



# What New Modern Programming Languages You Can Use to Develop Your Next Application?

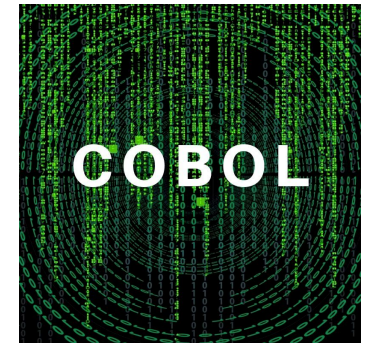
PETR PLAVJANIK | SOFTWARE ARCHITECT, BROADCOM

# Goal

- Show few popular modern programming languages
- Their benefits
- Use cases
- Personal experience

# My Story

- Petr Plavjaník – developer at Broadcom, Mainframe Software Division, located in Prague
- 1987 – BASIC
- 199x – Pascal, C, C++, Visual Basic
- 200x – PHP, SQL, Bash, Prolog, Haskell, x86 assembler
- 2004 – C#, Java
- 2005 – I have joined Computer Associates – REXX, HLASM, COBOL
- At this point, I thought that I know enough programming languages
- But ...



# What is “Modern” Programming Language?

- Quite a broad term
- You will see today:
  - Languages that gained a **significant popularity** in last decade
  - Languages that provide a **differentiation** to other programming language
- I apologize for missing your favorite programming language
- You will learn about **strengths** of these programming languages





# Evaluation Criteria

- Popularity
- Differentiating features
- Success stories
- Use cases
- Community
- Tooling
- Platform support
  - Mainframe (z/OS)
- Personal experience



# Selected Modern Programming Languages

POLL

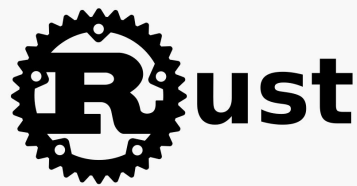
- Python
- TypeScript
- Kotlin
- Go
- Rust
- Swift



**Golang**

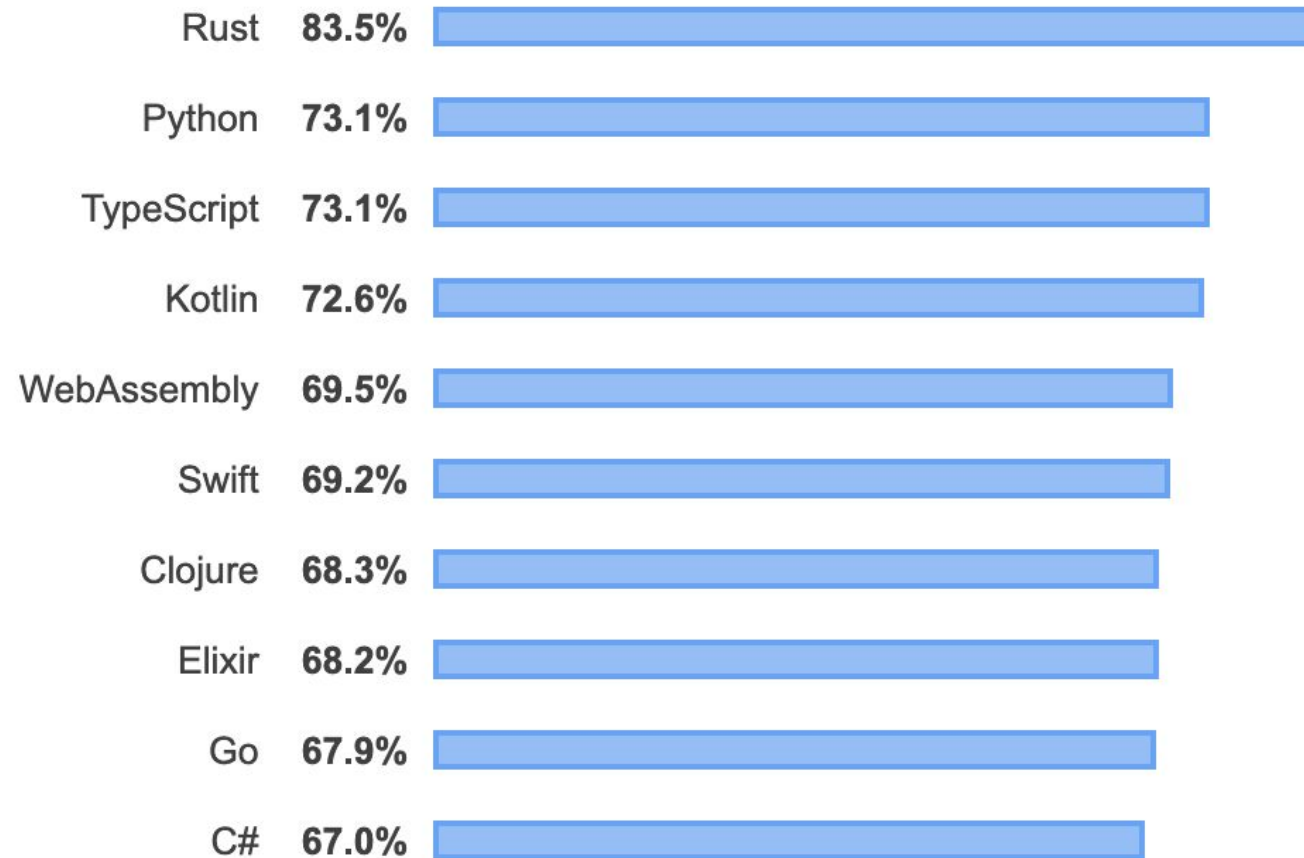


**Swift**



# Most Loved Programming Languages

<https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted>



# TIOBE Popularity Index for February 2020

<https://www.tiobe.com/tiobe-index/>

Feb 2020	Feb 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.358%	+1.48%
2	2		C	16.766%	+4.34%
3	3		Python	9.345%	+1.77%
4	4		C++	6.164%	-1.28%
5	7	⬆	C#	5.927%	+3.08%
6	5	⬇	Visual Basic .NET	5.862%	-1.23%
7	6	⬇	JavaScript	2.060%	-0.79%
8	8		PHP	2.018%	-0.25%
9	9		SQL	1.526%	-0.37%
10	20	⬆	Swift	1.460%	+0.54%
11	18	⬆	Go	1.131%	+0.17%



