.0. The initial name of PHP clearly suggests that it was not designed to be what it appears as today – the king of web programming languages. In fact, its creator Rasmus Lerdorf built it as a set of CGI binaries which would help him program his home page better with some database support. It actually wasn't meant to become a full-fledged programming language.

The initial public release of PHP in 1995 had, quite surprisingly, a large set of the basic capabilities that it still has today, and made mostly internal improvements across versions. It added object oriented programming capabilities in its version 5.0 release, done in year 2004. Since then a lot of features have been added to PHP including namespace, late static binding and closure support in version 5.3 and traits in 5.4 with a large number of features in other releases bringing it at par with other programming languages.

## Significance and programming

PHP is known for being one of the easiest programming (or scripting) languages to start off with, and is thus used by a large number of programmers who are starting out.

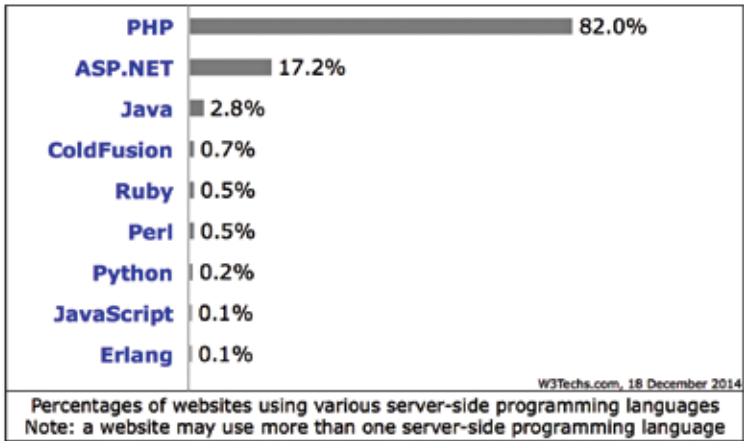
Another point where PHP shines out is being an almost-web-exclusive language. While PHP can be easily and effectively used for system scripts, it was



The world's most popular social networking website runs on PHP

built for handling websites and flaunts features which address issues specific to web development such as handling forms submission, user sessions, cookies and loose data types. PHP is designed to work in a 'single request-single response' environment where the script starts working when a request comes in, processes the request,

generates the response, sends it to the client and then exits. The next time the same script is requested, it runs again, serves the request and again ends the execution. This model suits the architecture of the web where a user makes requests sporadically to the server. At the same time, it can also be used to create sustainable services, which keep running indefinitely. This makes PHP suitable for working with WebSockets API introduced in PHP5.



PHP powers more than 80% of sites across the globe!

PHP allows the developer to mix HTML and PHP together while also being suitable for coding in a number of software patterns. Primary among those is the MVC software pattern, which suits the architecture of web very well (almost all PHP frameworks are modeled in MVC pattern). This flexibility helps in fixing

PHP powers more than 80% of all websites in the world with ASP.NET holding the second position with a little over 15% of the share! Most popular blogging tools and content management systems from wordpress and drupal to MediaWiki and OwnCloud run PHP. Also, some of the world's most popular websites including Facebook, Wikipedia, Wordpress.com and Yahoo! are powered by PHP. With the world moving towards the "Internet of Things" where a lot of server-side connections will be required, PHP might push its dominance even further.

## Future

It is planned that the next major version will be named version 7.0 (and not 6.0) to avoid confusion with content in books that were already written for PHP 6.0, which was later released as version 5.3. Version 7.0 will address a long known drawback – confusing order of parameters for built-in functions. Also, inspired by Facebook's HHVM interpreter for PHP, the PHP engine (parser and executor) is being overhauled to bring in lots of speed improvements. Certain tests indicate about 100% performance improvement

over the current version while largely maintaining the existing syntax and API. The future looks promising for PHP.

## Show me the code

The first entry to any programming language starts with a 'Hello world' program. Here is how it looks in PHP:

- ```
<?php echo "Hello World"; ?>
```

But that is too simple. Right? Let's try to print something little more interesting:

- ```
<?php
```
- ```
$x = 5;
```
- ```
$x_square = $x * $x;
```
- ```
$result = "The square of $x is: " . $x_square;
```
- ```
echo $result;
```
- ```
?>
```

This small program shows a good deal of features of PHP which make web development easy. The output of this program is:

- ```
Value of $x is 5 The square of 5 is: 25
```

Line 2 shows how you can dynamically declare a variable and assign it a value. Notice that there is no data type for the variable. Data type is automatically determined.

Line 3 shows you can perform simple mathematical operations as easily as you do in any other programming language. Once again, creating the variable is dynamic.

Line 4 shows that a string can be concatenated (using the '.' operator) with an integer (\$x) and printed on screen without any effort. PHP does the type conversion for display so you do not have to worry about it.

Line 5 shows that inside a double quoted string, a variable is evaluated before printing. Also, concatenation of a string and an integer can be stored into a variable and the resulting variable can be printed (line 6) and will be shown without an error.

Isn't it interesting?

Let's move to creating an array, sort it and display it:

```

• <?php
• $arr = [1.5, "4", "10.83", 2];
• sort($arr);
• foreach($arr as $x){
• echo $x . ', ';
• }
• ?>

```

The output of the above program is:

```

• 1.5, 2, 4, 10.83,

```

This program creates an array with 4 elements, which can all be read as numbers (floats and integers) but 2 of them are represented as strings (yes, an array can contain different data types in PHP allowing flexibility) and PHP converts them all to numerical form and then does the sort. Also, a 'foreach' construct shows that you can handle the elements of an array pretty easily. If you were to introduce a string in that array which cannot be converted to a number, PHP will treat all elements as strings and sorting will happen considering all elements as strings but it would not fail. You can alter the strictness of PHP's behavior in many cases by adding additional checks.

We have barely touched the surface of PHP. If it looks interesting, and you would like to learn more, go on reading.

