Functional Programming (FP) and Object-Oriented Programming (OOP) are two different paradigms used to structure and organize code. Here's a brief comparison of the two:

1. **Focus:** FP focuses on the evaluation of mathematical functions and avoids changing state and mutable data. OOP focuses on modeling real-world entities as objects that have state and behavior.

2. **Data Handling:** In FP, data is immutable, meaning it cannot be modified after creation. Functions operate on this immutable data and produce new data as output. In OOP, objects encapsulate both data (state) and behavior (methods), and their interactions manipulate and mutate the object's state.

3. **State and Side Effects:** FP emphasizes avoiding side effects and mutable state, promoting pure functions that always produce the same output given the same input. OOP allows objects to maintain state and supports side effects, such as modifying state or interacting with external resources.

4. **Composition:** FP relies heavily on composing functions to create more complex operations by combining simpler functions. OOP focuses on composing objects that interact with each other through message passing.

5. **Inheritance vs. Higher-order Functions:** OOP employs inheritance, where objects can inherit properties and behaviors from parent objects. FP emphasizes higher-order functions, which treat functions as first-class citizens, allowing them to be passed as arguments, returned as results, and stored in variables.

6. **Concurrency:** FP's emphasis on immutability and pure functions makes it easier to reason about and parallelize code, as functions don't rely on or modify shared state. OOP, with its emphasis on mutable state, can make concurrent programming more challenging.

7. **Typing:** FP languages often use static typing, where types are checked at compile-time. OOP languages can have static or dynamic typing, with some flexibility in runtime type changes.

It's worth noting that many programming languages, like Scala, can support both paradigms to varying degrees. The choice between FP and OOP depends on factors such as the problem domain, team preferences, and language capabilities. Each paradigm has its strengths and weaknesses, and understanding both can make you a more versatile programmer.