A Tour of JVM Languages

Eric Honsey

Leafwing Labs / Gateless

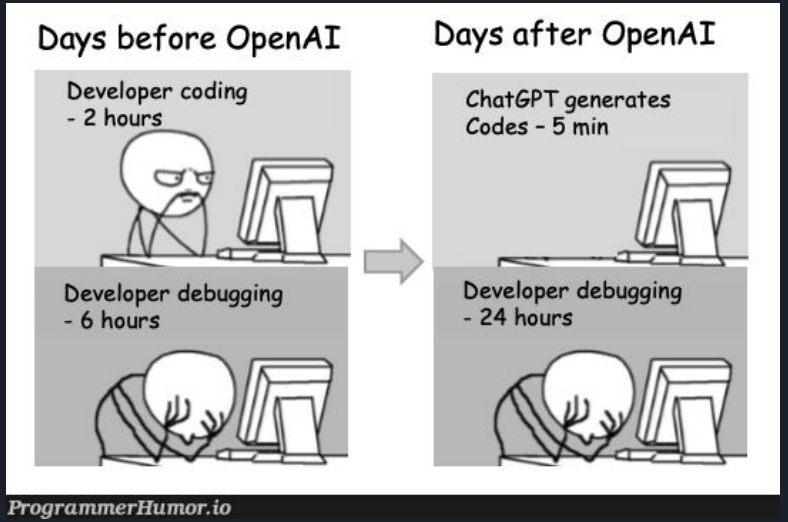
Sep 2024

Agenda

- 15 Minutes for each language
- Language Profile
  - History
  - Community
- Simple Use Cases
  - Sequence -> Min-Max Tuple
  - Database Interaction
- Professional Use Cases
  - Product Web API
  - Other Projects
- Resources

Why Programming Languages?

- LLMs => “Large Language Models”
  - AI writes code
  - Humans do more ?????
- Information Overload
  - Will I ever stop learning languages?
  - GPLs as Institutions
- Changing paradigms
  - MOAR DATA!!! => airflow :(
  - MOAR CLOUD!! => kubernetes :(



The Power of Expressiveness

“ A ***powerful*** programming language is more than just a means for instructing a computer to perform tasks. The language also serves as a ***framework*** within which we organize our ideas about processes. Thus, when we describe a language, we should pay particular attention to the means the language provides for ***combining simple ideas*** to form more complex ideas.”

- Abelson & Sussman, *Structure and Interpretation of Computer Programs §1.1*

* Emphasis added

Functional Programming

- Traits
  - Declarative Syntax
  - Immutability
  - Higher-Order Functions
  - Referential Transparency
  - Laziness
- Styles
  - Function Composition
  - Stream Processing
    - Map
    - Filter
    - Reduce
  - LISP => “LISt Processing”
  - DSLs => “Domain Specific Languages”
- Theory
  - Lambda Calculus
  - Algebras & Category Theory

Java

The Dinosaur

Java History

- 1991 => Not your Father's iPad
- 1995 => Browser Wars
  - First Dynamic Web Pages
- 1998 => JCP
- 1999 => J2EE / JEE / Java EE…
- 2007 => Sun JVM to FOSS
- 2014 => Java 1.8
- 2018 => Java 11
- 2021 => Java 17
- 2024 => Java 21

5 Principles:

- *Familiarity* => OOP
- *Secure*
- *Portable*
- *High-Performance*
- *Threaded*

Java Releases

- Progression
  - Generics & Collections
  - Functional
  - GC / NIO / Performance
  - Records / Pattern Matching
- Strategies
  - 1.*
  - LTS
  - JCP, JSRs & RIs
- Projects (Incubation)
  - Amber
  - Loom
  - Panama
  - Valhalla

Version	Release Date	Features	LTS
1.0	1996	OOP, Bytecode	
1.1	1997	Inner Classes, Reflection, Java Beans, JDBC API	
1.2	1998	Collections Framework, String Internment, JIT compiler, Swing API	
1.3	2000	HotSpot JVM, JNDI, Java Platform Debugger Architecture	
1.4	2002	Regular Expressions, Exception Chaining, NIO Interface, Logging API Changes	
1.5	2004	Generics, Enhanced for-loop, Autoboxing, Static Import, Varargs, Enumerations, Annotations	
1.6	2006	Scripting Language Support, Web Service Enhancements, JDBC 4.0 & SQL XML Support	
1.7	2011	try-with-resources, String Switch, Diamond Operator, `invokedynamic` , Exception Handling & File I/O Enhancements, G1GC	
1.8	2014	Lambda Expressions, Date and Time API Enhancements	X
9	2017	Java Platform Module System (JPMS), JShell, G1GC Official / CMS GC Deprecated	
10	Mar 2018	Local-Variable Type Inference, Immutable Collections	
11	Sep 2018	Official New HTTP Client, Open Java Flight Recorder, ZGC Linux, Java EE & Other Deprecations	X
12	Mar 2019	Switch Expressions, Minor Language Enhancements, Shenandoah Low-Pause GC	
13	Sep 2019	Switch Expression Yield, Text Blocks, Socket API Reimplementation	
14	Mar 2020	Records, Instance Pattern Matching, ZGC Multi-Platform, Native Memory API Enhancments	
15	Sep 2020	Records Official, Sealed Classes, Pattern Matching Types, ZGC & Shenandoah Official	
16	Mar 2021	Vector API Incubation Records / Sealed Classes / Pattern Matching Enhancements	
17	Sep 2021	LTS release with Minor Enhancements / Deprecations, Switch-Pattern Match Incubation	X
18	Mar 2022	Simple Web Server, Default UTF-8, Internet Address SPI, Other Enhancements	
19	Sep 2022	Record Patterns / Virtual Threads / Structured Concurrency Incubation, Other Enhancements	
20	Mar 2023	Scoped Values, Other Enhancements	
21	Sep 2023	String Templates, Sequence Collection, Key Encapsulation API	X
22	Mar 2024	Unnamed Variables, Statements before super, Stream Gather, FFI Official	
23	---	Markdown Docs,	

Java Min-Max Classic

- For => Procedural
  - i => side effect
  - if => predicate

- Predicate Abstraction
  - Mutation remains

```java
// classic for-if
public static int[] minMax1(int[] arr) {
  int[] result = new int[2];
  int min = arr[0];
  int max = arr[0];
  for (int i = 1; i < arr.length; i++) {
    if (arr[i] < min) {
      result[0] = arr[i];
    }
    if (arr[i] > max) {
      result[1] = arr[i];
    }
  }
  return result;
}

// classic for iteration
public static int[] minMax2(int[] arr) {
  int min = arr[0];
  int max = arr[0];
  for (int i = 1; i < arr.length; i++) {
    min = Math.min(min, arr[i]);
    max = Math.max(max, arr[i]);
  }
  return new int[]{min, max};
}
```

Java Min-Max Streams

- Streams
  - Terminal Value
  - Data Objects
  - API methods

```java
// APIs
public static int[] minMax5(int[] arr) {
  return new int[]{
      Arrays.stream(arr).min().getAsInt(),
      Arrays.stream(arr).max().getAsInt()};
}

public static int[] minMax6(int[] arr) {
  IntSummaryStatistics stats = Arrays.stream(arr).summaryStatistics();
  return new int[]{stats.getMin(), stats.getMax()};
}

public static int[] minMax7(int[] arr) {
  return new int[]{
      Collections.min(Arrays.stream(arr).boxed().toList()),
      Collections.max(Arrays.stream(arr).boxed().toList())
  };
}

public static int[] minMax8(int[] arr) {
  return Arrays.stream(arr).collect(
      () -> new int[]{Integer.MAX_VALUE, Integer.MIN_VALUE},
      (acc, x) -> {
        acc[0] = Math.min(acc[0], x);
        acc[1] = Math.max(acc[1], x);
      },
      (acc, x) -> {
        acc[0] = Math.min(acc[0], x[0]);
        acc[1] = Math.max(acc[1], x[1]);
      }
  );
}
```

Java Min-Max Functional

- reduce
  - Aggregation
  - Context
- Issues
  - Boxing
  - Stream-Collection Impedance Mismatch
  - Combiners

```java
// functional style
public static int[] minMax3(int[] arr) {
  return new int[]{
      Arrays.stream(arr).reduce(Integer.MAX_VALUE, Math::min),
      Arrays.stream(arr).reduce(Integer.MIN_VALUE, Math::max)
  };
}

// stream processing
public static int[] minMax4(int[] arr) {
  return Arrays.stream(arr)
               .boxed()
               .reduce(
                   List.of(Integer.MAX_VALUE, Integer.MIN_VALUE),
                   (List<Integer> acc, Integer x) -> List.of(
                       Math.min(acc.getFirst(), x),
                       Math.max(acc.get(index:1), x)
                   ),
                   (List<Integer> acc1, List<Integer> acc2) -> List.of(
                       Math.min(acc1.getFirst(), acc2.getFirst()),
                       Math.max(acc1.get(index:1), acc2.get(index:1))
                   )
               ).stream().mapToInt(i -> i).toArray();
}
```

Java 1.2 Database

- Hidden
  - Logic
  - Action
  - Value

```java
public List getProducts() {
  String url = "jdbc:mysql://localhost:3306/yourdatabase";                    // connection data
  String user = "yourusername";
  String password = "yourpassword";
  Connection connection = null;
  List productList = new ArrayList();                                         // STATE

  try {
    Class.forName(className:"com.mysql.jdbc.Driver");                         // class loading
    connection = DriverManager.getConnection(url, user, password);            // jdbc loading
    Statement statement = connection.createStatement();                       // transaction creation
    ResultSet resultSet = statement.executeQuery(sql:"SELECT * FROM product "); // LOGIC
    while (resultSet.next()) {
      productList.add(new Product(resultSet));                                // ACTION
    }
    resultSet.close();                                                        // reclaim
    statement.close();                                                        // reclaim
  } catch (ClassNotFoundException e) {                                        // exception handling
    System.err.println("JDBC Driver not found: " + e.getMessage());
  } catch (SQLException e) {
    System.err.println("SQL error: " + e.getMessage());
  } finally {
    if (connection != null) {
      try {
        connection.close();
      } catch (SQLException e) {
        System.err.println("Error closing connection: " + e.getMessage());
      }
    }
  }
  return productList;                                                         // EXIT
}
```