## Tools and resources

While PHP requires you to have a web server on machine, we cannot discuss its installation in detail. Depending on your OS, you can best install PHP as follows:

1. Windows: Install XAMPP – this is the best way to start off on Windows.
2. Mac OS X: PHP and Apache come bundled with OS X. We cannot detail the process here due to space constraints. You can read the official PHP guide for the same here: <http://php.net/manual/en/install.macosx.bundled.php>.
3. Linux: You can install PHP easily using your package manager. Most Linux DVDs also carry PHP with them. On Ubuntu, it is as easy as:
 

```

sudo apt-get install apache2 php5 libapache2-mod-php5
php5-mcrypt

```

Once installed, you can start writing code in no time. Any simple text editor will do the job. But in time you would need to get faster. Eclipse with

PHP tools is a good free IDE for PHP development. If you want a very professional tool, there is probably no IDE which can beat PhpStorm. However PhpStorm is a paid product which is available to students, lecturers and open source contributors for free – all you have to do is apply for a free license at JetBrains.com.

In the course of learning any language, one of the most important resources is the API documentation of that language. If you are keen to learn PHP, you are lucky – PHP has one of the most useful, easy and detailed API documentations with examples, warnings, best practices guide along with user comments for almost every function in the language (and many other popular extensions).

**strcmp**

(PHP 4, PHP 5)  
strcmp — Binary safe string comparison

**Description**

```
int strcmp ( string $str1 , string $str2 )
```

Note that this comparison is case sensitive.

**Parameters**

**\$str1**  
The first string.

**\$str2**  
The second string.


**Return Values**

Returns < 0 if \$str1 is less than \$str2; > 0 if \$str1 is greater than \$str2, and 0 if they are equal.

**Examples**

PHP's API is thoroughly documented

The popularity also means that you can search for your problems on the web and would usually get a good answer written already. If not, StackOverflow.com is a great place to ask questions. If reading books and understanding the underlying architecture is your style, Programming PHP, by Rasmus Lerdorf (creator of PHP) himself would be the best you can get for the language.

When you go for PHP development, unlike most other programming languages (except Java, probably), you will be exposed to an overwhelming number of frameworks. Remember to read about MVC (Model View Controller) architecture and how it works because moving forward, that might help you a lot. 



# PYTHON

Python is probably one of the best languages to start with if you are new to programming and it continues to be useful even as you mature as a developer

Python might seem like a recent development but at this point it is 20 years old. The first release of Python 1.0 goes as far back 1994. Even the Python 2 series which is the dominant version of Python to this day was released way back in 2000. The language was created by Guido van Rossum and he continues to have a central role in its development.

Python is in a rather odd situation currently where the most popular version of the language is the Python 2 variant despite the availability of Python 3 since 2008. Python 2 had its final major release in 2010 with the release of Python 2.7 and it will only be bug fixes from here. Python 3 itself is currently at 3.4 with 3.5 to release next year.



Guido van Rossum, the creator of Python worked for Google while they launched Google App Engine with Python support out of the box. He now works for Dropbox.

This preference for Python 2 comes from the fact that the Python 3 breaks compatibility with Python 2 code. Python 3 was a major update to the programming language and the developers took the opportunity to modernise the language in many ways and get rid of poor decisions made in the past.

The syntax differences between Python 2 and 3 kept a majority of Python libraries from instantly supporting it, and some are yet to support it. While there is an automated tool for converting Python 2 code to Python 3 code, it doesn't work in all cases. The tides are slowly changing as Python 3 gains new and enticing features, and more libraries are ported to it. To further ease this process Python 3.3 brought its syntax closer to 2 and made it easy to make code compatible with both versions. While this might make it sound like they are very different languages, they are really not, it's just the nature of the changes that has caused these issues rather than their intensity or quantity.

With talk of versions out of the way, let's look at the language itself, which is unconventional in many ways. Python takes a very different approach to structuring code, rather than denote blocks of code with curly braces {}, it uses whitespace indentation. Most languages keep the formatting of the code separate from syntax, which means you are likely to run into dozens of different coding styles and arguments over which is better. Python makes the indentation part of the language resulting in more consistent looking code.

Python has a core philosophy, and the more you code in Python the more you tend to understand what makes certain code more or less 'Pythonic'. Long time Python developers can be said to have gotten the 'The Zen of Python'. You can read 'The Zen of Python' on their website, or by typing ``import this`` in a Python console.

## Significance

Possibly one of Python's greatest strengths is its simplicity. It is easy to get into it. Python is widely used, to the point that it is present by default if you're running OSX. The same can be said of most Linux distributions. Installing Python on Windows isn't that hard either.

When you reach the limits of what can be accomplished using the Python's inbuilt libraries, there is an extensive ecosystem of packages out there to explore. The PyPI or Python Package Index is a website that lists all Python packages submitted to it. This index of packages is accessible from the command line with the ``easy_install`` and ``pip`` which lets you search this index, download packages and update them if need be. You can

just list all the packages you want for your project in a text file and `pip` can install them all automatically. Python packages are diverse, with libraries that integrate with APIs, GUI programming frameworks, mathematical and biological computation, web frameworks and more.

If your designs are more mercenary, you'll be glad to know that there is healthy demand for Python programmers, and it has been placed at #8 or above in the TIOBE Programming Community Index from many years. It is used by major organisations like Google — where its creator worked till 2012 — and scientific research organisations such as CERN.

A popular video hosting website called YouTube is also built on Python, and it is used behind the scenes on other Google products as well. DropBox also uses Python, and in fact currently employs Guido. Another high-traffic website, Reddit is built on Python and is in fact open source. The popular Discus commenting platform is also built on Python, using the Django framework.

Many popular applications — 3ds MAX, Maya, GIMP among others — support Python for automating the application or for creating plugins.

## Hello World

- `print("Hello, World")`

It's hard to make things any simpler than this, and Python is often about making things as simple as they can be.

## Sort

- `def insertion_sort(ulist):`
- `for idx in range(1, len(ulist)):`
- `curr_value = ulist[idx]`
- `pos = idx`
- `while pos > 0 and ulist[pos - 1] > curr_value:`
- `ulist[pos] = ulist[pos - 1]`
- `pos -= 1`
- `ulist[pos] = curr_value`

Python code tends to be self-documenting with the tendency of many Python programmers to use longer variable and function names in Snake Case rather than CamelCase.