# Simple HTTP Server Performance Test

- Simple URI: <https://host:port/greeting?name=TechSparX>
- Returns: Hello, TechSparX!
- Developed in the most popular framework (if needed)

```
from flask import Flask, request

app = Flask(__name__)

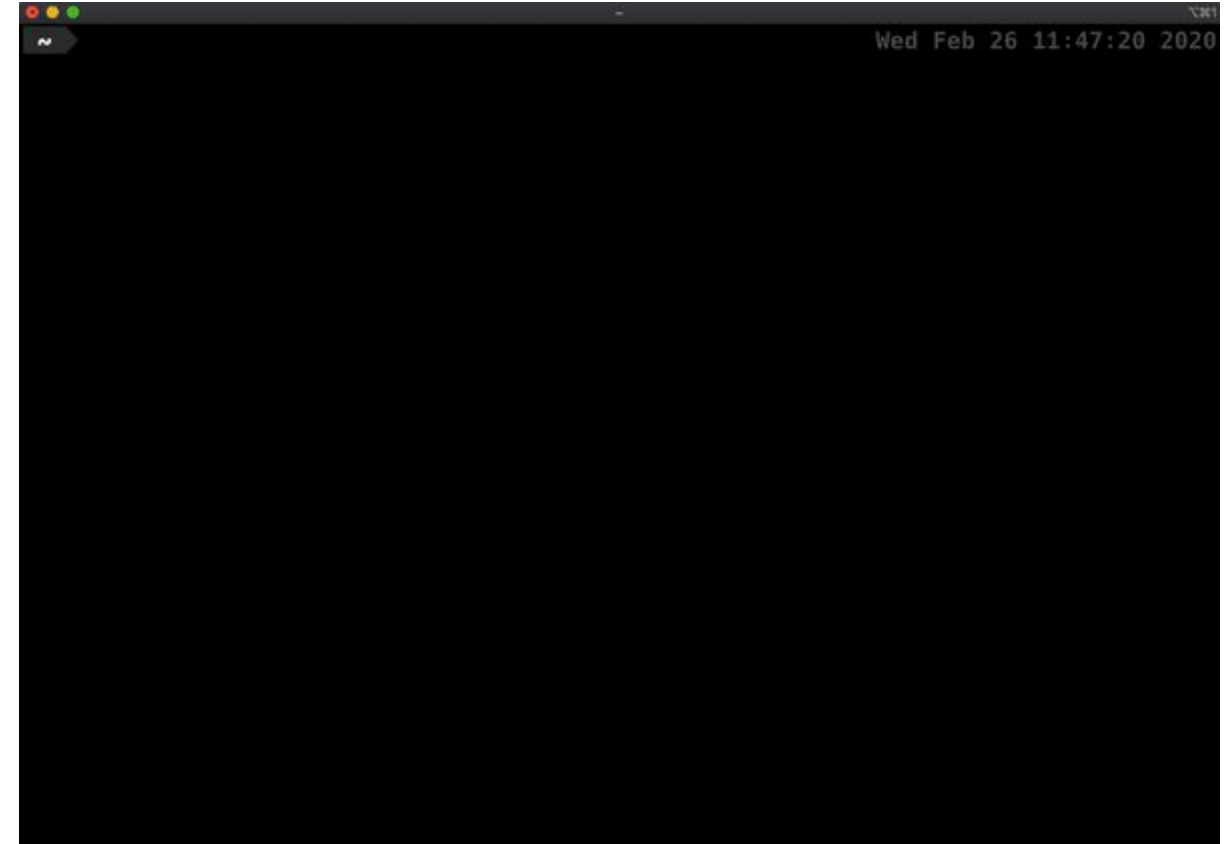
@app.route('/greeting')
def hello():
    name = request.args.get('name') or "world"
    return f"Hello, {name}!"
```

# Performance Testing using AutoCannon

`npm install autocannon --global`



```
~ http GET "http://ca32.lvn.broadcom.net:11054/greeting"
```



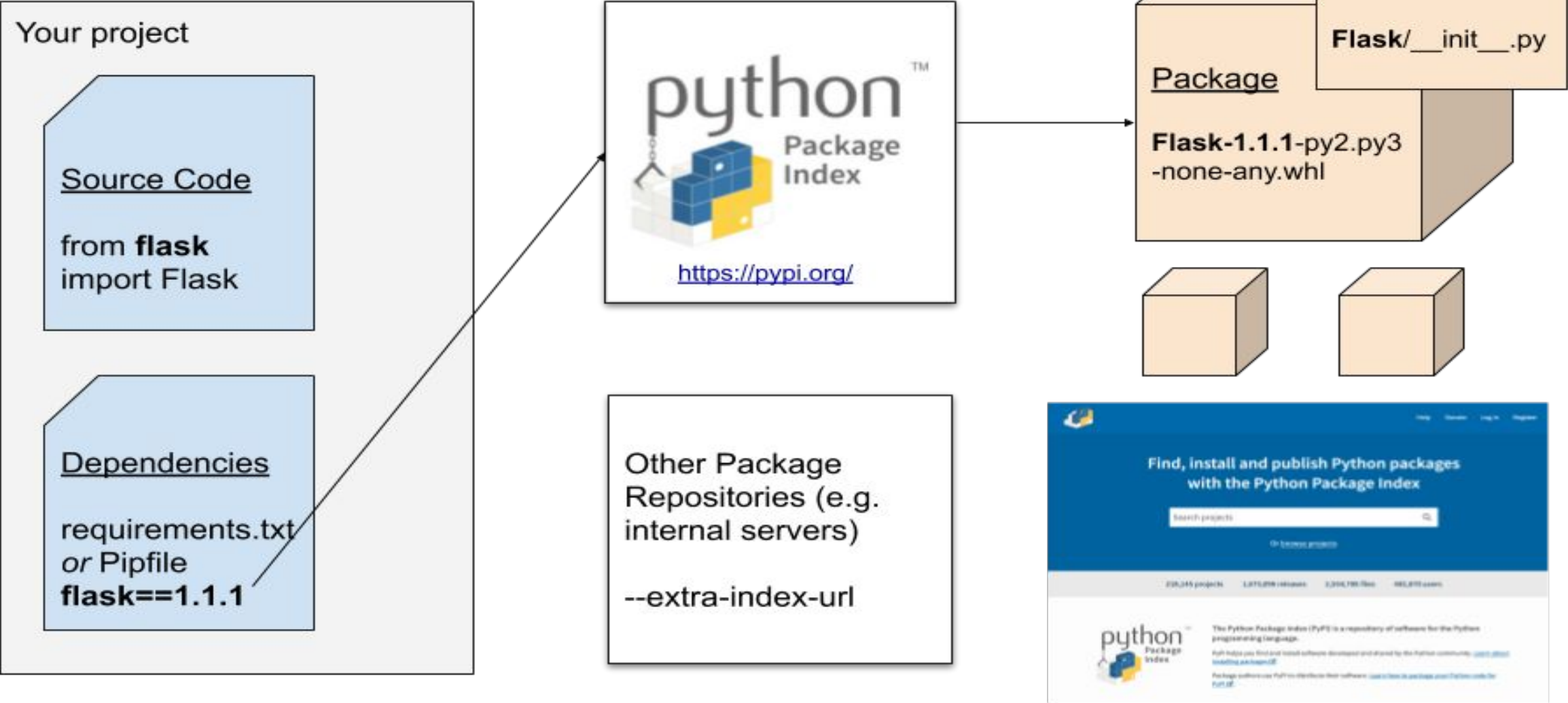
```
~ Wed Feb 26 11:47:20 2020
```



- I have resisted learning Python for some time
- Until a friend (non-developer, building architect) asked for a help with his Python code
- When my boss asked me to develop an internal tool in a short time, Python was the right tool
- **Easy-to-use** – Code is easy to understand and write even for people who are full-time programmers (Python is very popular in data science and machine learning for that reason)
- **Batteries included** – The standard library contains a lot of practical features
- **Package manager and ecosystem**
- **Fast** – Although it is interpreted language, applications in Python are fast
- **Platform support** – Almost every platform, including z/OS

