2 Months Challenge

With 1-2 hours per day, how can I learn Scala?

Figure 1: Learning Scala and cats week by week

Weeks 1-2: High frustration. You'll keep trying to write for loops and if/else blocks. You are fighting 10+ years of Java muscle memory.

Weeks 3-4: The "Scala" click. You start seeing everything as a match expression.

Weeks 5-6: The "Cats" wall. Category theory terms (Endofunctors!) sound scary, but the code starts getting remarkably shorter.

Weeks 7-8: Synthesis. You start writing code that feels more like "assembling a machine" than "writing instructions."

Month 1: The Scala Foundation & FP Thinking

Focus: Mastering the syntax, the Type System, and the "Function over Mutation" mindset.

Week 1: Syntax & The Object-Functional Hybrid

Theory: Immutability, val vs var, Case Classes, Pattern Matching, and "Everything is an Expression."

Practice: Rewrite a standard Java service (e.g., a User Validator) using only match statements and no nulls.

Key Concept: Use Option and Either instead of throwing exceptions.

Week 2: Functions & Collections

Theory: Higher-order functions (map, flatMap, filter, foldLeft). Currying and Partial Application.

Practice: Implement a "Discount Engine" that chains multiple pricing rules together using function composition.

Comparison: Notice how flatMap in Scala differs slightly from Stream.flatMap in Java 8+.

Week 3: The Type System Deep Dive
Theory: Traits (as Interfaces on steroids), Generics (Covariance/Contravariance), and the most important Scala feature: Implicits (or given/using in Scala 3).

Practice: Create a "JSON Serializer" trait and implement instances for String and Int using Type Classes (without using inheritance).

Week 4: Functional Design Patterns (ADTs)
Theory: Sum Types and Product Types. Modeling domain logic such that "illegal states are unrepresentable."

Practice: Model a Payment System (Credit Card, PayPal, Wire Transfer) where each state has its own specific data.

Month 2: Mastering the Cats Library
Focus: Category Theory concepts applied to code. Moving from "Scala" to "Typelevel Scala."

Week 5: Semigroups & Monoids
Theory: Why "adding things together" is a formal algebraic structure.

Practice: Use Monoid.combine to merge complex Maps of nested data (e.g., aggregating monthly statistics from multiple servers).

Week 6: Functors & Applicatives
Theory: Functor (mapping over a context) and Applicative (combining independent effects).

Practice: Use Validated from Cats to perform "Parallel Validation"—where you collect all errors in a form at once rather than failing on the first one (as Either does).

Week 7: The Monad & Monad Transformers
Theory: The Monad laws and why we use them to sequence operations. Dealing with nested types like Future[Option[User]] using OptionT.

Practice: Build a small CLI tool that fetches data from a mock DB and a mock API, using Monad to ensure they run in order, and OptionT to handle missing data cleanly.

Week 8: Error Handling & Tagless Final (The "Senior" Part)
Theory: Finalizing the architecture. How to abstract over the "Effect" (e.g., F[_]).

Practice: Implement a small "To-Do" repository. Write the logic once, then run it using Id for unit tests and IO (from Cats Effect) for production.