Java 1.5 Database

- Enterprise Edition
  - JNDI
  - JavaBeans

```java
@Stateless
public class DatabaseConnection_1_5 {

  public List<Product> getProducts() {
    List<Product> productList = new ArrayList<Product>();

    try {
      InitialContext ctx = new InitialContext();                                    // JNDI interface
      DataSource ds = (DataSource) ctx.lookup(name:"java:comp/env/jdbc/myDatasource"); // JNDI binding
      Connection connection = ds.getConnection();

      PreparedStatement ps = connection.prepareStatement(sql:"select * from product"); // LOGIC
      ResultSet rs = ps.executeQuery();
      productList.add(new Product(rs));                                             // ACTION

      rs.close();                                                                   // reclaim
      ps.close();
      connection.close();
      System.out.println(x:"Database connection closed.");
    } catch (Exception e) {
      System.out.println(x:"Failed to connect to the database");
      e.printStackTrace();
    }

    return productList;                                                           // EXIT
  }
}
```

Java 1.5 Spring DB

- Ports
  - Classes
  - Defined Interface

```java
public class DatabaseConnection_1_5_Spring {

  private DataSource dataSource;

  public void setDataSource(DataSource dataSource) {
    this.dataSource = dataSource;
  }

  public List<Product> queryProductTable() {
    JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);  // port
    List<Product> productList = new ArrayList<Product>();

    String selectQuery = "SELECT * FROM product";              // LOGIC
    jdbcTemplate.query(
      selectQuery,
      new RowCallbackHandler() {
        @Override
        public void processRow(ResultSet rs) {
          productList.add(new Product(rs));                    // ACTION
        }
      }
    );

    return productList;                                        // EXIT
  }

}
```

Java 1.8 Database

- New APIs
  - Stream
  - try-with-resources

```
public class DatabaseConnection_1_8 {

  public List<Product> getProducts() {
    try {
      InitialContext ctx = new InitialContext();                                    // JNDI interface
      DataSource ds = (DataSource) ctx.lookup(name:"java:comp/env/jdbc/myDatasource");
      try (                                                                         // resource acquisition
          Connection connection = ds.getConnection();
          PreparedStatement ps = connection.prepareStatement(sql:"select * from product"); // LOGIC
          ResultSet rs = ps.executeQuery( )                                         // ACTION
      ) {
        return Stream.of(rs)                                                        // EXIT
                     .map(Product::new)
                     .collect(Collectors.toList());
      }
    } catch (Exception e) {
      System.out.println(x:"Failed to connect to the database using J2EE and Java 1.8.");
      e.printStackTrace();
    }
    return Collections.emptyList();                                                 // EXIT
  }

}
```

Java 1.8
Spring DB

- Dirty Context
  - Objects
  - Metadata

```java
@Component
public class DatabaseConnection_1_8_Spring {

  private final JdbcTemplate jdbcTemplate;

  @Autowired
  public DatabaseConnection_1_8_Spring(JdbcTemplate jdbcTemplate) {
    this.jdbcTemplate = jdbcTemplate;
  }

  public List<Product> queryProductTable() {
    String selectQuery = "SELECT * FROM product";
    return jdbcTemplate.query(selectQuery, (rs, rowNum) -> new Product(rs));
  }

}
```

Java 21 Database

- Record
  - Data Container
  - Minimal

- Improvements??
  - Logging
  - Metrics
  - Error Handling…

```java
@Repository
public class DatabaseConnection_21 {


  public record Product(int id, String name, double price) {
    public Product(ResultSet rs) throws SQLException {
      this(rs.getInt("id"), rs.getString("name"), rs.getDouble("price"));
    }
  }

  private final JdbcTemplate jdbcTemplate;

  @Autowired
  public DatabaseConnection_21(JdbcTemplate jdbcTemplate) {
    this.jdbcTemplate = jdbcTemplate;
  }

  public List<Product> queryProductTable() {
    String selectQuery = "SELECT * FROM product";
    return jdbcTemplate.query(selectQuery, (rs, rowNum) -> new Product(rs));
  }
}
```

Java 21
Web API

- **Controller**
  - Context-Driven
  - Framework Adapter

```java
@RestController("/products")
public class ProductController {

  private ProductService productService;

  public ProductController(ProductService productService) {
    this.productService = productService;
  }

  @GetMapping
  public List<Product> getProducts() {
    return this.productService.getProducts();
  }

  @GetMapping("/{id}")
  public Product getProduct(@PathVariable Long id) {
    return this.productService.getProduct(id);
  }

  @PostMapping
  public Product createProduct(@RequestBody Product product) {
    return this.productService.createProduct(product);
  }

  @PutMapping("/{id}")
  public Product updateProduct(@PathVariable Long id, @RequestBody Product product) {
    return this.productService.updateProduct(id, product);
  }

  @DeleteMapping("/{id}")
  public void deleteProduct(@PathVariable Long id) {
    this.productService.deleteProduct(id);
  }

}
```

Java 21
Web API

- **Service**
  - Domain Logic
  - Orchestration
  - Framework Adapter

```java
@Service
public class ProductService {

  private ProductRepository productRepository;

  public ProductService(ProductRepository productRepository) {
    this.productRepository = productRepository;
  }

  public List<Product> getProducts() {
    return productRepository.getProducts();
  }

  public Product getProduct(Long id) {
    return productRepository.getProduct(id);
  }

  public Product createProduct(Product product) {
    return productRepository.createProduct(product);
  }

  public Product updateProduct(Long id, Product product) {
    return productRepository.updateProduct(id, product);
  }

  public void deleteProduct(Long id) {
    productRepository.deleteProduct(id);
  }

}
```

Java 21
Web API

- **Repository**
  - Domain Logic
  - Orchestration
  - Framework Adapter

```java
@Repository
public class ProductRepository {

  JdbcTemplate jdbcTemplate;

  public ProductRepository(JdbcTemplate jdbcTemplate) {
    this.jdbcTemplate = jdbcTemplate;
  }

  private static final RowMapper<Product> productRowMapper = (rs, rowNum) ->
      new Product(rs.getNString(0),
                  rs.getNString(1),
                  rs.getDouble(2));

  public List<Product> getProducts() {
    return jdbcTemplate.query("SELECT * FROM product", productRowMapper);
  }

  public Product getProduct(Long id) {
    return jdbcTemplate.queryForObject("SELECT * FROM product WHERE id = ?",
                                       new Object[]{id},
                                       productRowMapper);
  }

  public Product createProduct(Product product) {
    jdbcTemplate.update("INSERT INTO product (name, description, price) VALUES (?, ?, ?)",
                        product.getName(),
                        product.getDescription(),
                        product.getPrice());
    return product;
  }

  public Product updateProduct(Long id, Product product) {
    jdbcTemplate.update("UPDATE product SET name = ?, description = ?, price = ? WHERE id = ?",
                        product.getName(),
                        product.getDescription(),
                        product.getPrice(),
                        id);
    return product;
  }

  public void deleteProduct(Long id) {
    jdbcTemplate.update("DELETE FROM product WHERE id = ?", id);
  }

}
```

Java 21
Web API

- **Entity**
  - Boilerplate
  - Context-Driven
  - Framework Adapter

- Metaprogramming
  - `@AllArgsConstructor`
  - Challenge
    - ✓ Framework -> Code
    - X Code -> Framework

```java
@Entity
@Table(name = "products")
// @AllArgsConstructor  => bug with ResultSet constructor interaction
@NoArgsConstructor
@Getter
@Setter
@ToString
public class Product {

  private String name;
  private String description;
  private double price;

  public Product(ResultSet rs) {
    try {
      this.name = rs.getString("name");
      this.description = rs.getString("description");
      this.price = rs.getDouble("price");
    } catch (Exception e) {
      throw new RuntimeException(e);
    }
  }

  public Product(String name, String description, double price) {
    this.name = name;
    this.description = description;
    this.price = price;
  }

}
```

Java => NIO HTTP

- Kingdom of Nouns
  - Channel
  - Inbound
  - Handler
  - Adapter
  - Context

```java
public class TimeServerHandler extends ChannelInboundHandlerAdapter {

  /**
   * Allocates an asynchronous buffer for the current time and writes it to the channel.
   */
  @Override
  public void channelActive(final ChannelHandlerContext ctx) {
    final ByteBuf time = ctx.alloc().buffer(initialCapacity:4);
    time.writeInt((int) (System.currentTimeMillis() / 1000L + 2208988800L));

    final ChannelFuture f = ctx.writeAndFlush(time);
    f.addListener(new ChannelFutureListener() {
      @Override
      public void operationComplete(ChannelFuture future) {
        assert f == future;
        ctx.close();
      }
    });
  }

  /**
   * Logs the exception and closes the channel.
   */
  @Override
  public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) {
    cause.printStackTrace();
    ctx.close();
  }
}
```

Java => Flux

```
public class ProductStream {

  private ProductService productService;
  private CacheService cacheService;                                    // fallbacks
  private SuggestionService suggestionService;

  private ProductStream(ProductService productService, CacheService cacheService,
      SuggestionService suggestionService) {
    this.productService = productService;
    this.cacheService = cacheService;
    this.suggestionService = suggestionService;
  }

  public Mono<List<Product>> streamProducts() {                         // LAZY STREAM
    return Mono.fromCallable(() -> productService.getProducts())
               .timeout(Duration.ofMillis(millis:800))                  // time bound
               .onErrorResume(e -> Mono.just(cacheService.cachedProducts()))   // error fallback
               .switchIfEmpty(Mono.just(suggestionService.getSuggestions()))   // empty fallback
               .take(Duration.ofSeconds(seconds:5))                     // time slice
               .publishOn(Schedulers.boundedElastic());                 // orchestration
  }

  public Disposable subscribeProducts() {
    return streamProducts().subscribe(UiUtils::show, UiUtils::errorPopup);      // ACTION
  }

}
```

Java Resources

- **Web**
  - Baeldung
  - Leetcode
  - Codewars
  - dev.to
  - Stack Overflow :D
- **Books**
  - Oracle
  - Pragmatic Bookshelf
  - O'Reilly
  - MOOCs

Domain	Sample Libraries
Framework	Spring Boot, Micronaut, Quarkus
Web	Servlets, Spring, Reactor, Dropwizard
Cloud	Spring Cloud, Micronaut, Netflix OSS
Networking	Netty, okhttp, Jetty, Tomcat
Scheduling	Quartz, Spring Batch, Akka
Data	Hibernate, JPA, JDBI
Caching	Caffeine, Guava, Ehcache
Encoding	Jackson, Gson, Protobuf
Testing	JUnit, Mockito, TestNG
Build	Maven, Gradle
Logging	Log4j, Logback, SLF4J
Security	Spring Security, Bouncy Castle, Keycloak
Messaging	Kafka, RabbitMQ, ActiveMQ
Monitoring	Prometheus, Grafana, Jaeger

Java Rap Sheet

- ★ Portability
- ★ Scalability
- ★ Threading
- ★ Type Safety
- ★ Community => JCP
- ★ Maturity
- ★ Spring

- ❑ Verbosity
- ❑ Incidental Complexity
- ❑ Limited Expressiveness
- ❑ Everything is an Object
- ❑ Yet Another Framework…

Everything comes from a Class…

Groovy

Far Out...