# Package Manager Concept

`pip install Flask`



# Use Case: DevTest Automation using Python

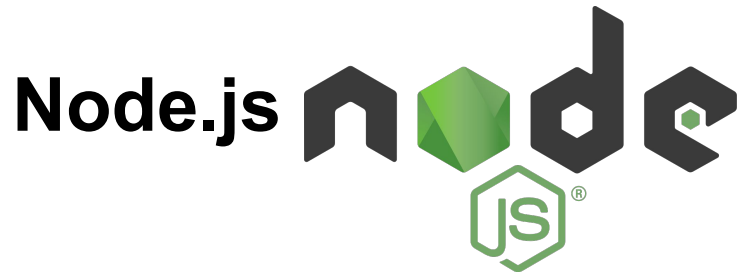
- Over years, we have developed an internal mainframe automation and testing library in Python
  - Batch, 3270, Db2
- Became popular even among non-developers (QA engineers)
- Used from workstations and Linux Jenkins machines
- **Co-location** (Linux on z connected to z/OS)
  - Tests were executed 10-100x faster since the Linux on z was on the same machine
  - Some users reported “bugs” that it is too fast before looking into results :-)

# Python Summary

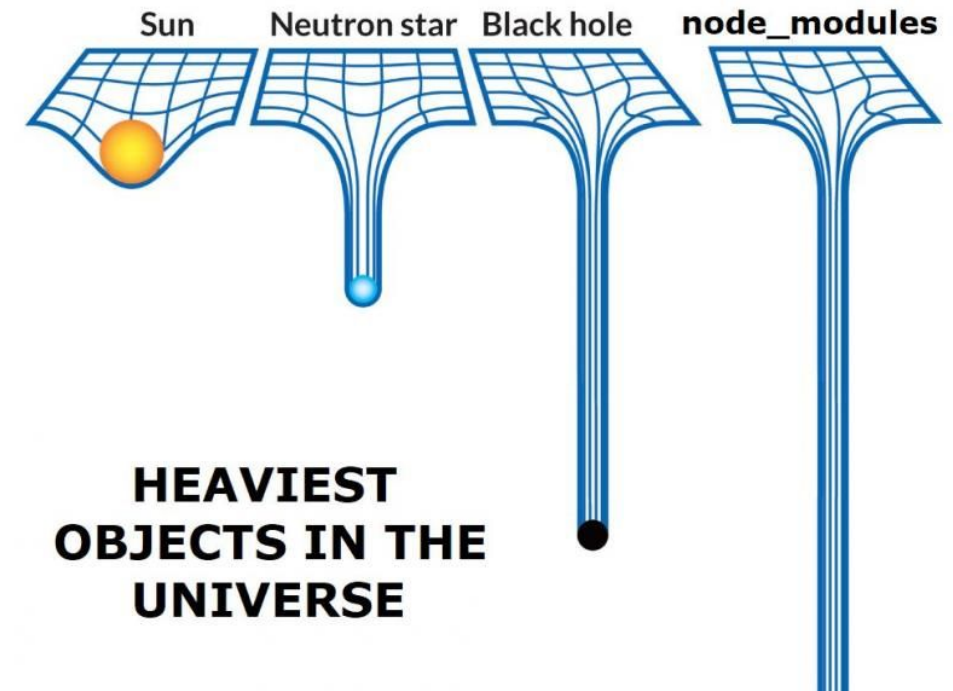
- Easy to use
- Batteries included
- Good ecosystem
- Broad platform support - including IBM mainframes
- Great for:
  - automation and any scripting
  - machine learning, data science
  - backend for web applications



