The Nature of the Language

In this introductory lesson, we'll take a look at the purpose of ReasonML.

WE'LL COVER THE FOLLOWING

The Bridge Between OCaml and JavaScript

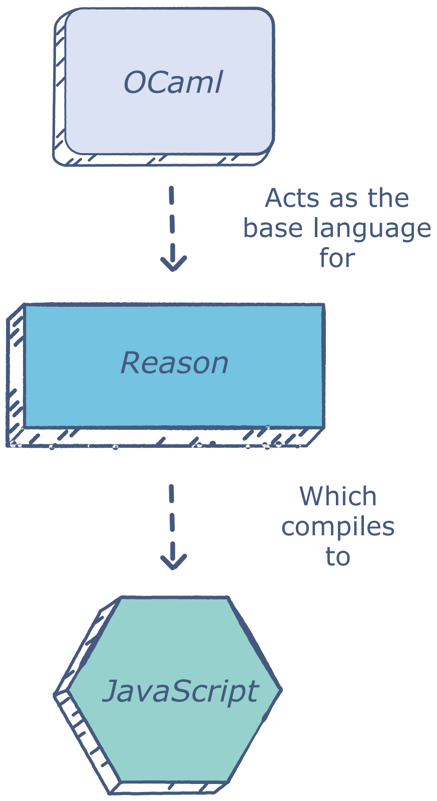
ReasonML, also known simply as **Reason**, was developed in 2016 by Facebook. It provides a new syntax and toolchain for the meta-language, **OCaml**. OCaml is an extremely efficient industrial-level language which focuses on type-driven functional programming and code optimization.

The purpose of Reason is to compile OCaml into JavaScript through the BuckleScript language. Bucklescript's syntax simply allows us to translate OCaml into clean and efficient JavaScript. This is why Reason adopts BuckleScript.

The distinguishing factor between Reason and JavaScript is that Reason is a **strongly typed** language. Unlike JavaScript, where we do not need to specify

strongly-typed language. Unlike JavaScript, where we do not need to specify data types, Reason contains explicit type definitions.

Its foundation in the strongly-typed OCaml makes Reason a faster and safer way to write JavaScript code.



Static vs. Dynamic

The original documentation for ReasonML calls it a “solidly statically typed cousin of JavaScript”. JavaScript is one of the most popular dynamic languages today. A dynamic language checks types during **runtime**.

In a **static** language like Reason, type-checking is performed at **compile time**. Although we have to be more careful when writing static languages, they are less prone to runtime crashes.

However, the beauty of Reason is that it has access to both OCaml (static) and JS (dynamic) ecosystems.

The Future of ReasonML

It is worth noticing that Facebook is heavily investing in Reason. The language has become fully compatible with **React** through **ReasonReact**, making it an extremely efficient platform for app development.

Reason’s community and functionality continue to grow rapidly with the latest stable release introduced in 2018. It will surely be relevant in the following years due to its optimization of JavaScript.

The next lesson will explore the primary operators of Reason.