Of course, the above code need not be written in Python since it comes with an inbuilt function called `sort`. Python's `sort` happens to implement an excellent sorting algorithm that was invented for Python itself. It is called Timsort after its inventor Tim Peters and is widely used.

## Tools and Learning Resources

You would be forgiven for believing that a major part of the internet was actually composed of Python tutorials. Even so, Mark Pilgrim's "Dive into Python" and "Dive into Python 3" books are great introductory texts.

Another great book is "Think Python: How to Think Like a Computer Scientist". It too is available for free online at [www.greenteapress.com](http://www.greenteapress.com)

The folks at [www.interactivepython.org](http://www.interactivepython.org) have also converted the above book, among others, into an interactive online version that lets you run code samples and run your own code right in the browser.

Browser-based tools for Python are now abundant, thanks to Skulpt, a JavaScript 'port' of the Python interpreter. You can run basic Python code online at [www.codeskulptor.org](http://www.codeskulptor.org) and [www.repl.it](http://www.repl.it)

There are also some specialised Python-based online tools available. [www.morph.io](http://www.morph.io) lets you write Python scrapers for extracting data from other websites; PythonAnywhere focuses on hosting Python code and Wakari lets you do scientific computing using Python.

Speaking of scientific computing, packages like NumPy allow one to perform extensive mathematical operations, SymPy allows for symbolic mathematics, Pandas is great for data manipulations, and matplotlib is brilliant for plotting data.

One of the greatest tools though is probably IPython, an enhanced Python shell that integrates well with the above to let you show plots embedded in your Python console. IPython notebook lets you create interactive documents that contain a mixture of rich text, plots and calculations.

There are even specialised versions of Python that come with these packages pre-installed, such as Enthought's Canopy, Continuum Analytics' Anaconda, and Python(x,y).

Speaking of specialised versions, it's important to note that the Python installers you get on python.org are just one implementation of the Python language. The three implementations mentioned above bundle tonnes of scientific tools with Python, then there is eGenix Python that's a single exe!

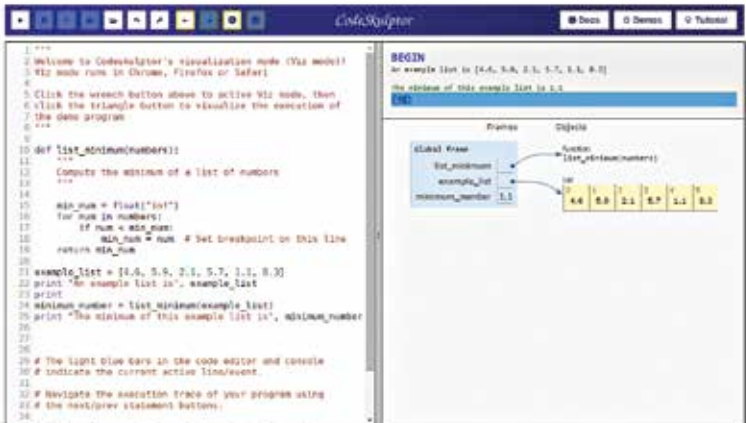
There are also complete re-implementations of Python. For instance JPython is a Python interpreter that runs on the Java VM, and as such it can access Java functionality. Similarly, you have IronPython that runs on .NET

Most interesting is PyPy, a Python implementation that was originally written in Python itself. Now it's a JIT compiler — the kind you find for JavaScript in modern browsers — for Python. In many cases it performs significantly faster than the standard Python implementation.



You should definitely prefer the interactive version of the book, unless you have an excessively slow computer or internet connection.

We would never forgive ourselves if we didn't mention virtualenv, a very popular package for Python that makes developing apps much simpler. It



The Viz Mode in online Python IDE CodeSkulptor lets you visualise your Python code as it runs.

allows you to create virtual Python environments which lets you develop and test your code in a clean environment that isn't tainted by all the Python packages you have installed globally. 

# R

## Everything you need to know about the R programming language.

**R** is the most popular language used for data analysis, modelling and visualisation. It is great for linear and non-linear models, parametric and non-parametric tests, clustering and time-series models. It also provides advanced graphics functionality for presentations and analysis.


### How was R created?

R was created in 1991 by New Zealand based Ross Ihaka (who is currently working on a new programming language that is quicker and holds more data) and Robert Gentleman. R is based on S, which was developed by John Chambers and others at Bell Labs in 1976. S was originally developed as a series of Fortran libraries to help with internal statistical requirements but was rewritten in C in 1988. This essentially makes R a 40-year-old technology. It was released to the public in 1993 and soon became the de facto language for visualisation and statistical analysis.



R is named after the first alphabets of the first names of its creators – Ross and Robert.

R is free and open source, like Python. It follows the terms of the Free Software foundation under the GNU license agreement. This means that you have full access to the source code to study how programs work, adapt them your own needs, and share your improvements with other people on sites like GitHub. This has led to the formation of an active R developer community, including a



```
R version 3.1.0 (2014-04-10) -- "Spring Dance"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.63 (6734) x86_64-apple-darwin10.8.0]
[Workspace restored from /Users/danieldmello/.RData]
[History restored from /Users/danieldmello/.Rapp.history]
> |
```

The default R language download is very basic.

core group of 20 developers who, beginning in 1997, keep releasing updates to R every year, and provide feedback on packages you submit to the group.

Packages are programs stored in script files that can run automatically without any additional coding. There are 4000 packages available on the R website at the moment, though some of these are for extremely specialised use and may only benefit one particular community, like people who sequence genomes. Whatever you need, chances are there is a package for it. The packages on the R website go through a quality control process by the core group, but you can always write your own package for personal use and share it online for others to use.

## Why is R so important today?

R has applications in healthcare, pharmaceuticals, biochemistry, genetics, molecular biology, computer science, AI and the social sciences. It has packages for almost anything an analyst needs.

Healthcare and pharmaceutical professionals can use R to model the efficacy of drug treatments, biologists can compare and contrast genomes and therapies, social scientists can use it to model behaviour, computer scientists can use it to build better predictive algorithms, and business researchers can use it to model marketing, finance, HR and operations data to help find patterns in data and help businesses.