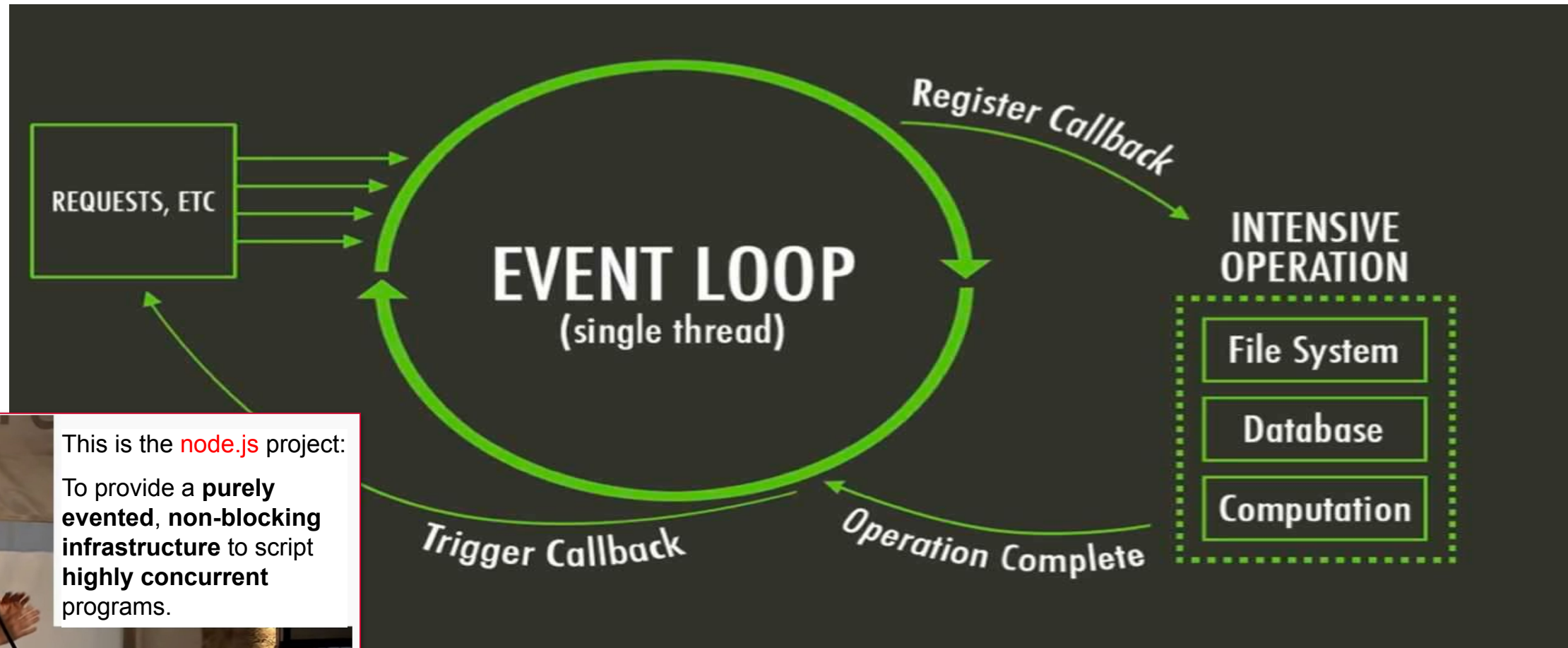




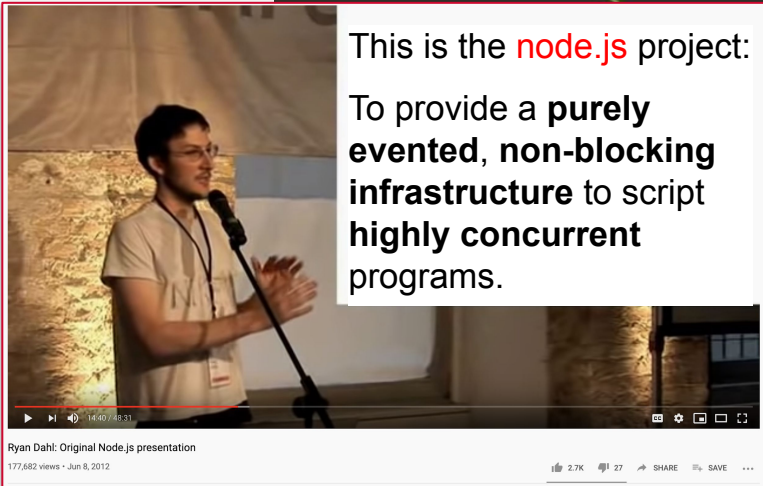
- 95% of websites are using JavaScript
- Node.js allows to use JavaScript to develop backend applications
  - Used for CLI tooling – e.g. web development tools (Angular CLI, Webpack)
  - Can be used for Desktop applications (Electron framework – VS Code)
- JavaScript runtime built on [Chrome's V8 JavaScript engine](#)
  - JIT compilation to provide good performance
- **NPM** – Node Package Manager
  - Largest repository of packages
- **Lightweight** – ideal for microservice architecture
- Netflix, LinkedIn, PayPal, Uber, eBay, Medium, Fidelity
- [OpenJS Foundation](#) – Google, IBM, Microsoft, Joyent
- Supported on many platforms, including mainframe (z/OS)



# Asynchronous Non-blocking I/O model



This is the **node.js** project:  
To provide a **purely evented, non-blocking infrastructure** to script **highly concurrent** programs.



Ryan Dahl (creator of Node.js)

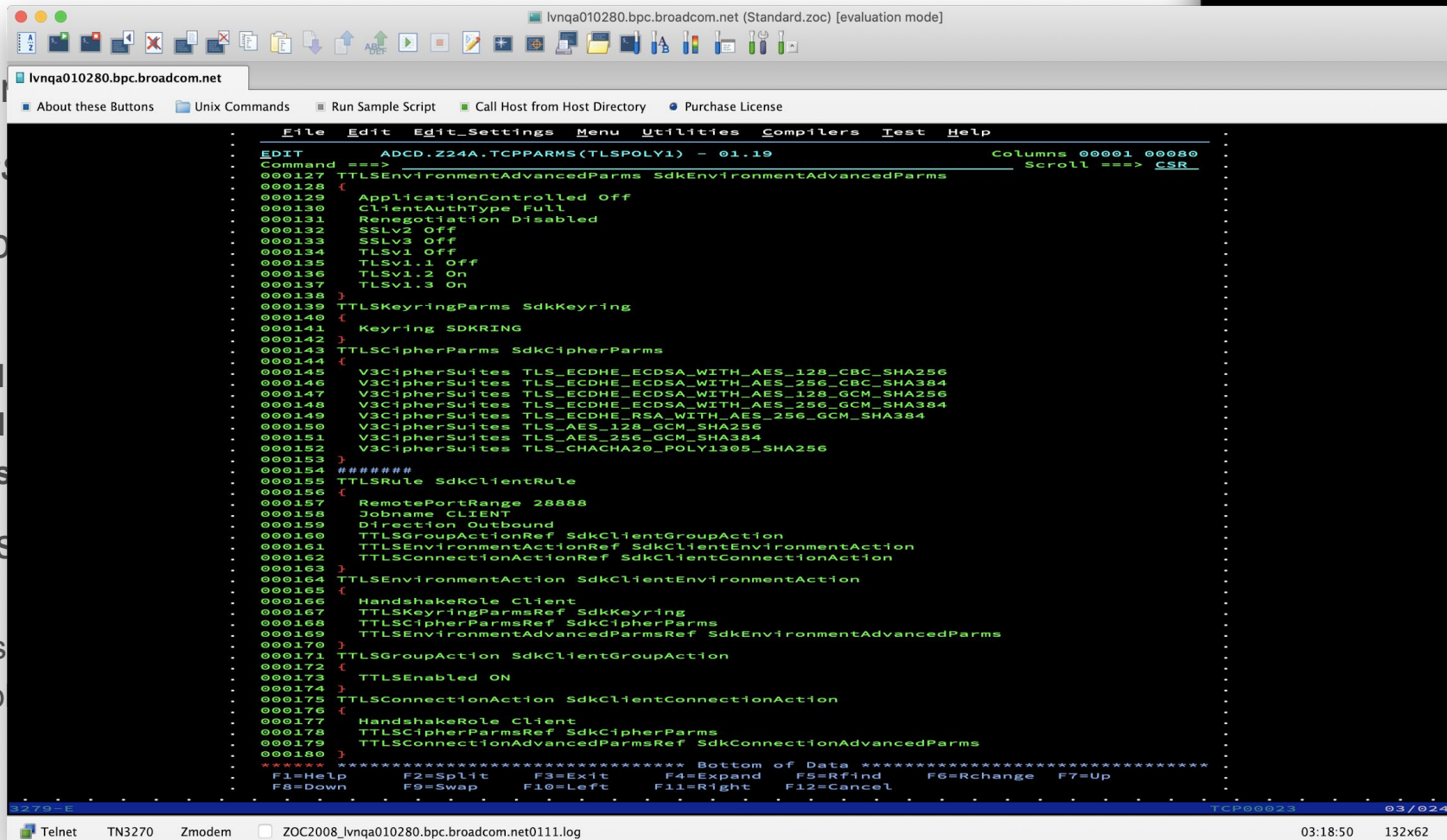
# TypeScript

- Open-source language developed by Microsoft
- First appeared in 2012
- Superset of JavaScript
- Adds optional static typing
- Popular because of static typing that catches some problems at compile time, new features (classes), better support in IDEs, and static analysis
- Some new features of JavaScript (e.g. classes) were available earlier in TypeScript
- “Transpiled” by TypeScript compiler to JavaScript that can then run on Node.js or in browsers

Zowe™, and the Zowe™ logo, and the Open Mainframe Project™ are trademarks of the Linux Foundation.

