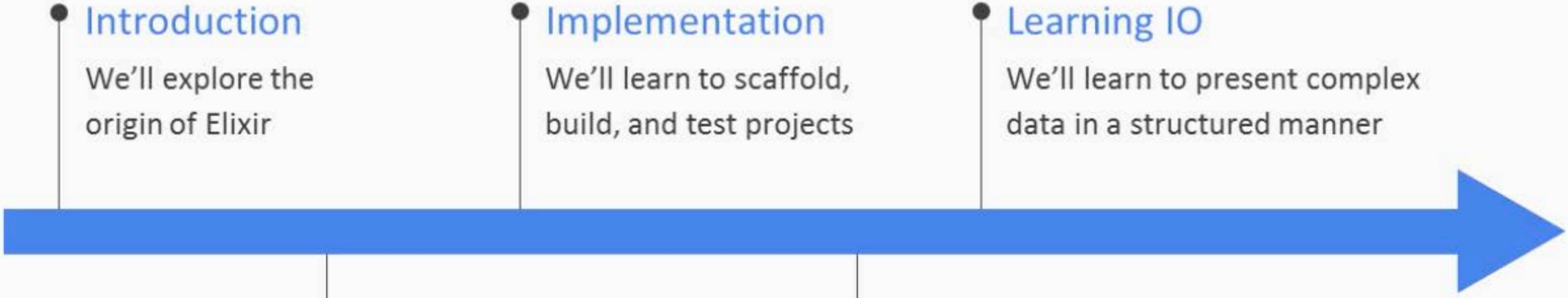
Packt

Elixir: Scalable and Efficient Application Development

About This Course

The goal of this course is to help you gain an in-depth understanding of the core Elixir programming language.
You will learn to build your own Elixir applications from scratch.

What we cover



Introduction

We'll explore the origin of Elixir

Implementation

We'll learn to scaffold, build, and test projects

Learning IO

We'll learn to present complex data in a structured manner

Core programming

We'll learn the fundamentals of Elixir

Computations

We'll learn about data types and collections

Our Learning Process

Step 1

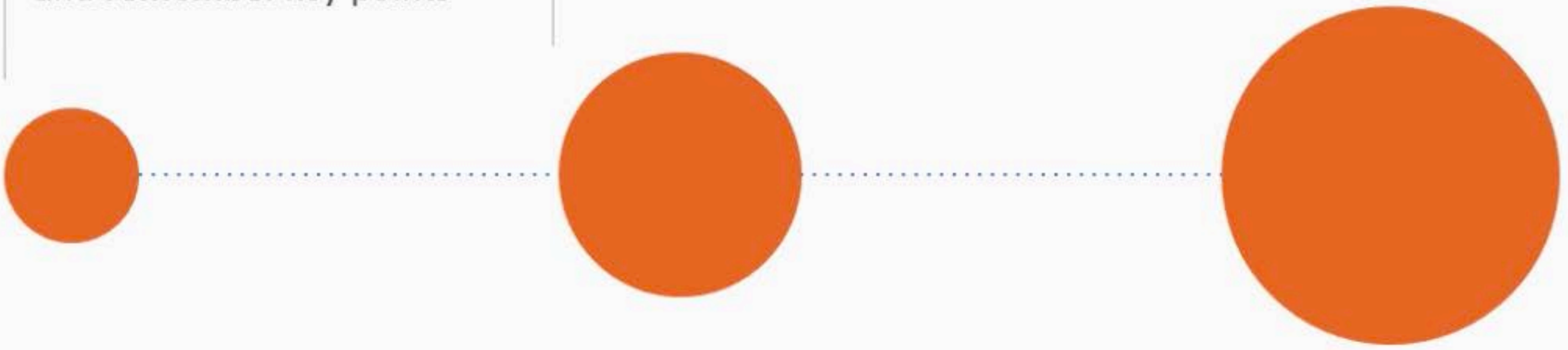
We help you **understand** and **remember** key points

Step 2

We show you how to **analyse** and **apply** what you've seen

Step 3

We challenge you to **create** and **build** your own examples



Your Instructor Team

- **Joao Goncalves** is a professional software engineer with over 7 years of experience in various areas of software development.
- **Kenny Ballou** is a life-long learner, developer, mathematician, and overall thinker. He enjoys solving problems, learning about technologies, and discussing new and different ideas.

Course Editor

Your specialist editor is here to help you progress through the entire course from start to finish.

You can get in touch with your editor, our expert instructors, and other active learners via the built-in **Q&A** section.

Stay Relevant

We've carefully assembled tried-and-true technical content to deliver you the best learning experience possible.

Now it's up to you!

Elixir: Scalable and Efficient Application Development

João Gonçalves

What is Elixir?

Packt>

In this video, we are going to take a look at...

- Exploring the history of Elixir
- Discussing language characteristics
- Observing notable use cases

History of Elixir

Erlang

Design Goals

Compatibility Productivity Extensibility

History of Elixir

2014

Characteristics

- General Purpose
  - Suitable for multiple application types
- Functional
  - Adheres mostly to the functional paradigm, but not only
- Actor Model
  - Processes are akin to Actors, encapsulating data (state), receiving, and sending messages

Characteristics

- Dynamically Typed
  - Types are inferred in compile and run-time
- Compiled
  - The compiler generates code that is executed by the BEAM (Erlang VM)

Use Cases

Use Cases

- General applications
- Large scale services
- Web applications

- Systems programming
- Scripting

Elixir: Scalable and Efficient Application Development

João Gonçalves

Functional Programming

Packt>

In this Video, we are going to take a look at...

- Defining functional programming
- Discussing motivation
- Exploring characteristics and traits

What is Functional Programming?

- Evaluation of expressions
- Promotes immutable state

What is Functional Programming?

- Behaves more closely to their mathematical counterpart
- Is a first-class citizen (i.e. can be returned from functions and passed as argument)

Function

$$f(x) \rightarrow y$$

Motivation

- Better structuring discipline
- No side-effects reduces number of bugs
- Suitable for parallelism

Characteristics

- Higher order functions:
  - Functions as return values and as parameters to other functions

$$f(g(x)) \rightarrow z(x)$$

- Recursion:
  - Allowing a function to call itself or looping algorithms

$$\begin{cases} f(x) \rightarrow x + f(x-1) \\ f(1) = 1 \end{cases}$$

Characteristics

- Referential Transparency:
  - Same evaluation = same outcome

$$y = f(x)$$

$$g(y, y) \Leftrightarrow g(f(x), f(x))$$

Quiz Time!

Quiz 1 | 2 Questions

Start Quiz Skip Quiz >

Question 1:

Choose the correct core Elixir language design goal from the following options:

- Flexibility of Erlang
- Compatibility of Erlang
- Interactivity of Erlang
- Efficiency of Erlang

Question 2:

Which of the following are the characteristics of Elixir programming language?

- It is a non-functional programming language
- It is a statically typed language
- It has the actor model programming paradigm
- It is an interpreted languages