## What makes R better?

In a word - customisation. There are other languages and applications that analyse data, but none offer the flexibility or range in packages that R does. Applications like MATLAB, S-PLUS, Stata, SPSS and SAS hide their command prompts behind fancy GUIs. These products are owned and developed by corporations like IBM that design and copyright the source code behind their GUIs.

This admittedly makes most data modelling easier as the learning curve is quicker. The user has only to learn which button to click on to get desired results along with how to interpret results. However, this only works as long as the user does not need any additional customisation requiring a change in the source code, which would be impossible to access as this is proprietary software. In this respect R has a huge advantage as it is open source and the source code is freely available.

Languages like Python, Java and Julia can also model and visualise data, but they have not been around as long as R has and so lack the large library of statistical and graphics packages that R offers. For example, Python still lacks the ability to run multiple regression scripts. It will take a few years for its library to catch up with that of R.

## How do you get R?

The base R programming language is available as a free download from <http://www.r-project.org>.

R can be installed on Windows, Mac, Unix, Linux systems and even on the PlayStation 3 (though this is only for serious geeks who don't mind missing out on gaming time). After downloading R you can also choose to download R Studio, which has a better interface, with multiple screens to work between. You can write scripts in a separate text box, check a list of defined variables, past commands, look at graphs and help files, and code from the command prompt all at the same time.

To download R, go to <http://www.r-project.org> > Click on 'download R' > click on a mirror link to download from the server closest to where you live. For India, this will be located at IIT Chennai (still referred to as Madras on the R site).

To download R Studio, go to <http://www.rstudio.com/products/rstudio/download/> > click on the latest installer for your system. Remember that you need to have already downloaded R first.

The R download, called the base system, contains the R programming language, a compiler, debugger, useful datasets, and a number of common

data analysis and graphics packages that you will need. You can download additional packages directly from the command prompt in R using the `install.packages` and `library` commands.

```
> install.packages("package name")
> library(package name)
```

## Learning resources

A huge benefit to using R is the built-in help function. For example, to find out more about boxplots, open R and type “`?boxplot`” into the command console. You will get details about boxplot usage and arguments to pass along with examples and references. A great way to learn R is through a short interactive learning resource called Swirl (<http://swirlstats.com>) that allows you to learn about basic R operations through the R console.


To install Swirl, open R > type `install.packages("swirl")` > `library("swirl")` > `swirl()`

A longer way to learn R applied to data Science is to work through the nine courses in the Data Science specialisation offered by John Hopkins University on Coursera.

Go to <https://www.coursera.org> > Click Specialisations > click Data Science

If you are looking for a more basic course, try the Data Analysis and statistical inference course by Duke University on Coursera, or the Foundations of Data Analysis course by the University of Texas on EdX.

Some of the best books to use to learn R are:

- ◆ Victor A. Bloomfield. Using R for Numerical Analysis in Science and Engineering. Chapman & Hall/CRC, 2014. ISBN 978-1439884485
- ◆ Sarah Stowell. Using R for Statistics. Apress, 2014. ISBN 978-1484201404.
- ◆ Steven Murray. Learn R in a Day. SJ Murray, 2013.
- ◆ Robert J Knell. Introductory R: A Beginner's Guide to Data Visualisation and Analysis using R. March 2013. ISBN 978-0-9575971-0-5.
- ◆ J.C. Nash. Nonlinear Parameter Optimization Using R Tools. Wiley, 2014. ISBN 9781118883969. 

# RUBY

Ruby is a language that takes the term object-oriented very seriously and was built from the ground up to be object-oriented.

**T**he first public release of Ruby dates as far back in 1995, with the release of Ruby 1.0 just a year later. Since then it has taken nearly 18 years for the language to reach v 2.0.

This isn't a slight against the language though! It has continued to evolve in the duration, with the release of 1.x versions. In fact Ruby 1.9, which released in 2007, changed the language in incompatible ways, such that many scripts that worked with 1.8 needed to be rewritten to support it. On the other hand the more major-sounding release, Ruby 2.0 was largely compatible with previous versions, and the same can be said of 2.1 — both released in 2013. Ruby is the creation of Yukihiro Matsumoto, who felt that there was a lack of proper object-oriented languages, and that on most languages



Yukihiro Matsumoto, the creator of Ruby hard at work on Ruby, we assume.

it felt like the feature was tacked on. Ruby was his attempt at such a language. The result of that decision is that everything is an object in Ruby. While some languages differentiate between ‘raw’ data types such as integers and floats, and more complex object-oriented datatypes it supports, that is not the case with Ruby. Even integers and floats are objects which allows for odd looking things such as calling methods on integer or float values (`3.1412.to_s()` for instance will convert the number 3.1412 to a string).

Another aspect of everything being an object in Ruby, is that even your own code is an object that can be manipulated. In other words, Ruby supports meta-programming. Even classes and functions are objects in Ruby, and can as such be created, modified, passed around and used as return values in Ruby.

Ruby even has ‘open’ classes, so a class created elsewhere, by someone else, in a different module can be edited by you in your own code to add features. This is not the same as subclassing, which is also supported, you are modifying the class itself. You can even modify core classes, such as the inbuilt Integer class to add features to integer numbers. For instance you could add an ‘`is_prime`’ method to the Integer class, and then be able to quickly check if any number is prime as `17.is_prime` or `9.is_prime`.

Given what we’ve just said about how easy Ruby makes it to modify things, it might seem odd that Ruby also supports functional programming. Ruby goes through the trouble of marking any method that has side-effects. Any method on an object that changes the internal state of that object is marked with a ‘!’ at the end. So if you have an array called ‘`arr`’, calling ‘`arr.sort`’ will return a sorted version of the array, while ‘`arr.sort!`’ will sort the array itself in place. This allows people who wish to program in a functional paradigm to do so.

Where Python takes the view that it is best to provide one way to do things (i.e. not have many alternate forms of syntax for the same thing) Ruby takes the opposite view, that having different ways of doing the same thing is good. Sometimes doing things a different way makes them clearer if not better. Which view is better? We say that’s a silly question, and its silly to rank subjective views on an objective scale.