# Using Node.js in Zowe

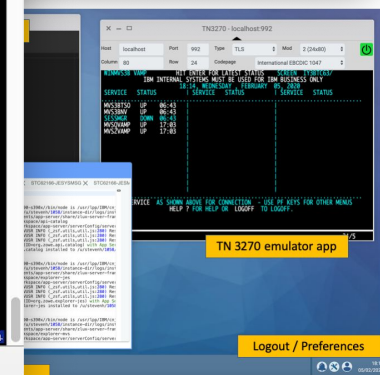
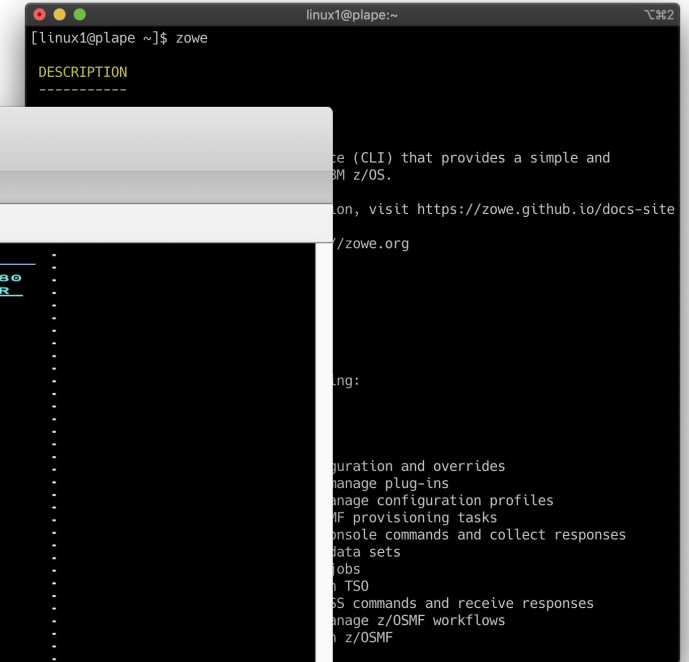
- Zowe – open
- Make access
- Initial contrib
- Provides:
  - Zowe CLI
  - Zowe API
  - Zowe Des
- Node.js is us
  - Zowe CLI
  - Zowe Des
  - Zowe Exp



```

lvnqa010280.bpc.broadcom.net (Standard.zoc) [evaluation mode]
About these Buttons  Unix Commands  Run Sample Script  Call Host from Host Directory  Purchase License

File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT ADCD.Z24A.TCPPARMS(TLSPOLY1) - 01.19 Columns 00001 00080
Command ==> Scroll ==> CSR
000127 TTLSEnvEnvironmentAdvancedParms SdkEnvEnvironmentAdvancedParms
000128 <
000129 ApplicationControlled Off
000130 ClientAuthType Full
000131 Renegotiation Disabled
000132 SSLV2 Off
000133 SSLV3 Off
000134 TLSV1 Off
000135 TLSV1.1 Off
000136 TLSV1.2 On
000137 TLSV1.3 On
000138 >
000139 <
000140 <
000141 Keyring SDKRING
000142 >
000143 <
000144 <
000145 V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
000146 V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
000147 V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
000148 V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
000149 V3CipherSuites TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
000150 V3CipherSuites TLS_AES_128_GCM_SHA256
000151 V3CipherSuites TLS_AES_256_GCM_SHA384
000152 V3CipherSuites TLS_CHACHA20_POLY1305_SHA256
000153 >
000154 <
000155 <
000156 <
000157 RemotePortRange 28888
000158 Jobname CLIENT
000159 Direction Outbound
000160 <
000161 <
000162 <
000163 <
000164 <
000165 <
000166 <
000167 <
000168 <
000169 <
000170 <
000171 <
000172 <
000173 <
000174 <
000175 <
000176 <
000177 <
000178 <
000179 <
000180 <
***** Bottom of Data *****
F1=Help F2=Split F3=Exit F4=Expand F5=RFind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
  
```



# Node.js Summary

- Lightweight
- Ideal for backend services or CLIs
- Good ecosystem
- Supported on z/OS
- TypeScript — ideal language for backend serv

```
1  import express from "express";
2
3  const app = express();
4  const port = process.env.PORT || 8080;
5
6  app.get("/greeting", (req, res) =>
7  |   res.send(`Hello, ${req.query.name || "world"}!`)
8  );
9
10 app.listen(port, () =>
11 |   process.stdout.write(`server started at http://localhost:${port}\n`)
12 );
```





- First appeared in 2014
- Modern alternative to Objective-C for Apple platforms
  - I did not want to learn Objective-C, so it motivated me to create first
- Safe by design
- Fast
  - Swift is 2.6x faster than Objective-C and 8.4x faster than Python (*source: apple.com*)
- Expressive
  - Concise syntax with popular features from other modern languages

```
1  import Kitura
2
3  let router = Router()
4
5  router.get("/greeting") { request, response, next in
6      response.send("Hello, world!")
7      next()
8  }
9
10 Kitura.addHTTPServer(onPort: 11055, with: router)
11 Kitura.run()
```



# Safety

- Pointers not accessible by default
- Optional types – no NullPointerException as in Java or S0C4
  - It is like a “box” that needs to be unwrapped

```
if let constantName = someOptional {  
    statements  
}
```
- Definitive initialization
- Array bounds checking
- Arithmetic overflow checking
- Automatic reference counting (ARC)



# Golang

- First appeared in 2009
- Created by Google engineers Robert Griesemer, Rob Pike, and Ken Thompson
- Similar syntax as C but with memory safety, garbage collection, CSP-style concurrency, strong typing
- Does **not** have many features or syntactical goodies as other languages
- **Simplicity** is one of the core principles
- **Fast** – fast to learn, fast to compile, fast to run
- Used by Google, YouTube, Apple, Dropbox, BBC, **Docker**, The Economist, The New York Times, IBM, Twitter, Facebook
- Supported on Linux for z, older version 1.6 ported to z/OS, it should be updated soon

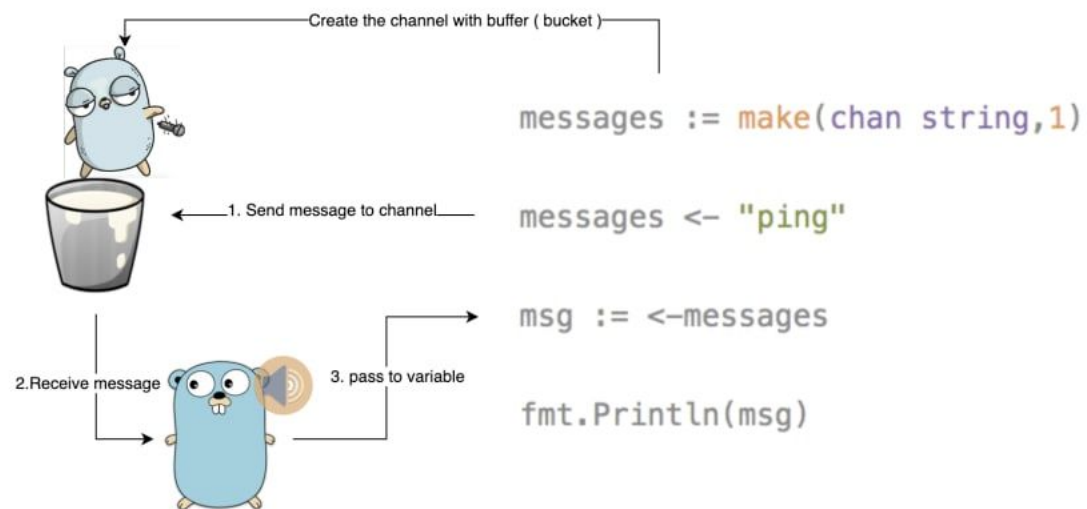
# Simplicity and Concurrency

## Missing Features

- No function or operator overloading
- No implicit conversions
- No classes and types inheritance
- No dynamic code loading
- No dynamic libraries
- No generics
- No exceptions
- No assertions
- No immutable variables
- ...