Groovy History

- Timeline
  - Development => 2003
  - Release => 2007
- Traits
  - More Than Classes
  - Scriptable
  - Grape
  - Closures
  - Metaprogramming
  - DSLs

Version	Release Date	Features
1.0	2007-01-03	Initial Release
1.5	2007-12-07	Interactive shell, Java interop enhancements, Language enhancements
1.6	2009-02-18	Grape, AST Transformations, Multi-assignment, Mixins, `@Immutable`, `@Lazy`, `@Delegate`
1.7	2009-12-22	Annotations Everywhere, Class/Closure interop, `ASTBuilder`
1.8	2011-04-27	DSL Command Chains, AST Transformation Enhancements, JDK 7 Alignment
2.0	2012-06-28	`@TypeChecked`, `@CompileStatic`, Java 7 interop, Modularization
2.1	2013-01-24	Annotation Aliasing, Full `invokedynamic` support, `@DelegatesTo`, `@TypeChecked` extensions
2.2	2013-11-06	Closure coercion, `@Memoized`, Type-Checking pre-compilation, Script Delegates
2.3	2014-05-05	JDK 8 Support, Traits, `@TailRecursive`, `@Builder`, Tool ehancements
2.4	2015-01-21	`@SelfType`, Performance Improvements
2.5	2018-05-27	Macro Support, `ASTMatcher`, AST improvements
3.0	2020-02-07	Parrot Parser, GDK Improvements
4.0	2022-01-25	Switch Expressions, Sealed Types, Records
5.0-alpha-9	2024-06-27	Wildcard Type, String format type-checker, Extension Method improvements

Groovy Lang => Basic

- Economy
  - Type Inference
  - Interpolation
  - Collection Literals
  - Default / Named Args
  - Optional Parentheses
- Closures
  - Ad-Hoc Abstraction
  - Shared Context

```groovy
// methods
def main() {
    def name = "Groovy"                             // inference
    def answer = 42
    println "$name, ${new Random().nextInt(100)}"   // interpolation

    def lst = [1, 2, 3, 4, 5]                       // collection literals
    def map = [name: "Groovy", answer: 42]
    def dbl = lst*.toDouble()                       // spread operator
    def even = lst.findAll { it % 2 == 0 }          // streams

    println greet()                                 // error
    println greet("Eric")                           // Eric: 0
    println greet(answer = answer, name = name)     // Groovy: 42

    // closures
    def isLowerCase = {
        c ->
            try {
                Character.isLowerCase(c as char)
            } catch (ignored) {
                false
            }
    }
}

def greet(name, answer = 0) {
    "$name: $answer."                               // default & named arguments
}
```

Groovy Lang => Meta

- Economy
  - Type Inference
  - Interpolation
  - Collection Literals
  - Default / Named Args
  - Optional Parentheses
- Closures
  - Ad-Hoc Abstraction
  - Shared Context

```groovy
def meta() {
    // metaprogramming
    String.metaClass.repeatString = { n ->
        def result = ''
        n.times { result += delegate }
        result
    }
    println "abc".repeatString(3)                        // abcabcabc

    // DSLS
    please show the square_root of 100                    // 10.0
}

// DSLs
def please(action) {
    [the: { what ->
        [of: { n -> action(what(n)) }]
    }]
}
def show = { it -> println it }
def square_root = { Math.sqrt(it) }

@Macro
static Expression upper(MacroContext macroContext, ConstantExpression constX) {
    if (constX.value instanceof String) {
        return new ConstantExpression(constX.value.toUpperCase())
    }
    macroContext.sourceUnit.addError(new SyntaxException("Can't use upper with non-String", constX))
}
```

Groovy => Min-Max

- For => Procedural
  - i => side effect
  - if => predicate

- Predicate Abstraction
  - Mutation remains

```groovy
// classic
def minMax1(arr) {
    def min = arr[0]
    def max = arr[0]
    for (int i = 1; i < arr.size(); i++) {
        if (arr[i] < min) {
            min = arr[i]
        }
        if (arr[i] > max) {
            max = arr[i]
        }
    }
    return [min, max]
}

// closure
def minMax2(arr) {
    def min = arr[0]
    def max = arr[0]
    arr.each { element ->
        min = Math.min(min, element)
        max = Math.max(max, element)
    }
    return [min, max]
}
```

Groovy => Min-Max

- Streams
  - Inject
  - Collect
- DSL
  - With
  - First
  - Last
- Metaprogramming
  - @Target
  - @DelegatesTo
  - @ClosureParams

```groovy
// reduce
def minMax3(arr) {
    return arr.inject([arr[0], arr[0]]) { result, element ->
        [Math.min(result[0], element),
         Math.max(result[1], element)]
    }
}

// functions
def minMax4(arr) {
    return arr.collect { it }.sort().with { [first(), last()] }
}

// API
def minMax5(arr) {
    return [arr.min(), arr.max { it }]
}
```

```groovy
public static <T, U> T with(
        @Target("self") U self,
        @DelegatesTo(value = DelegatesTo.Target.class, target = "self", strategy = 1)
        @ClosureParams(FirstParam.class)
                Closure<T> closure) {
    if (self == NullObject.getNullObject()) {
        self = null;
    }
    Closure<V> mutableClosure = (Closure) closure.clone();
    mutableClosure.setResolveStrategy(1);
    mutableClosure.setDelegate(self);
    V rv = mutableClosure.call(self);
    return rself : rv;
}
```

Groovy => DB Script

- Scripts!!!
  - Utilities
  - Reduce Overhead
  - Fast Feedback

```groovy
#!/usr/bin/env groovy

@GrabConfig(systemClassLoader=true)
@Grapes([
        @Grab(group = 'org.apache.groovy', module = 'groovy-sql', version = '4.0.22'),
        @Grab(group = 'org.postgresql', module = 'postgresql', version = '42.2.23')
])

import groovy.sql.Sql

def url = 'jdbc:postgresql://localhost:5432/mydatabase'
def user = 'myuser'
def password = 'mypassword'

def sql = Sql.newInstance(url, user, password, "org.postgresql.Driver")

sql.eachRow('SELECT * FROM product') { row ->
    println "${row.name} - ${row.description} = ${row.price}"
}

sql.close()
```

```
❯ ./GroovySql.groovy
Product 1 - Description 1 = 10.00
Product 2 - Description 2 = 20.00
Product 3 - Description 3 = 30.00
```

Groovy => Grails / Gorm

- New API
  - ORM
  - Constraints
  - Mapping
  - Framework Methods

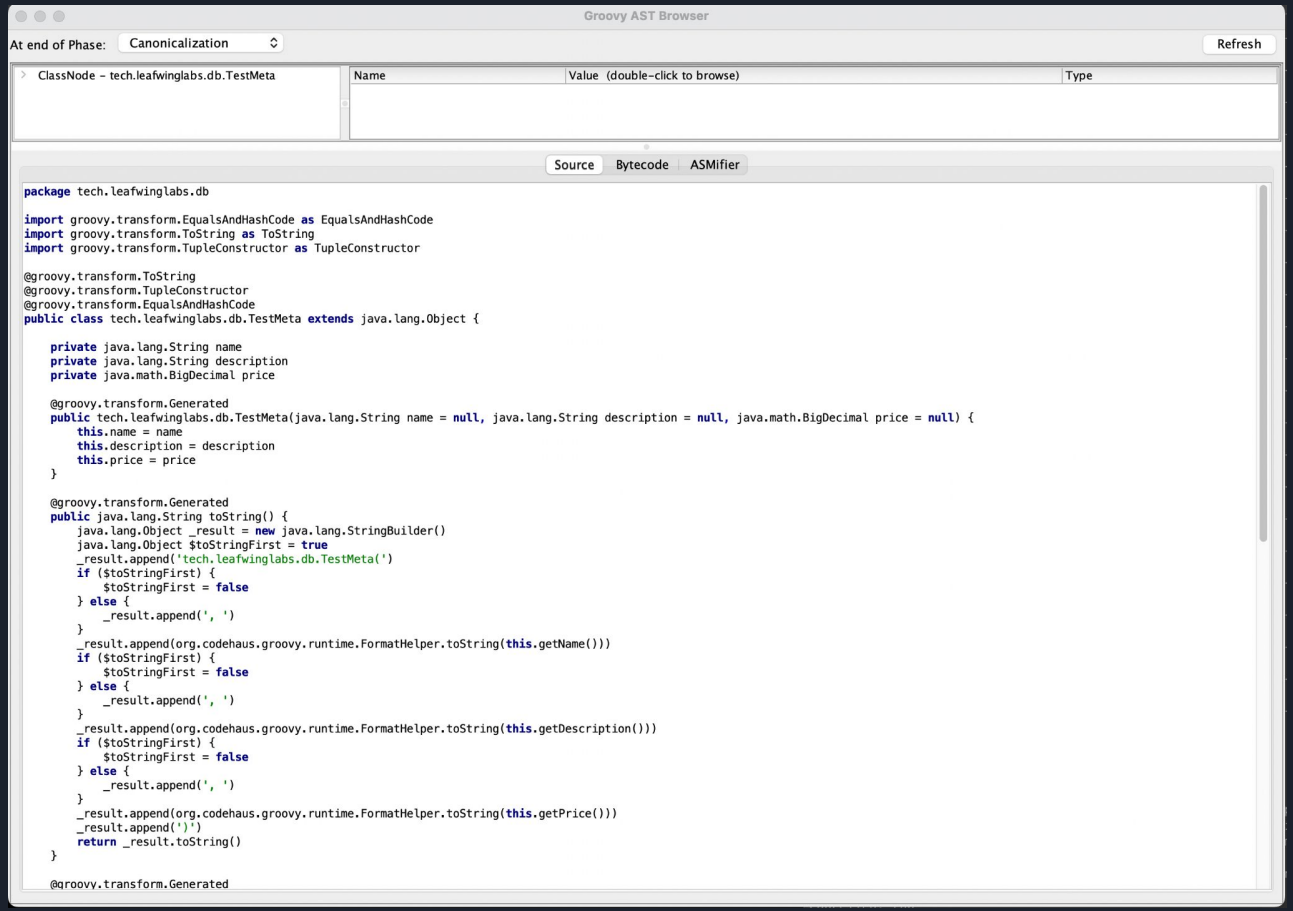
```groovy
@Entity
class Product {
    String name
    String description
    BigDecimal price

    static constraints = {
        name nullable: false, blank: false
        description nullable: false, blank: false
        price nullable: false, min: 0.0
    }

    static mapping = {
        price scale: 2
        sort name: "asc"
    }
}

// create
def p = new Product(name: "Book", description: "A good book", price: 19.99)
p.save(flush: true)

// retrieve
def clone = Product.get(1)

// update
p.price = 29.99
p.save(flush: true)

// delete
p.delete(flush: true)
```