## Significance

If we had one word to explain the significance of Ruby, that word would be ‘Rails’. Ever since the release of ‘Ruby on Rails’ in 2005, the framework has taken the world of web development by storm. Like it or hate it, it’s impossible to deny that it has had a huge impact. In fact it was initially the framework of choice of Twitter — they have since moved to other technologies — and today

it runs popular websites such as GitHub and Hulu.

Rails is a framework for creating websites that is built on Ruby, that embraces and encourages many software engineering best practises. It greatly simplifies building a RESTful website by automating a large



The diaspora\* social network introduced features like contact groups (circles) that are now available in both Google+ and Facebook.

part of the process using auto-code-generation tools and a philosophy of “convention over configuration”.

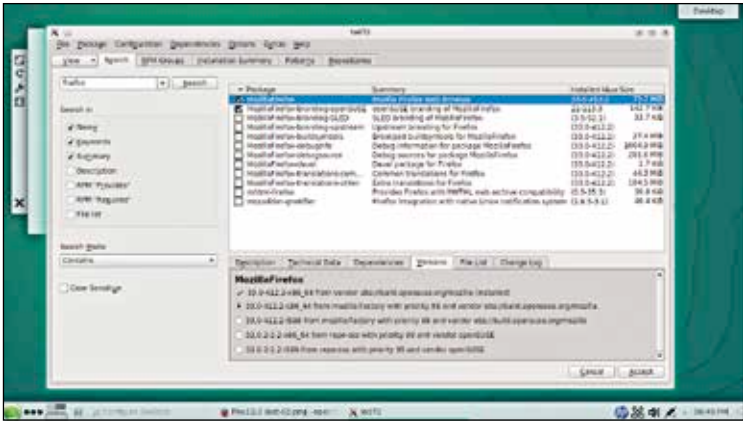
Ruby is also the language of choice of the open-source social network software diaspora\*. While the project never reached critical mass — or it hasn't yet — the choice of technology for such a project is critical. The diaspora\* is not only open-source, but also decentralised, allowing people hosting their profile on different servers to be friends.

Another major consumer of Ruby is YaST (Yet another Setup Tool), a collection of tools for Linux administrators to configure different parts of the system. It is software developed by the OpenSUSE community for their distro. YaST was originally written in a custom programming language called YCP, which had the obvious disadvantage that only developers of YaST knew it or cared about it. For their 13.1 release in 2013 that developers decided to switch to Ruby due to its popularity with developers and the chance to inject fresh blood to this important project.

Ruby, while fluctuating up and down on the TIOBE rankings, still happens to stay in the top 20 languages. It ranks highly (4th) among the languages on GitHub though, with over a hundred thousand active repositories.

Perhaps the most popular and significant thing to come with Ruby though is Rails. Its flexibility and ease of use, along with easy deployment to hosts such as Heroku, has given it a huge boost and put it in great demand.

It's a simple language to learn, yet quite powerful. It has also has the advantage of a large community, and thousands of great libraries and packages.



The Software Installer component of YaST is a powerful way to search, filter and install applications on OpenSUSE Linux.

## Hello World

- `puts "Hello, World"`

Ruby is yet another one of those languages that doesn't need large amounts of boilerplate just to push basic things like a "Hello, World" message to screen.

## Sort

- `def insertion_sort(uarr)`
- `1.upto(uarr.length - 1) do |idx|`
- `curr_value = uarr[idx]`
- `pos = idx`
- `while pos > 0 and uarr[pos - 1] > curr_value`
- `uarr[pos] = uarr[pos - 1]`
- `pos -= 1`
- `end`
- `uarr[pos] = curr_value`
- `end`
- `end`

This little piece of code might not be the best representation of the Ruby approach to problem solving.

A similar 'real' example would embrace Ruby features like open classes by adding the feature directly to Ruby's Array class. Also since this func-

tion is trying to modify the array, and as such has side effects it should end with a '!'.

That said, this code does show some of the elegance of Ruby, such as the use of constructs like `num1.upto(num2)` for iteration. Ruby has many such idioms that make programming pleasant.

## Tools and Learning Resources

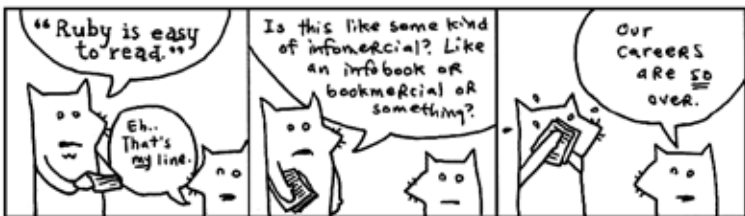
Given the vast community Ruby enjoys, there is no shortage of tutorials dedicated to learning the language. Ruby happens to have a website dedicated to offering detailed interactive tutorials that are not only brilliant, but also free. You can find them at [rubymonk.com](http://rubymonk.com)

Another great website that tests you as you go along is [rubykoans.com](http://rubykoans.com). However this one is best experienced after you have at least the basic fundamentals of some programming language down. We feel it works better when introducing you to Ruby as a second language rather than the first.

One of the most unconventional (and as such most interesting) tutorials to the language you will find is called “Why’s (Poignant) Guide to Ruby”. If you don’t want to read it for Ruby, read it for the cute illustrations of anthropomorphised foxes. Also it’s the only programming language tutorial / guide we can think of that comes with its own soundtrack. You can find this book at: [mislav.uniqpath.com/poignant-guide/](http://mislav.uniqpath.com/poignant-guide/)

So now that know where to go if you want to get started with Ruby, you probably want to get it installed on your computer. If you are using a Linux or Unix distro, you’ll be happy to know that your distro’s repositories will most definitely have it available.

