LangChain Docs

Get started › Philosophy

Get started

Philosophy

Copy page

LangChain exists to be the easiest place to start building with LLMs, while also being flexible and production-ready.

LangChain is driven by a few core beliefs:

Large Language Models (LLMs) are great, powerful new technology.

LLMs are even better when you combine them with external sources of data.

LLMs will transform what the applications of the future look like. Specifically, the applications of the future will look more and more agentic.

It is still very early on in that transformation.

While it's easy to build a prototype of those agentic applications, it's still really hard to build agents that are reliable enough to put into production.

With LangChain, we have two core focuses:

1. **We want to enable developers to build with the best models.**

   Different providers expose different APIs, with different model parameters and different message formats. Standardizing these model inputs and outputs is a core focus, making it easy for developer to easily change to the most recent state-of-the-art model, avoiding lock-in.

2. **We want to make it easy to use models to orchestrate more complex flows that interact with other data and computation.**

LangChain Docs

Get started › **Philosophy**

History

Given the constant rate of change in the field, LangChain has also evolved over time. Below is a brief timeline of how LangChain has changed over the years, evolving alongside what it means to build with LLMs:

2022-10-24

v0.0.1

A month before ChatGPT, **LangChain was launched as a Python package**. It consisted of two main components:

- LLM abstractions
- "Chains", or predetermined steps of computation to run, for common use cases. For example - RAG: run a retrieval step, then run a generation step.

The name LangChain comes from "Language" (like Language models) and "Chains".

2022-12

The first general purpose agents were added to LangChain.

These general purpose agents were based on the **ReAct paper** (ReAct standing for Reasoning and Acting). They used LLMs to generate JSON that represented tool calls, and then parsed that JSON to determine what tools to call.

2023-01

OpenAI releases a 'Chat Completion' API.

LangChain Docs

Get started › **Philosophy**

2023-01

LangChain releases a JavaScript version.

LLMs and agents will change how applications are built and JavaScript is the language of application developers.

2023-02

LangChain Inc. was formed as a company around the open source LangChain project.

The main goal was to “make intelligent agents ubiquitous”. The team recognized that while LangChain was a key part (LangChain made it simple to get started with LLMs), there was also a need for other components.

2023-03

OpenAI releases ‘function calling’ in their API.

This allowed the API to explicitly generate payloads that represented tool calls. Other model providers followed suit, and LangChain was updated to use this as the preferred method for tool calling (rather than parsing JSON).

2023-06

LangSmith is released as closed source platform by LangChain Inc., providing observability and evals

The main issue with building agents is getting them to be reliable, and LangSmith, which provides observability and evals, was built to solve that need. LangChain was updated to integrate seamlessly with LangSmith.

LangChain releases 0.1.0, its first non-0.0.x.

The industry matured from prototypes to production, and as such, LangChain increased its focus on stability.

2024-02

LangGraph is released as an open-source library.

The original LangChain had two focuses: LLM abstractions, and high-level interfaces for getting started with common applications; however, it was missing a low-level orchestration layer that allowed developers to control the exact flow of their agent. Enter: LangGraph.

When building LangGraph, we learned from lessons when building LangChain and added functionality we discovered was needed: streaming, durable execution, short-term memory, human-in-the-loop, and more.

2024-06

LangChain has over 700 integrations.

Integrations were split out of the core LangChain package, and either moved into their own standalone packages (for the core integrations) or `@langchain/community` .

LangChain Docs

Get started › **Philosophy**

As developers tried to improve the reliability of their applications, they needed more control than the high-level interfaces provided. LangGraph provided that low-level flexibility. Most chains and agents were marked as deprecated in LangChain with guides on how to migrate them to LangGraph. There is still one high-level abstraction created in LangGraph: an agent abstraction. It is built on top of low-level LangGraph and has the same interface as the ReAct agents from LangChain.

2025-04

Model APIs become more multimodal.

Models started to accept files, images, videos, and more. We updated the `@langchain/core` message format accordingly to allow developers to specify these multimodal inputs in a standard way.

2025-10-20

v1.0.0

LangChain releases 1.0 with two major changes:

1. Complete revamp of all chains and agents in `langchain`. All chains and agents are now replaced with only one high level abstraction: an agent abstraction built on top of LangGraph. This was the high-level abstraction that was originally created in LangGraph, but just moved to LangChain.

   For users still using old LangChain chains/agents who do NOT want to upgrade (note: we recommend you do), you can continue using old LangChain by installing the `@langchain/classic` package.

2. A standard message content format: Model APIs evolved from returning messages with a simple content string to more complex output types - reasoning blocks, citations, server-side tool calls, etc. LangChain evolved its message formats to standardize these across providers.

LangChain Docs

Get started › **Philosophy**

Was this page helpful? Yes No

LangChain Docs

Resources

Forum

Changelog

LangChain Academy

Trust Center

Company

About

Careers

Blog

Powered by mintlify

LangChain Docs

Get started › **Philosophy**