Groovy => Grails MVC

- New API
  - ORM
  - Constraints
  - Mapping
  - Framework Methods

```groovy
@Entity
class Product {
    String name
    String description
    BigDecimal price

    static constraints = {
        name nullable: false, blank: false
        description nullable: false, blank: false
        price nullable: false, min: 0.0
    }

    static mapping = {
        price scale: 2
        sort name: "asc"
    }
}

// create
def p = new Product(name: "Book", description: "A good book", price: 19.99)
p.save(flush: true)

// retrieve
def clone = Product.get(1)

// update
p.price = 29.99
p.save(flush: true)

// delete
p.delete(flush: true)
```

Groovy => Meta


Groovy AST Browser
At end of Phase: Canonicalization
Refresh
ClassNode - tech.leafwinglabs.db.TestMeta
Name
Value (double-click to browse)
Type
Source
Bytecode
ASMifier
package tech.leafwinglabs.db
import groovy.transform.EqualsAndHashCode as EqualsAndHashCode
import groovy.transform.ToString as ToString
import groovy.transform.TupleConstructor as TupleConstructor
@groovy.transform.ToString
@groovy.transform.TupleConstructor
@groovy.transform.EqualsAndHashCode
public class tech.leafwinglabs.db.TestMeta extends java.lang.Object {
private java.lang.String name
private java.lang.String description
private java.math.BigDecimal price
@groovy.transform.Generated
public tech.leafwinglabs.db.TestMeta(java.lang.String name = null, java.lang.String description = null, java.math.BigDecimal price = null) {
this.name = name
this.description = description
this.price = price
}
@groovy.transform.Generated
public java.lang.String toString() {
java.lang.Object _result = new java.lang.StringBuilder()
java.lang.Object $toStringFirst = true
_result.append('tech.leafwinglabs.db.TestMeta(')
if ($toStringFirst) {
$toStringFirst = false
} else {
_result.append(', ')
}
_result.append(org.codehaus.groovy.runtime.FormatHelper.toString(this.getName()))
if ($toStringFirst) {
$toStringFirst = false
} else {
_result.append(', ')
}
_result.append(org.codehaus.groovy.runtime.FormatHelper.toString(this.getDescription()))
if ($toStringFirst) {
$toStringFirst = false
} else {
_result.append(', ')
}
_result.append(org.codehaus.groovy.runtime.FormatHelper.toString(this.getPrice()))
_result.append(')')
return _result.toString()
}
@groovy.transform.Generated

Groovy Gradle

- Scripting is hard…
  - … behold Groovy!
- Convention or Configuration
  - Maven => Convention
  - Ant => Configuration
  - Gradle => Middle??

```
plugins {
    id 'org.springframework.boot' version '3.1.2'
    id 'io.spring.dependency-management' version '1.1.3'
    id 'application'
    id 'groovy'
}

repositories {
    mavenCentral()
}

dependencyManagement {
    imports {
        mavenBom "io.projectreactor:reactor-bom:2023.0.8"
    }
}      You, 2 days ago • Initial Commit

dependencies {
    testImplementation libs.groovy
    testImplementation libs.spock.core
    testImplementation libs.junit
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'

    implementation libs.guava
    implementation libs.jakartaee
    implementation libs.lombok
    implementation libs.netty
    implementation libs.spring.web
    implementation libs.spring.jdbc
    implementation libs.spring.reactor
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

application {
    mainClass = 'org.example.App'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

Groovy Spock

- BDD
  - Given
  - When
  - Then

- Mixins
  - Specification
  - DomainUnitTest

```groovy
class ProductSpec extends Specification implements DomainUnitTest<Product> {

    def "product should have id, name, description, and price"() {
        given:
        def productId = 1
        def productName = "Sample Product"
        def productDescription = "Sample description"
        def productPrice = 100.00

        when:
        def product = new Product(
            id: productId,
            name: productName,
            description:
            productDescription,
            price: productPrice
            )

        then:
        product.id == productId
        product.name == productName
        product.description == productDescription
        product.price == productPrice
    }

}
```

Groovy Resources

- **Web**
  - Baeldung
  - Leetcode
  - Codewars
  - dev.to
  - Stack Overflow :D
- **Books**
  - Oracle
  - Pragmatic Bookshelf
  - O'Reilly
  - MOOCs

Domain	Sample Libraries
Web	Grails, Ratpack
Data	GORM, JDBI, Groovy SQL, Groovy Couchbase
Testing	Spock, Geb, JUnit, TestNG
Build	Gradle, Gant, Griffon
Scripting	Groovy Shell, Groovy Console, Groovy Scripts

Groovy Rap Sheet

- ★ Dynamic
- ★ Scriptable / REPL
- ★ Closures
- ★ Functional Programming
- ★ Metaprogramming
- ★ DSLs

- ❑ Metaprogramming Complexity
- ❑ Code Proliferation
- ❑ Adoption
- ❑ Competition

Groovy inspired JVM programmers to reach beyond object-oriented programming when needed.

Kotlin

The New Kid

Kotlin History

- Timeline
  - Development => 2011
  - Release => 2016
- Traits
  - Multiplatform
    - JVM
    - Native
    - JS
    - WASM
  - Type Inference
  - FP Semantics
    - Immutability
    - Streams
  - Smart cast

Version	Release Date	Features
1.0	2016-02-15	Initial Release on JVM
1.1	2016-02-15	Coroutines Experimental, JavaScript, Type Aliases, Bound Callables, std-lib enhancements
1.2	2017-11-28	Multi-platform Experimental, top-level `lateinit`, Type Inference Improvements, std-lib enhancements
1.3	2018-10-29	Coroutines Release, Experimental Contracts, Inline Classes, JVM companion fields
1.4	2020-08-17	Single Abstract Method, Mixed Named/Positional Args, Callable Reference Improvements
1.5	2021-05-25	`@JvmRecord`, `@JvmInline`, Sealed Interfaces, Stable JVM IR backend, `invokedynamic` improvements
1.6	2021-11-30	Exhaustive `when`, Suspend Function Types, Annotated Generic Types
1.7	2022-06-09	Wildcard Types, Inline Delegation K2 Alpha, Gradle & K/Native
1.8	2022-12-28	Java 19 support, Gradle 7.3 support, reflection optimizations
1.9	2023-07-06	Multiplatform Enhancements, Reduce WASM artifact size
2.0	2024-05-21	K2 Compiler, Extended Smart Casts

Kotlin Lang => Types

- “Loose” Typing
  - Mutable
  - Immutable
  - Typed
  - Inference
  - Smart Cast
- Collections as Streams
  - Map
  - Filter
  - Reduce

```kotlin
// data
var name: String = "Kotlin"
name = "Kotlin 1.5.21"          // mutable
val version = "1.5.21"          // immutable
val t2: String? = null          // null checking
t2?.length                      // null safety

// type-checking
"1" is String                   // true
'1' !is Char                    // false
"1" as CharSequence             // correct
"1" as Int?                     // exception
"1" as? Int                     // null

// smart casts
val t3: Any = "nonce"
when (t3) {
    is String -> print("Length: ${t3.length}")
    is Int -> print("Increment: ${t3.inc()}")
    else -> print("Unknown type")
}

// collections => stream processing
val lst = listOf(1, 2, 3, 4, 5)
val map = mapOf(1 to "one", 2 to "two", 3 to "three")
val seq = sequenceOf("1", "2", "3", "4", "5")
    .map(String::toInt)
    .filter { n -> n % 2 == 0 }
    .map { it * it }
    .sortedByDescending { it.toString() }
    .joinToString(separator = "|") { it.toChar().toString() }
```

Kotlin Lang => Functions

- Value-Oriented
  - Named Parameters
  - Default Parameters
  - Expressions
- No Arg Callables
  - Static => “Unbound”
  - Variable => “Bound”

```
// functions
fun greet(id: String = "World", n: Int = 1): String {
    return "Hello, $id!".repeat(n)
}
greet()
greet("Eric")
greet(n = 3)
greet(n = 5, id = "Eric")

// function expressions
fun goodbye(id: String = "World", n: Int = 1): String =
    "Goodbye, $id!".repeat(n)

// callables
val rng = 1..100
rng.reduce(Int::plus)

fun odd(n: Int) = n % 2 == 0
rng.filter(::odd).forEach(::log)

// bounded callables
val logger = LoggerFactory.getLogger("KotlinLang")
(1..Random.nextInt(100))
    .map { it.toString() }
    .forEach(logger::info)
```

Kotlin Lang => Functions II

- Function Patterns
  - Higher-Order
  - Extension
  - Scope

```kotlin
// higher-order functions
fun <T> consumer(fn: (T) -> Unit, x: T) {
    println("Consuming function: ${fn.javaClass.simpleName}")
    fn(x)
}

val greet3: (String) -> String = { id: String -> greet(id, 3) }   // partial application

// extension functions
fun String?.isNullOrEmpty(): Boolean {
    return this == null || this.isEmpty()
}

// scope functions => object context
fun finalMessage(s: String) = s.takeIf {
    it.length < 20
}?.let {
    it.uppercase() + "!!!"
}?: """
    |Too much effort to say
    |Try again later
""".trimIndent()

finalMessage(goodbye())                    // GOODBYE, WORLD!!!
finalMessage(goodbye("Cruel World"))      // Too much effort to say\nTry again later
```

Kotlin Lang => Data & Objects

- Data
  - Accessors
  - Stringify
  - Copy
- Sealed
  - Bottom Type
  - Exhaustive Pattern Match
    - `When`
- Abstract
  - Class Hierarchy
  - Works with `interface` too

```kotlin
// log abstraction
fun log(obj: Any) {
    println(obj)
}

// what happens during a request??
data class ServiceRequestStatus(
    val status: Int,
    val message: String
)

// middleware
sealed class ServiceRequest<T> {
    abstract fun `do`(): ServiceRequestStatus
    fun middleware(): ServiceRequestStatus {
        when (this) {
            is CacheRequest<*> -> log("Cache Request: ${this.data}")
            is RecRequest -> log("Recommendation Request: ${this.serviceName}")
            else -> log("Unmanaged Service Request")
        }
        return this.`do`()
    }
}