For Windows and OSX users, the best place to start is [ruby-lang.org](http://ruby-lang.org) the official website of the Ruby programming language. For Windows users the best version to get is probably the one available at [railsinstaller.org](http://railsinstaller.org) since



A great deal of effort and love seems to have been put into “Why’s (Poignant) Guide to Ruby” to try and make something as dull as learning a new programming language entertaining. The fact that it’s free and under Creative Commons is ever more heartening.


it also includes Rails (which you'll probably want to check out) and also a number of additional Ruby third-party components. If you just want to dive into Rails, and want a full stack including a web server and database server check out Bitnami's Ruby stack.

A must-have tool if you want to do serious work with Ruby is RVM (Ruby Version Manager). This brilliant package lets you use multiple Ruby versions, and multiple Ruby package versions without getting into a confusing mess. You are likely to find this very useful if you have multiple projects using different versions of the same packages. Installing RVM on Windows is possible, but not directly, it will only work under Cygwin, a Linux compatibility layer for Windows.

Like most programming languages that have been around for a while, Ruby too has a number of alternative implementations. These can be more appropriate than the offi-

cial implementation based on what you need to do with Ruby. JRuby for instance runs on the JVM and gives you access to Java classes. IronRuby runs *on .NET* and lets you use it's features. Rubinius is another alternative implementation of Ruby that uses a JIT compiler for better performance.

If you're looking to develop for iOS and Android, RubyMotion is the implementation to check out, although it is not free like the ones mentioned above. Also unlike the above, it compiles your Ruby code into an app for those platforms — which it would have to for iOS to not run afoul of its rules. If you like Ruby and are looking for a cross-platform way to build a mobile app, it might be worth the money.

One last tool we'd like to mention, is Pry, which is an alternative to the default Ruby shell (irb). The default Ruby shell is — to put it kindly — functional, Pry greatly improves on the default shell, and it is just a single command away (`gem install pry`). 



Bitnami's stack is great if you are getting into Ruby for web development, since it includes web and database servers, and a simple tool to manage them.



# RUST

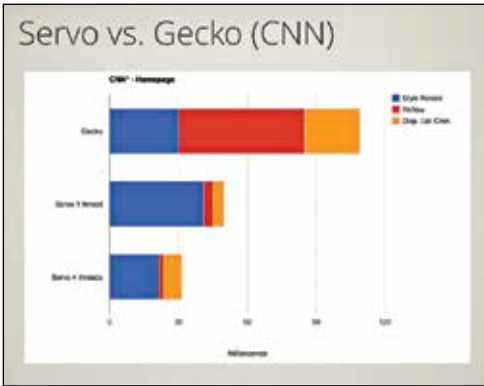
Rust is a language that is very much in development, but if things go well it will have a bright future.

Rust has an interesting history, especially since it is very much still in the making. Seeing the limitations of browser engines that started development back in the days when text was the main component of web pages, Mozilla began development of Servo, a next-generation web engine that would be highly parallelised. It would take advantage of the growing trend of multi-core CPUs on devices, and modernising the way web content is rendered.

Current generation language such as C and C++ also began to show their age and inadequacy when faced with the task of coding such an engine. Rust was the answer to a problem Mozilla created for itself; where will the web be in another decade?

Just as Rust is currently in constant development, so is its most major project, Servo. A complex project like a web rendering engine is the perfect testbed for a language. Ten years from now, Servo might just be what is behind Firefox — or Mozilla’s next browser — and Rust will be what is behind Servo.

Rust claims to have a focus on “high-level, bare-metal programming” and this is quite evident once you start exploring the language. First of all the official implementation of Rust is a compiler that produces a binary file. Secondly it includes many features found in languages like Python and Ruby that greatly improve developer productivity. Despite this it also offers



Servo is already passing the ACID2 test, and has remarkably better performance than current the Firefox Gecko engine.

many of the power features of low-level languages like C, such as pointers. In fact since it has started development at a time when multi-core computers are so popular, it also makes it simpler than ever to write multi-threaded applications.

One of the big goals of Rust is to be a safe language despite being low-level; to not let

performance compromise safety. This has lead to some interesting design choices for the language.

Rust takes many cues from functional programming languages, and has many features — such as patten matching — that will be familiar to those who like functional languages. In fact variables in Rust are immutable (meaning they can't be modified after assignment) by default.

While Rust was started by Graydon Hoare as a personal project at Mozilla at this point Rust is a community driven project. This is the perfect opportunity then, to get onto Rust on the ground floor, and not only learn a language that might be significant some day, but to shape it.

## Significance

Rust is a language that is still very much in development, so its future is hard to evaluate right now. Given that it has been around for 5 years, is still going strong, and that it has a large organisation like Mozilla behind it, it won't simply disappear off the face of the Earth suddenly one day. It is also a language strongly tied to the development of Mozilla's future browser, so it has at least one major software baked in.

The syntax and semantics of Rust are still in flux. This can certainly be a negative for those trying to learn a language. Especially to find that the perfectly good program run one day and fail in the very next release. However, for those interested in studying programming languages, and their development, this is a great opportunity.

While Go is new, and publicly developed with community contributions, Rust is in an earlier and more exciting phase where there is more scope for change, and backward compatibility isn't much of an issue. For someone who wants to be involved in the development of a programming language, this is a perfect opportunity. Even if you not interested in this aspect of programming, seeing the development of a language can give you insight into the programming language you currently use.

Even before releasing a first stable version, or reaching a stable specification, it has begun to influence other languages. For instance it was one of the language that influenced Swift, Apple's recently released programming language for iOS and OSX development.

The most interesting thing about Rust is that it is incredibly simple to do seemingly complex things. For example creating concurrent applications that run multiple threads where each works on bits of the same problem while coordinating with a main thread; this can be a pain in many languages. However solving this challenge is something that Rust is built around and as such you might find the syntax for doing such things simpler than in many other languages.

For example if you wanted to simply create an application that prints "Hello World" from a separate thread, it would be as simple as: `'spawn(move || {println!("Hello, world!");})'` Note that the above syntax may change (it already has from the stable 0.12 to 0.13 in development).

```

1 fn main() {
2     for x in range(01..101) {
3         spawn(move || {
4             println!("Hello, this is thread {}" + x);
5         })
6     }
7 }
8
9 Hello, this is thread 3
10 Hello, this is thread 2
11 Hello, this is thread 4
12 Hello, this is thread 1
13 Hello, this is thread 5
14 Hello, this is thread 6
15 Hello, this is thread 8
16 Hello, this is thread 7
17 Hello, this is thread 9
18 Finished in 0.1s
  
```

The above simple code sample runs each iteration of the loop in a separate thread. Each time you run this code, the order of the output will be different depending on the order in which the threads finish executing.