## Concurrency

- Functions can be started as *goroutines* that run concurrently
- Communicate via channels



<https://talks.golang.org/2012/concurrency>

# Built-in Tooling

- Modern programming languages are not just about better syntax and compiler
- Many tools are provided with the language:
  - Building
  - Code documentation
  - Code formatting
  - Package management
  - Testing

```
~ go
Go is a tool for managing Go source code.

Usage:

    go <command> [arguments]

The commands are:

    bug          start a bug report
    build        compile packages and dependencies
    clean        remove object files and cached files
    doc          show documentation for package or symbol
    env          print Go environment information
    fix          update packages to use new APIs
    fmt          gofmt (reformat) package sources
    generate      generate Go files by processing source
    get          add dependencies to current module and install them
    install      compile and install packages and dependencies
    list         list packages or modules
    mod          module maintenance
    run          compile and run Go program
    test        test packages
    tool        run specified go tool
    version      print Go version
    vet         report likely mistakes in packages

Use "go help <command>" for more information about a command.
```



- First appeared in 2011
- Created by JetBrains (company based in Prague, Czechia behind IntelliJ IDEA)
- It is language for JVM (Java Virtual Machine)
  - So does Scala, Clojure, Groovy, and others
- In 2019, Google has announced that Kotlin is preferred language for Android app development
- Free and open-source, paid support by JetBrains, and by Google (for Android)
- Fully interoperable with Java
- Cleaner, more expressive, and practical syntax than Java
- Used by many companies Pinterest, Coursera, Uber, Netflix, Trello, Square, and open-source projects

# Expressiveness

- Java is good
- But no one could expect how it will be used in 20 years
- Modern languages try to have defaults and simple syntax for good patterns
- Kotlin does it for the Java ecosystem

## Java

```
1  import java.util.Objects;
2
3  public class Person {
4      private String name;
5
6      public String getName() {
7          return name;
8      }
9
10     public void setName(String name) {
11         this.name = name;
12     }
13
14     @Override
15 > public String toString() { ...
16
17
18
19
20
21     @Override
22 > public boolean equals(Object o) { ...
23
24
25
26
27
28
29     @Override
30 > public int hashCode() { ...
31
32
33 }
```

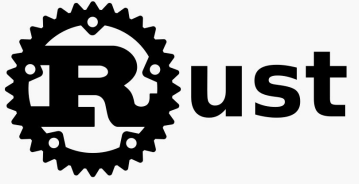
```
public void createAndPrintPerson() {
    String name = "Dan";
    Person person = new Person(name);
    printName(person.getName());
    // Prints: Dan
}
```

## Kotlin

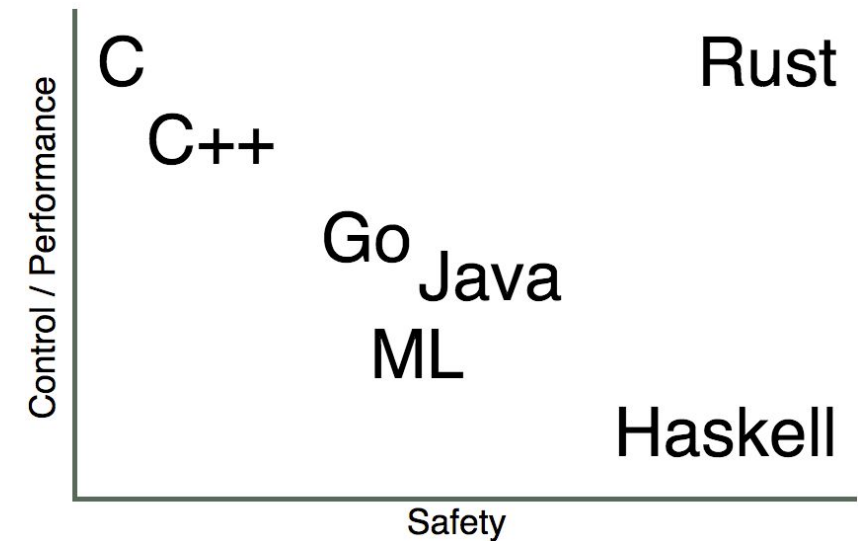
```
1  data class Person(val name: String)
```

```
fun createAndPrintPerson() {
    val name = "Petr"
    val person = Person(name)
    printName(person.name)
    // Prints: Petr
}
```





- First appeared in 2010
- Created by Mozilla
- *"Most loved programming language"* in the [Stack Overflow](#) Developer Survey every year since 2016
- Focused on safety while preserving high-performance
  - Memory safety
  - Concurrency safety
  - New concept of "ownership" in the language
- Good expressiveness
- Great tooling
  - s390x is supported
  - Used by Mozilla, Dropbox, npm Inc., and many startups



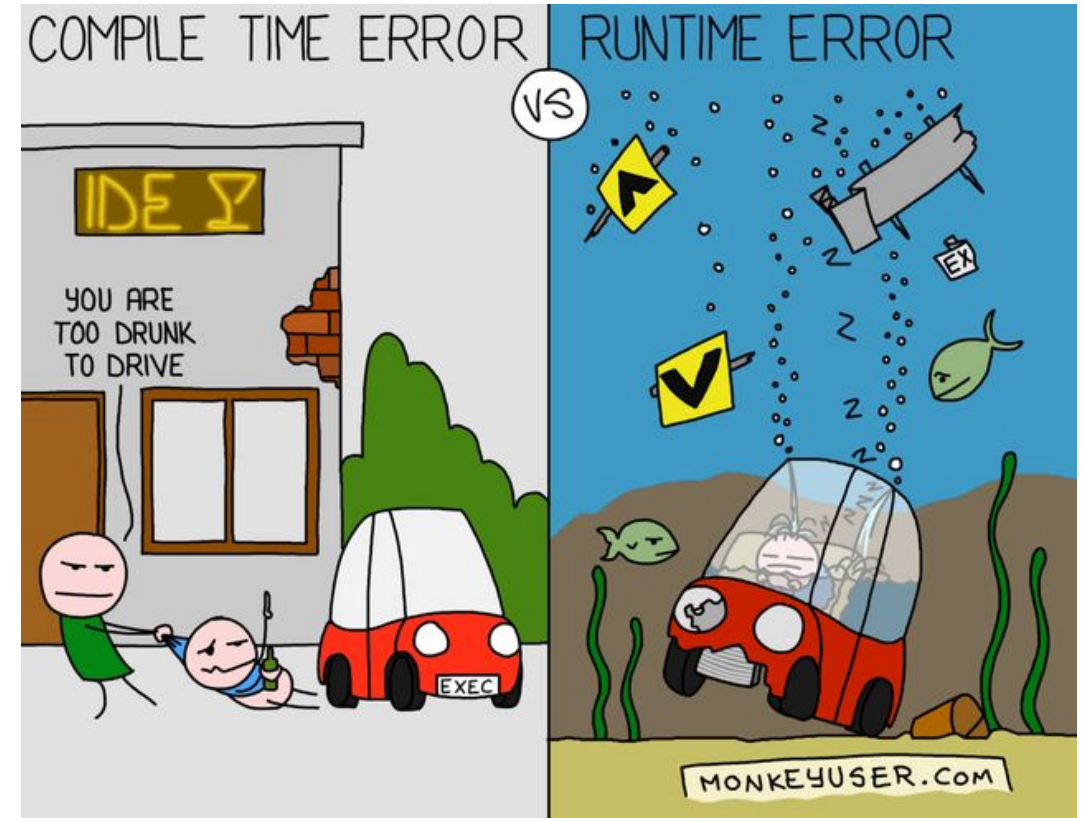
# Safety<sup>2</sup> and Zero-Overhead Features