// cache adapter
data class CacheRequest<T>(val data: T) : ServiceRequest<T>() {
    override fun `do`(): ServiceRequestStatus {
        return ServiceRequestStatus(200, "Cache Request Success")
    }
}

// recommendation adapter
data class Recommendation(val id: String)
data class RecRequest(val serviceName: String) : ServiceRequest<Recommendation>() {
    override fun `do`(): ServiceRequestStatus {
        return ServiceRequestStatus(200, "Recommendation Request Success")
    }
}
```

Kotlin => Min-Max

- Type Alias
  - Semantics
  - Ubiquitous Language

- Streams
  - Reduce
  - Fold

```kotlin
typealias MinMax = Pair<Int, Int>

// classic iteration
fun minMax1(xs: List<Int>): MinMax {
    var min = Int.MAX_VALUE
    var max = Int.MIN_VALUE
    for (x in xs) {
        if (x < min) min = x
        if (x > max) max = x
    }
    return MinMax(min, max)
}

// collection api
fun minMax2(xs: List<Int>): MinMax {
    val min = xs.minOrNull() ?: Int.MAX_VALUE
    val max = xs.maxOrNull() ?: Int.MIN_VALUE
    return MinMax(min, max)
}

// reduce => accumulate starting at first value
fun minMax3(xs: List<Int>): MinMax =
    MinMax(
        xs.reduce { acc, x -> minOf(acc, x) },
        xs.reduce { acc, x -> maxOf(acc, x) }
    )

// fold => accumulate starting at seed value
fun minMax4(xs: List<Int>): MinMax =
    xs.fold(MinMax(Int.MAX_VALUE, Int.MIN_VALUE)) { (min, max), x ->
        MinMax(
            minOf(min, x),
            maxOf(max, x)
        )
    }
```

Kotlin => Reactive Web

- Minimization
  - Data
  - Metadata
  - Expressions
  - Marker Classes
- DSLs
  - Type-Safe Builders
  - Block-Oriented

```kotlin
data class Product(
    @Id val id: Int,
    val name: String,
    val description: String,
    val price: Double
)

interface ProductRepository : CoroutineCrudRepository<Product, Int>

@RestController
class ProductController(private val repo: ProductRepository) {
    @GetMapping("/products/{id}")
    suspend fun productsById(@PathVariable id: Int): Product =
        this.repo.findById(id) ?: throw IllegalArgumentException("Product not found")
}

@SpringBootApplication
class KotlinSpringFlux

fun main(args: Array<String>) {
    runApplication<KotlinSpringFlux>(*args) {
        addInitializers(beans {
            bean {
                val repo = ref<ProductRepository>()
                coRouter {
                    GET("/products") {
                        val products: Flow<Product> = repo.findAll()
                        ServerResponse.ok().bodyAndAwait(products)
                    }
                }
            }
        })
    }
}
```

Kotlin => Reactive Web

- Minimization
  - Data
  - Metadata
  - Expressions
  - Marker Classes
- DSLs
  - Type-Safe Builders
  - Block-Oriented

```kotlin
data class Product(
    @Id val id: Int,
    val name: String,
    val description: String,
    val price: Double
)

interface ProductRepository : CoroutineCrudRepository<Product, Int>

@RestController
class ProductController(private val repo: ProductRepository) {
    @GetMapping("/products/{id}")
    suspend fun productsById(@PathVariable id: Int): Product =
        this.repo.findById(id) ?: throw IllegalArgumentException("Product not found")
}

@SpringBootApplication
class KotlinSpringFlux

fun main(args: Array<String>) {
    runApplication<KotlinSpringFlux>(*args) {
        addInitializers(beans {
            bean {
                val repo = ref<ProductRepository>()
                coRouter {
                    GET("/products") {
                        val products: Flow<Product> = repo.findAll()
                        ServerResponse.ok().bodyAndAwait(products)
                    }
                }
            }
        })
    }
}
```

Kotlin => Ktor

- Web
  - Client
  - Server
- Minimal Framework
  - Routing
  - Serialization

```kotlin
@Serializable
data class Product(
    val id: Int,
    val name: String,
    val description: String,
    val price: Double
)

fun main() {
    val products = mutableListOf(
        Product(1, "Product 1", "Product 1 description", 100.0),
        Product(2, "Product 2", "Product 2 description", 200.0),
        Product(3, "Product 3", "Product 3 description", 300.0)
    )

    val server = embeddedServer(Netty, port = 8080) {
        install(ContentNegotiation) {
            json()
        }
        routing {
            get("/products") {
                call.respond(products)
            }
            get("/products/{id}") {
                val id = call.parameters["id"]?.toInt()
                val product = products.find { it.id == id }
                if (product != null) {
                    call.respond(product)
                } else {
                    call.respond("Product not found")
                }
            }
            post("/products") {
                val product = call.receive<Product>()
                products.add(product)
                call.respond("Product added")
            }
        }
    }
    server.start(wait = true)
}
```

Kotlin Resources

- Web
  - Kotlin Documentation
  - mcxiaoke/awesome-kotlin
  - Exercism
  - Kotlin Koans
  - Leetcode
  - Codewars
- Books
  - Programming Kotlin
  - Programming DSLs in Kotlin
  - Kotlin and Android Development with Jetpack

Domain	Sample Libraries
Device	Android
Framework	Ktor, Spring, Koin
Cloud	RxKotlin, Spring Cloud
Web	Spring, Javalin
Networking	ktor-http-client, Fuel, Retrofit
Scheduling	Coroutines, Arrow
Data	Exposed
Testing	Kotlin Test, MockK
Build	Gradle
Logging	Timber

Kotlin Rap Sheet

- ★ Multiplatform
- ★ Scriptable / REPL
- ★ Closures
- ★ Functional Programming
- ★ Metaprogramming
- ★ DSLs

- ❑ Large Lexicon
- ❑ Smaller Adoption Rate
- ❑ Compilation Speed
- ❑ Artifact Size
- ❑ Frequent Changes

Kotlin is the next-generation language
Inspired by programming on the JVM.

Scala

The Academic

Scala History

- Timeline
  - Development => 2001
  - Release => 2003 / 2006
- Traits
  - Advanced FP Typing
    - ADTs
    - Pattern Matching
    - Contexts
  - Distributed Computing
    - Petri Nets
    - Concurrency
  - Lazy Evaluation
- Scala Improvement Process

Version	Release Date	Features
2.0	2006-03-12	Initial Release
2.3.0	2006-11-23	Bottom Types, Syntax Enhancements
2.4.0	2007-03-09	Tuple, Extractor & Self-Type Syntax Enhancements
2.5.0	2007-05-02	For-Comprehensions, Implicit Functions
2.6.0	2007-07-27	Lazy Values, Existential Typing
2.7.0	2008-02-07	Case Class Enhancements, Java Generics Support
2.8.0	2010-07-14	Collection Enhancements, REPL, New Tools
2.9.0	2011-05-12	Parallel Collections, `App`, Scala Runner
2.10.1	2013-03-13	`AnyVal` "Struct", `implicit` Classes, Promises, Akka Actors Macros Experiment
2.11.0	2014-04-21	Performance Improvements
2.12.0	2016-11-03	JDK 8 Support, Scala.js, Type Inference Improvements
2.13.0	2019-06-11	Collections Redesign, Ammonite REPL, Futures Enhancements, Partial Unification
3.0.0	2021-05-14	New Syntax, DOT Compiler, Clauses, Contextual Abstraction, Macros
3.1.0	2021-10-18	Performance Improvements, Stable APIs
3.2.0	2022-09-05	Clause Syntax improvements, Experimental APIs
3.3.0	2023-05-30	Syntax Improvements, Experiments
3.4.0	2024-02-29	Type System & Performance Improvements
3.5.0	2024-08-22	Scala CLI Runner, Named Tuples & Context Bounds Syntax Experiments

Scala Lang => Data

- Concision
  - _ Wildcard
  - Uniform Access Principle
- => Higher Order
  - Currying
  - Partials
- Implicits
  - Extension
  - Context

```scala
// data
var name: String = "Scala"
name = "Scala 3"                // mutable
val version = "3"               // immutable
s"Hello, $name!"                // string interpolation

// collections => stream processing
val lst = List(1, 2, 3, 4, 5)
val map = Map(1 -> "one", 2 -> "two", 3 -> "three")
val seq = List("1", "2", "3", "4", "5")
  .map(_.toInt)
  .filter(_ % 2 == 0)
  .map(x => x * x)
  .sortBy(_.toString.reverse)
  .map(_.toChar.toString)
  .mkString("|")

// higher-order functions
def consumer[T](fn: T => Unit, x: T): Unit =
  println(s"Consuming function: ${fn.getClass.getSimpleName}")
  fn(x)

val greet2: String => String = greet.curried("Scala") // Currying
val greet3: String => String = greet(_, 3) // Partial application

// extension functions => implicits
implicit class StringOps(s: Option[String]):
  def isNullOrEmpty: Boolean = s.isEmpty || s.exists(_.isBlank)
```

Scala Lang => Patterns

- Option => Monad Contract
  - Map
  - FlatMap
- Destructuring => patterns
  - Tuples
  - Case Classes
- Pattern Matching
  - Destructuring
  - Partial Functions

```scala
// monads
def shout(s: Option[String]): Option[String] =
  s.filter(_.endsWith("!")).map(_.toUpperCase)

println(shout(Some("Scala")))      // None
println(shout(Some("scala!!!")))   // Some(SCALA!!!)