Rust also has an intricate system for controlling access to pointers such that you never end up in a situation that causes crashes due to incorrect use of pointer. It's also better at tracking pointer memory usage without major performance penalties.

Finally, Rust also makes it easy to integrate with legacy C code with efficient C bindings.

## Hello World

```
• fn main() {
•     println!("Hello, World!");
• }
```

Will this still work as a Hello World app in a year, we don't know — it most probably will — but it sure does right now! Rust keeps things simple when you're trying to do simple things while still having enough power to accomplish complex things when needed.

## Sort

```
• fn insertion_sort(uarr: &mut [int]) {
•     for idx in range(1u, uarr.len()) {
•         let curr_value = uarr[idx];
•         let mut pos = idx;
•         while pos > 0 && uarr[pos - 1] > curr_value {
•             uarr[pos] = uarr[pos - 1];
•             pos -= 1;
•         }
•         uarr[pos] = curr_value;
•     }
• }
```

On a glance this might look like just some C or C++ code, but there are some interesting differences. For one, there are no brackets around the expressions following `for` and `while`. Another noticeable difference is that you do not need to declare the type of a variable if that is obvious from the declaration.

You will also notice that we declare `'curr_value'` as `'let curr_value = uarr[idx];'` while for `'pos'` we declare `'let mut pos = idx;'`. This is because by default variables in Rust cannot be changed once set; so if we want a variable that need to be constantly modified, we need to explicitly declare it as a mutable variable. Since we want to modify the incoming array to sort it, the array needs to be mutable as well.

## Tools and Learning Resources

Given that Rust is still heavily in development, and the syntax of today might not work tomorrow, the best place to learn the language is the official website. While other tutorials and blog posts that introduce the

language might exist, they are likely to be out of date by the time you read them.

The official documentation and guides that are offered on [rust-lang.org](http://rust-lang.org) are kept up to date as the language evolves, so you can be sure that the information there is correct, and up to date. It also helps that the official documentation of the language on its website is very good and detailed.


Rust is also so early in development currently that there are no IDEs that officially support it. The closest you will get are plugins for Eclipse and JetBrains' IDEA IDEs that add support for the Rust language.

The IDEA plugin can be installed from the the IDE's plugin management system, and can be found by searching for Rust. Currently this plugin does not support anything other than syntax highlighting for Rust source code, although that is planned and coming soon. The Eclipse plugin is called RustyCage and offers a few more features.

Atom, Sublime Text and Emacs also have plugins available that add syntax highlighting and autocompletion support for Rust. For Vim lovers there is a syntax highlighting plugin available.

Currently the best support for Rust is available for Sublime Text. Its plugins allow for not only syntax highlighting and autocompletion, but also compiling and running applications from within the editor.

If you're looking to download the Rust compiler itself, you'll find that it's currently a bit harder than installing most other languages due to the language still being in development. The only Linux repositories where you will find the rust compiler are unofficial ones, that may or may not keep it updated.

The best way to get the rust compiler and libraries then is through the official website that offers 32bit and 64bit bit packages for Linux, OSX and Windows. The website also offers two versions of the language, a 'stable' version (currently at 0.12) and a nightly version that is constantly updated. The recommended version is the nightly version since the syntax of the 'stable' version is usually already outdated with the continual development of the nightly version. 

# SWIFT

An alternative language to develop iOS and OS X applications removes the C(omplexity) from Objective-C

**T**he birth of Swift comes after a realisation that many programming neophytes find it difficult to code in Objective-C because of its untangled pointers, file handling and the punctuation-rich syntax. Announced in June 2014 at the World Wide Developers Conference, Swift makes its place next to Objective-C to work with Apple's Cocoa and Cocoa Touch frameworks and the large body of existing Objective-C code written for Apple products.

Chris Lattner - the primary author of LLVM (Low Level Virtual Machine) project, began development in 2010 in collaboration with many other programmers. It took ideas from Objective-C, Rust, Haskell, Ruby, Python, C#, CLU and many others. With the newly introduced language, Apple hopes that programmers who have learnt Java, JavaScript, or Ruby will find many similarities in Swift that will help them master the language without much difficulty.

The basic structure of Swift is similar to Java. The entire code is divided into classes which comprises of fields and methods. What sets it apart from Java is the declaration of the syntax and other simplified features. For example a function declaration starts with a keyword (func) and return value comes after the method name, not before it, unlike Java.



Apple highlighting the important features of Swift at WWDC 14

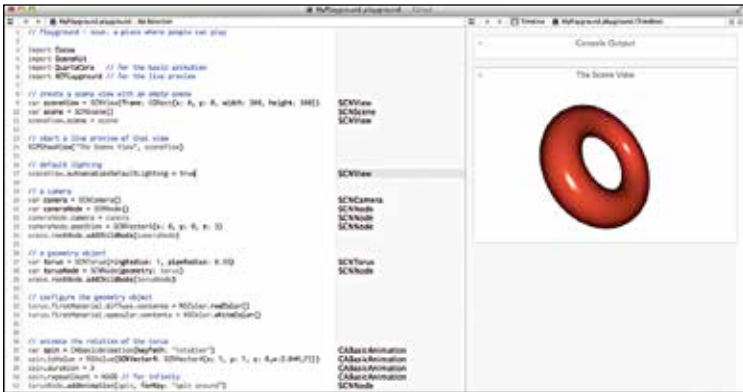
The WWDC app is the first publicly released app written in Swift. Apple introduced version 1.0 of Swift with the Golden Master of Xcode 6.0 for iOS in September 2014. Apple isn't killing Objective-C, it will stay along with Swift. Many old developers will stick to Objective-C rather than learning an entirely new language to do the same thing that they already know how to. Swift is aimed at attracting newcomers.