## Rust's Ownership and Borrowing

- Compiler enforced
- Every resource has a unique owner
- Others can borrow from owner with restrictions
- Owner cannot free or mutate its resource while it is borrowed
- As result:
  - No need for runtime
  - Memory safety
  - Data-race freedom

One of the Rust design criteria are **Zero-Overhead** features:

- It must not slow down code that is not using it
- It needs to be as fast as you would implement it yourself



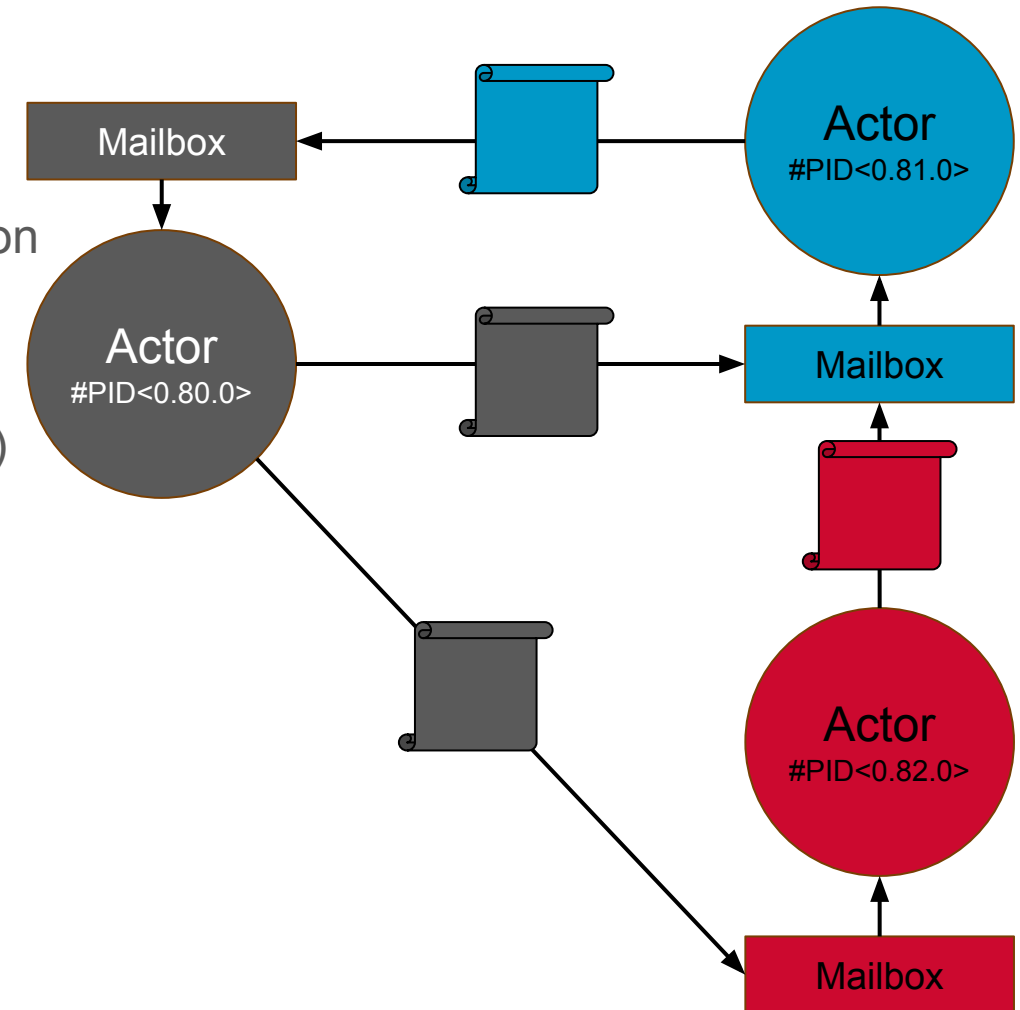
© 2020 Monkey User. All rights reserved.



- First appeared in 1984
- [Erlang](#) runtime system designed for applications that are:
  - Distributed
  - Fault-tolerant
  - Soft real-time
  - High-available
  - Hot-swappable
- Purely functional programming language
- BEAM – virtual machine (VM) for Erlang and Elixir
- [Elixir](#) – modern alternative to Erlang using the same VM with better syntax and tooling (built-in package manager and build tool)

# Actor Model

- Actors are processes (lightweight threads)
- They send messages among themselves
- Erlang VM manages creation, execution, and communication
- Their memory is isolated - no shared state
- Each process has PID (unique in the world)
- Mailbox
- Processes are supervised by other processes (supervisors)



# Functional Programming

- Pure functions
  - Always produce same output for the same input arguments (immutability)
  - Deterministic (have no side-effects)
  - Capable compiler can memorize results, parallelize, do lazy evaluation

- Values are immutable

- Advantages:

- The code are easy to understand
- Debugging and testing is easier
- Implementing concurrency is easy

```
init(Req0, State) ->  
    Req = cowboy_req:reply(200,  
        #{<<"content-type">> => <<"text/plain">>},  
        <<"Hello Erlang!">>,  
        Req0),  
    {ok, Req, State}.
```

- Disadvantages:

- Immutable values and recursion can lead to reduced performance
- I/O and programs where that use loops in procedural style is more difficult

# Erlang Success Story

- [Why WhatsApp Only Needs 50 Engineers for Its 900M Users](#)
  - Acquired by Facebook for \$19B
- I have learned Erlang while working at GoodData
  - They had some core component for analytics and it was difficult to maintain and extend it without introducing new bugs
  - They have evaluated Haskell that allowed to get complex program right
  - They used Erlang to create a working and scalable solution
- It was easier to understand and modify Erlang programs than other components written in Perl or Java



# Other Languages Summary

- **Kotlin**
  - Solid language, popular for Android and backend development
  - Easier and more powerful than Java
  - Works on JVM
- **Golang**
  - Ideal for backend development
  - Simple, concurrency support
- **Rust**
  - Very safe while efficient
- **Swift**
  - Solid language, starting to be used outside of Apple platforms
- **Erlang**
  - Proven in production for distribute, high-available, and fault-tolerant systems



# Golang

```

1 package main
2
3 import (
4     "fmt"
5     "log"
6     "net/http"
7     "os"
8 )
9
10 func HelloHandler(w http.ResponseWriter, r *http.Request) {
11     name := r.URL.Query().Get("name")
12     if name == "" {
13         name = "world"
14     }
15     fmt.Fprintf(w, "Hello, %s!", name)
16 }
17
18 func main() {
19     http.HandleFunc("/greeting", HelloHandler)
20     log.Println(os.ExpandEnv("Listening on port: ${PORT}"))
21     err := http.ListenAndServe(os.ExpandEnv(":${PORT}"), nil)
22     if err != nil {
23         log.Fatal("ListenAndServe: ", err)
24     }
25 }

```

```

-module(hello_world_app).
-behaviour(application).

-export([start/2]).
-export([stop/1]).