// destructure
def stats(xs: List[Int]): (Int, Int) =
  (xs.sum, xs.sum / xs.size)

val (a, b) = stats(List(1, 2, 3, 4, 5))
println(a)                    // 15
println(b)                    // 3

// case classes => tuple operations
case class Pair[T, U](a: T, b: U)
type SumAvg = Pair[Int, Int]
val (sum, avg) = stats(List(1, 2, 3, 4, 5))

def score(stats: SumAvg): String =
  stats match
    case Pair(s, a) if s >= 20 && a >= 4 => s"High score: $s"
    case Pair(s, a) if s < 15 && a < 3 => s"Low score: $s"
    case _ => s"Average score: $s"

println(score(Pair(sum, avg))) // Average score: 15
```

Scala Lang => Contexts

- Implicit => Scala 2
  - “Hidden” Context
  - Compiler Errors
    - Missing
    - Conflict
- Contexts => Scala 3
  - Extension
  - Using
  - Given

```scala
object ScalaContexts:

  // Scala 2

  // implicits => ordering
  val xs = List("apple", "banana", "cherry", "date", "elderberry")
  xs.sortBy(_.length)
  // List(date, apple, banana, cherry, elderberry)

  object ReverseOrdering extends Ordering[Int]:
    def compare(x: Int, y: Int): Int = y.compareTo(x)

  xs.sortBy(_.length)(ReverseOrdering)
  // List(elderberry, banana, cherry, apple, date)


  // Scala 3

  // extensions
  extension (s: String)
    def isPalindrome: Boolean = s == s.reverse

  // using => term inference
  xs.sortBy(_.length)(using ReverseOrdering)

  // given => canonical value
  case class Config(host: String, port: Int)
  given config: Config = Config("localhost", 8080)
```

Scala Lang => Min-Max

- For-Comprehension
  - Coordination

- Functional
  - Fold
  - Reduce
  - API

```scala
type MinMax = (Int, Int)

// classic for
def minMax1(xs: List[Int]): MinMax =
  var p = Int.MaxValue
  var q = Int.MinValue
  for x <- xs do
    p = p min x
    q = q max x
  (p, q)

// fold
def minMax2(xs: List[Int]): MinMax =
  (xs.fold(Int.MaxValue)(_ min _), xs.fold(Int.MinValue)(_ max _))

// reduce
def minMax3(xs: List[Int]): MinMax =
  (xs.reduce(_ min _), xs.reduce(_ max _))

// convenience
def minMax4(xs: List[Int]): MinMax = (xs.min, xs.max)
```

Scala => Coffee Bar

```
// An {{ORDER}} is noted with an espresso shot numbe
// `x` ranging between 1 and 4, and an optional letter
// `L` that indicates that the student wants to have a
// `x`-shot latte. For example, an order 2 means a 2-shot
// espresso, and an order 3L means a 3-shot latte:
//
//  - Making an x-shot espresso consumes x ounces of
//    water
//  - Making a latte requires steaming milk and
//    consumes one additional ounce of water
//
// The espresso machine at the micro kitchen has a water
// {{TANK}} of `s` ounces that is full at the beginning
// of the day. John refills the water tank to s ounces
// whenever the remaining water in the tank is not
// enough to fulfill the next student's order.
//
// How many times do John have to {{REFILL}} the water
// tank today in order to serve all `n` students?
```

```scala
object CoffeeBar:

  // {{ORDER}}
  case class Order(waterOunces: Int)

  def parseOrder(order: String): Order =
    order.takeRight(1) match
      case "L" => Order(order.dropRight(1).toInt + 1)
      case _ => Order(order.toInt)

  // {{TANK}}
  case class Tank(capacity: Int, refills: Int = 0)

  // {{REFILL}}
  def makeDrink(tank: Tank, order: Order, max: Int): Tank =
    if (tank.capacity - order.waterOunces < 0) tank.copy(
      capacity = max,
      refills = tank.refills + 1
    )
    else tank.copy(
      capacity = tank.capacity - order.waterOunces
      )

  // Context
  def countTankRefills(orders: List[String], tankCapacity: Int): Int =
    orders
      .map(parseOrder)
      .foldLeft(Tank(tankCapacity)):
        (tank, order) => makeDrink(tank, order, tankCapacity)
      .refills
```

Scala => Play Domain

- Product Table
  - Internal Representation
  - Tuple Mapping
  - Low Impedance
- ExecutionContext
  - Async Environment
  - Future

```
# conf/application.conf
slick.dbs.default.profile = "slick.jdbc.PostgresProfile$"
slick.dbs.default.driver = "slick.driver.PostgresDriver"
slick.dbs.default.db.driver = "org.postgresql.Driver"
slick.dbs.default.db.url = "jdbc:postgresql://localhost:5432/mydatabase"
slick.dbs.default.db.user = "myuser"
slick.dbs.default.db.password = "mypassword"
```

```
case class Product(id: Int,
                   name: String,
                   description: String,
                   price: Double)

@Singleton
class ProductRepository @Inject()(dbConfigProvider: DatabaseConfigProvider)
                                 (implicit ec: ExecutionContext) {
  private val dbConfig = dbConfigProvider.get[JdbcProfile]

  import dbConfig._
  import profile.api._

  private class ProductTable(tag: Tag) extends Table[Product](tag, "product") {
    def id = column[Int]("id", O.PrimaryKey, O.AutoInc)

    def name = column[String]("name")

    def description = column[String]("description")

    def price = column[Double]("price")

    def * =
      (id, name, description, price) <>
        ((Product.apply _).tupled, Product.unapply)
  }

  private val products = TableQuery[ProductTable]

  def list(): Future[Seq[Product]] = db run {
    products.result
  }

  def get(id: Int): Future[Option[Product]] = db run {
    products.filter(_.id === id).result.headOption
  }

  def create(product: Product): Unit = db.run(products += product)
}
```

Scala =>
Play Controller

- Implicits
  - Future => Async
  - Request => HTTP
  - JSON => Encoding
- ProductForm
  - Separation of Concerns
  - Separate Router

```
# conf/routes
POST    /products                   controllers.ProductController.create()
GET     /products                   controllers.ProductController.list()
GET     /products/:id               controllers.ProductController.get(id: Int)
```

```
You, 15 minutes ago | 1 author (You)
@Singleton
class ProductController @Inject()(val productRepository: ProductRepository,
                                  val cc: ControllerComponents)
                                  (implicit ec: ExecutionContext)
  extends AbstractController(cc) {

  import play.api.libs.json._
  implicit val productJson: OFormat[Product] = Json.format[Product]

  def list() = Action.async { implicit request =>
    productRepository.list().map { products =>
      Ok(Json.toJson(products))
    }
  }

  def get(id: Int) = Action.async { implicit request =>
    productRepository.get(id).map {
      case Some(product) => Ok(Json.toJson(product))
      case None => NotFound
    }
  }

  val productForm = {
    import play.api.data.Forms._
    import play.api.data.format.Formats.doubleFormat
    Form(
      mapping(
        "id" -> number,
        "name" -> text,
        "description" -> text,
        "price" -> of(doubleFormat)
      )(Product.apply)(Product.unapply)
    )
  }

  def create = Action.async { implicit request =>
    productForm.bindFromRequest.fold(
      errorForm => {
        Future(BadRequest("Invalid data"))
      },
      product => {
        productRepository.create(product)
        Future.successful(Created)
      }
    )
  }

}
```

Scala => SBT

- Simple Build Tool
  - Scala Programs
  - Interactive
  - Lazy

- Giter8
  - Templates
  - `sbt new ...`

```
import Dependencies._

ThisBuild / organization := "tech.leafwinglabs"
ThisBuild / scalaVersion := "3.5.0"

lazy val `jvm-languages-tour-sbt-sc3` =
  project
    .in(file("."))
    .settings(name := "jvm-languages-tour-sbt-sc3")
    .settings(commonSettings)
    .settings(autoImportSettings)
    .settings(dependencies)

lazy val dependencies = Seq(
  libraryDependencies ++= Seq(
    // main dependencies
  ),
  libraryDependencies ++= Seq(
    com.eed3si9n.expecty.expecty,
    org.scalacheck.scalacheck,
    org.scalameta.`munit-scalacheck`,
    org.scalameta.munit,
    org.typelevel.`discipline-munit`,
  ).map(_ % Test),
)
```

Scala => Ammonite

- Ivy
  - Dependency Manager
  - Like Groovy Grapes
- CLI Runner
  - @main

```scala
import $ivy.`com.github.tototoshi:scala-csv_2.13:1.3.8`
import scala.io.Source
import com.github.tototoshi.csv._

case class Product(
  id: Int,
  name: String,
  description: String,
  price: Double
)

@main def processCsvData(csvFilePath: String): Unit = {
  val reader = CSVReader.open(Source.fromFile(csvFilePath))
  val products: List[Product] = reader.allWithHeaders().map { row =>
    Product(row("id").toInt, row("name"), row("description"), row("price").toDouble)
  }
  reader.close()
  products.foreach(println)
}
```

```
❯ amm src/main/scala/tech/leafwinglabs/ScalaAmmonite.sc src/main/resources/products.csv
https://repo1.maven.org/maven2/com/github/tototoshi/scala-csv_2.13/1.3.8/scala-csv_2.13-1.3.8.pom
  100.0% [##########] 2.0 KiB (4.1 KiB / s)
https://repo1.maven.org/maven2/com/github/tototoshi/scala-csv_2.13/1.3.8/scala-csv_2.13-1.3.8-sources.jar
  100.0% [##########] 10.8 KiB (88.7 KiB / s)
Compiling /Users/ehonsey/Dropbox/engineering/presentations/jvm-languages-tour/src/scala/jvm-languages-tour-sbt-sc3/
Product(1,apple,fruit,1.0)
Product(2,banana,fruit,2.0)
Product(3,carrot,vegetable,3.0)
```

Scala => Category Theory

- Monads
  - Common Contract
    - Pure
    - FlatMap
  - Different Contexts
    - Option
    - List
    - Future
    - Either
    - Reader...
- Scalatest
  - Test DSLs

```scala
import cats.*
import cats.implicits.*
import org.scalatest.flatspec.AnyFlatSpec
import org.scalatest.matchers.should.Matchers

class MonadTest extends AnyFlatSpec with Matchers {

 implicit def optionMonad(implicit app: Applicative[Option]): Monad[Option] =
   new Monad[Option] {
     def pure[A](x: A): Option[A] = app.pure(x)

     def flatMap[A, B](fa: Option[A])(f: A => Option[B]): Option[B] =
       fa.flatMap(f)
   }

 "Monad Option" should "return value" in:
   Monad[Option].pure(1)shouldBe Some(1)
   Monad[Option].flatMap(Some(1)) {
     x => Some(x + 1)
   } shouldBe Some(2)
   Monad[Option].flatMap(None: Option[Int]) {
     x => Some(x + 1)
   } shouldBe None

implicit def listMonad(implicit app: Applicative[List]): Monad[List] =
   new Monad[List] {
     def pure[A](x: A): List[A] = List(x)

     def flatMap[A, B](fa: List[A])(f: A => List[B]): List[B] =
       fa.flatMap(f)
   }

 "Monad List" should "return value" in:
   Monad[List].pure(1) shouldBe List(1)
   Monad[List].flatMap(List(1, 2, 3)) {
     x => List(x, x)
   } shouldBe List(1, 1, 2, 2, 3, 3)

