

# Emerging Technologies

---

ian.mcloughlin@gmit.ie

Languages

Go – Getting Started

Build tools

Web Application Architectures

# Languages

---

# Popular programming languages

**JavaScript** “high-level, dynamic, untyped, and interpreted”

**SQL** “special-purpose programming language”

**Java** “general-purpose, concurrent, class-based, object-oriented”

**C#** “multi-paradigm programming language”

**PHP** “ server-side scripting”

**Python** “high-level, general-purpose, interpreted, dynamic”

**C++** “general-purpose, imperative, object-oriented and generic”

**C** “general-purpose, imperative”

**Others** Node.js, AngularJS, Ruby, Objective-C (in order).

<http://stackoverflow.com/research/developer-survey-2016>

# Kinds of programming languages

**Interpreted** Software interprets the language at runtime.

**Compiled** Software translates the language into machine code, which is then run.

**Systems** Designed with operating system, device drivers development in mind.

**Applications** Designed with user applications development in mind.

**High-level** Abstraction from the nitty-gritty computer details.

**Imperative** Statements change the program state.

# New languages

**Go** 2009 at Google.

**Rust** 2010 at Mozilla.

**Swift** 2014 at Apple.

**Hack** 2014 at Facebook, variant of PHP.

**Scala** 2004 at EPFL (Martin Odersky).

**Julia** 2012 at MIT.

**Dart** 2011 at Google.



*I'm always delighted by the light touch and stillness of early programming languages. Not much text; a lot gets done. Old programs read like quiet conversations between a well-spoken research worker and a well studied mechanical colleague, not as a debate with a compiler. Who'd have guessed sophistication bought such noise?*

— Dick Gabriel

## People: Dennis Ritchie



Dennis Ritchie 1941-2011 (right)

- Helped Ken Thompson (left in above photo) to create UNIX.
- Created C, wrote book with Brian Kernighan.



## People: Ken Thompson



Ken Thompson (left)

- Created UNIX.
- One of the creators of Go.

## People: Brian Kernighan



- Wrote *The C Programming Language* with Dennis Ritchie.
- Coined Hello, world!.
- Wrote *The Go Programming Language* (with Alan Donovan).

## People: Bjarne Stroustrup



- Created C++.
- Former head of Large-scale Programming Research at Bell Labs.

## Places: Bell Labs



- Pretty much set up by Alexander Graham Bell.
- Eight Nobel prizes, two Turing awards.
- Owned by Alcatel-Lucent, who were bought by Nokia.

## **Go – Getting Started**

---

**Concurrency** is builtin with light-weight goroutines, channels.

**Fast compiling** is a goal.

**Packages** are easily managed and dependencies are quickly resolved.

**Type inference** is available (sometimes).

**C-like** in syntax.

**Tools** like go fmt and godoc are builtin.

**Garbage collection** is builtin.

# Hello, World!

```
package main

import "fmt"

func main() {
    fmt.Println("Hello, world!")
}
```

**go build** is the Go compiler.

**go build hello.go** created an executable called hello (or hello.exe on windows).

**Dependencies** are automatically built.

**go run** is an alternative that also runs the program after.

**Building** is fast in Go.



# Functions

```
package main

import "fmt"

func add(x int, y int) int {
    return x + y
}

func main() {
    fmt.Println(add(42, 13))
}
```

## for loops

```
package main

import "fmt"

func main() {
    sum := 0
    for i := 0; i < 10; i++ {
        sum += i
    }
    fmt.Println(sum)
}
```

## if and else

```
func pow(x, n, lim float64) float64 {  
    if v := math.Pow(x, n); v < lim {  
        return v  
    } else {  
        fmt.Printf("%g >= %g\n", v, lim)  
    }  
    // can't use v here, though  
    return lim  
}
```

```
func say(s string) {  
    for i := 0; i < 5; i++ {  
        time.Sleep(100 * time.Millisecond)  
        fmt.Println(s)  
    }  
}  
  
func main() {  
    go say("world")  
    say("hello")  
}
```

## Build tools

---

```
> go help [command]
```

---

**go help** prints out help for the go command

**Optionally** you can provide another command line argument to get help about a specific command.

```
> export GOPATH=/Users/john/go
```

---

**GOPATH** is a variable that must be set properly to use go packages.

**It's a list** of directories to look for packages in.

**Directories** listed must have `src`, `pkg` and `bin` subdirectories.

**Colons** are used to separate the directories listed in GOPATH (but semicolons on Windows).

```
> go fmt [-n] [-x] [packages]
```

---

**go fmt** formats go code in a standard way.

**gofmt** does the same, but reads and writes to and from stdin and stdout.

**Tabs** are used to clean up the code, with one tab equal to 8 spaces.

**Writing** code can be done in developers own style, then reformatted.

**Reading** code is a bit easier, as there's a standard.

**Diffs** are cleaner.

[golang.org/cmd/go](https://golang.org/cmd/go)



```
> go get ... [-u] [build flags] [packages]
```

---

**go get** downloads packages and installs them into the first entry in GOPATH.

**Dependencies** are taken care of.

**Source code** is put into the src subdirectory.

**Object files** are put into the pkg directory.

**Any executables** are put in bin.

# Web Application Architectures

---

# What is a web application?

**Web applications** are web pages that are interactive, thanks to JavaScript.

**JavaScript** is used to capture user events, and typically responds to them by firing off HTTP Requests in the background and changing the displayed resource based on the HTTP Response.

**Resources** are typically generated on the fly by a web server, which usually interacts with a database.

**LAMP** is an older architecture stack that follows this pattern.

**MEAN** is a more modern stack.

**Discoverable** Identifiable as applications, findable by search engines.

**Linkable** Easily shareable via URLs, without requiring complex installation.

**Progressive** Works for every user, regardless of browser choice.

**Responsive** Fits any form factor: desktop, mobile, tablet, or whatever comes next.

**Network** Works offline or on low quality networks.

# HTTP and CRUD