```


```

start(_Type, _Args) ->
    Dispatch = cowboy_router:compile([
        {'_', [{"/", hello_handler, []}]}
    ]),
    {ok, _} = cowboy:start_clear(my_http_listener,
        [{port, 8080}],
        #{env => #{dispatch => Dispatch}}
    ),
    hello_erlang_sup:start_link().

```



oadcom" rel



```

1 package com.example
2
3 import io.ktor.application.*
4 import io.ktor.response.*
5 import io.ktor.routing.*
6
7 fun main(args: Array<String>): Unit = io.ktor.server.netty.EngineMain.main(args)
8
9 @Suppress("unused") // Referenced in application.conf
10 @kotlin.jvm.JvmOverloads
11 fun Application.module(testing: Boolean = false) {
12     routing {
13         get("/greeting") {
14             val name: String =
15                 if (call.request.queryParameters["name"] != null)
16                     call.request.queryParameters["name"]!! else "world"
17             call.respondText("Hello, $name!")
18         }
19     }
20 }

```



```

1 use actix_web::{get, web, App, HttpServer, Responder};
2 use serde::Deserialize;
3 use std::env;
4
5 #[derive(Deserialize)]
6 pub struct GreetingRequest {
7     name: String,
8 }
9
10 #[get("/greeting")]
11 async fn greeting(web::Query(info): web::Query<GreetingRequest>) -> impl Responder {
12     format!("Hello, {}!", info.name)
13 }
14
15 #[actix_rt::main]
16 async fn main() -> std::io::Result<()> {
17     let bind_address = match env::var("PORT") {
18         Ok(port) => format!("0.0.0.0:{}", port),
19         Err(e) => format!("0.0.0.0:8080"),
20     };
21     HttpServer::new(|| App::new().service(greeting))
22         .bind(bind_address)?
23         .run()
24         .await
25 }

```

# HTTP Server Test Results

GitHub Repository:

<https://github.com/plavjanik/simple-http-servers>

## z/OS

Name	Avg Req/Sec
Golang	5973
Java Spring Boot	5057
Kotlin Ktor	4959
Typescript Express	4891
Python Flask	355

## LinuxONE (Linux on z)

Name	Avg Req/Sec
Golang	14532.5
Rust Actix	14262.34
Kotlin Ktor	7802.59
Java Spring Boot	6040.45
Node.js Typescript Express	2452.95
Python Flask Gunicorn	412.82

## Notes

- Test is using Flask only on z/OS, Python is usually put behind another HTTP server (e.g. Apache)
- Golang server using only standard library that is good for production use
- Better benchmark -

<https://www.techempower.com/benchmarks/#section=data-r18&hw=ph&test=db>

# Summary

POLL

Programming Language	Strengths	Use Cases
Python	Easy to use, flexible	Automation, rapid-prototyping, machine learning, web applications backend
Swift	Clean, fast, safe	Apps in Apple ecosystem, server-side applications
Golang	Simple design, fast	Server-side applications, utilities
Kotlin	Interoperable with Java	Everywhere as Java
TypeScript	Adds type checks to JavaScript, versatile	Web applications (front-end, backed), desktop applications, utilities
Rust	Safety, performance	Server-side applications, utilities
Erlang	Functional programming language, fault-tolerant	Distributed server-side applications

# Conclusion

- A lot of choices today
- Start with understanding the problem, then choose the programming language
- Learning new language is fun and can make you a better programmer
- My email: [petr.plavjanik@broadcom.com](mailto:petr.plavjanik@broadcom.com)



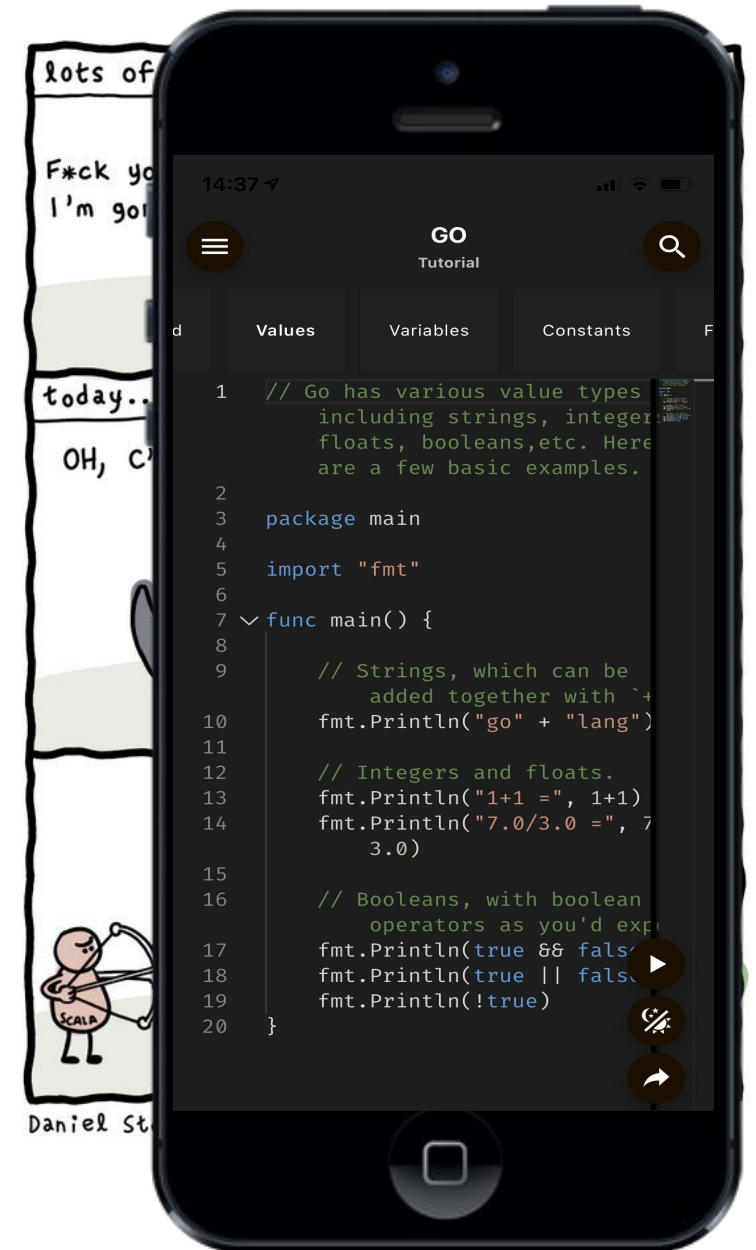
## Code Playground: Learn to Code 4+

Learn coding fundamentals

[Hemanta Sapkota](#)

★★★★★ 4.6, 1.6K Ratings

Free





# Thank You

[petr.plavjanik@broadcom.com](mailto:petr.plavjanik@broadcom.com)