}
```

Scala Resources

- **Exercises**
  - Exercism
  - Scala Exercises
  - Codewars
  - Leetcode
  - stkeky/best-of-scala
- **Books**
  - Scala 3 Book
  - Scala 2 Book
  - Scala Algorithms
  - Functional Programming in Scala
  - Pragmatic Scala
  - Functional Programming Patterns in Scala and Clojure

Domain	Sample Libraries
Web	Play, finatra
Data	Apaache Spark, Flink, Doobie, Slick
Data Science	Breeze, Smile,
Tools	sbt, Ammonite, Mill, Metals
Distributed	Kafka, Finagle, Akka / Pekko
Caching	scalacache, scaffeine, zio-cache
Graphs	cassoary, scala-graph
Category Theory	Cats, scalaz
Effects	ZIO, fs2, Circe, Monocle

Scala Rap Sheet

- ★ Rich Abstractions
- ★ Concurrency First
- ★ Type Safe
- ★ Contextual Abstraction
- ★ Continuous Innovation

- ❑ Compilation Times
- ❑ Compiler Battles
- ❑ Cognitive Overload
- ❑ Over-Abstraction
- ❑ Niche Communities

Scala is a continuation of the Haskell lineage for the JVM-oriented developer.

Clojure

Mad Hatter Wizard

Clojure History

- Timeline
  - Development => 2005
    - *jfli* => Foreign Language Interface
    - *FOIL* => Foreign Object Interface for Lisp
    - *Lisplets* => Java Servlets
  - Release => 2007
- Traits
  - LISP Dialect => “List Processing”
  - Functional Thinking
  - Dynamic Typing
  - Immutable Data Structures
  - REPL => Read-Eval-Print-Loop

Version	Release Date	Features
0.x	2007-10-17	Initial Release
1.0	2009-05-04	Persistent Data Structures, Protocols, Multimethods, Agents, Atoms
1.1	2009-12-31	Futures/Promises, `clojure.test` , `clojure.walk`
1.2	2010-08-19	`defprotocol` , `defrecord` , `case` , Destructuring, Enhancements
1.3	2011-09-23	Enhanced Primitive Support, Error Reporting, Performance Improvements
1.4	2012-04-15	`edn` , Reader Literals, Java Interop and Collections enhancements
1.5	2013-06-07	`core.async` , Reducers, Agent / Threading Improvements , Performance Enhancements
1.6	2014-03-25	Java API Enhancements, Namespaced Key Destructuring, Performance Enhancements
1.7	2015-06-30	Transducers, Reader Conditionals, Performance Enhancements
1.8	2016- 01-19	Direct Linking, Core String Functions, Socket Server REPL
1.9	2017-12-08	`clojure.spec` , Qualified Maps, New Predicates, Instant Support
1.10	2018-12-17	JDK 1.8+, Error Introspection, Graph Navigation, `tap`
1.11	2022-03-22	`clojure.math` , `update-keys` , `update-vals`
Clojure 1.12.0-rc2	2024-08-28	Hot Load Libs, `clojure.java.process` , Qualified Methods, Performance Enhancements

Clojure Lang => Data

- Namespaces
  - S-Expressions
    - Data
    - Functions
- Data => Immutable Only
  - State => STM
  - `nil` Safety
- Functions => Data
  - Uniformity
- Special Forms
  - Minimal

```clojure
(ns tech.leafwinglabs.lang                          ;; namespaces encapsulate data
  (:require [clojure.string :as str])               ;; import Clojure namespaces
  (:import [java.util Date]))                       ;; import Java classes

;; Everything is a s-expr
(print                                              ;; call fn form => *out*
  "a"                                               ;; data
  :b                                                ;; keyword => identity [function]
  (fn [x y] (+ x y))                                ;; build other fns
  (str 1 " and " 2)                                 ;; call site
  [1 2 3])                                          ;; vector

; Data
(def m {:a "foo" :b 1 :c "bar"})                    ;; keywords mark data
(:a m)                                              ;; keywords are functions
(m :a)                                              ;; maps are functions too

;; Functions
(str/split "a,b,c" #",")                            ;; call a function in other ns
(defn inc' [x] (+ x 1))                             ;; intern a function
(println (inc' 1))                                  ;; 2
(def inc'' (partial + 1))                           ;; partial application
(println (inc'' 11))                                ;; 12
((comp not str/blank? :name) {:name "eric"})        ;; composition

;; special forms
(if '(:pred) true false)                            ;; branch on predicate
(do '(:body))                                       ;; evaluate forms and return last
(let [x 1] '(:body))                                ;; local bindings
(fn f [x] '(:do))                                   ;; higher order function
#(':do)                                             ;; anonymous functions
(loop [acc '()] (do '(:body) '(recur)))               ;; tail recursion
(try '(:body)                                       ;; exception handling
     (catch Exception e (throw e))
     (finally '(:body)))
```

Clojure Lang => Collections

- **Seq** => Collections
  - List
  - Vector
  - Map
  - Set
- **Streams** => Threads
  - Function Composition
  - | => Unix Pipes

```clojure
;; collection data structures
(seq '(:a))                                          ;; base => lazy lists
'(:a :b :c)                                          ;; lists => linked
[1 2 3]                                              ;; vectors => eager arrays
{:a 1 :b 2 :c 3}                                     ;; maps => seq of pairs
#{:a :b :c}                                          ;; sets => domains
(defn fib [a b] (lazy-seq (cons a (fib b (+ a b))))) ;; infinite lists => streams

;; collection functions
(map inc [1 2 3])                                    ;; map => transform
(filter odd? [1 2 3])                                ;; filter => select
(reduce + [1 2 3])                                   ;; reduce => fold
(apply min [1 3 0 2])                                ;; apply => variadic

;; streams
(def seq
  (->> '("1" "2" "3" "4" "5")                        ;; chain of responsibilities
       (map #(Integer/parseInt %))                   ;;  |
       (filter even?)                                ;;  |....
       (map #( * % %))
       (sort-by #(apply str (reverse (str %))))
       (map #(str (char %)))
       (clojure.string/join "|")));; => 1 2 3
```

Clojure Lang => Meta

- **Macros**
  - *Start* Evaluation
  - *Stop* Evaluation
- **Threading**
  - `->` *First* Arg
  - `->>` *Last* Arg
- **Do**
  - Side Effects
  - Flavors

```clojure
;; metaprogramming
(defmacro when' [t b] (list 'if t (cons 'do b)))          ;; macros transform symbols
(macroexpand-1 '(when' true (println "hi")))              ;; (if true (do println "hi"))

(defmacro unless [pred & body]
  `(if (not ~pred)
     (do ~@body)))                                        ;; syntax quotes stop evaluation

;; threading / do => stream processing
(->> [1 2 3]                                             ;; thread last => collections
     (map inc)
     (filter odd?))                                       ;; => (3)
(-> " one , two, three "                                  ;; thread first => objects
    str/trim
    (str/replace #"\s?,\s?" "|")
    #_(str/split #"|"))
;; => "one|two|three"

;; object side effects
(doto (StringBuilder.)
  (.append "a")
  (.append "b"))
;; => #object[java.lang.StringBuilder 0x2c708440 "ab"]

;; consumer
(doseq [x [1 2 3]] (println x))
```

Clojure Reader

- **Homoiconicity**
  - Reader <> Data
  - Extensible => edn
  - Custom Readers
- # => Dispatch Macro
  - Read Table
- ` => Syntax Quote
  - Non-Evaluation
  - Custom Macros

,	Comma reads as white space. Often used between map key/value pairs for readability.
'	quote: '*form* → (quote *form*)
/	Namespace separator (see Primitives/Other section)
\	Character literal (see Primitives/Other section)
:	Keyword (see Primitives/Other section)
;	Single line comment
^	Metadata (see Metadata section)
foo	'earmuffs' - convention to indicate dynamic vars, compiler warns if not dynamic
@	Deref: @*form* → (deref *form*)
`	Syntax-quote
foo#	'auto-gensym', consistently replaced with same auto-generated symbol everywhere inside same `(...)
~	Unquote
~@	Unquote-splicing
->	'thread first' macro ->
->>	'thread last' macro ->>
>!! <!! >! <!	core.async channel macros >!! <!! >! <!
(	List literal (see Collections/Lists section)
[	Vector literal (see Collections/Vectors section)
{	Map literal (see Collections/Maps section)

#'	Var-quote: #'*x* → (var *x*)
#"	#"*p*" reads as regex pattern *p* (see Strings/Regex section)
#{	Set literal (see Collections/Sets section)
#(	Anonymous function literal: #(...) → (fn [args] (...))
%	Anonymous function argument: %N is value of anonymous function arg N. % short for %1. %& for rest args.
#?	Reader conditional: #?(:clj x :cljs y) reads as x on JVM, y in ClojureScript, nothing elsewhere. Other keys: :cljr :default
#?@	Splicing reader conditional: [1 #?@(:clj [x y] :cljs [w z]) 3] reads as [1 x y 3] on JVM, [1 w z 3] in ClojureScript, [1 3] elsewhere.
#foo	tagged literal e.g. #inst #uuid
#:	map namespace syntax e.g. #:foo{:a 1} is equal to {:foo/a 1}
##	(1.9) symbolic values: ##Inf ##-Inf ##NaN
$	JavaContainerClass$InnerClass
foo?	conventional ending for a predicate, e.g.: zero? vector? instance? (unenforced)
foo!	conventional ending for an unsafe operation, e.g.: set! swap! alter-meta! (unenforced)
_	conventional name for an unused value (unenforced)
#_	Ignore next form

Clojure => Min-Max

```clojure
(ns tech.leafwinglabs.min-max)

;; seed
(defn min-max
  [xs]
  (reduce
    (fn [[min max] x]
      [(if (or (nil? min) (< x min)) x min)
       (if (or (nil? max) (> x max)) x max)])
    [nil nil]
    xs))

;; seeds
(defn min-max-2
  [xs]
  (reduce (fn [[min max] x]
            [(min min x) (max max x)])
          [Integer/MAX_VALUE Integer/MIN_VALUE]
          xs))
```

```clojure
;; local functions
(defn min-max-3
  [xs]
  (letfn [(min-max [[p q] x]
            [(min p x) (max q x)])]
    (reduce min-max
            [Integer/MAX_VALUE Integer/MIN_VALUE]
            xs)))

;; threading
(defn min-max-4
  [xs]
  (->> xs
       (map #(* % %))
       (reduce (fn [[min max] x]
                 [(min min x) (max max x)])
               [Integer/MAX_VALUE Integer/MIN_VALUE])))

;; tail recursion
(defn min-max-5
  [xs]
  (loop [xs xs, min Integer/MAX_VALUE, max Integer/MIN_VALUE]
    (if (empty? xs)
      [min max]
      (let [x (first xs)]
        (recur (rest xs)
               (min min x)
               (max max x))))))
```

Clojure => Web DB

- Simple Abstractions
  - Functions
  - Vectors
  - Maps => Spec

- Rich Comment
  - REPL
  - Testing

```clojure
(ns tech.leafwinglabs.product.db
  (:require [next.jdbc :as jdbc]))

(declare db-spec)