## Significance

Apple calls it "Objective-C without the C". It is simpler and easier to use than Objective-C. A new programming language takes years to develop a large community and that's no secret. Swift, however, has a different story here. Because of its simplicity and the ever growing ranks of iOS and OS X developers it could achieve mass adoption with an unprecedented speed. Swift is expected to surpass the uptake of Sun Microsystems' Java programming language and Microsoft's C# in the late 1990s and early 2000s.

Swift isn't as attractive as the popular programming languages out there like Ruby, Python and C#. Its only advantage is its simplicity over Objective-C which uses the complex mid-80s style, which is not easy to use as modern programming languages. Swift brings a new tool called "Playground", which lets you see the program as you write enabling you to teach yourself to code.

With Swift you won't need to waste time and keystrokes mentioning the type of each variable; the compiler will infer it from initialization. In addition to this there are some interesting and unusual features like:-



Using Playground to see the code results then and there.

**Multiple return values** - Swift allows multiple return values for a function. Looking at it in another way, it returns only single value which is a tuple that can be changed during the runtime.

**Generics** - Like Java, Swift also supports generic classes and functions.

**Class like Structures** - Swift supports structures as well as classes in the same manner as Objective-C. However, one way Swift stands apart is by providing support for functions in structures.

**Trailing Closures** - A new feature that adds closures at the end of the code which arranges it in a more natural way.

**Operator Overloading** - In addition to the predefined operators, Swift lets you overload entirely new characters from a limited character set.

## Libraries

Among the popular Swift libraries or repositories on GitHub are:

**Swiftz** - For functional programming that defines purely functional data structures and functions. Download: [digit.in/SwiftzRepo](https://github.com/digit-in/SwiftzRepo)

**ReactKit** - For reactive programming in Swift. Download: [digit.in/ReactKit](https://github.com/digit-in/ReactKit)

**Alamofire** - For elegant HTTP networking in Swift. Download: [digit.in/Alamofire](https://github.com/digit-in/Alamofire)

**Animated tab bar** - Animated Tab Bar is a Swift module for animating tab bar elements. Download: [digit.in/AnimatedTabBar](https://github.com/digit-in/AnimatedTabBar)

**Quick** - The Swift testing framework. Download: [digit.in/SwiftQuick](https://github.com/digit-in/SwiftQuick)

**FlappySwift** - Implementation of Flappy Bird in Swift. Download: [digit.in/FlappySwift](https://github.com/digit-in/FlappySwift)



**Swifter** - A Twitter framework for iOS and OS X written in Swift.

Download: [dgit.in/SwifterLibrary](http://dgit.in/SwifterLibrary)

**Chats** - An open source native iPhone messaging app. Download: [dgit.in/SwiftChats](http://dgit.in/SwiftChats)

**SwiftWeather** - A weather app for iOS developed in Swift. Download: [dgit.in/SwiftWeather](http://dgit.in/SwiftWeather)

In comparison to other programming languages there are very few libraries available for Swift. This newly released language needs more developers to match up to other popular languages.



Libraries and frameworks creating reusable code.

## Tools and Resources

Apple's new language lies within Apple's ecosystem. Development for Apple products always happens in Apple's environment, Swift is no exception. To develop in Swift you need a Mac based computer running OS X Yosemite. Like Objective-C, the development in Swift happens using Xcode IDE. Apple announced Xcode 6 this year with Swift support.

Unlike other programming languages, there is no alternative IDE available at the moment. If you have a Mac, you can go to the App Store and download Xcode 6 for free.

The newly introduced language has a plethora of learning resources available. Many websites offer free as well as paid resources that include tutorials and articles. Additionally, StackOverflow is also flooded with queries related to Swift.

## Apple

Apple has published "The Swift Language Reference" ebook on the iTunes Store for free. Alternatively, you can get your hands on the same ebook in HTML format on Apple's website.

[https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/](https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift_Programming_Language/)

## SoSoSwift

SoSoSwift is a great website dedicated for Swift language. It's an index consisting of Swift tutorials sourced from all over the Internet.

[sososwift.com](http://sososwift.com)

## Bloc

Bloc offers free tutorials which will guide you in building your very first 2D game in Swift using Spire kit.

[bloc.io](http://bloc.io)

## LearnSwift

LearnSwift consists of a curated list of helpful resources to learn Swift, which includes tutorials, code samples, code libraries and references.

[learnsswift.tips](http://learnsswift.tips)



Using Xcode 6 to develop an iOS app with Swift language.

## Ray Wenderlich

Ray Wenderlich is a popular website that offers iOS development tutorials. With the launch of the new language, the website has updated its database with tutorials that cover the basics of the language as well as

simple apps written in Swift using several iOS frameworks.

<http://www.raywenderlich.com/tutorials>

## Objective-C classes in Swift

If you are interested in using the old Objective-C classes in Swift this is a neat step by step guide for creating a bridging header.

<http://ios-blog.co.uk/tutorials/ios8-how-to-use-objective-c-classes-in-swift/>

## Higher Order Functions in Swift

Learn everything about closures in Swift and how they can be used to build filters, maps, and reduce sequence operations.

<http://www.weheartswift.com/higher-order-functions-map-filter-reduce-and-more/> 



All this and more in the  
world of Technology

VISIT  
NOW

▶ **digit.in**



# Giveaway!



FIRST **10,000**  
PARTICIPANTS WILL  
GET 12 MONTHS  
**ProDot Antivirus**  
Worth **₹465/-**  
**FREE**

Team Digit will send key to get 12 months  
ProDot Antivirus subscription to first 10,000 participants.

**Last Date to Participate:**  
**31st Jan 2015**

Actual product images and prices may vary.  
\* Regular SMS charges apply



**How to Claim:**

**SMS DIGIT <space> <CODE> <space> <your-emailid>**  
**to 92200 92200**

i.e. DIGIT ABCD1234 amit.kumar@email.com to 92200 92200

YOUR  
CODE ►

In case of any problem, feel free to contact us at [help@digit.in](mailto:help@digit.in)  
You can find antivirus installation file from Alpha DVD.

The keys will be given away to first 10,000 entries submitted by 31 Jan 2015