;; queries
(defn list-products []
  (with-open [conn (jdbc/get-connection db-spec)]
    (jdbc/execute! conn ["SELECT * FROM product"])))

(defn create-product [product]
  (with-open [conn (jdbc/get-connection db-spec)]
    (jdbc/execute! conn ["INSERT INTO product (name, description, price) VALUES (?, ?, ?)"
                         (:name product)
                         (:description product)
                         (:price product)])))

(defn get-product [product-id]
  (let [product-id (Integer/parseInt product-id)]
    (with-open [conn (jdbc/get-connection db-spec)]
      (jdbc/execute! conn ["SELECT * FROM product WHERE id = ?" product-id]))))

;; db mappings
(def db-spec
  (jdbc/get-datasource
    {:dbtype  "postgresql"
     :dbname   "mydatabase"
     :user     "myuser"
     :password "mypassword"
     :host     "localhost"
     :port     5432}))

;; repl
(comment
  (list-products)
  (create-product {:name "Product 5" :description "Description 5" :price 50})
  (get-product 5)

  :rcf)
```

Clojure => Web Routing

- Functional Abstractions
  - Utils
  - Route Handlers
  - Route Orchestration

- Middleware
  - Function
  - Chain of Responsibility

```clojure
(ns tech.leafwinglabs.product.api
  (:require [reitit.ring :as ring]
            [ring.adapter.jetty :as jetty]
            [ring.middleware.json :refer [wrap-json-body wrap-json-response]]
            [tech.leafwinglabs.product.db :as db]))

;; response util
(defn response [status body]
  {:status  status
   :headers {"Content-Type" "application/json"}
   :body    body})

;; Route handlers
(defn list-products-handler [_]
  (response 200 (db/list-products)))

(defn create-product-handler [request]
  (let [product (-> request :body)]
    (response 201 (db/create-product product))))

(defn get-product-handler [request]
  (let [product-id (-> request :path-params :id)]
    (if-let [product (db/get-product product-id)]
      (response 200 product)
      (response 404 {:error "Product not found"}))))

;; API Routes
(def handler
  (-> (ring/ring-handler
        (ring/router
          [["/products"
            {:get  list-products-handler
             :post create-product-handler}]
           ["/products/:id"
            {:get get-product-handler}]]))
      (wrap-json-body)
      (wrap-json-response)))
```

Clojure => Web Server

- State Management
  - `Defonce`
  - `Atom`
  - `Swap!`

- Rich Comment
  - REPL
  - Testing

```clojure
;; server lifecycle
(defonce ^:private server (atom nil))

(defn restart []
  (swap! server (fn [x]
                  (when x (.stop x))
                  (jetty/run-jetty
                    #'handler
                    {:port 3000 :join? false})))
  (println "server running in port 3000"))

;; repl
(comment
  (restart)

  (handler {:request-method :get :uri "/products"})
  (handler {:request-method :get :uri "/products/1"})
  (handler {:request-method :post :uri "/products"
            :body           {:name "Product 5"
                             :description "Product 5 description"
                             :price 50.0}})

  :rcf)
```

Clojure DS&A =>Tree Walking

```clojure
(require '[clojure.walk :as walk])

(defn postwalk
  [f form]
  (walk (partial postwalk f) f form))

(walk/postwalk-demo {:a 1 :b {:c 2 :d 3} :e [4 5]})
;Walked: :a
;Walked: 1
;Walked: [:a 1]
;Walked: :b
;Walked: :c
;Walked: 2
;Walked: [:c 2]
;Walked: :d
;Walked: 3
;Walked: [:d 3]
;Walked: {:c 2, :d 3}
;Walked: [:b {:c 2, :d 3}]
;Walked: :e
;Walked: 4
;Walked: 5
;Walked: [4 5]
;Walked: [:e [4 5]]
;Walked: {:a 1, :b {:c 2, :d 3}, :e [4 5]}
```

```clojure
(require '[clojure.walk :as walk])

(defn prewalk
  [f form]
  (walk (partial prewalk f) identity (f form)))

(walk/prewalk-demo {:a 1 :b {:c 2 :d 3} :e [4 5]})
;Walked: {:a 1, :b {:c 2, :d 3}, :e [4 5]}
;Walked: [:a 1]
;Walked: :a
;Walked: 1
;Walked: [:b {:c 2, :d 3}]
;Walked: :b
;Walked: {:c 2, :d 3}
;Walked: [:c 2]
;Walked: :c
;Walked: 2
;Walked: [:d 3]
;Walked: :d
;Walked: 3
;Walked: [:e [4 5]]
;Walked: :e
;Walked: [4 5]
;Walked: 4
;Walked: 5
```

Core.async => CSP

- Channels
  - Go Blocks
  - Non-Blocking
    - <! Take
    - >! Put
  - Blocking
    - <!! Take
    - >!! Put
- Pipelines
  - Transducers
- Alts
  - Fastest Response

```clojure
(ns tech.leafwinglabs.async
  (:require [clojure.core.async :as async :refer [<!! >!]]))

;; product => could be clojure.spec
(def product {:id          1
              :name        "Product 1"
              :description "Product 1 description"
              :price       100})

;; all products
(def product-stream (async/chan))
;; high price products
(def high-price-products (async/chan))

;; domain logic
(defn high-price? [product]
  (> (:price product) 50))

(comment

  (def product-stream (async/chan))
  (def high-price-stream (async/chan))
  (def high-price-pipe (async/pipeline 4
                                       high-price-products
                                       (filter high-price?) ; transducer
                                       product-stream))

  (async/go
    (let []
      (doseq [id (range 100)]
        (let [product {:id          id
                       :name        (str "Product")
                       :description (str "Product " id " description")
                       :price       (rand-int 100)}]
          (>! product-stream product)))))

  (println (str "Product : " (<!! product-stream)))
  (println (str "High price product: " (<!! high-price-products)))

  (async/close! product-stream)
  (async/close! high-price-products)

  :rcf)
```

Clojure => Systems

- **System => Map**
- Component
  - Start
  - Stop
- Integrant
  - Init-key
  - Halt-key!
  - Suspend-key!
  - Resume-key

```clojure
;; datasource
(defn ->datasource
  []
  (hikari/make-datasource
   {:adapter "postgresql"
    :username "user"
    :password "password"
    :subname "//localhost:5432/mydb"}))

;; component mixin
(defrecord Database [datasource]
  component/Lifecycle
  (start [this]
    (println "Starting the database connection pool")
    (assoc this :datasource datasource))
  (stop [this]
    (println "Stopping the database connection pool")
    (hikari/close-datasource (:datasource this))
    (dissoc this :datasource)))

(def database (->Database (->datasource)))
(def comp-system (component/system-map :database database))

;; start
(component/start comp-system)

;; stop
(component/stop comp-system)



;; Integrant mixin
(def config {:datasoure (->datasource)})

;; Start the component
(def int-system (ig/init config))

;; Stop the component
(ig/halt int-system)
```

Clojure Resources

- Web
  - Exercism
  - 4clojure
  - functional-koans/clojure-koans
  - Clojure Guides
  - Clojure Docs
  - Clojurians Slack
  - Codewars
- Books
  - Grokking Simplicity
  - Functional Programming Patterns in Scala and Clojure
  - Any Clojure Pragmatic Bookshelf
  - LISP Books

Domain	Sample Libraries
System	Component, Integrant, Mount, Polylith
Config	edn, aero
Web Server	Ring, Reitit, Compojure
Web Client	Clojurescript, Pedestal, Om, Duct
Async	core.async, aleph
Data	next.jdbc, honeysql, korma
Data Science	Incanter, Neanderthal, Scicloj
Graphs	Pathom, Specter, Datomic
Testing	clojure.test, kaocha
Spec	clojure.spec, metosin/malli, plumatic/schema
Matching	core.match, Meander
Tools	deps, clj-kondo, Flow Storm

Clojure Rap Sheet

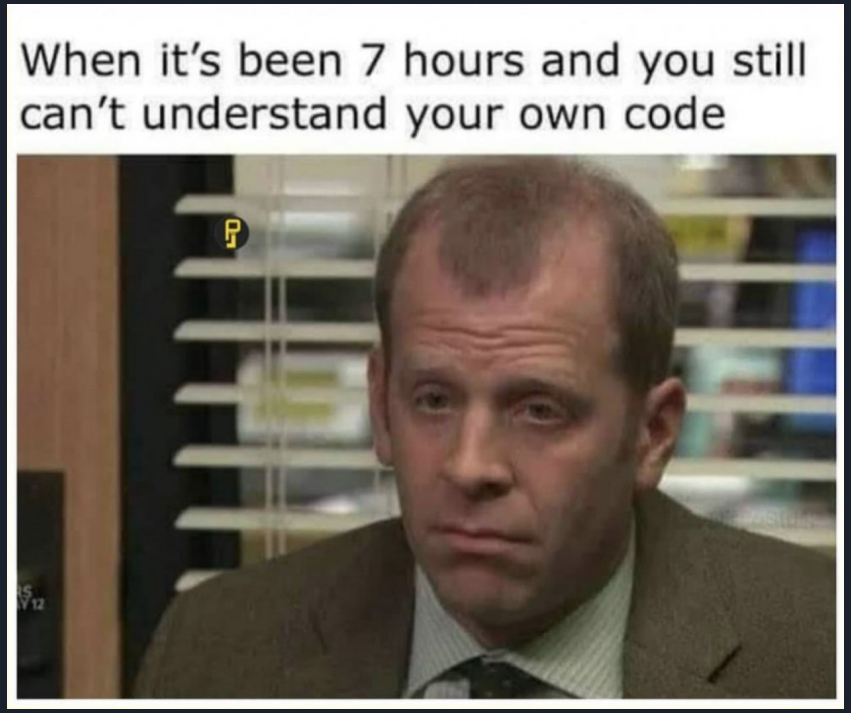
- ★ Everything is a *List <> Seq <> s-expr*
- ★ Immutability First
- ★ Concurrency First
- ★ Flexible Abstractions
- ★ Powerful Metaprogramming
- ★ Many Use Cases

- ❑ Large Learning Curve
- ❑ Esoteric
- ❑ Smaller Community
- ❑ Tooling Difficulties

Clojure aims to free code and data by using the simplest forms possible

Takeaways

- *Common Patterns*
  - Data Containers
  - Streams
  - Contexts
- *Functional Mindset*
  - Immutability
  - Concise Declarations
  - Traits
- *Languages*
  - Java => Lower-Level
  - Groovy => Spectator
  - Kotlin => Future
  - Scala => Architect
  - Clojure => Engineer



Thank You

Eric Honsey
Leafwing Labs LLC
eric@leafwinglabs.tech
LinkedIn
Toptal